ulm university universität
uulm

Universität Ulm | 89069 Ulm | Germany

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

Effiziente Kontexterkennung mittels aktuellen Cross-Plattform-Frameworks im Rahmen therapeutischen Interventionen

Masterarbeit an der Universität Ulm

Vorgelegt von:

Sabrina Friedl
sabrina.friedl@uni-ulm.de

Gutachter:
Prof. Dr. Manfred Reichert
Dr. Rüdiger Pryss

Betreuer:
Marc Schickler

2018

Fassung 5. Oktober 2018

Satz: PDF-LaTeX 2$_\varepsilon$

Kurzfassung

Durch die große Akzeptanz der modernen mobilen Endgeräte können Anwendungen aus dem Bereich der Gesundheit den Alltag aktiv unterstützen. Die Masterarbeit beschäftigt sich mit der Analyse der Umgebung, um im Rahmen therapeutischer Anwendungen Kontexte zu erkennen. Die in die modernen Endgeräte integrierten Sensoren (z.B GPS-Sensor oder Mikrofon) können die Umgebung für die Kontexterkennung messen.

Mithilfe einer plattformübergreifenden Anwendung soll der Zugriff auf native Sensoren und Hardware verschiedener Betriebssysteme überprüft werden. Dafür wurde eine hybride mobile Anwendung mit dem Framework Ionic entwickelt, die den Benutzer bei seinen Hausaufgaben bestmöglich betreuen und motivieren soll. Dafür sammelt die Anwendung mittels der Sensoren Umgebungsdaten, die regelbasiert miteinander verknüpft werden. Zur Beschreibung der Umgebung oder einer Situation werden Kontexte auf einem Server zentral gespeichert. Wird ein Kontext erfolgreich erkannt, wird der Benutzer informiert und zur Durchführung der Hausaufgabe motiviert.

Da die in der Arbeit beschriebene mobile Anwendung auf die bekanntesten Sensoren eines mobilen Endgerätes zugreifen kann, können beliebige Kontexte und Situationen erkannt werden. Durch die stetige Weiterentwicklung des Frameworks ist den hybriden Anwendungen ein uneingeschränkter Zugriff auf viele native Funktionen möglich. Der Einsatz hybrider Anwendungen ist dennoch vom Anwendungsfall abhängig.

Inhaltsverzeichnis

1 Einleitung

Die vollständige Akzeptanz von modernen mobilen Endgeräten, wie Smartphones oder Tablets, bietet einen großen Markt für IT-Dienstleistungen in Form von mobilen Anwendungen. Diese Anwendungen können aus unterschiedlichen Bereichen stammen, die von Spielen und Unterhaltung über Business bis hin zum Fitness- und Gesundheitssektor reichen. Vor allem im Bereich der Gesundheit wird mit einer vermehrten Integration in den Alltag in zehn Jahren gerechnet, die durch eine Unterstützung der IT erzielt werden soll [1].

Im Rahmen therapeutischer Interventionen verbindet beispielsweise der Einsatz von modernen mobilen Endgeräten die Therapie mit dem Alltag, indem der Patient über die Durchführung einer Hausaufgabe informiert und motiviert wird. Diese Unterstützung ist insbesondere durch die stetige Weiterentwicklung der modernen mobilen Endgeräte möglich. Ausgestattet mit vielfältigen Funktionen und Sensoren sind sie heutzutage ein aktiver Begleiter in einer Vielzahl von alltäglichen Situationen [2]. Mithilfe der Sensoren können unterschiedliche physikalische Größen erfasst und analysiert werden. Zum Bei-

spiel liefert der GPS-Empänger die geografische Position des Endgeräts, die meistens für Navigationsanwendungen verwendet wird. Zur Raumorientierung werden oft der Beschleunigungssensor und der Gyroskope (Rotationssensor) miteinander kombiniert. Des Weiteren wird die Helligkeit der Umgebung mit dem Helligkeitssensor registriert, um die Bildschirmhelligkeit des mobilen Endgeräts anzupassen. Ferner werden Thermometer und Feuchtigkeitsmesser in Smartphones integriert, die Temperatur- und Feuchtigkeitswerte in der Umgebung messen.

Aus diesen Daten lassen sich Rückschlüsse auf die Umgebung und auf den Benutzer ziehen. Um Situationen maschinenlesbar beschreiben zu können, müssen gewonnene Daten aus unterschiedlichen Quellen miteinander verknüpft werden. Diese kontextsensitiven Anwendungen reagieren selbständig auf erkannte Situationen und können ihr Verhalten dementsprechend anpassen oder dem Benutzer gefilterte Informationen darstellen. Im Beispiel von Hausaufgaben kann eine definierte Situation oder Kontext, in der die Hausaufgabe ausgeführt werden soll, zu einem effizienteren Therapieverlauf führen.

1.1 Zielsetzung

Das Ziel dieser Arbeit ist die Konzeption und Umsetzung einer prototypischen mobilen Anwendung mit Anbindung eines Webservices für die Benutzer- und Datenverwaltung. Das zu entwickelnde System soll Patienten im Rahmen von therapeutischen Interventionen bei der Motivation und Durchführung von Hausaufgaben unterstützen. Der Fokus des Systems liegt auf der Kontexterkennung, die für die Durchführung einer Hausaufgabe in einer bestimmten Situation oder Umgebung genutzt wird. Als Basis der Kontexterkennung dienen vordefinierte Kontexte, durch die eine Situation beschrieben werden kann. Erkannt werden die Situationen durch Überwachung und Analyse von nativen Sensoren oder anderen Quellen.

Das Ziel der Arbeit ist zu überprüfen, wie gut native Sensoren oder Funktionen bei der Verwendung von aktuellen Cross-Plattform-Frameworks ansprechbar sind. Für die Entwicklung der mobilen Anwendung wurde das Ionic-Framework verwendet, mit dem Anwendungen u.a. für die Betriebssysteme iOS und Android generierbar sind.

1.2 Struktur der Arbeit

Diese Arbeit untergliedert sich in sieben Kapitel, die aufeinander aufbauen (siehe Abb. 1.1).

Abbildung 1.1: Kapitelübersicht

Nach dem einleitenden Teil folgt das **Kapitel 2**, in dem die grundlegenden Begriffe erläutert werden, die das Verständnis der hier vorliegenden Arbeit erleichtern. Dazu werden zunächst therapeutische Interventionen im Umfeld der Psychologie und die Möglichkeiten der IT-Unterstützung betrachtet. Danach folgen die Definition von Kontexten in mobilen Anwendungen und die Funktionen einer kontextsensitiven Anwendung. Abgeschlossen wird das Grundlagenkapitel durch den Vergleich von plattformübergreifenden und nativen Anwendungen.
Das **Kapitel 3** umfasst die Analyse von Anforderungen. Dazu müssen zunächst verwendete Begriffe erklärt und der Rahmen abgegrenzt werden, in dem die zu entwickelnde Anwendung einzuordnen ist. Diese bilden die Grundlage der aufgelisteten funktionalen und nichtfunktionalen Anforderungen.
Aus den definierten Anforderungen abgeleitet folgt in **Kapitel 4** das Konzept der mobilen Anwendung. Dazu wird das erstellte Modell zur Strukturierung der Daten vorgestellt

sowie der Aufbau der Anwendung anhand der Dialogstruktur gezeigt.
Anschließend folgt das **Kapitel 5**, welches die prototypische Umsetzung der mobilen Anwendung zeigt. Aufbauend auf der Dialogstruktur folgt zu Beginn die Gestaltung der Dialoge, die einzeln ausführlich beschrieben werden. Die nachfolgende Skizzierung der Architektur beschreibt das Zusammenwirken von Client und der einzelnen Komponenten des Webservices. Danach wird der Aufbau eines Kontextes verdeutlicht und auf die Kontexterkennung und -validierung eingegangen. Abschließend werden die verwendeten Technologien und Frameworks dargelegt, die die Umsetzung des Systems vereinfachen.
Um den Erfüllungsgrad jeder Anforderung bewerten zu können, werden in **Kapitel 6** die funktionalen und nichtfunktionalen Anforderungen mit dem aktuellen Stand der umgesetzten mobilen Anwendung verglichen.
Der abschließende Teil dieser Arbeit in **Kapitel 7** bietet eine Zusammenfassung mit einem Ausblick auf zukünftige Erweiterungen oder Verbesserungen.

2

Grundlagen

In diesem Kapitel werden die Grundlagen erläutert, die für das weitere Verständnis der Arbeit notwendig sind. Da die Arbeit im Rahmen von therapeutischen Interventionen entwickelt worden ist, wird dieser Begriff neben den Kontexten in mobilen Anwendungen beschrieben. Abgeschlossen wird das Kapitel durch die Spezifikation der plattformübergreifende Entwicklung.

2.1 Therapeutische Interventionen

Eine therapeutische Intervention wird während einer Therapie vom Therapeuten mithilfe des Patienten und dessen Anamnese in einem persönlichen Gespräch geplant, um aktiv eine Therapie in ihrer Wirksamkeit zu steigern. Die therapeutische Intervention kann eine einfache Medikamenteneinnahme oder eine komplexe Aufgabe sein, die der Patient außerhalb der Therapiesitzung durchführen soll. Therapeutische Interventionen sind sehr vielfältig und kommen nicht nur in der Psychotherapie zum Einsatz, sondern in jedem

Prozess, bei dem Hausaufgaben nützlich sind (z.B. Physiotherapie oder Sozialarbeit) [3].

Einer therapeutischen Intervention liegt eine strukturierte Planung zu Grunde, um die Wirksamkeit einer Therapie positiv zu beeinflussen. In der Psychotherapie beispielsweise gelten therapeutische Hausaufgaben als Standardtechnik, um die Therapie zu intensivieren und zu unterstützen [4]. Hausaufgaben werden in einem persönlichen Behandlungsgespräch dem Patienten vom Psychotherapeut zugewiesen, gemeinsam können zusätzliche Kontexte (z.B. jeden Tag um 19 Uhr) für die Aufgaben abgesprochen werden. Hausaufgaben werden anhand ihrer Funktion in zwei Oberkategorien *Kognitive Aufgaben* und *Verhaltensbezogene Aufgaben* eingeteilt. Dabei gehören zu den Kognitiven Aufgaben z.B. das Beobachten und Protokollieren von Symptomen oder die Reflexion eines bestimmten Themas. Konfrontationsaufgaben, kreative Aufgaben oder Aufgaben das Verhalten zu ändern oder zu festigen, gehören unter anderem zu den Verhaltensbezogenen Aufgaben [4].

Durch Hausaufgaben werden Therapieinhalte außerhalb der eigentlichen Therapiestunde erneut ins Bewusstsein gerufen und der Patient kann erlebte Fähigkeiten im Alltag festigen. Durch eine Verbindung zwischen Therapie und Alltag kann außerdem der Therapieprozess durch neue Informationen bereichert werden. Der Therapieprozess kann ebenfalls durch einen motivierten Patienten effizient gesteigert werden, wenn eine Aufgabe leicht durchgeführt werden kann und dadurch der Patient einen Fortschritt erlebt [4].

Jedoch ist die Erledigung der Hausaufgaben ohne eine Überwachung oder eine Rückmeldung nur in der nächsten persönlichen Behandlung retrospektiv überprüfbar [5, 6]. Bei einer erforderlichen Anpassung der Hausaufgabe kann sich durch Warten auf die nächste Sitzung die Therapiedauer verlängern. Der Therapieprozess kann durch den Einsatz von mobilen Interventionen optimiert werden, in dem technologischen Ansätze zur Durchführung von Hausaufgaben außerhalb der Therapiesitzung genutzt werden. Im Gegensatz zu herkömmlichen (z.B. auf Papier basierten) Hausaufgaben können Daten durch den Einsatz von modernen mobilen Endgeräten in Echtzeit zwischen den

Therapiesitzungen erfasst werden. Die Daten geben Aufschluss über den aktuellen Zustand des Benutzers oder über den Status der Hausaufgabe [5, 7].

Grundsätzlich lassen sich solche mobile Interventionen nach der Art der Kommunikation zwischen Therapeut und Patient kategorisieren. Eine direkte Kommunikation besteht bei live-therapeutischen Gesprächen mittels Smartphones, wie es z.B. bei der Telefonpsychotherapie der Fall ist. Dagegen gehört der Austausch von Informationen über Textnachrichten bereits zu der asynchronen Kommunikation. Auch die Art der Kommunikation selbst ist ein wichtiger Teil von mobilen Interventionen. Diese kann z.B. wie bei einfachen Medikamentenerinnerungen standardisiert oder auf den Benutzer und sein Umfeld angepasst werden. Auch Therapeuten unterstützen eine informationstechnische Lösung, die es ihnen ermöglicht Hausaufgaben flexibel und individualisiert zu definieren und den gesamten Therapievorgang effizienter zu gestalten [6].

Aus diesem Grund ist das MobileTx-Projekt (Albatros) entwickelt worden und bildet die Grundlage dieser hier vorliegenden Arbeit, indem es therapeutische Prozesse durch IT-Systeme unterstützt (siehe Abb. 2.1). Dessen Ziel ist es durch den Einsatz von modernen mobilen Endgeräten (wie z.B. Smartphones) therapeutische Interventionen bestmöglich in den Alltag des Patienten zu integrieren [9].

Beispielsweise kann das moderne mobile Endgerät den Patienten über eine zugewiesene Hausaufgabe informieren und zu einer regelmäßigen Durchführung motivieren. Durch die Verwendung von modernen Endgeräten können außerdem verschiedene Medien für die Hausaufgabe genutzt werden. Beispielsweise ist ein Video eine bessere Unterstützung für den Patienten, als eine textuell beschriebene Anweisung einer Hausaufgabe. Auf der anderen Seite hilft der Einsatz von IT-Systemen dem Therapeuten eine effiziente Überwachung der Behandlung in der natürlichen Umgebung des Benutzers. Er kann gegebenenfalls einschreiten und den Therapieplan dynamisch anpassen [8].

Die Herausforderung des Projektes ist die Entwicklung eines Systems, das die Vielfalt der therapeutischen Interventionen abdeckt und sich nicht nur auf die Beziehung zwischen Therapeut und Patient konzentriert. Werden Forscher in das System mit einbezogen, können sie von den gesammelten Daten profitieren und neue Therapieformen entwickeln. Außerdem ist es sinnvoll, die vielfältigen Funktionen der modernen mobilen

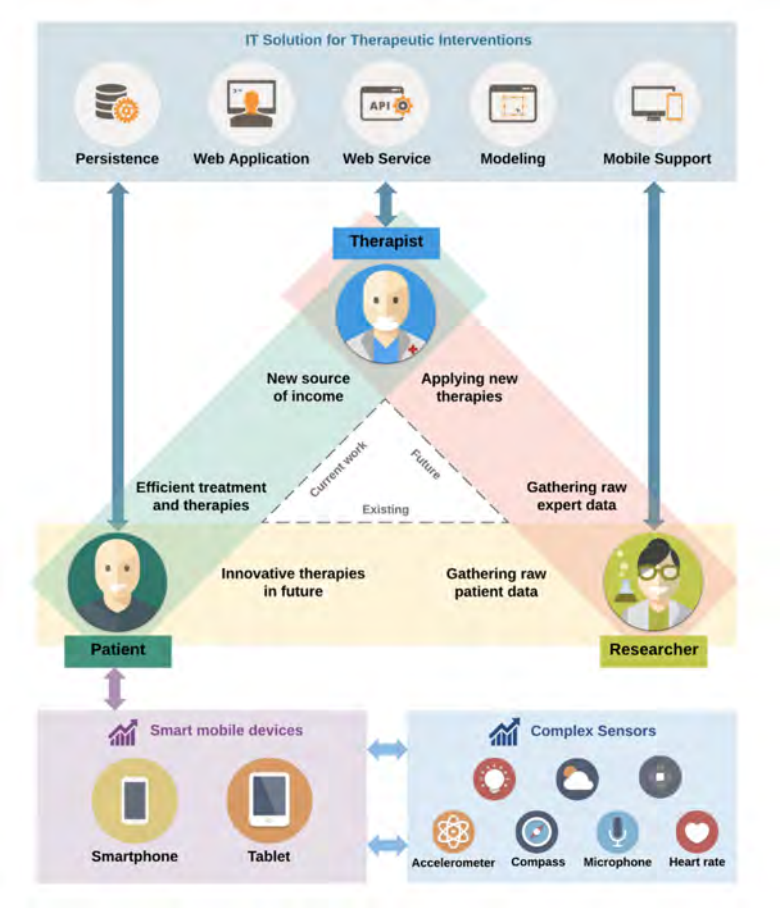

Abbildung 2.1: IT-Unterstützung von therapeutischen Interventionen [8]

Endgeräte für therapeutische Interventionen bestmöglich zu nutzen [8]. So bieten die mobilen Endgeräte nicht nur den Vorteil einer direkten Interaktion zwischen Patient und Therapeut, sondern ermöglichen die Nutzung der integrierten Sensoren. Durch die Sensoren können Daten gesammelt werden, die der Forschung als Grundlage dienen können. Sensoren können aber auch für eine Kontexterkennung genutzt werden, wenn die Wirksamkeit der Hausaufgabe durch einen passenden Kontext (z.B. jeden Tag um 19 Uhr) gesteigert werden soll.

2.2 Kontext in mobilen Anwendungen

> *"Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and application themselves [10]."*

Allgemein ist eine Unterstützung des Benutzers, insbesondere eine Verbesserung des Benutzererlebnisses, die Grundidee den Kontext in mobilen Anwendungen zu integrieren.

Als Kontexte sind Informationen zu verstehen, die zur Beschreibung einer Situation eines Benutzers in einer Interaktion verwendet werden können. Ein Kontext besteht aus mindestens einer Information oder aus einer Kombination vieler Informationen aus unterschiedlichen Quellen. Es gibt eine Vielzahl von Kontextarten, die zur Beschreibung einer Situation möglich sind. Beispielsweise kann eine Situation durch Ortsinformationen, den Zeitkontext, die physikalische Umwelt, den Gerätekontext oder durch Aktivitäten und Interaktionen definiert werden [11].

Mobile Anwendungen haben ein großes Potential bezüglich der Integration von Kontexten und Kontexterkennung. Schließlich bieten die modernen mobilen Endgeräte von heute kostengünstig leistungsfähige Sensoren, die das Sammeln von kontextuellen Informationen erleichtert. Aufgrund der Verbreitung durch App Stores oder das World Wide Web steht eine mobile Anwendung einer großen Anzahl an Benutzers weltweit zur Verfügung. Wissenschaftlern ist es dadurch möglich, viele Daten global zu analysieren und zu sammeln [12].

Eine mobile Anwendung kann Kontexte und somit Situationen nicht sofort erkennen und verstehen, um das Benutzerverhalten zu optimieren. Deshalb muss sie einen Prozess an Funktionen durchlaufen, der in der Abb. 2.2 dargestellt ist. Zuerst müssen Rohdaten gesammelt werden, die im nächsten Schritt so weiter verarbeitet werden, so dass im dritten Schritt eine Entscheidung getroffen werden kann, ob ein Kontext erkannt wurde. Bei einer erfolgreichen Kontexterkennung können Aktionen oder Anpassungen der mobilen Anwendung ausgeführt werden.

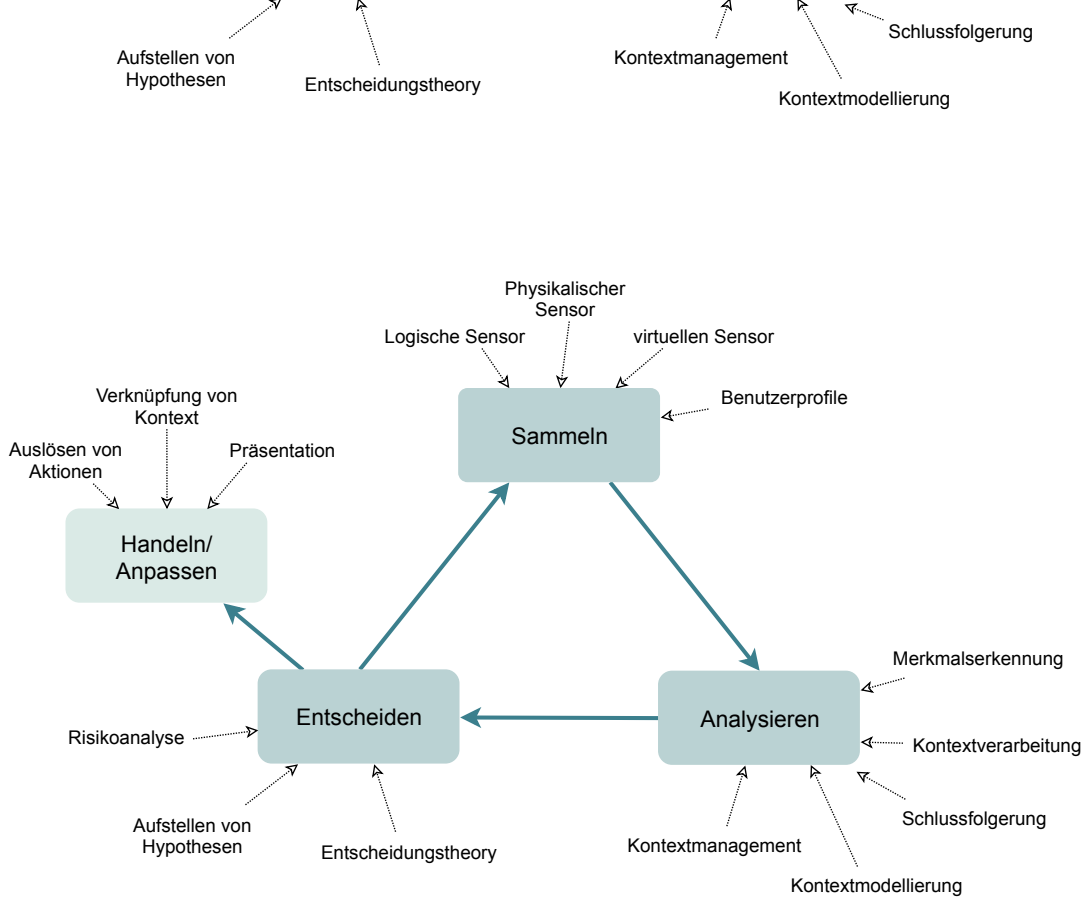

Abbildung 2.2: Funktionen einer kontextsensitiven Anwendung (vgl. [11])

2.2.1 Kontexterfassung

Die gesammelten Rohdaten können aus verschiedenen Sensortypen oder aus dem Benutzerprofil stammen. Mobile Anwendungen haben aufgrund der in das mobile Endgerät integrierten Sensoren verschiedene Kontextquellen, mit denen der Zustand der Umgebung beobachtet werden kann. Unterschieden werden physikalische, logische und virtuelle Sensoren. Die am häufigsten verwendeten physikalische Sensoren sind meist Hardware-Sensoren, die ein mobiles Endgerät besitzt. Das sind beispielsweise GPS-Sensoren, Beschleunigungssensoren, Mikrofone, Thermometer oder Barometer. Virtuelle Sensoren hingegen beziehen ihre Daten aus Softwareanwendungen oder Diensten. Der elektronische Kalender kann z.B. weitere Informationen zum Standort des Benutzers geben [13]. Auch das Überwachen der Benutzerinteraktionen, z.B. Eingaben über die Tastatur geben Aufschluss über die Aktivitäten des Benutzers. Die logischen Sensoren kombinieren mehrere Informationsquellen aus physikalischen und virtuellen Sensoren und verknüpfen diese mit zusätzlichen Informationen aus Datenbanken oder

verschiedenen anderen Quellen. Mithilfe der logischen Sensoren können abstrakte Kontexte dargestellt werden, wie z.B. Emotionen oder die Stimmungslage [14]. Ein schnell ansteigender Puls kann beispielsweise ein Stresssymptom sein. Eine weitere Bezugsquelle für Kontextinformationen können Benutzerprofile sein. Aus individuellen Einstellungen, Interessen oder Gewohnheiten können ebenfalls Kontextinformationen abgeleitet werden, wie z.B. der Wohnort des Benutzers.

Beispiel "Benutzer schläft" Ein System kann diese reale Alltagssituation nur durch Sensoren erkennen. Die Umgebungshelligkeit, die Umgebungslautstärke, der Standort und die Uhrzeit sind mögliche relevante Sensoren, die einem mobilen Endgerät zur Verfügung stehen. Um eine genauere Bestimmung des Kontextes zu ermöglichen, sind Informationen wie die Raumtemperatur, die Herzfrequenz und die horizontalen Lage des Benutzers ebenfalls nützlich [15].

2.2.2 Kontextanalyse

Die Anwendung kann alleine durch das Erfassen und Sammeln von Messdaten noch keinen Kontext erkennen. Dafür müssen die Messdaten so aufbereitet und interpretiert werden, dass ein Informationsgewinn und Analyse möglich ist. Eine Kontexterkennung anhand der Rohdaten wäre ohne diese Vorverarbeitung nicht möglich, da die (physikalischen) Messgrößen zunächst in eine elektrische Größe umgewandelt werden müssen [16]. Die umgewandelten Daten können dann vom System wie im folgenden Beispiel interpretiert werden:

- Temperatur: 18 ° C
- Uhrzeit: 23:30 Uhr
- Umgebungshelligkeit: 0,02 lx

Da typischerweise der genaue elektrische Wert für den Kontext irrelevant ist, muss das Abstraktionslevel erhöht werden. Dadurch entstehen aus den elektrischen Werten bereits einfache Kontexte. Um den Kontext "Schlafen" zu erkennen, ist der exakter

Wert der Umgebungshelligkeit nicht relevant. Entscheidend für das Schlafen ist die dunkle Umgebung, die ideale Schlaftemperatur und die Tageszeit. Aus der Kombination einfacher Kontexte kann ein komplexer Kontext, wie das Schlafen, abgeleitet werden. Je mehr Informationsquellen einer Situation zugrunde liegen, desto komplexer ist der Kontext.

Für die Darstellung, Speicherung und zum Austausch von Kontextinformationen gibt es verschiedene Kontextmodelle, die Kontextdaten für Maschinen lesbar definieren [17, 14].

Key-Value Modell:
Daten werden durch Key-Value Paare repräsentiert, indem einer Kontextinformation (Value) ein eindeutiger Schlüssel (Key) zugeordnet wird. Das Modell stellt die einfachste Art der Datenstruktur dar und ermöglicht eine leichte Verwaltung von Key-Value-Paaren, die allerdings für eine effiziente Kontextabfrage nicht optimal strukturiert ist.

Markup-Schema Modell:
Alle Markup-basierten Modelle verwenden eine hierarchische Datenstruktur, die aus Markup-Tags mit Attributen und Inhalten bestehen. Aufgrund der verwendeten Markup-Sprachen (z.B. XML) ist die Darstellung des Kontextes über ein festgelegtes Schema möglich.

Grafisches Modell:
Auf Basis von UML (Unified Modelling Language) können Kontextinformationen aufgrund ihrer generischen Struktur graphisch beschrieben werden.

Objektorientiertes Modell:
Die Modellierung des Kontextes mittels objektorientierter Techniken bietet Möglichkeiten der Vererbung, Wiederverwendbarkeit und Kapselung. Kontexttypen (wie z.B. Ort oder Helligkeit) können durch verschiedene Objekte dargestellt werden. Details der Kontextdarstellungen können gekapselt werden. Der Zugriff auf den Kontext erfolgt über klar definierte Schnittstellen.

Logisches Modell:
Um ein Kontextmodell logisch zu definieren, können Fakten, Ausdrücke und Regeln für die Darstellung eines Kontext verwendet werden. Logikbasierte Modelle haben ein hohes Maß an Formalität, das erlaubt, kontextuelle Fakten und Zusammenhänge präzise und nachvollziehbar zu beschreiben. Durch eine genaue Begriffsdefinition werden Daten und deren Bedeutung auf die gleiche Art und Weise interpretiert.

Ontologie-basiertes Modell:
Ontologien stellen eine Beschreibung der Konzepte und Zusammenhänge dar. Kontexte, die als Ontologien modelliert werden, sind besonders für die Beschreibung von Alltagssituationen geeignet und bieten der Anwendung Schlussfolgerungstechniken auf Basis der Ontologien.

Schlussendlich sollte das Kontextmodell so definiert sein, dass Kontexte einfach zu definieren sind, jedoch sollte das Modell flexibel sein und das Hinzufügen neuer Kontextelemente ermöglichen. Außerdem sollten mithilfe des Modells möglichst viele Kontextzustände beliebig detailliert abgebildet werden können, um Fehlinterpretationen zu vermeiden.

2.2.3 Schlussfolgerung

Nachdem die Sensoren die Umgebungsdaten aufgezeichnet haben und die Kontexte entsprechend modelliert wurden, muss eine Entscheidung getroffen werden, ob ein definierter Kontext vorliegt.

Zur Entscheidungsfindung können verschiedene Methoden aus dem Bereich der künstlichen Intelligenz und wissensbasierter Systeme eingesetzt werden. Diese Methoden sind vor allem vom Anwendungsbereich, den verfügbaren Informationen und dem gewählten Kontextmodell abhängig. Insbesondere werden Trainingsdaten, Kenntnisse über bedingte Wahrscheinlichkeiten oder historische Zusammenhänge zur Erkennung von

Benutzeraktivitäten benötigt. Solche Kontexte benötigen maschinelles Lernen, um z.B. von Rohdaten eines Beschleunigungssensors auf eine komplexe Bewegung schließen zu können.

Zur Erkennung von Situationen bieten sich Verfahren an, die auf Fakten und Regeln basieren, da mögliche Zustände mithilfe von Vorwissen definiert werden können. Die Faktenbasis wird aus den aufbereiteten Informationen durch eine stetige Überwachung generiert. Bei jeder Aktualisierung werden die Fakten auf passende Bedingungen durch vordefinierte Regeln überprüft. Wie in dem Beispielkontext ”Schlafen” ist Vorwissen nötig, um von den absoluten Eingangswerten des Helligkeitssensors auf eine dunkle Umgebung zu schließen. Eine Regel definiert beispielsweise den Schwellenwert, der zwischen hell und dunkel unterscheidet. Solche booleschen Ausdrücke verhindern Ungenauigkeiten und ermöglichen eine Kontextmodellierung mithilfe des logikbasierten Modells.

2.2.4 Handeln und Anpassen

Bei einer erfolgreichen Kontexterkennung können Aktionen oder Anpassungen der mobilen Anwendung ausgeführt werden. Dabei kann die Anwendung auf drei verschiedene Arten reagieren: Kontextpräsentation, Kontextausführung oder Kontextverknüpfung. In Abhängigkeit des Kontextes kann die Anwendung gefilterte oder speziell auf den Benutzer angepasste Informationen darstellen. Die Anwendung kann aber auch ihre Verhaltenslogik dem Kontext anpassen, indem automatisch eine Aktion ausgelöst wird. Bei der letzten Reaktionsmöglichkeit werden der Kontext oder die kontextuelle Informationen mit einem Inhalt verknüpft, das für eine spätere Verarbeitung nützlich sein kann [11].

2.3 Aktuelle Cross-Plattform Frameworks

Neben der Möglichkeit *native mobile Anwendungen* für eine spezifische mobile Plattform (z.B. iOS oder Android) zu entwickeln, sind mobile *Web-Anwendungen* und *hybride*

mobile Anwendungen betriebssystemunabhängig und gehören zu den Cross-Plattform Frameworks.

Die spezielle Anpassung an das Betriebssystem ermöglicht den nativen Anwendungen über das *Software Development Kit (SDK)* eine direkte Nutzung der nativen Plattform und aller plattformspezifischen Hard- und Software-Funktionen, wie z.B. der Zugriff auf den lokalen Speicher oder auf die Sensoren des modernen mobilen Endgeräts. Diese Spezialisierung erfordert einen hohen Arbeitsaufwand und somit erhöhte Kosten, wenn für jede Plattform eine eigene native Anwendung parallel entwickelt werden muss. Wegen den unterschiedlichen Betriebssystem typischen Programmiersprachen (z.B. Swift oder Java) müssen mehrere *Codebases* verwaltet und gewartet werden [18].

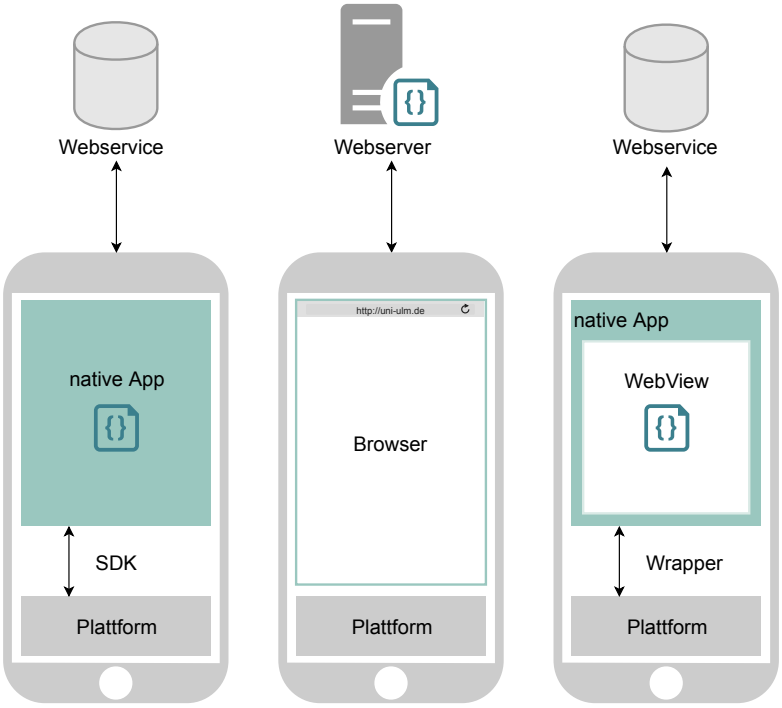

Abbildung 2.3: Vergleich zwischen nativer, Web-und Hybrid-Anwendungen (vgl. [19, 20])

Aufgrund verschiedener führender mobilen Betriebssysteme bieten sich Cross-Plattform Frameworks an, um die Kosten zu senken [18]. Die Entwickler erstellen von einer Codebasis ausgehend eine mobile Anwendung, die mit minimalem Aufwand auf mehreren Plattformen eingesetzt werden können. Plattformübergreifende Anwendungen können in

jedem (mobilen) Browser, als Progressive Web App (PWA) oder als mobile Anwendung gestartet werden.

Mobile Web-Anwendungen sind Webseiten, die über einen integrierten Webbrowser des modernen Endgeräts aufgerufen werden. Da keine Installation auf dem Endgerät notwendig ist, ist die Anwendung auf jedem mobilen Endgerät mit einem Browser verwendbar. Die Entwicklung von mobilen Web-Anwendungen beruht auf Webstandards, wie HTML5 und Javascript, somit ist das Verhalten äquivalent zu Webseiten. Die Web-Anwendung wird im Browser mit einer eingeschränkte Benutzeroberfläche dargestellt, so dass dem Benutzer die Darstellung als Webseite nicht unbedingt ersichtlich ist.

Bei der Benutzung werden alle Daten für die visuelle Darstellung und den Inhalt der mobilen Web-Anwendung von einem Webserver geladen. Aus diesem Grund entsteht ein höherer Datenverkehr als bei nativen Anwendungen, die für das Aussehen und die Programmlogik selbst verantwortlich sind und nur inhaltsbezogene Daten mit einem Webservice austauschen (siehe Abb. 2.3) [19]. Der erhöhte Datenverkehr kann sich bei einer begrenzten oder langsamen Internetverbindung zu einem Nachteil auswirken. Ebenfalls fehlt der Zugriff auf alle nativen Gerätefunktionen, Ausnahmen bilden die vom Browser bereitgestellten Schnittstellen [19, 20]. So unterstützen die aktuellen Browser beispielsweise die Funktion der Geolokalisierung.

Als eine Besonderheit der Web-Anwendung haben sich *Progressive Web Apps* (PWAs) etabliert [21]. Diese verbinden alle Vorteile einer Web-Anwendung mit Funktionen nativer Anwendungen (siehe Tabelle 2.1). Da die Publikation von PWAs über eine URL erfolgt und nicht über einen App Store, ist die Anwendung ohne manuelle Updates immer auf dem aktuellen Stand. Eine PWA verhält sich für den Benutzer durch die Möglichkeit einer Installation auf dem Homescreen ähnlich einer nativen Anwendung. Ein weiterer Vorteil bringt die Unabhängig der Internetverbindung durch die Offline-Nutzung von PWAs.

Verantwortlich dafür ist ein Script (Service Worker), das im Hintergrund getrennt von der Anwendung auf Events reagiert. Durch den Service Worker sind Offline-Funktionen, Push-Notifications, Aktualisierung von Inhalten im Hintergrund oder Caching möglich [22]. Aber auch die PWAs haben keinen Zugriff auf Hardware spezifische Funktionen.

	Progressive Web App	Web Anwendung	Native Anwendung
Plattformübergreifend	✓	✓	
Publikation ohne App-Store	✓	✓	
Keine Updates notwendig	✓	✓	
Hardwarefunktionen			✓
Installation (Homescreen)	✓	(✓)	✓
Offline-Funktion	✓		✓
Push-Notifications	✓		✓

Tabelle 2.1: Vergleich von Progressive Web Apps gegenüber Web Anwendungen und nativen Anwendungen (vgl. [23])

Hybride mobile Anwendungen vereinen die Betriebssystemunabhängigkeit mit der Möglichkeit auf native Gerätefunktionen zu zugreifen. Die hybride Form der mobilen Anwendung enthält eine Browser ähnliche Komponente (*WebView*), die innerhalb der nativen Umgebung eingebunden wird. Aufgrund der benötigten Webtechnologien ist die Entwicklung hybrider Anwendung ähnlich der Web-Anwendung, bzw. Webseiten mit HTML, CSS und Javascript. Um trotzdem einen Zugriff auf plattformspezifische Funktionen zu erhalten, verbindet ein *Wrapper* die nativer Geräteplattform mit der WebView. Der Wrapper wird von Frameworks wie Apache Cordova zur Verfügung gestellt, mit dem zusätzlich das Auslesen verschiedener Sensoren möglich ist. Jedoch haben hybride Anwendungen gegenüber den nativen Anwendungen Einschränkungen. Trotz der nativen Integration ist die Leistung von der Qualität des Browsers der Plattform abhängig [21]. Ein Zugriff auf native Schnittstellen ist nicht immer gewährleistet, da dafür meist ein zusätzlich entwickeltes Plugin von Drittanbietern eingebunden werden muss. Ein weiteres Hindernis ist die Benutzerfreundlichkeit. Benutzer unterschiedlicher Betriebssysteme erwarten die gewohnten nativen Bedienelemente. Allerdings gibt es für diese Probleme Frameworks, wie z.B. *Ionic*, die sich auf die Gestaltung der verschiedenen Benutzeroberflächen spezialisieren.

Die Tabelle 2.2 fasst alle wesentlichen Unterschiede bezüglich der nativen und plattformübergreifenden Entwicklung zusammen. Vor allem bei Anwendungsfällen, bei denen eine hohe Grafikleistung oder allgemein eine hohe Performance erforderlich ist, sind native Anwendungen besser geeignet. Grundsätzlich muss dafür ein hoher Entwicklungsaufwand in Kauf genommen werden. Vor allem in diesen Punkten bieten sich Cross-Plattform Anwendungen an. Bei geringen Aufwand kann schnell ein großer Benutzerkreis abgedeckt werden [18].

	Nativ	**Hybrid**
Benötigtes Entwickler-Know-How	Swift/Objective-C, Java, iOS SDK, Android SDK	HTML, CSS, Javascript
Art der Publikation	App Store	App Stores, Desktop/Mobile Browser
Entwicklungsaufwand	hoch	gering
Wartungskosten	hoch	mäßig/gering
Grafikleistung	hoch	mäßig
Performance	hoch	abhängig vom Anwendungsfall
Zugriff auf native Funktionen	vollständige native Library	vollständige native Library (erfordert Plugins von Drittanbietern)
UX-Konsistenz über Plattformen und Geräte hinweg	erfordert mehrere Anwendungen	Ja

Tabelle 2.2: Zusammenfassende Gegenüberstellung der Entwicklung nativer und plattformübergreifender Anwendungen (vgl. [18])

2.3.1 Ionic

Ionic ist ein Open-Source-SDK, das zur Erstellung hybrider mobiler Anwendungen basierend auf den bekannten Webtechnologien HTML5, CSS und Javascript/Typescript verwendet wird. Das Framework baut auf Angular (ein Web Application Framework) und Cordova auf, das für die Erstellung nativer Anwendungen benötigt wird [24, 19].

Das Hauptziel von Ionic ist die Anpassung der Benutzeroberfläche und der Benutzerinteraktionen an die nativen Endgeräte. Dazu erweitert Ionic die HTML-Bibliothek um native mobile Benutzerelemente, wie z.B. Tabbars, um plattformspezifische Besonderheiten zu unterstützen. Dazu zählen auch die Funktionsweise von z.B. Listen- oder Formularelementen. Diese Benutzerelemente bilden eine nahezu native Schnittstelle innerhalb einer hybriden Anwendung.

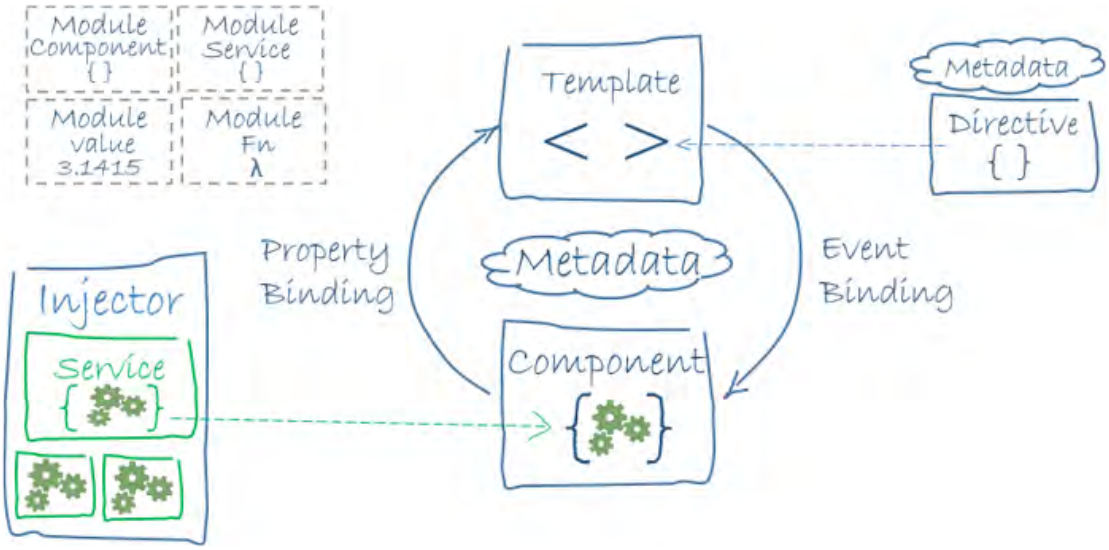

Abbildung 2.4: Architektur einer Angular-Anwendung [25]

Während das Framework Ionic selbst hauptsächlich für das Visuelle verantwortlich ist, bietet Angular die Architektur einer hybriden Ionic-Anwendung (siehe Abb. 2.4). Sie ermöglicht die Entwicklung komplexer, leistungsfähiger Single-Page-Webanwendungen, die dem Prinzip des Model-View-ViewModel folgt [26]. Bei einer Ionic-Anwendung läuft Angular innerhalb des WebViews und ist z.B. für das Routing und für die Verwaltung der

Logik verantwortlich.

Der Hauptteil der Angular-Architektur besteht aus Komponenten, die eine baumförmige Programmstruktur bilden. Dabei verzweigt sich die Root-Komponente in die einzelnen Sub-Komponenten. Komponenten definieren Views, die Angular entsprechend der Programmlogik auswählen und modifizieren kann. Jede Komponente definiert eine *Komponentenklasse*, der ein *Template* und optional ein oder mehrere *Stylesheets* zugeordnet sind. Die Klasse stellt die View-Logik zur Verfügung, die zur Verwaltung der Benutzeroberfläche benötigt wird. Die Präsentation und Strukturierung der Benutzeroberfläche wird im Template definiert, das HTML5 mit Angular Markup kombiniert. Die im Template dargestellten Logikdaten werden via *Property Binding* von der Komponentenklasse eingebunden. Mit *Event Binding* kann die Anwendung auf Benutzereingaben reagieren und die Logikdaten aktualisieren. Die eigentliche Geschäfts-Logik einer Anwendung wird in separaten Services ausgelagert, die Komponenten über definierten Schnittstellen ansprechen können [25].

Der Zugriff auf die nativen Gerätefunktionen und die Erstellung in die nativen Anwendungen verschiedener Betriebssysteme erfolgt mit Apache Cordova. Dabei ist der *Cordova App Wrapper* die native Anwendung, die auf dem mobilen Endgerät installiert wird und den Code der Anwendung in die WebView lädt. Der Wrapper bildet über eine Javascript Schnittstelle die Verbindung zwischen der Webanwendung und der nativen Plattform.

2.3.2 Xamarin

Das Framework *Xamarin* ermöglicht das Entwickeln von nativen Anwendungen u.a. für Android und iOS mittels einer Codebasis (*C#*) und plattformspezifischen Benutzeroberflächen [27]. Xamarin-Anwendungen verhalten sich den nativen Anwendungen sehr ähnlich, da vor allem die Codebasis für Geschäftslogik, Datenbankzugriff und Netzwerkkommunikation plattformübergreifend genutzt wird. Die Benutzeroberfläche wird plattformspezifisch entwickelt, indem native Bedienelemente und Layouts verwendet werden (siehe Abb. 2.5 links).

Das Framework bietet das Werkezeug *Xamarin.Forms* an, mit dem plattformübergreifender UI-Code entwickelt werden kann (siehe Abb. 2.5 rechts). Die Perfomance der Anwendung wird durch diese breite Codegenerierung reduziert. Xamarin bietet sich nicht bei einer kleinen Menge an gemeinsam genutztem Code an. Das ist z.B. beim Erstellen von Spielen, bei einer umfangreichen benutzerdefinierten Benutzeroberfläche oder bei komplexen Animationen der Fall [28].

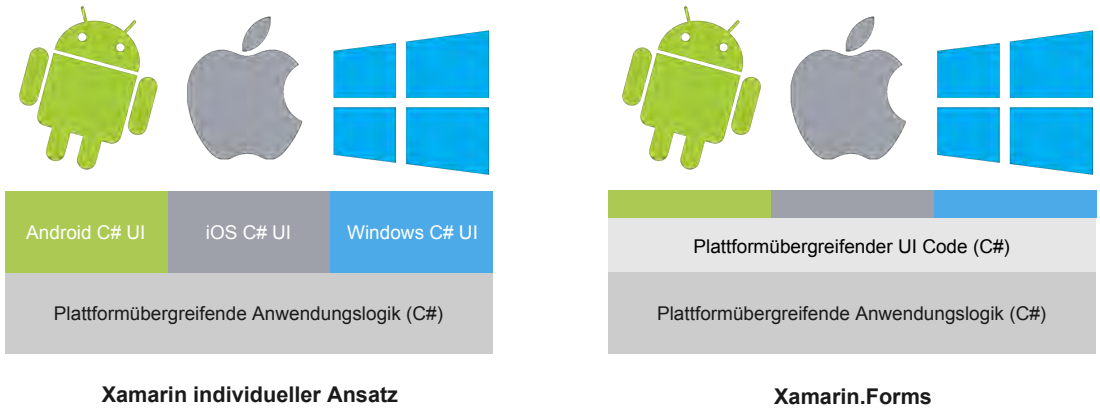

Abbildung 2.5: Architektur von Xamarin und Xamarin.Forms (vgl. [28])

Xamarin wird nativ kompiliert und ist deshalb besonders für komplexe Anwendungsfälle geeignet, bei denen eine hohe Performance für eine schnelle und flüssige Interaktion wichtig ist. Die Kompilierung erfolgt plattformspezifisch. In iOS wird der Quellcode direkt in nativen ARM-Assembly-Code vor dem Start der Anwendung kompiliert *(Ahead-of-time (AOT) Kompilierung)*, in Android wird ein zunächst erstellter Zwischencode zur Laufzeit in nativen Assembler-Code kompiliert *(Just-in-time (JIT) Kompilierung)* [29].

Mit Xamarin kann die Anwendung auf alle nativen Funktionen zugreifen. Neben dem Zugriff auf plattformspezifische APIs unterstützt Xamarin die Verknüpfung mit nativen Bibliotheken.

Die Tabelle 2.3 fasst alle Unterschiede zwischen den Cross-Plattform Frameworks Xamarin und Ionic zusammen. In der Tabelle wurde ebenfalls Xamarin.Forms berücksichtigt, das sich in der Verwendung des plattformübergreifenden Codes, in der Perfomance und in den Anwendungsfällen von Xamarin unterscheidet.

	Xamarin	**Ionic**
Entwicklungssprache	C#	HTML, CSS, Typescript, Javascript
Kompilierung	iOS: AOT Android: AOT/JIT	JIT
UI-Rendering	Native UI-Elemente	HTML, CSS
Plattformübergreifender Code	Geschäftslogik, Datenbankzugriff, Netzwerkkommunikation[1]	bis zu 98% wiederverwendbare Code
Performance	nahezu nativ[2]	mäßig/gering (abhängig vom Anwendungsfall)
Anwendungsfälle	alle Anwendungen[3]	einfache Anwendungen/ Unternehmensanwendungen

Tabelle 2.3: Zusammenfassende Gegenüberstellung der Frameworks Xamarin und Ionic

Der Einsatz von Xamarin ist vor allem geeignet, wenn der Fokus auf der nativen Benutzeroberfläche liegt und diese plattformspezifisch entwickelt werden kann. Ionic bietet für eine Anwendung mit ganzheitlichem plattformübergreifendem Code die gleiche Performance wie Xamari.Forms. Die Entwicklung einer Ionic-Anwendung wird zudem durch die Verwendung bekannter Webtechnologien erleichtert.

[1] Bis zu 96% wiederverwendbare Code mit Xamarin.Forms

[2] Mäßig/geringe Perfomance mit Xamarin.Forms

[3] Einfache Anwendungen/ Unternehmensanwendungen mit Xamarin.Forms

3

Anforderungsanalyse

Zu Beginn der Anforderungsanalyse werden grundlegende Begriffe, sowie Anforderungen an die Anwendung und das System allgemein definiert. Bevor die Anforderungen ermittelt und beschrieben werden, wird der Anwendungskontext und die Architektur des Gesamtsystems erläutert, in dem die in diesem Rahmen entwickelte Anwendung eingebunden ist. Danach folgt die Auflistung der Anforderungen, die in funktionale und nichtfunktionale Anforderungen aufgeteilt werden.

3.1 Fachwissen

Für ein einheitliches Verständnis werden verwendete Begriffe zum Thema therapeutische Hausaufgaben und Kontexte beschrieben und erläutert. Obwohl die Anwendung generisch entwickelt werden soll, werden Begriffe aus der Psychotherapie verwendet. Jedoch lassen sich die Begriffe auch auf andere mögliche Anwendungsfälle übertragen, wie z.B. Lehrer-Schüler- anstatt Therapeut-Patienten-Beziehung.

Begriff	Betreuer
Synonym	Therapeut
Beschreibung	Der Therapeut kann sowohl aus medizinischer Sicht ein behandelnder Arzt oder ein Psychotherapeut, als auch im Allgemeinen ein Betreuer (z.B. ein Lehrer) sein. Allgemein betreut der Therapeut einen Patienten bei einer Behandlung. Dabei plant der Betreuer die Behandlung, erstellt und verwaltet die Hausaufgaben.

Begriff	Benutzer
Synonym	Patient
Beschreibung	Der Patient ist im Rahmen dieser Arbeit der Benutzer der mobilen Anwendung. Ein Patient hat dabei mindestens einen Therapeuten mit dem eine oder mehrere Behandlungen durchgeführt werden.

Begriff	Behandlung
Synonym	Therapie
Beschreibung	Eine Behandlung oder auch Therapie beschreibt einen Prozess, um ein bestimmtes Ziels eines Patienten zu erreichen. Jede Behandlung wird von einem Betreuer begleitet, der den Therapieplan jederzeit anpassen kann. Ein Benutzer kann sich in einer gerade laufenden Behandlung befinden oder hat Behandlungen bereits abgeschlossen.

Begriff	Aufgabe
Synonym	Hausaufgabe
Beschreibung	Jede Behandlung hat eine oder mehrere Hausaufgaben, die der Benutzer außerhalb der eigentlichen Therapiesitzung ausführen soll. Jede Aufgabe ist individuell durch die Auswahl eines Kontextes, sowie

bestimmte Übungen und einem Feedback auf den Benutzer angepasst. Der Benutzer wird darüber informiert, dass eine Aufgabe durchgeführt werden muss.

Begriff Übung

Beschreibung Die Durchführung einer Aufgabe erfolgt durch das Absolvieren verschiedener Übungen, die sequenziell abgearbeitet werden. Eine Übung besteht aus einer textuellen Beschreibung und einem digitalen Medium, wie Bilder, Audios oder Videos. Übungen werden allgemein definiert und können verschiedenen Aufgaben zugeordnet werden. Nach der Durchführung der Übung erfolgt ein Feedback, das der Benutzer seinerseits dem Betreuer geben soll.

Begriff Feedback

Beschreibung Das Feedbackformular besteht aus Fragen oder Angaben, die der Benutzer nach Absolvieren aller Übungen beantworten soll. Das Feedback gibt dem Betreuer eine direkte Rückmeldung, wie der Benutzer die Übung absolvieren konnte. In Abhängigkeit des Ergebnisses kann die Therapieplanung angepasst und bestenfalls beschleunigt werden.

Begriff Kontext

Beschreibung Ein Kontext beschreibt eine Situation, die für eine erfolgreiche Durchführung einer Aufgabe notwendig sein kann. Ein Kontext besteht aus der Verknüpfung einzelner Kontextmerkmale oder weiterer Kontexte, die einen komplexen Kontext bilden.

Begriff Kontextmerkmale

Beschreibung Ein einzelnes Kontextmerkmal beschreibt eine Bedingung in der

	Umgebung, die in elementaren, symbolischen oder sensorischen Bedingungen gegliedert werden können. Während die symbolischen Bedingungen Fakten beschreiben, die vom Benutzer ausgewertet werden müssen, können die sensorischen und elementaren Bedingungen von den modernen mobilen Endgeräten (mithilfe ihrer Sensoren) gemessen werden.
Beispiel	symbolisch: Standort (zu Hause), Aktivität (beim Duschen) sensorisch: Mikrofon (< 20 db), GPS (48.387 | 9.872) elementar: Zeit (ab 6 Uhr), Konnektivität (WiFi)

Begriff	Sensoren
Beschreibung	Die modernen mobilen Endgeräte verfügen über unterschiedliche Sensoren, die die gegenwärtige Situation durch Messen der Kontextmerkmale erkennen. Zu den häufigsten Sensoren gehören der Beschleunigungssensor, Rotations-, Helligkeits- und Näherungssensoren sowie GPS-Empfänger.

3.2 Anwendungsrahmen

Mit dem Fokus den therapeutischen Prozess von Therapeuten und Patienten im Anwendungsfall Psychotherapie zu unterstützen, kombiniert das MobileTx-Projekt vier Komponenten (siehe Abb. 3.1). Dabei sind der Psychotherapeut, der Patient und der Wissenschaftler die Hauptakteure des Systems, die jeweils mithilfe einer mobilen oder Web-Anwendung untereinander oder mit der Server-Anwendung kommunizieren können.

Die Aufgabe des Therapeuten ist es Hausaufgaben für den Patienten zu erstellen und zu verwalten, die an den Server übermittelt werden. Der Server bildet die zentrale Komponente des Systems, die alle Daten und Erkenntnisse von den Akteuren speichert und analysiert. Ein Datenaustausch zwischen den Akteuren verläuft immer über den Server, der für die Sicherheit des Systems verantwortlich ist. Die Patienten-Anwendung fordert

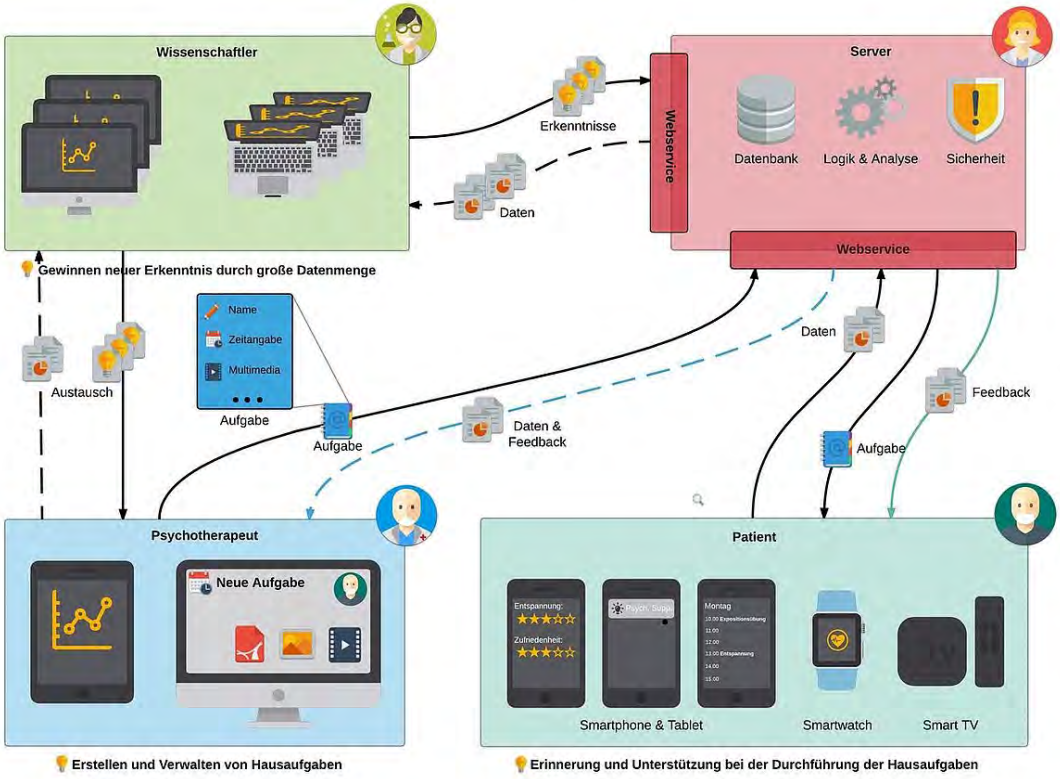

Abbildung 3.1: Architektur des Gesamtsystems MobileTx [9]

vom Server alle relevanten Daten an, die für die Erinnerung und Unterstützung bei der Durchführung der Hausaufgaben benötigt werden. Aufgrund der verschiedenen Endgeräte kann die Anwendung für den Patienten sowohl auf verschiedenen Smartphones, als auch auf Smartwatches oder sogar auf Smart TVs verfügbar sein. Daten, wie z.B. ein Feedback der Hausaufgabe oder Messdaten, werden an den Server geschickt, damit sowohl der Psychotherapeut als auch der Wissenschaftler diese auswerten können. Durch große Mengen an Messdaten (z.B. der Herzfrequenz) können Wissenschaftler neue Erkenntnisse gewinnen. Dafür stehen sie im direkten Kontakt mit dem Therapeuten.

Patienten-Anwendung Der Fokus dieser Arbeit liegt auf der mobilen Anwendung für die Patienten, die die Anwendung zur Erinnerung und Durchführung von Hausaufgaben benutzen. Um ein breites Spektrum an Benutzern ansprechen zu können, wird ein hybrider Ansatz gewählt. Durch eine entwickelte Anwendung werden verschiedene

mobile Betriebssysteme (wie z.B. Android und iOS) abgedeckt. Jedoch kann weder eine Web-Anwendung noch eine Progressive Web App den Benutzer optimal unterstützen, wenn die Hausaufgabe in einem bestimmten Kontext durchgeführt werden muss. Denn die Anwendung hat nicht nur eine darstellende Funktion, sondern beinhaltet Anwendungslogik zur Kontexterkennung. Nur nativen oder hybriden Anwendungen ist der Zugriff auf native Soft- oder Hardwarefunktionen möglich, die für das Beobachten der Umgebung notwendig sind.

Server Im Rahmen dieser Arbeit ist der Webservice als zentrale Schnittstelle, der mit den Anwendungen kommuniziert sehr wichtig. Der Server muss der Anwendung alle Daten und Informationen zur Präsentation zur Verfügung stellen, weil die Anwendung selbst nur Informationen zur Strukturieren besitzt, nicht aber die Inhalte selbst. Da für die Entwicklung der Patienten-Anwendung die Validierung von Datensätze und somit die Geschäftslogik nicht relevant ist, werden in diesem Rahmen nicht relevante Server-Funktionen vernachlässigt betrachtet. Dennoch sollen Anfragen von der mobilen Anwendung und eine persistente Datenhaltung möglich sein. Außerdem muss der Server eine sichere Authentifizierung bereit stellen, um dem Benutzer einen Zugriff auf das System und seinen Daten zu gewähren.

3.3 Funktionale Anforderungen

Die funktionalen Anforderungen sind gegliedert in die Anforderungen, die im direkten Zusammenhang mit dem Benutzer und somit der Benutzeroberfläche stehen (Anforderungen I -VI) und in die Anforderungen, die sich mit der Anwendungslogik befassen.

I **Anmeldung und Registrierung:**
Der Benutzer sollen sich beim System registrieren und anmelden können, um einen vollen Funktionsumfang der mobilen Anwendung nutzen zu können. Dafür muss das System verschiedene Benutzer verwalten und authentifizieren können.

II **Neue Therapie hinzufügen:**
Der Benutzer soll neue Behandlungen mit einem Therapeuten hinzufügen können. Das Hinzufügen soll durch eine manuelle Eingabe des Behandlungsnamen oder mittels QR-Codes erfolgen.

III **Behandlungsübersicht:**
Der Benutzer soll alle seine Behandlungen einsehen und abgeschlossene Behandlungen löschen können. Die Behandlungen sollen zusätzliche Informationen anzeigen, wie z.B. den behandelnden Therapeuten und den Status einer Behandlung.

IV **Aufgabenübersicht:**
Jede Behandlung hat mindestens eine Aufgabe mit Namen, Beschreibung des Kontextes und der Häufigkeit der Durchführung der Aufgabe. Der Benutzer soll seine gemessenen Daten während der Aufgabe graphisch in einer Statistik einsehen können.

V **Darstellung der Übungen:**
Jede Aufgabe hat eine Detailansicht, in der die Übung beschrieben wird. Die Beschreibung kann mit einem Medien-Element (Bild, Audio, Video) und einem erklärenden Text gestaltet sein. Der Benutzer soll die Möglichkeit haben, Übungen von Aufgaben für die Offline-Nutzung vom Server zu laden.

VI **Feedbackformular:**
Nach Abschließen einer Aufgabe soll das System dem Benutzer ein Formular anzeigen, um der Aufgabe und somit dem Betreuer ein Feedback geben zu können. Das Formular verschiedene Eingabemöglichkeiten unterstützen soll (siehe 5.1.8 im Kapitel Umsetzung). Das Schema wird vom Betreuer im Rahmen der Erstellung der Hausaufgabe definiert.

VII **Kontexterkennung:**
Das System soll Daten (u.a. sensorisch) jederzeit im Hintergrund erfassen und sammeln können. Das Sammeln von Daten soll für jeden einzelnen Sensor aktiviert und deaktiviert werden können. Das System soll aus den gesammelten Daten einen vorgegeben Kontext erkennen können. Die Anwendung soll eine mögliche fehlerhafte Kontexterkennung behandeln können, indem der Benutzer im Fall eines falsch erkannten Kontext dem System eine Rückmeldung geben kann.

VIII **Unterstützung nativer Funktionen:**
Das System soll auf alle nativen Sensoren des modernen mobilen Endgeräts zugreifen können. Verschiedene Hardware-Sensoren erfassen

- verschiedene Umgebungsparameter (Barometer, Thermometer, Mikrofon, oder Photometer)
- Bewegungen (Accelerometer, Gyroskope)
- Position (GPS, Kompass)
- Aktivität des Benutzers (Pedometer)

Ebenso soll das System die native Kalenderanwendung als weitere Datenquelle nutzen können.

IX **Benachrichtigungen:**
Das System soll den Benutzer mittels einer Nachricht informieren, falls ein Kontext erkannt wurde. Die Benachrichtigung soll zu jedem Zeitpunkt dem Benutzer angezeigt werden. Der Benutzer soll in regelmäßigen Abständen an die Durchführung einer Aufgabe erinnert werden, bis die Aufgabe absolviert wurde.

X **Generische Darstellung:**
Der Inhalt der Anwendung soll generisch gehalten sein und dynamisch generiert

werden. Sowohl die verschiedenen Darstellungselemente der Übungen und des Feedbacks, als auch die benutzerbezogene Daten werden textuell in JSON beschrieben. Das System muss diese interpretieren können, um die entsprechenden Inhalte oder Bedienkomponenten anzeigen zu können. Für die generische Darstellung werden die JSON-Dateien auf einem Server gespeichert. Die Anwendung muss mit dem Server kommunizieren können, um alle Informationen zu erhalten oder auf dem Server zu aktualisieren.

XI **Download und Offline-Funktion:**
Das System soll dem Benutzer eine Offline-Nutzung der Anwendung ermöglichen. Wenn die Anwendung mit dem WLAN verbunden ist, soll das System neue Aufgaben automatisch vom Server laden können. Diese Funktion soll in den Einstellungen vom Benutzer aktiviert und deaktiviert werden können.

3.4 Nichtfunktionale Anforderungen

Nichtfunktionale Anforderungen beschreiben, wie die geforderten Funktionalitäten an das System auszuführen sind. Zu den nichtfunktionale Anforderungen gehören beispielsweise Qualitätsanforderungen, Sicherheitsanforderungen oder Performanceanforderungen.

I **Funktionalität:**
Die Anwendung soll die definierten funktionalen und nichtfunktionalen Anforderungen vollständig umsetzen. Dadurch soll die Anwendung den Benutzer so unterstützen, dass er die Hausaufgaben möglichst effektiv und effizient erledigen kann. Dazu soll das System die Umgebung in regelmäßigen Abständen abtasten und daraus Kontexte korrekt ableiten und Fehlinterpretationen vermeiden.

II **Benutzerfreundlichkeit:**
Die Anwendung soll leicht verständlich und zufriedenstellend bedienbar sein, so

dass keine Anleitungen oder Hilfestellungen möglich sind. Die Erwartungen an das Verhalten der Anwendung soll den betriebssystemabhängige Konventionen entsprechen, die Benutzeroberfläche soll aber plattformübergreifend einheitlich sein.

III **Zuverlässigkeit:**
Die Anwendung soll fehlertolerant sein und auf falsche Benutzereingaben reagieren können. Der Benutzer muss die Möglichkeit haben, fehlerhafte Eingaben zu korrigieren. Die Anwendung muss zuverlässig mit einem Server kommunizieren können, um Daten verlustfrei auszutauschen.

IV **Effizienz:**
Die Bedienung der Anwendung soll flüssig sein, indem jegliche Aktionen performant bearbeitet werden, insbesondere die Kontexterkennung. Ebenfalls sollen die Ladezeiten der Daten vom Server möglichst kurz sein.

V **Portierbarkeit und Wartbarkeit:**
Die Anwendung soll für die Betriebssysteme iOS und Android zur Verfügung stehen. Weitere Plattformen sollen möglichst einfach ergänzt werden können. Außerdem soll die Anwendung durch weitere Zugriffe auf Sensoren erweiterbar sein.

VI **Sicherheit und Datenschutz:**
Der Schutz von vertraulichen, personenbezogenen Daten muss durch eine Authentifizierung jeder Zeit gewährleistet sein.

4

Konzeption

Das Konzept wurde aus den Anforderungen abgeleitet, die im vorangegangenen Kapitel definiert wurden. Das Konzept besteht aus dem Datenmodell, das die Daten der Anwendung strukturiert und aus der Dialogstruktur, die einen Überblick über den Aufbau der mobilen Anwendung gibt.

4.1 Datenmodell

Zur Strukturierung und Organisation der für die Anwendung benötigten Daten wird konzeptionell ein *Entity-Relation-Diagramm* mithilfe der Barker's Notation erstellt. Diese Schreibweise erlaubt eine Auflistung der Attribute direkt unter dem Namen einer Entität. Jedes Attribut enthält eine Eigenschaft, die Aufschluss gibt, ob das Attribut eindeutig (#), zwingend erforderlich (*) oder optional (o) ist [30].

Das Diagramm in Abb. 4.1 zeigt die sieben Entitäten mit ihren Attributen, die miteinander verknüpft sind. Der Benutzer kann keine, eine oder mehrere Behandlungen (Therapien)

haben. Die Therapie ist genau einem Benutzer zugeordnet. Jede Therapie hat aber mindestens eine *Aufgabe*. Diese Aufgabe hat neben einem eindeutig referenzierbaren Namen auch mindestens eine *Übung*, ein *Feedbackformular* und einen *Kontext*. Diese drei Entitäten sind nicht nur genau einer Aufgabe zugehörig, sie können mehreren Aufgaben zugeordnet sein. Der Kontext selbst hat nur einen Namen und einen optionalen Operator. Die Entität *Kontextmerkmal* wird dem Kontext eindeutig zu geordnet. Ein Kontextmerkmal enthält alle Eigenschaften, die für eine Kontexterkennung benötigt werden.

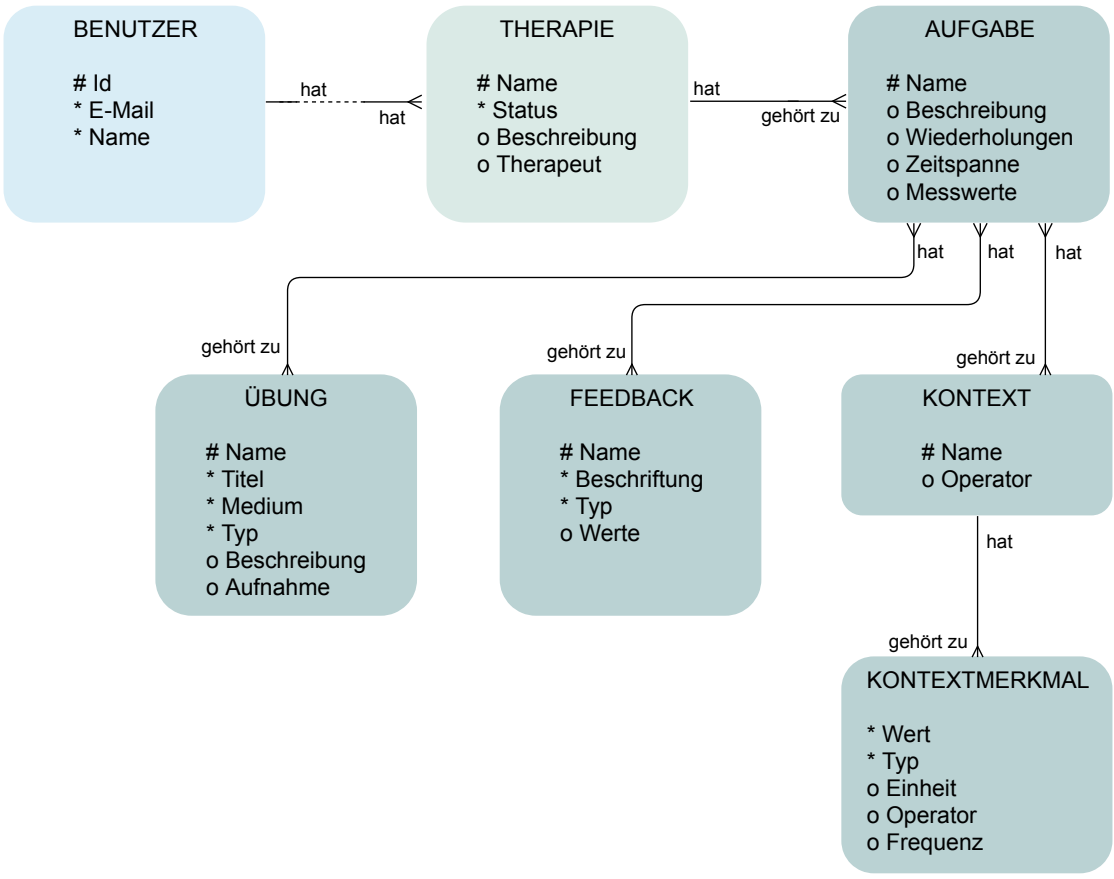

Abbildung 4.1: Datenmodell

Entität Benutzer Jeder Benutzer, der sich am System registriert, bekommt eine eindeutige Identifikationsnummer zugewiesen. Der Benutzer benötigt zur Authentifizierung

am System E-Mail-Adresse und Passwort, die bei der Registrierung angegeben werden müssen.

Entität Therapie Eine Therapie wird eindeutig über ihren Namen referenziert. Der Status ist ein erforderliches Attribut, um Therapien in zwei Kategorien ”offen” oder ”abgeschlossen” unterteilen zu können. Optionale Attribute, wie die Beschreibung und der Name des Therapeuten werden für den Benutzer zusätzlich angezeigt, sofern sie definiert sind.

Entität Aufgaben Therapien werden durch Aufgaben bestimmt. Diese Aufgaben werden durch einen eindeutigen Namen gekennzeichnet. Eine Aufgabe hat Attribute, die dem Benutzer als zusätzliche Informationen dienen, aber für das System nicht relevant sind, wie z.B. die Beschreibung oder die Messwerte. Daten, die der Benutzer während einer Übung aufnimmt, werden in den Messwerten gespeichert, um sie in der Anwendung dem Benutzer graphisch darstellen zu können. Die Attribute Wiederholungen und Zeitspanne ermöglichen eine begrenzte Kontextüberwachung. Wiederholungen ein Rhythmus (z.B. 7x täglich) oder eine Anzahl von Wiederholungen bis zu einem bestimmten Datum sein.

Entität Übung Die Datenstruktur Übung dient der generischen Anzeige von Anweisungen zur Durchführung einer Übung. Dabei ist jede Übung eindeutig über den Namen referenzierbar. Im Titel wird die Überschrift der Übung festgehalten. Die Attribute Medium und Typ legen den Inhalt des Mediums fest. Entweder durch das Medium selbst oder ein Verweis auf die Quelle und die Art des Mediums (Bild, Video, Audio). Das Medium kann durch eine Beschreibung der Übung in Textform ergänzt werden, die dem Benutzer bei der Durchführung zusätzlich helfen soll. Zur Strukturierung des Textes sind HTML-Tags erlaubt, die z.B. eine Auflistung ermöglichen. Das Attribut Aufnahme definiert Sensoren, die während der Übung aktiviert werden können, um Messwerte aufzunehmen (z.B. die Dauer einer Übung).

Entität Feedback Für die Kommunikation zwischen Betreuer und Benutzer werden Feedbackformulare genutzt, die nach Absolvieren einer Aufgabe eine Rückmeldung z.B. über den Schwierigkeitsgrad einer Übung geben. Die Entität Feedback beschreibt dabei nur ein Formularelement, das eine Überschrift, einen Typ und davon abhängige Werte enthält. Aus einem Feedback oder mehreren Feldbackelementen entsteht ein Formular.

Entität Kontext Eine Aufgabe muss mindestens einen Kontext enthalten, der die Situation beschreibt, in der die einzelnen Übungen durchgeführt werden sollen. Wie auch beim Feedback wird jeder Kontext durch den Namen eindeutig referenziert. Das ermöglicht eine Mehrfachzuordnung zu verschiedenen Aufgaben. Der Operator hat die Funktion einen Wahrheitswert von den Kontextmerkmalen zu liefern, dafür werden boolesche Operatoren benötigt. Der booleschen Operator *UND* ist voreingestellt.

Entität Kontextmerkmal Ein Kontext wird durch ein Kontextmerkmal oder die Verknüpfung von mehrere Kontextmerkmale beschrieben. Dafür benötigt jedes Merkmal einen Wert, einen Typ und optional eine Frequenz, mit der die Abtastrate des Sensors eingestellt wird. Der Typ beschreibt die Art des Sensors, der benötigt wird, um den definierten Schwellenwert oder einen Wertebereich vergleichen zu können, wie z.B. die Geolocation oder auch Kalender. Die Struktur der Werte sind vom Typ und vom Vergleichsoperator abhängig, der voreingestellt auf Gleichheit testet. Grundsätzlich sind alle Operatoren möglich. Objekte und Zeichenketten können jedoch nur auf Gleichheit überprüft werden, wie z.B. Kontextmerkmale vom Typ ”Accelerometer”, die ein JSON-Objekt der Form $\{x : 0, y : 0, z : 0\}$ benötigen. In Ausnahmefällen, wie z.B. beim Barometer bietet sich aufgrund von Rundungsfehlern an, Operatoren, wie ”between”, ”min” oder ”max” zu verwenden.

Das Attribut Einheit hat Einfluss auf die Darstellung der Werte. Beim Typ Geolocation beispielsweise kann der Wert ein Objekt mit Latitude und Longitude, eine Zeichenkette für die Adresse oder ein Zahlenwert sein. Die Adresse besteht entweder aus einem Städtename, der kompletten Adresse oder dem symbolischen Wert ”zuHause”. Die Tabelle 4.1 zeigt im Überblick die unterstützten Typen, die erwarteten Werte und die Einheiten, die für eine eindeutige Validierung notwendig sind.

Typ	**Wert**	**Einheit**
Accelerometer, Gyroskop	$\{x : Zahlenwert,$ $y : Zahlenwert,$ $z : Zahlenwert\}$	-
Kompass	Zahlenwert	magneticHeading
Pedometer (Schrittzähler)	Zahlenwert, Datum	numberOfSteps, distance, floorsAscended, floorsDescended, startDate, endDate
Mikrofon	Zahlenwert	-
Geolocation	$\{latitude : Zahlenwert,$ $longitude : Zahlenwert\}$, Zeichenkette, Zahlenwert	coordinates, address, altitude, speed, accuracy
Kalender	Zeichenkette, Datum	title, location, notes, startDate, endDate
Barometer	Zahlenwert	pressure, relativeAltitude
Zeit	Zahlenwert, Zeichenkette	H, yyyy, d MMM, EEEE[1]

Tabelle 4.1: Übersicht der Darstellung von Kontextmerkmalen

[1] Eine vollständige Auflistung ist auf der Angular API definiert: `https://angular.io/api/common/DatePipe`

4.2 Dialogstruktur

Die Dialogstruktur zeigt den Aufbau der Anwendung und die mögliche Navigation durch das System. Dabei repräsentiert ein Dialog eine *View* (siehe Abb. 4.2), die durch die Komponentenklasse und dem Template definiert wird (siehe Grundlagenkapitel 2.3.1).

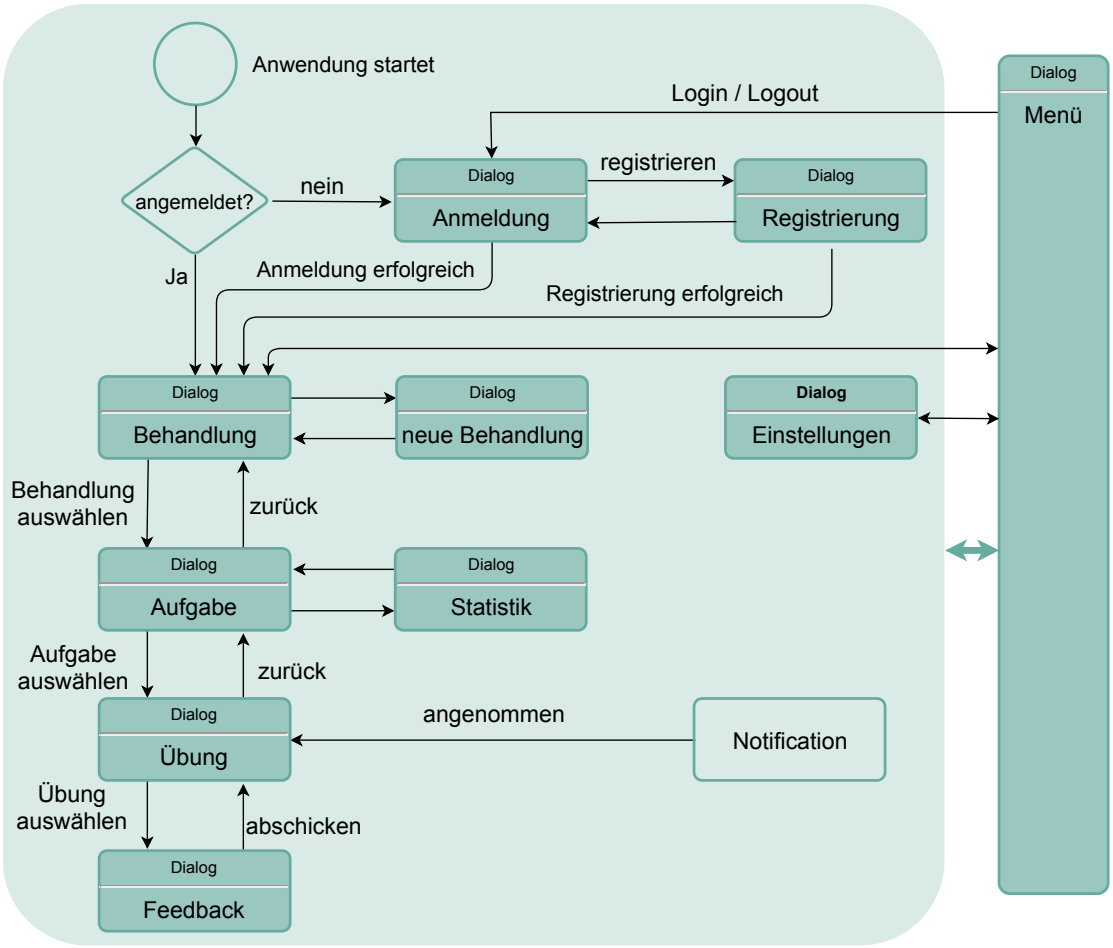

Abbildung 4.2: Dialogstruktur der mobilen Anwendung

Zunächst muss die Anwendung auf einem mobilen Endgerät mit passendem Betriebssystem installiert werden. Hat der Benutzer das System bereits genutzt, kann er sich mit seiner E-Mail Adresse und Passwort anmelden und bleibt solange am System angemeldet, bis der Benutzer sich selbständig vom System abmeldet. Ist dem System der

Benutzer unbekannt ist eine Registrierung notwendig. Bei der Registrierung legt der Benutzer seine Anmeldedaten fest.

Nach einer erfolgreichen Anmeldung oder Registrierung gelangt der Benutzer zum Dialog, der die Behandlungen des Benutzers in der Übersicht zeigt. Der Benutzer sieht auf dieser Seite alle wichtigen Informationen über seine offenen und abgeschlossenen Behandlungen.

Im Dialog Behandlung kann der Benutzer über einen Button das Menü öffnen. In allen anderen Dialogen öffnet sich das Menü durch eine Wisch-Geste vom linken Bildschirmrand in die Mitte des Bildschirmes. Im Menü hat der Benutzer die Möglichkeit zu den Behandlungen zu gelangen, den Einstellungs-Dialog zu öffnen oder sich vom System abzumelden. Das Menü schließt sich bei einer Auswahl eines Benutzers, durch ein Wischen zum Bildschirmrand oder durch das Klicken des Menü-Buttons im Dialog Behandlungen, wo der Benutzer zurück zu seinen Behandlungen kommt.

In den Einstellungen kann der Benutzer sein Profil bearbeiten und Umgebungsbedingungen, wie z.B. seinen Wohnort angeben. Diese Informationen können zur Kontexterkennung hinzugezogen werden.

Wählt der Benutzer eine Behandlung aus, öffnet sich der Dialog Aufgaben, der alle Aufgaben der ausgewählten Behandlung zeigt. Der Benutzer kann von hier aus zu der Übung oder zu dem Statistik-Dialog navigieren. Die Statistik zeigt grafisch die gemessene Werte mit Datum aus den Übungen an, wie z.B. die Dauer einer Übung. Der Dialog Übung führt den Benutzer durch verschiedene Übungen, die er mithilfe einer Anleitung durchzuführen hat. Eine Aufgabe gilt erst als erledigt, wenn der Benutzer zum Schluss das angezeigte Feedbackformular ausgefüllt hat. Nach dem Abschicken des Feedbackformulars gelangt der Benutzer zurück zur Aufgabenübersicht.

Der Benutzer kann zu jederzeit Benachrichtigungen erhalten, falls ein Kontext erkannt wurde und dem Benutzer empfohlen wird die zum Kontext passende Übung durchzuführen. Nach dem Annehmen der Benachrichtigung gelangt der Benutzer direkt zur ersten Übung der Aufgabe.

5

Realisierung

Dieses Kapitel beschreibt die Erstellung einer prototypischen Anwendung, die auf den Anforderungen und dem Konzept in den vorangegangen Kapiteln basiert. Dabei wird zuerst die Gestaltung der Dialoge veranschaulicht, die aus der Dialogstruktur bekannt sind. Im Weiteren wird die Architektur der Gesamtanwendung, der schematische Aufbau eines Kontextes, die Kontexterkennung und Kontextvalidierung vorgestellt. Abschließend werden die verwendeten Technologien und Frameworks, wie *Ionic Native* oder *Firebase* erläutert.

5.1 Dialoggestaltung

Nachdem die Struktur der Dialoge im Konzept definiert wurden, wird in diesem Kapitel näher auf die Gestaltung eingegangen. In den folgenden Abbildungen werden jeweils Bildschirmfotos der Betriebssysteme iOS (links) und Android (rechts) gegenüber gestellt. Durch unterschiedliche *User-Interface-Styleguides* werden die Grundstruktur

der Anwendung, Farben, Typografie und Benutzerelemente an die unterschiedlichen Betriebssystemen angepasst.
Alle Betriebssysteme verwenden die gleichen Farben durch die Festlegung eines eigenen Farbschemas. Empfohlen werden miteinander harmonierende Farben auszuwählen und auf ausreichend Kontrast von Schriftfarbe und Hintergrund für eine gute Lesbarkeit zu achten. Neben den Farben ist das Aussehen der Anwendung durch die Verwendung von sechseckigen Formen charakterisierend. Die Sechsecke kennzeichnen ein Element besonders, wie z.B das Auslösen einer Funktion. Um die Entwicklung zu erleichtern, werden viele Standardkomponenten der Betriebssysteme iOS und Android ohne eine weitere Individualisierung übernommen. Deshalb ist die Grundstruktur der Oberflächen der beiden Betriebssysteme sehr ähnlich, während z.B. das Verhalten der Anwendungen unterschiedlich sein kann.

5.1.1 Login

Die Funktion der Loginseite (sieh Abb. 5.1) beschränkt sich auf die Anmeldung am System. Die Authentifizierung erfolgt über die E-Mail-Adresse des Benutzers und sein Passwort, die in Textfelder eingetragen werden.
Das Erscheinungsbild des Formulars ist betriebssystemspezifisch, aber sehr ähnlich. Abgeschickt wird das Formular durch den Anmelde-Button, der in Form eines Hexagons dargestellt wird. Durch das Auslösen der Aktion, werden die Benutzerangaben validiert. Bei einer falschen Eingabe erscheint auf dem Bildschirm eine Fehlermeldung, in Form eines *Alerts*. Im erfolgreichen Fall der Anmeldung gelangt der Benutzer zum Dialog Behandlung.

In der Fußzeile der Seite ist eine Schaltfläche, die den Benutzer zum Dialog Registrieren führt. Der Übergang zwischen den Dialogen ist bei den Betriebssystemen standardmäßig unterschiedlich, so schiebt sich die neue Seite bei iOS von links, bei Android von unten über die alte Seite, in dem Fall der Login-Dialog.

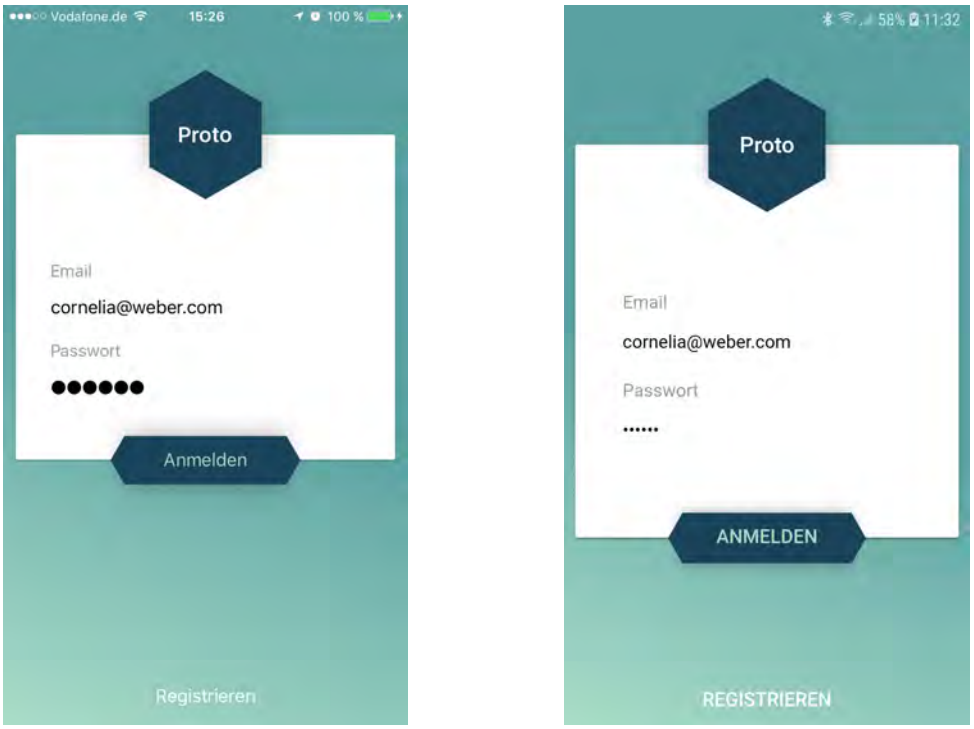

Abbildung 5.1: Dialog Login

5.1.2 Registrieren

Der Aufbau des Dialogs Registrieren (siehe Abb. 5.2) ist analog zum Dialog Login. Das Formularfeld wird um ein weiteres Textfeld ergänzt, in das der Benutzer seinen Namen eintragen kann. Für eine Identifizierung des Benutzers ist die Angabe der E-Mail-Adresse und ein Passwort notwendig. Das Passwort muss mindestens sechs Zeichen lang sein. Aus dem Anmelde-Button des- Dialogs Login wird ein Registrier-Button, der bei einer erfolgreichen Registrierung den Benutzer am System anmeldet und zum Dialog Behandlungen führt. Der Benutzer wird über die erfolgreiche Anmeldung zusätzlich mittels eines *Toasts* dezent hingewiesen.

Tritt bei der Registrierung ein Fehler auf, z.B. wenn die angegeben E-Mail-Adresse bereits im System hinterlegt ist, wird dem Benutzer erneut ein Alert angezeigt.

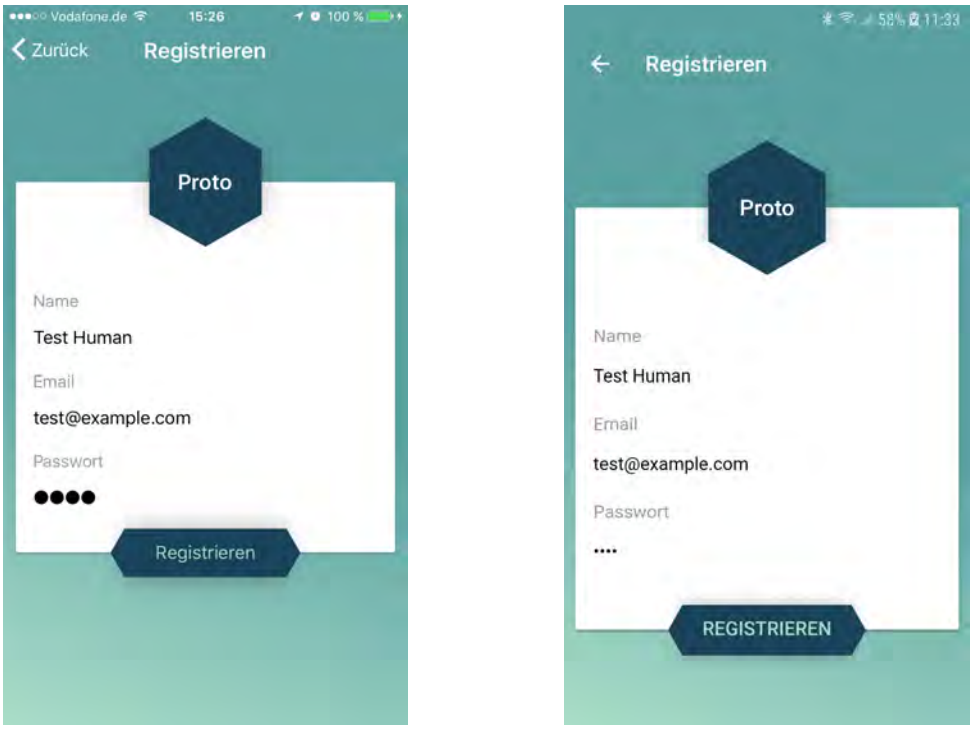

Abbildung 5.2: Dialog Registrieren

5.1.3 Therapie

Der Dialog Behandlung zeigt sowohl die offenen (siehe Abb.5.3 links), als auch die schon abgeschlossenen Behandlungen des angemeldeten Benutzers. Um zwischen den Ansichten wechseln zu können, wird ein *Segmented Control* angepasst, indem wieder die Form des Hexagons verwendet wird. Für solch eine Gruppe von Schaltflächen verwendet Android entsprechende Tabs (Registerkarten). Durch die Individualisierung wird die Form des Hexagons in beiden Betriebssystemen gleich übernommen.

Wie auch in der restlichen Anwendungen werden für ganze Listen oder Listeneinträge sogenannten *Karten* (engl. Cards) verwendet. Jede Behandlung wird mithilfe einer Karte dargestellt, in der der Name der Behandlung, die Beschreibung und weitere Informationen, wie der behandelnden Therapeut vermerkt wird. Für eine schnelle Erkennung der Behandlung, kann diese zusätzlich mit einem Icon optisch hervorgehoben. Durch

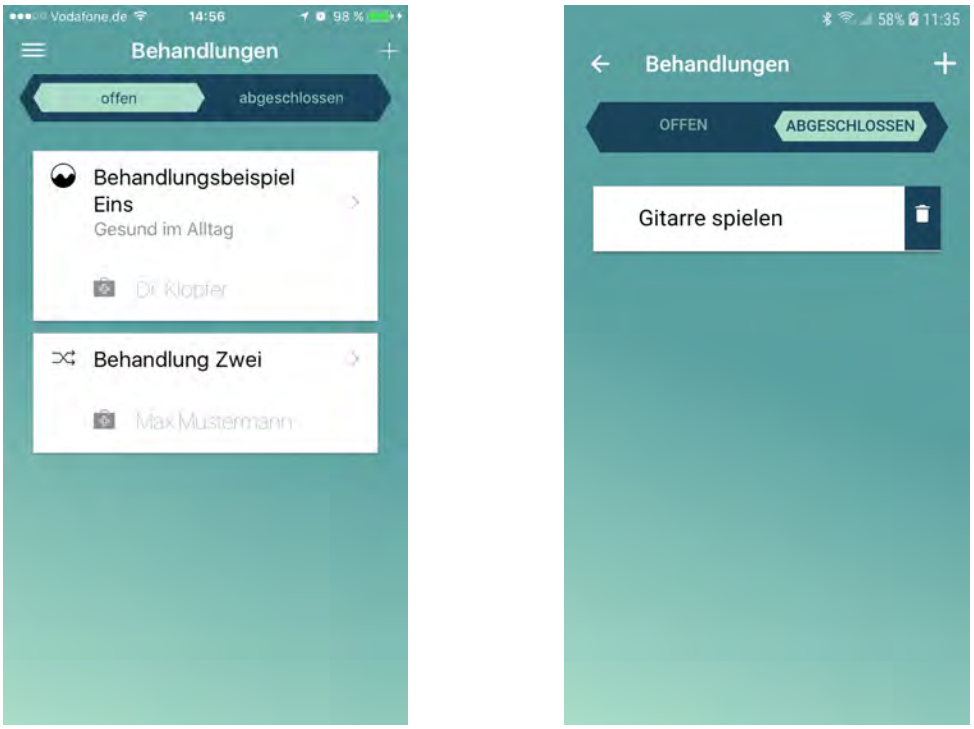

Abbildung 5.3: Dialog Behandlungen

Auswahl des Behandlungsnamens wird der Benutzer zur zugeordneten Aufgabe geleitet. Diese Möglichkeit der Aktion wird in iOS durch einen Pfeilindikator angezeigt, während es in Android keine Unterscheidung in der Darstellung von informativen Listeneinträgen (Name des Therapeuten) und Aktionen gibt.
Da sich die Karteneinträge innerhalb der Karten wie Listeneinträge verhalten, können Schaltflächen, wie z.B ein Lösch-Button, durch Wischen der Einträge sichtbar gemacht werden. Das Verhalten wird bei den abgeschlossenen Behandlungen (siehe Abb.5.3 rechts) benutzt, damit der Benutzer die Möglichkeit hat Behandlungen aus seiner Anwendung zu löschen.

In der Kopfzeile der Seite werden links neben der Überschrift ein Menü-Button und rechts neben der Überschrift ein Button für das Hinzufügen neuer Behandlungen mittels platzsparender Icons dargestellt.

5.1.4 Neue Therapie

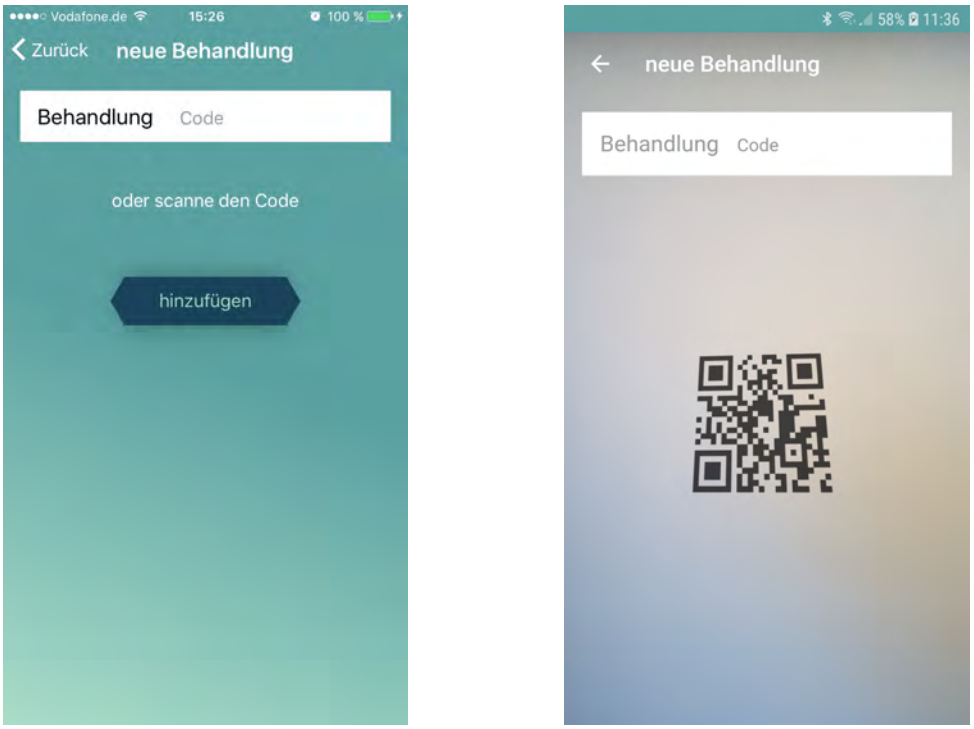

Abbildung 5.4: Dialog neue Therapie

In diesem Dialog können neue Therapien manuell hinzugefügt werden können (siehe Abb. 5.4). Dafür kann der Benutzer einen Code, der eine Behandlung eindeutig referenziert, in das Textfeld eintragen (linke Abb.) oder mittels der integrierten Kamera des modernen mobilen Endgeräts einen QR-Code scannen (rechte Abb.). Beim Scannen werden alle Bedienelemente entfernt, die für den Vorgang nicht relevant sind, sodass nur das obere Eingabefeld sichtbar ist. Sobald das Gerät den Code gescannt hat, wird der Code automatisch in das Textfeld eingetragen und der Benutzer kann die Behandlung durch Betätigen des sechseckigen Buttons hinzufügen. Der Benutzer wird automatisch zu dem Dialog Behandlungen geführt.
Der Benutzer kann jeder Zeit ohne eine neue Behandlung hinzuzufügen mittels dem ”Zurück”-Button zum vorherigen Dialog gelangen. Der Menü-Button fehlt aufgrund des

geringen Darstellungsraumes. Das Menü öffnet sich durch die Wischbewegung vom linken Bildschirmrand.

5.1.5 Aufgabe

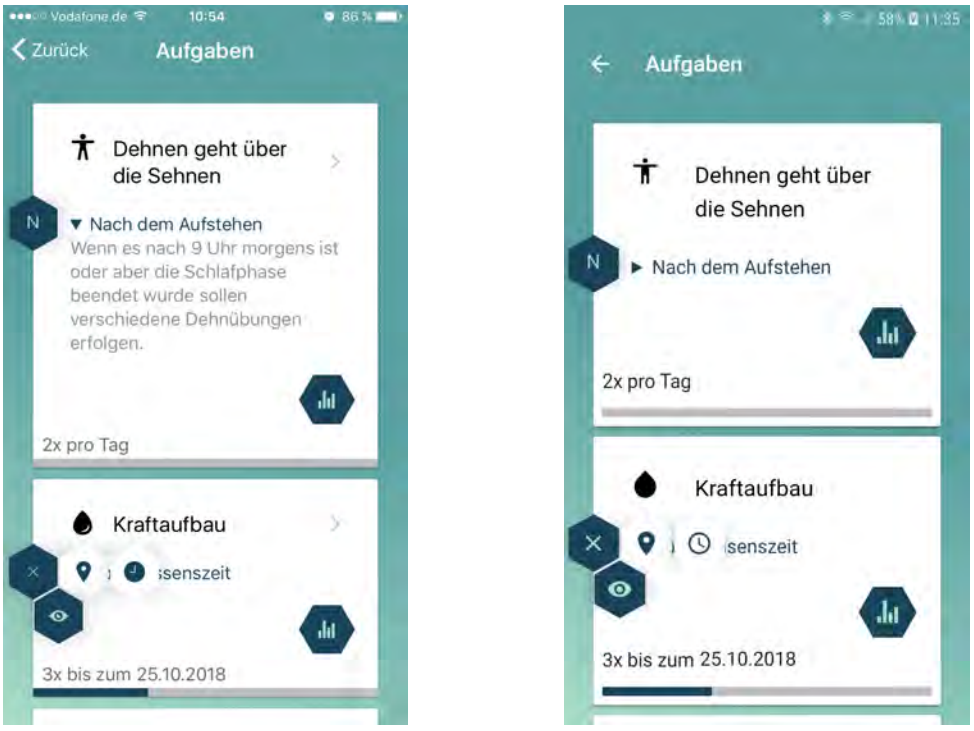

Abbildung 5.5: Dialog Aufgabe

Jede Behandlung hat mindestens eine therapeutische Hausaufgabe, die der Benutzer unter bestimmten Bedingungen auszuführen hat. Diese Aufgaben sind wieder in Form von Karten dargestellt. Jede Karte hat einen Namen und eine Beschreibung, die aufklappbar mehr Informationen über den Kontext und die Aufgabe gibt. Am linken Rand der Karte befindet sich eine Schaltfläche, die sich an einem *FAB* (Floating Action Button) orientiert. Dieses Bedienelement ist für Android charakteristisch und soll die Einhandbedienung von mobilen Anwendungen erleichtern, indem das Element flexibel auf dem Bildschirm angeordnet werden kann. In iOS wird ein *Call-to-Action-Button* verwendet, der sich in der Mitte einer Tabbar (Registrierkarte) befindet. Nach Betätigung des FABs öffnen sich

weitere Elemente. Die weißen Hexagons zeigen die Sensortypen in Form von Icons an, die für die Kontexterkennung der Übung benötigt werden. Mit der blauen Schaltfläche in der zweiten Reihe kann die Überwachung eines Kontext an- oder ausgeschaltet werden.

Der Betreuer kann neben dem Kontext einen Rhythmus oder eine Anzahl von Wiederholungen bis zu einem bestimmten Datum vorgeben. So muss der Benutzer z.B. eine Aufgabe zweimal pro Tag durchführen. Dieser Fortschritt wird dem Benutzer im unteren Teil der jeweiligen Karte in einem Balken dargestellt. Über dem Balken befindet sich ein Hexagon-Button, der den Benutzer zum Dialog Statistik führt.

5.1.6 Statistik

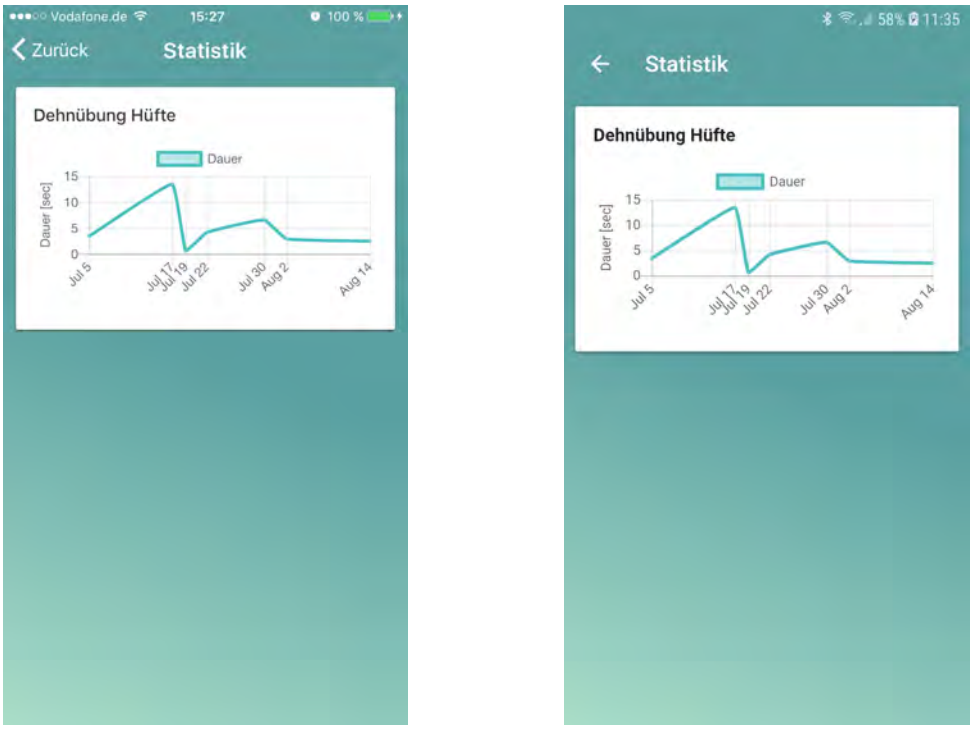

Abbildung 5.6: Dialog Statistik

Im Dialog Statistik werden alle Messwerte grafisch dargestellt, die der Benutzer während einer Übung aufgenommen hat. Im Vorfeld definiert der Therapeut die Art der Messwerte.

Die Werte werden in Liniendiagrammen abgebildet, um den Fortschritt der Therapie zu veranschaulichen. Dabei wird ein Liniendiagramm für jede Übung in einer Karte analog zu den Dialogen Aufgabe und Behandlung dargestellt, sofern dem System Messwerte vorliegen. Kennzahlen, wie z.B. in der Abbildung 5.6 die Dauer der ”Dehnübung Hüfte”, werden auf der y-Achse und die Dimension (das zugehörige Datum) wird auf der x-Achse dargestellt.
Gibt es viele Messwerte einer Übung empfiehlt es sich das Gerät aufgrund der geringen Bildschirmbreite von z.B.Smartphones ins Querformat zu wechseln.

5.1.7 Übung

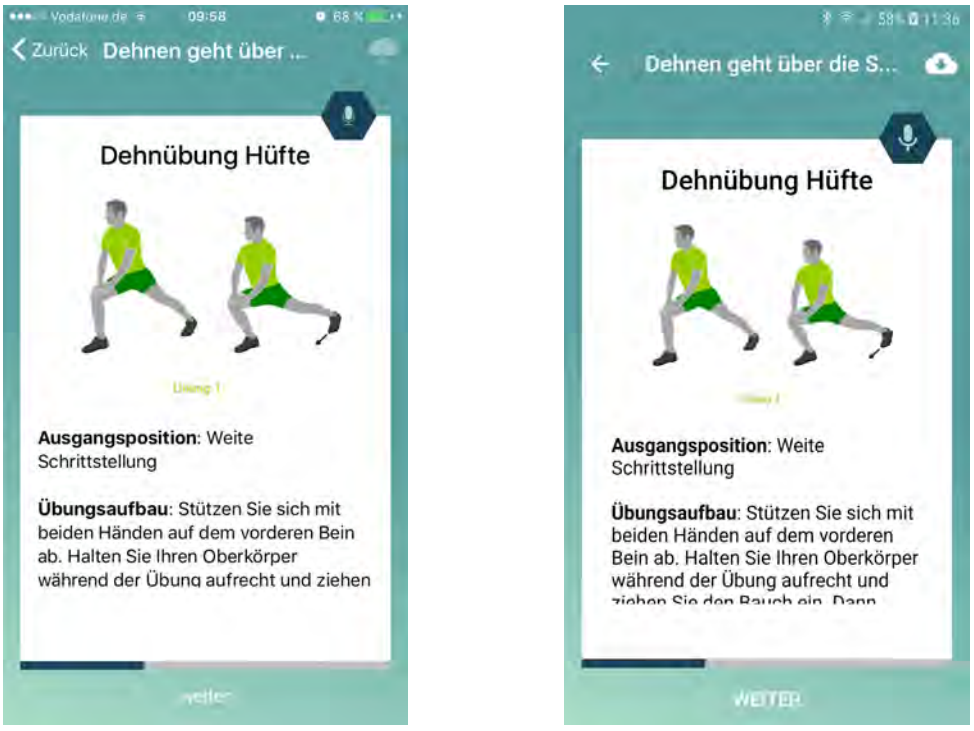

Abbildung 5.7: Dialog Übung

Der Dialog Übung zeigt in vereinfachter Form, wie Übungen einer Aufgabe dargestellt werden können. Obwohl eine Übung viel komplexer strukturiert sein kann, wird eine sequenzielle Durchführungen der Übungen angenommen, um die Entwicklung der

Anwendung zu vereinfachen. Die sequenzielle Durchführung wird mithilfe der *Slides-Komponente* umgesetzt, bei der zwischen den einzelnen Seiten durch Wischen navigiert werden kann. Der Balken am unteren Rand zeigt die Anzahl der bereits absolvierten und noch ausstehende Übungen an.
Jede Karte einer Übung hat einen Titel, ein mediales Element und einen beschreibenden Text. Die Anwendung unterstützt die Medien Bild, Video und Audio. Abhängig von der Beschreibung der Übung, kann der Benutzer Messwerte (z.B. Durchführungsdauer) aufnehmen.

Diese Aufnahme wird durch das Betätigen des Hexagon-Buttons am oberen Rand der Karte gestartet. Mithilfe einer Wischgeste oder über den Weiter-Button in der Fußzeile wird die nächste Übung sichtbar. Eine laufende Aufnahme wird dann automatisch gestoppt. Auf der letzten Karte der Sequenz ist ein Feedbackformular dargestellt. Der Benutzer kommt durch eine Wischgeste zur vorangegangenen Übung. Eine Aufgaben kann über den Zurück-Button abgebrochen werden, dadurch wird die Aufgabenübersicht einer Behandlung erneut angezeigt.

In der Kopfzeile befindet sich rechts ein Button, der das Laden aller Übungen vom Server für die Offline-Nutzung startet. Das Icon ist deaktiviert (siehe Abb. 5.7 links), falls die Übungen bereits heruntergeladen wurden.

5.1.8 Feedback

Der Dialog Feedback (siehe Abb. 5.8) ist der letzte Teil der Slides-Komponente aus dem Dialog Übungen. Jedes Feedbackformular besteht aus einzelnen Formularelementen, die jeweils aus einer Beschriftung und dem Formularfeld bestehen. Das Formularfeld kann ein einfaches Eingabefeld, eine Ratingskala oder eine Einfach- bzw. Mehrfachauswahl sein. Der Typ sowie mögliche Auswahlmöglichkeiten werden vom Betreuer definiert und auf dem Server gespeichert.
Für die Mehrfachauswahl verwendet Android Checkboxen, die auch auf Webseiten oder Desktop-Betriebssystemen verwendet werden (siehe Abb. 5.8 rechts). Das mobile Betriebssystem iOS unterstützt weder Checkboxen noch Radiobuttons für die Einfach-, bzw. Mehrfachauswahl. Anstatt der üblichen Darstellung verwendet iOS für die Einfach-

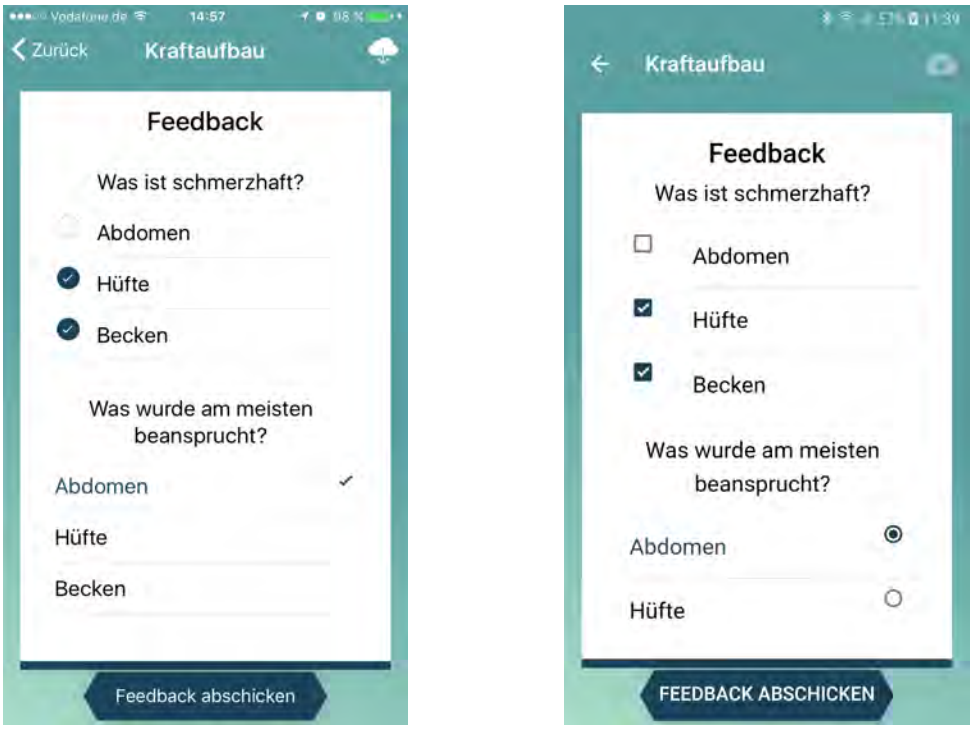

Abbildung 5.8: Dialog Feedback

auswahl Häkchen und für die Mehrfachauswahl einen Kreis mit einem Häkchen, wie in 5.8 (links) abgebildet ist.

Der Benutzer hat die Möglichkeit, seine in den Übungen zuvor aufgenommene Messdaten mit dem Formular an den Betreuer zu übermitteln. Das Bedienelement *Switch* ermöglicht das Aktivieren, bzw. Deaktivieren der Funktion ”Messdaten mitschicken”.

Am Ende aller Übungen ist der Fortschrittsbalken komplett ausgefüllt. Der Weiter-Button aus dem Dialog Übungen wird zu einem sechseckigen Button, der das Feedback an den Server schickt, somit ist die Übung erfolgreich beendet.

5.1.9 Menü

Das Menü bietet dem Benutzer die Möglichkeit sich vom System abzumelden oder die Behandlungen, bzw. die Einstellungen zu öffnen. Der Benutzer kann in jedem Dialog

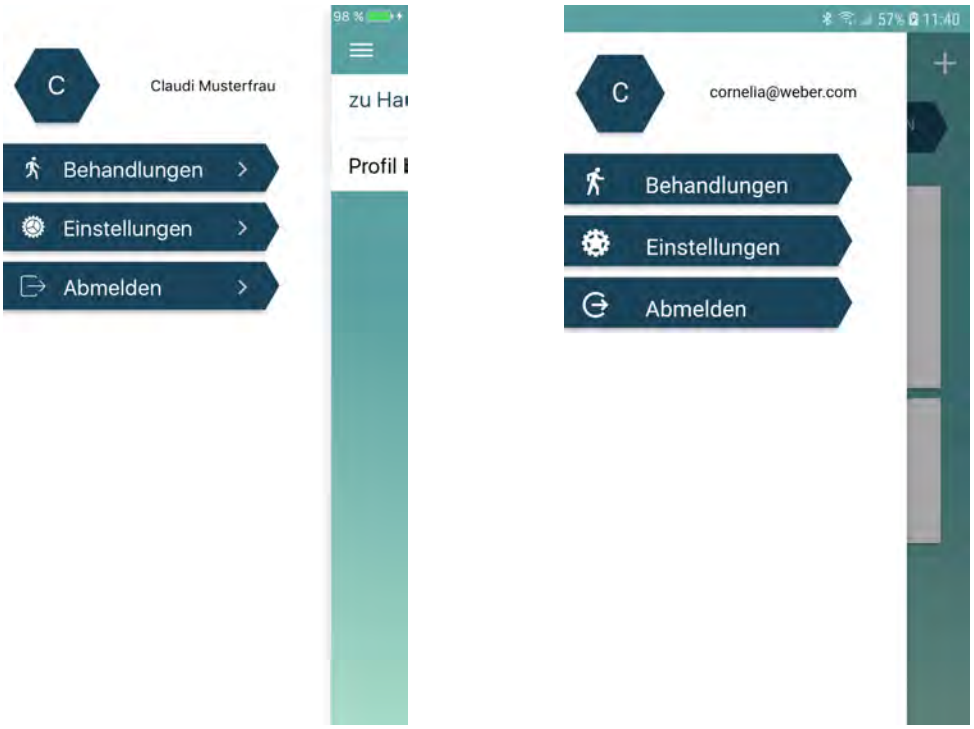

Abbildung 5.9: Dialog Menü

das Menü öffnen. Im Dialog Behandlungen gibt es sichtbar einen Button dafür, in allen anderen Dialogen wird das Menü durch eine Wischbewegung vom rechten Rand in die Mitte des Bildschirmes geöffnet. Öffnet ein Benutzer das Menü, wenn er noch nicht am System angemeldet ist, kann er ausschließlich den Dialog Login auswählen.

Es gibt zwei Möglichkeiten das Menü zuschließen:

1. durch entgegengesetztes Wischen
2. durch einen Klick auf die ursprüngliche Seite, die vom Menü nur teilweise bedeckt ist

5.1.10 Einstellungen

In den Einstellungen kann der Benutzer persönliche Informationen einsehen und bearbeiten. Er kann z.B. sein zu Hause manuell oder über die Positionserkennung des

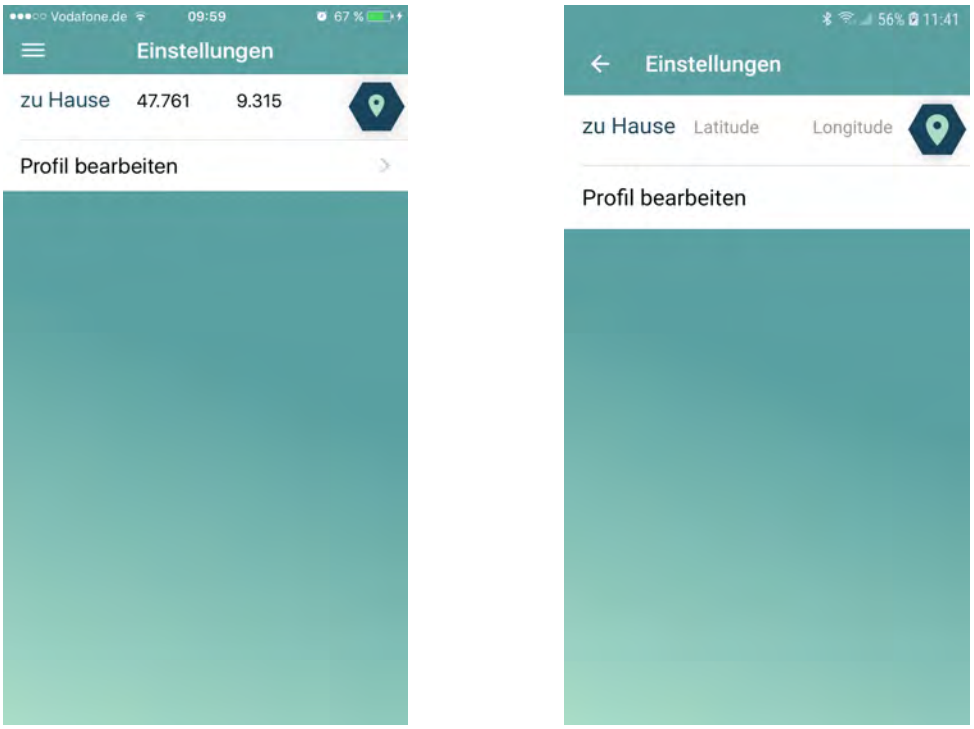

Abbildung 5.10: Dialog Einstellungen

modernen mobilen Endgeräts eintragen. Diese Information kann als Quelle für die Kontexterkennung dienen.

5.1.11 Benachrichtigungen

Das System verschickt Benachrichtigungen (*Notifications*), um den Benutzern an die Durchführung einer Übung zu erinnern, falls die Aufgabe noch nicht abgeschlossen ist. Diese Benachrichtigung werden zu jeder Zeit angezeigt, auch im gesperrten Modus des mobilen Endgerätes. Zwei Schaltflächen unter dem Nachrichtentext ermöglichen eine direkte Rückmeldung über die Kontexterkennung, ohne die Anwendung öffnen zu müssen. Während in Android die Schaltflächen sofort sichtbar sind, muss bei iOS die Benachrichtigung erst geöffnet werden (siehe Abb. 5.11).

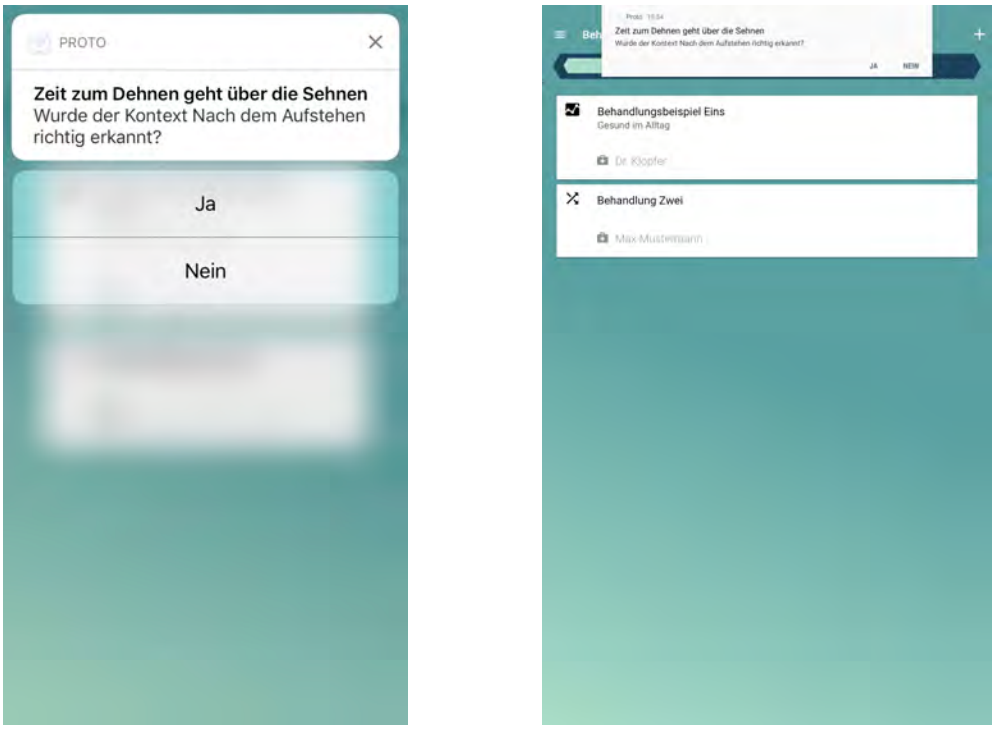

Abbildung 5.11: Notification mit Aktionbuttons

5.2 Architektur

Aus den Anforderungen an das System ergibt sich eine Client-Server Architektur, wie sie in Abb. 5.12 zu dargestellt ist. Der Fokus liegt auf der mobilen hybriden Anwendung, die als *FAT-Client* sowohl für die Darstellung der Daten als auch für die Logik der Kontexterkennung zuständig ist. Der Webservice ist einfach gehalten und wird für die Benutzerverwaltung und Authentifizierung, sowie für die Datenhaltung benötigt.

Echtzeit-Datenbank Zur Datenhaltung wird eine NoSQL-Cloud Datenbank benutzt, die die Anwendungsdaten in Echtzeit clientseitig synchronisiert. Die Daten selbst werden in JSON strukturiert, die für den Client über eine REST-API zugänglich sind. Aufgrund der Nutzung einer Echtzeitdatenbank entfallen die typischen HTTP-Anfragen, die vom Client an den Server geschickt werden. Veränderungen von Daten über eine

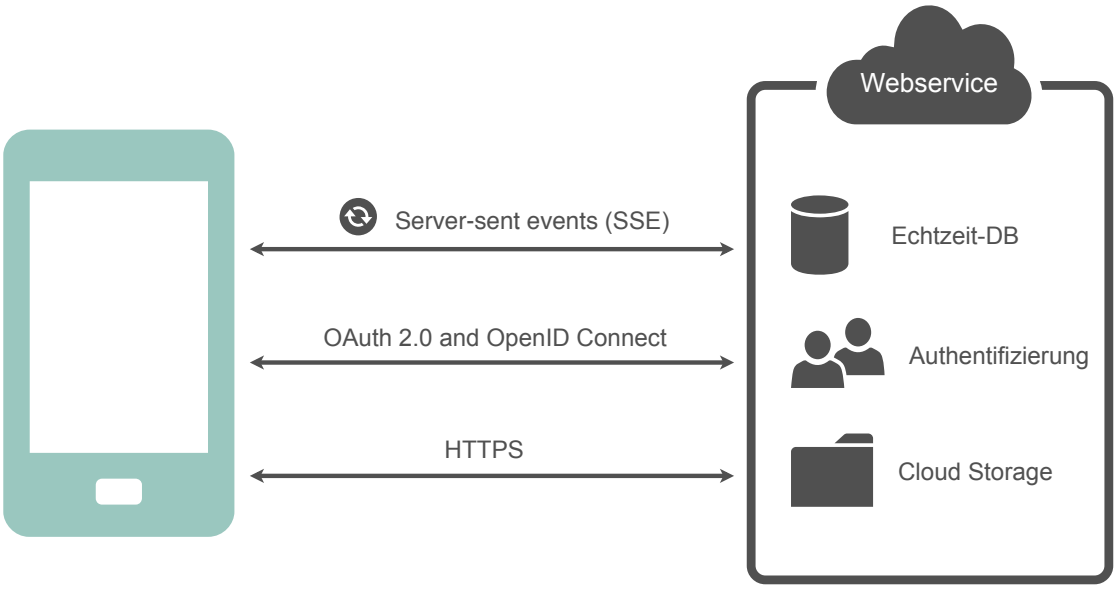

Abbildung 5.12: Abstrahierte Systemarchitektur

HTTP-Verbindung können durch den Einsatz des *Server-Sent Events* (SSE) Protokolls automatisch erfasst werden. Zudem werden die Daten lokal zwischengespeichert, damit einer Offline-Nutzung der Anwendung möglich ist. Sobald eine Verbindung zum Server hergestellt werden kann, werden die Datenänderungen mit der Datenbank synchronisiert.

```
https://proto-ma.firebaseio.com/users/EHhpCtoMyHxIH3/treatments
\___/   \_____________________/       \_____________/ \_________/
  |               |                         |             |
Schema          Host                      UID      Behandlungen
                                    \_____________________________/
                                                |
                                              Pfad
```

Listing 5.1: Beispiel URL zur Abfrage von Behandlungen

Über eine URL werden die Daten aus der Datenbank angefordert, wie beispielsweise alle Behandlungen eines Benutzers. Der Aufbau der URL setzt sich zusammen aus dem verwendeten Netzwerkprotokoll (HTTPS), dem Host und dem Pfad, über den die

Behandlungen des Benutzers mit der Nutzer-ID *EHhpCtoMyHxIH3* lokalisiert werden können (siehe Listing 5.1).

Die Datenbank liefert beispielsweise drei Behandlungen (siehe Listing 5.2). Im Listing werden jedoch die Aufgaben (activity Z. 2) der Übersichtlichkeit nicht dargestellt.

```
[ {
  "activity" : [...],
  "description" : "Gesund im Alltag",
  "name" : "Behandlungsbeispiel Eins",
  "status" : 0,
  "therapist" : "Dr. Klopfer"
}, {
  "icon" : "shuffle",
  "name" : "Behandlung Zwei",
  "status" : 0,
  "therapist" : "Max Mustermann"
}, {
  "name" : "Gitarre spielen",
  "status" : 1
} ]
```

Listing 5.2: Beispiel Behandlungen in JSON

Der Zugriff auf die Daten wird durch Sicherheitsregeln und Benutzerauthentifizierungen bestimmt. Aufgrund sensibler Daten sind sowohl die Lese- als auch die Schreibzugriffe nur authentifizierten Benutzers möglich. Umgesetzt sehen die Regeln wie in Listing 5.3 dargestellt aus.

```
{
  "rules": {
    ".read": "auth != null",
    ".write": "auth != null"
  }
}
```

Listing 5.3: Sicherheitsregeln der Datenbank

Authentifizierung Aufgrund der personalisierter Benutzung des Systems durch verschiedene Benutzer (z.B. Patienten) wird ein globales Benutzerverwaltungssystem benötigt. Dieses System ermöglicht die Benutzerauthentifizierung mit E-Mail-Adresse und Passwort, um die Benutzerdaten vor unauthorisiertem Zugriff zu schützen. Dafür muss das Verwaltungssystem die Anmeldedaten des Benutzers verifizieren und nach einer erfolgreichen Anmeldung dem Benutzer zur Identifikation einen Authentifizierungstoken zuweisen.
Für die Authentifizierung werden Industriestandards, wie *OAuth 2.0* und *OpenID Connect* genutzt. Da OpenID Connect auf OAuth 2.0 basiert, können Clients die Identität des Benutzers überprüfen oder Informationen über authentifizierte Sitzungen erhalten.

Cloud Storage Auf dem Server werden die verschiedenen Medienelemente, wie Bilder, Videos oder Audios, gespeichert. Medienelemente können direkt von den Clients auf das Ordner/Dateisystem zugreifen Die Clients können direkt über das Ordner/Dateisystem auf die Medienelemente zugreifen und via Download-URL (siehe Listing 5.4) geladen werden. Dateien oder ganze Ordner können frei zugängig sein oder geschütz werden.

```
1 Storage-Speicherort:
2 gs://proto-ma.appspot.com/Dehnuebung1.jpg
3
4 Download-Url:
5 https://firebasestorage.googleapis.com/v0/b/proto-ma.appspot.co
6 m/o/Dehnuebung1.jpg?alt=media&token=9be48e01-973d-4a2b
```

Listing 5.4: Speicherort eines Beispiel-Bildes

5.3 Aufbau Kontext

Ein Kontext kann in einer abstrakten Baumstruktur (siehe Abb.5.13) dargestellt werden, bei dem die Wurzel der Kontext eine Situation ist [31]. Von der Wurzel führt genau eine Kante zu einem inneren Knoten, der nachkommende innere Knoten miteinander

verknüpft. Die Verknüpfung besteht aus den logischen Operatoren Konjunktion (UND) oder Disjunktion (ODER).
Ein innerer Knoten der Tiefe 2 kann entweder ein Vergleichsoperator oder ein weiterer Kontext sein und kann als die Wurzel der in Abb.5.13 markierten Teilbäume betrachtet werden. Die Teilbäume können auf der einen Seite ein Kontextmerkmal oder andererseits ein weiterer Kontext sein.

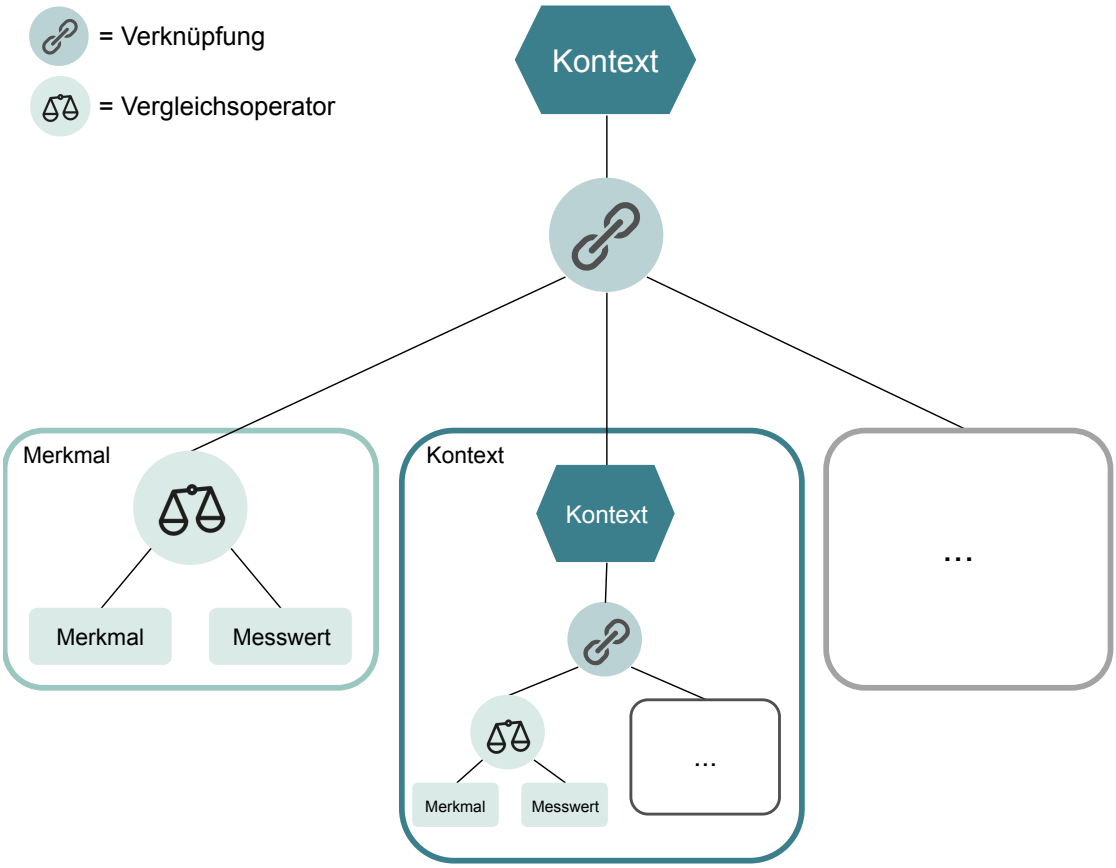

Abbildung 5.13: Schematischer Aufbau eines Kontextes

Der linke Teilbaum in der Abbildung besteht aus einem Vergleichsoperator mit zwei Blättern, die aus dem vom Betreuer definierten Vergleichswert und dem gemessenen Wert bestehen. Der Vergleichsoperator stellt die zwei Werte, die auch ein Objekt oder eine Liste von Werten sein können, gegenüber und liefert einen Wahrheitswert. Neben

der Verwendung des Gleichheitszeichens, des Ausrufezeichens, des Größer- oder Kleinerzeichens (auch in Kombination) sind *Min-/Max-* Operatoren, sowie *Between-*Operatoren notwendig, um die Baumstruktur weiterhin erhalten zu können.
Ein Teilbaum kann wiederum einen weiteren Kontext beschreiben, der analog zum übergeordneten Kontext aufgebaut ist. Direkt nach der Wurzel folgt eine Verknüpfung, der beliebig viele Kinder-Knoten nachfolgen können. Ein Kontext, der weitere Kontexte als Teilbäume hat, wird *komplexer Kontext* genannt. Grundsätzlich kann ein Kontext beliebig tief durch weitere Kontexte geschachtelt werden. Durch die vielen Kontextebenen nimmt auch die Komplexität zu. Die Breite des Baumes wird ebenfalls größer, je mehr Informationsquellen zur Beschreibung einer Situation herangezogen werden.

In der Datenbank wird ein Kontext im JSON-Format strukturiert, um einen einfachen Datenaustausch zu ermöglichen. Das Listing 5.5 zeigt exemplarisch zwei Kontexte, bei denen es sich um einen komplexen Kontext (Z. 1 - 11) und einen einfachen Kontext (Z. 12-18) handelt.

```
1  "startWeekend" : {
2     "features" : [ {
3       "type" : "date",
4       "value" : "Saturday",
5       "valueUnit" : "EEEE"
6     }, {
7       "type" : "context",
8       "value" : "at_home"
9     } ],
10    "operator" : "and"
11  },
12  "at_home" : {
13    "features" : [ {
14      "type" : "geolocation",
15      "value" : "Ulm",
16      "valueUnit" : "address"
17    } ]
18  }
```

Listing 5.5: Beispiel einfacher und komplexer Kontext

Der Kontext ßtartWeekendïst ein komplexer Kontext, weil er aus einem Kontextmerkmal und aus einer Referenz (Z.7 + 8) auf einen weiteren Kontext besteht. Um auf einen anderen Kontext zu referenzieren, muss der Typ ”context” und der referenzierte Name des Kontextes im ”Value”-Attribut angegebenen werden. Der einfache Kontext kann sowohl als Referenz eines anderen Kontextes aufgerufen oder auch direkt als eigenständiger Kontext genutzt werden. Im Beispiel ist dem einfachen Kontext (”at_home”) kein Operator (im Vgl. zu Z. 10) explizit zugewiesen. Aufgrund der Eindeutigkeit ist das auch nicht nötig, da es keine Verknüpfung mit weiteren Kontexten gibt.

5.4 Kontexterkennung

In diesem Abschnitt soll näher auf die Anwendungslogik für die Kontexterkennung eingegangen werden, in dem die einzelnen Aktivitäten des Prozesses näher erläutert werden. Im Aktivitätsdiagramm (Abb. 5.14) sind die drei Schichten

1. Präsentation der Benutzeroberfläche
2. Anwendungslogik
3. Datenhaltung

dargestellt.

Nach der erfolgreichen Anmeldung des Benutzers am System, wird der Prozess der Kontexterkennung für jede einzelne Aufgabe gestartet. Dafür wird ein Kontext mit eindeutigem Namen beim *Kontext Service* einmalig registriert, sodass die Liste der registrierten Kontexte keine Duplikate enthält. Bei der Registrierung wird eine Aktion einem Kontext zugeordnet, die das Senden einer Benachrichtigung oder die rekursive Kontextvaldierung bei komplexen Kontexten sein kann. Die Aktion wird bei einer erfolgreichen Kontexterkennung ausgelöst, bzw. es können auch mehrere Aktionen ausgelöst werden.

Für die eigentliche Kontexterkennung benötigt das System die Kontextmerkmale, die vom Server angefordert werden. Dabei kann einem Kontexttyp eine Liste von Kontextmerkmalen zugeordnet werden. Durch diese Key-Value-Struktur werden keine doppelten Kontexttypen gespeichert, um einen Mehrfachzugriff auf die Sensoren zu vermeiden.

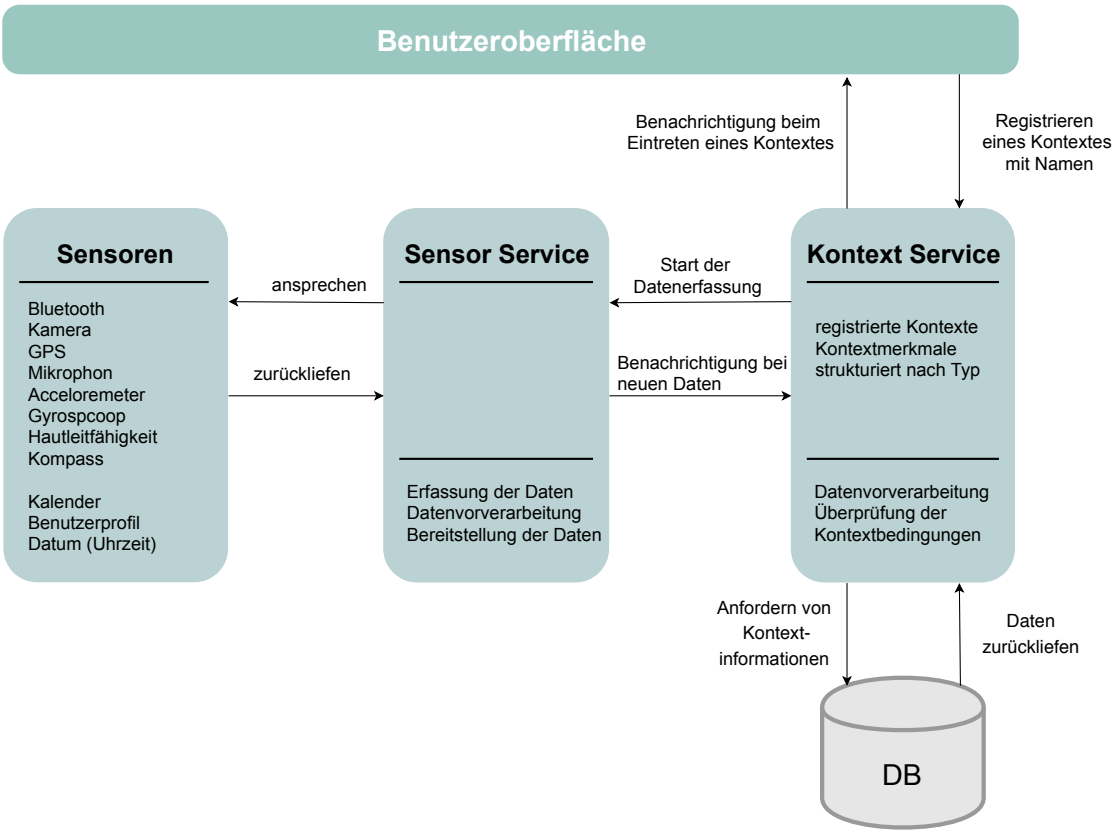

Abbildung 5.14: Abstrahiertes Aktivitätsdiagramm

Das Erfassen, sowie die Bereitstellung der Daten übernimmt der *Sensor Service*. Sobald ein neuer Kontexttyp der Liste hinzugefügt wird, spricht der Sensor Service den entsprechenden Sensor an. Zum Beispiel greift der Sensor Service auf den GPS-Sensor des mobilen Endgerätes zu, wenn im Kontextmerkmal der Typ *Geolocation* definiert ist. Das System muss nicht zwangsweise auf einen nativen Sensoren zugreifen, wenn es sich um einen elementaren Kontextmerkmal (WLAN oder Zeit) handelt. Symbolische Kontexte können ebenfalls nur bedingt sensorisch erfasst werden, wie z.B. der Zugriff auf das Benutzerprofil, um den Standort *zuHause* zu erkennen. Das System kann hingegen die Aktivitäten, wie z.B *Duschen* derzeit nicht erfassen.
Es gibt zwei Arten, die Daten zu sammeln. Der Sensor Service enthält die Daten automatisch von den Sensoren (z.B. GPS) oder muss diese aktiv anfordern (z.B. Uhrzeit). Er selektiert zunächst die Daten und strukturiert sie zur weiteren Verarbeitung.

Die neuen Daten werden dem Kontext Service sofort bereitgestellt und für die Validierung erstmalig verarbeitet (siehe Kapitel 5.5). Die Vorverarbeitung ermöglicht später einen Vergleich der Messdaten mit dem Schwellenwert, die abhängig vom Kontext- und Sensortyp eine unterschiedliche Struktur haben können.

Beispielsweise sind die Daten, die der GPS-Sensor und der Sensor Service liefern, als Objekt mit Latitude und Longitude strukturiert. Während der Betreuer einen Standort nicht unbedingt über Koordinaten, sondern auch über eine Adresse beschreiben kann, der Typ Geolocation ist hier ein Spezialfall. Andere Typen können aufgrund verschiedener möglichen Einheiten mehrdeutig sein, Beispiele sind die verschiedenen Zeit- oder Datumsformate. Während durch die Angabe der Einheit ”EEEE” beim Vergleichswert eine Zeichenkette (Wochentag) oder durch ”H” einen Zahlenwert (Stunde) erwartet wird, ist der gemessene Wert ein *JavaScript Date Object* und muss infolgedessen transformiert werden.
Durch die Vorverarbeitung sind die Werte miteinander vergleichbar und somit kann eine Situation durch einen erkannten Kontext beschrieben werden. Bei einem erkannten Kontext wird die Präsentationsschicht informiert. Es gibt auch Fälle, in denen der Benutzer nicht sofort informiert wird, wenn z.B. der Patient die Aufgabe in einem gewissen Zeitraum hinreichend oft ausgeführt hat.

Außerdem wird in diesem Zusammenhang die Zeitspanne überprüft, in der eine Aufgabe durchzuführen ist. Da eine Aufgabe außerhalb der definierten Zeitspanne nicht durchgeführt werden soll, gilt die Aufgabe somit als abgeschlossen und der Kontext muss für diese Aufgabe nicht weiter überwacht werden.

Für das Pausieren oder Beenden einer Kontexterkennung, muss der Kontext aus der Liste der registrierten Kontexte und der Kontextmerkmale gelöscht werden. Die Überwachung des Sensors muss außerdem gestoppt werden, falls kein weiterer Kontext auf diesen Sensor zugreift. Um die Kontexterkennung wieder starten zu können, muss der Kontext wieder beim Kontext Service registriert werden.

Sobald der Benutzer das System durch seine Abmeldung verlässt, muss die Kontexterkennung komplett eingestellt werden. Dafür werden alle laufenden Sensoraufnahmen gestoppt und alle registrierten Kontexte und Kontextmerkmale aus den Listen entfernt.

5.5 Kontextvalidierung

Die Validierung der Kontexte ist Aufgabe des Kontext Services. Der Kontext Service ist die Schnittstelle zwischen Datenbank, Benutzeroberfläche und dem Sensor Service und damit zu den Sensoren.

Wie im vorangegangem Kapitel beschrieben werden dem Kontext Service die gemessenen Daten übergeben, indem ein registrierter *Callback* aufgerufen wird. Dadurch wird der Kontext bei jeder Datenänderung neu validiert werden. Da dem Kontext Service lediglich der Sensor- bzw. der Kontexttyp und der gemessene Wert übergeben wird, benötigt er zusätzlich alle Kontextmerkmale vom übergebenen Kontexttyp. Jedes Kontextmerkmal setz sich zusammen aus Vergleichswert, Vergleichsoperator und Einheit.
Sind alle Daten vor verarbeitet vorhanden und Spezialfälle abgedeckt, werden die logikbasierten Regeln konstruiert. Dazu ist die abstrakte Baumstruktur des Kontextes nützlich (siehe Aufbau eines Kontextes Abb. 5.13). Die Regeln werden schrittweise für die Teilbäume, angefangen bei den Blättern, aufgestellt. Jede Regel wird durch Key-Value Paare strukturiert. Hierbei ist der ”Key” der Operator und der ”Value” eine Liste von den zu vergleichenden Werten. Eine einfache Regel zur Überprüfung der Gleichheit sieht beispielsweise folgendermaßen aus: $\{\text{“} == \text{“} : [1, 1]\}$.
Wird das Ergebnis der Regel als wahr validiert, wird es beim Kontextmerkmal zwischengespeichert. Anschließen wird das darüber liegenden Niveau des Regelbaums betrachtet. Auf diesem Niveau werden die einzelnen Kontextmerkmale miteinander verknüpft. Für diese Regel werden die zwischengespeicherten Ergebnisse aus der vorangegangenen Validierung benutzt. Diese Regel sieht durch die zwischengespeicherten Ergebnisse folgendermaßen aus: $\{\text{“}and\text{“} : [true, false]\}$, während die komplette Regel im Listing 5.6 dargestellt ist.

```
1        {"and" : [
2                 { "==" : [1,1] },
3                 { ">" : [1,3] }
4        ] }
```

Listing 5.6: Regel einer UND-Verknüpfung

Das Ergebnis dieser logischen Verknüpfung wird ebenfalls zwischengespeichert. Das ist vor allem bei der Validierung von komplexen Kontexten wichtig, damit der übergeordnete Kontext erfolgreich validiert werden kann. Die weitere Validierung bei komplexen Kontexten wird durch die Aktion ausgelöst, die bei der Registrierung von komplexen Kontexten übergeben wird. Diese Aktion wird natürlich nur dann ausgelöst, wenn die logische Regel den Wahrheitswert *wahr* liefert. Die letzte Aktion ist die Benachrichtigung über die erfolgreiche Erkennung eines Kontextes an den Benutzer.

5.6 Verwendete Technologien und Frameworks

Für die Umsetzung der Anwendung, insbesondere der Kontexterkennung muss das System auf die nativen Funktionen und Sensoren des mobilen Endgerät zugreifen. Für eine vereinfachte Entwicklung werden die Cordova-Plugins durch *Ionic Native* integriert. Des Weiteren sind *JsonLogic* und *Geohash* bei der Validierung der Kontexte nützlich.
Für die Umsetzung des Servers wird *Firebase* verwendet. Die Entwicklungs-Plattform für mobile und Web-Anwendungen stellt Funktionen und Infrastruktur zur Verfügung. Für die Integration von Firebase empfiehlt sich hier die offizielle Angular-Bibliothek *AngularFire*.

5.6.1 Ionic Native

Für den Zugriff auf native Funktionen der unterschiedlichen Betriebssysteme wird Ionic Native benötigt, das ein TypeScript-Wrapper für den JavaScript-Code des Cordova-Plugins ist. Dieser ermöglicht eine einfache Integration nativer Funktionen in die mobile Anwendung.Ionic Native besteht aus einer Reihe von Plugins von Drittanbietern und bietet eine gemeinsame Schnittstelle, um sicherzustellen, dass native Ereignisse Änderungen in der Anwendungslogik auslösen. Durch die Entwicklung der Plugins durch Drittanbieter können Probleme bei Updates der Betriebssysteme auftreten. Manche Funktionen eines Betriebssystems können nicht mehr ansprechbar sein.

Ionic Native wird automatisch bei der Erstellung einer Ionic-Anwendung installiert. Für die Installation der einzelnen Plugins muss ein Befehl mit *Ionic CLI* (Command Line

Interface) ausgeführt werden.Im Linsting 5.7 (Zeile 1) ist z.B. der CLI-Befehl zum Installieren des Kamera-Plugins zu sehen. Bevor allerdings das Cordova-Plugin genutzt werden kann, muss noch das spezielle Ionic Native-Paket für das Kamera-Plugin installiert (vgl. Listing 5.7 (Z. 2)) und im *App Module* importiert werden [32].

```
1 npm install @ionic-native/camera --save
2 npm install --save @ionic-native/camera
```

Listing 5.7: CLI-Befehle zum Installieren vom Cordova und Ionic Native Kamera-Plugin

Alle verfügbaren Plugins (über 200) sind auf der Webseite des Ionic Frameworks unter Ionic Native dokumentiert [32]. Die Anwendung verwendet eine Reihe von nativen Plugins zur sensorischen Messung des Kontextes, z.B. das *Calender*-, *DB Meter*-, *Device Motion*-, *Device Orientation*-, *Geolocation*- oder das *Gyroscope*-Plugin. Nicht alle Plugins oder alle Funktionen der Plugins sind für alle Betriebssysteme verfügbar. Die meisten können in iOS und Android verwendet werden. Ein Beispiel ist das Plugin für die Umgebungshelligkeit, das zum Zeitpunkt der Entwicklung lediglich für Android verfügbar war.

5.6.2 Firebase

Firebase ist eine Plattform zur Entwicklung von mobilen und Webanwendungen, die die Serverinfrastruktur und verschiedene Werkzeuge zur Verfügung stellt. Mit Firebase kann die Entwicklung (*Realtime Database*), die Anwendungs-Qualität (*Performance Monitoring*) und die Skalierbarkeit, bzw. das Wachstum (*Google Analytics*) der Anwendung verbessert werden.
Insgesamt stehen dem Entwickler derzeit 15 Funktionen zur Verfügung, die in Abb. 5.15 dargestellt sind.

Das Kernstück von Firebase ist die Echtzeitdatenbank, die das entwickelte Anwendung nutzt. Die Firebase Datenbank strukturiert die Daten in JSON-Dateien und synchronisiert sie in Echtzeit mit dem Client. Firebase bietet auch verschiedene andere Dienstleistungen, beispielsweise die Bereitstellung von Web-Ressourcen, einen unbegrenzten

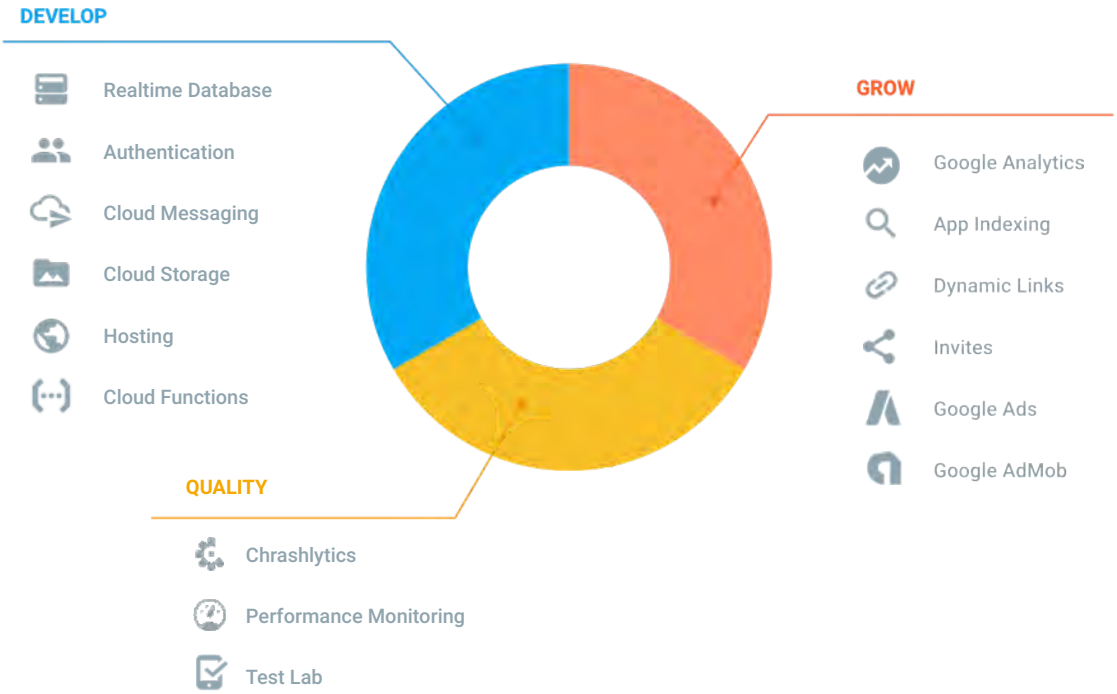

Abbildung 5.15: Übersicht der Funktionen von Firebase (vgl. [33])

Analyse-Dienst (*Firebase Analytics*) und *Firebase Cloud Messaging*, das zum Versenden von Benachrichtigungen und zur Bereitstellung von Instant Messaging verwendet wird.

Der *Authentication Service* bietet dem Entwickler einen Backend-Service, ein Software Development Kit (SDK) und fertige UI-Bibliotheken zur Authentifizierung der Benutzer. Die Authentifizierung kann mit Hilfe von E-Mail und Passwort erfolgen, möglich sind auch Anmeldungen über Google-, Facebook-, Twitter- und GitHub-Konten [34].

Der Entwickler kann mit Hilfe eines SDKs auf Funktionen zugreifen, die Hauptplattformen iOS, Android und das Web unterstützen. Im Fall der *Realtime Database* übernimmt das SDK die Synchronisation der Daten zwischen Client und dem Google Server, dabei muss der Entwickler nur die Daten über eine Schnittstelle austauschen.

AngularFire2 AngularFire gilt als die offizielle Bibliothek für Firebase, die das offizielle Firebase-Paket integriert hat und dem Entwickler eine einfache Schnittstelle für die Integration der Firebase-Funktionen bietet. AngularFire bietet das Echzeit-Synchronisieren der Daten mit der Firebase Realtime Database, die Authentifizierung bei verschiedenen Anbietern und das Verwalten von Daten im *Cloud Storage* an. Die Bibliothek beinhaltet ein Hosting von Angular-Projekte (z.B. Webanwendungen) [35].

Für die Integration von AngularFire müssen Initialisierungsinformationen (siehe Listing 5.8) angegeben werden, um das Firebase JavaScript SDK für die Verwendung von der Echtzeitdatenbank, der Authentifizierung und Cloud Storage zu konfigurieren.

```
var config = {
  apiKey: "AIz[...]lmQ",
  authDomain: "proto-ma.firebaseapp.com",
  databaseURL: "https://proto-ma.firebaseio.com",
  projectId: "proto-ma",
  storageBucket: "proto-ma.appspot.com",
  messagingSenderId: "132[...]331"
}
```

Listing 5.8: Konfiguration von Firebase

5.6.3 JsonLogic

Die Bibliothek JsonLogic ermöglicht das Erstellen komplexer logischer Regeln, die in JSON serialisiert werden. Durch die JSON-Struktur können die Regeln in einer Datenbank gespeichert und einfach zwischen Client und Server ausgetauscht werden [36]. JsonLogic bietet eine konsequente Schreibweise, mit der komplexe Strukturen einfach aufgebaut und erweitert werden können. Außerdem bietet JsonLogic die Möglichkeit neue Operatoren hinzuzufügen. Neben den logischen und booleschen Operationen werden unter anderem Array- und String-Operationen unterstützt. Ebenfalls kann JsonLogic Daten aus bereitgestellten Objekten bearbeiten, die typischerweise zur Laufzeit geschrieben werden.

Die Daten müssen nicht in der Regel stehen, sondern können ausgelagert werden (Listing 5.9 (Z. 6)). Diese komplexe Regel wird durch den Funktionsaufruf in Zeile 8 überprüft, der in diesem Beispiel den Wert *wahr* zurückliefert.

```
1 var rules = { "and" : [
2   {"<" : [ { "var" : "temp" }, 110 ]},
3   {"==" : [ { "var" : "pie.filling" }, "apple" ] }
4 ] }
5
6 var data = { "temp" : 100, "pie" : { "filling" : "apple" } }
7
8 var result = jsonLogic.apply(rules, data)
```

Listing 5.9: Komplexe Beispiel-Regel mit JsonLogic

5.6.4 Geohash

Das Geokodierungssystem *Geohash* gruppiert nahegelegene Orte auf der Erde mit unterschiedlicher Auflösung. Das System kodiert Längen- und Breitengrade eines geographischen Ortes, indem die Erde rekursiv in kleinere Regionen aufgeteilt wird. Der Geohash selber ist eine alphanumerischen Zeichenkette und wird durch das Konvertieren der Koordinaten und der Länge der Zeichenkette gebildet. Die Länge der Zeichenkette bestimmt die Auflösung, bzw. den Genauigkeitsgrad.

Ein Geohash identifiziert eine rechteckige Zelle (siehe Abb. 5.16 blaues Rechteck) und definiert somit die Entfernung zu benachbarten Orte. Nachbarn haben somit den gleichen Präfix (siehe Abb. 5.16). Je länger ein gemeinsamen Präfix ist, desto näher liegen die Orte beieinander. Die Länge bestimmt die Zeichenkette die Größe der Zelle. Im Beispiel wird die Präzision 5 gesetzt, d.h. die Zelle hat eine Breite und Höhe von 4.89km hat. Je länger die Zeichenkette ist, desto kleiner ist die Zelle.
Die verwendete JavaScript-Bibliothek zum Kodierungsverfahren ermöglicht das Kodieren von Breiten- und Längengraden zu einem Geohash, das Decodieren eines Geohashs, das Bestimmen der Grenzen einer Geohashzelle und das Finden von Nachbarn eines Geohashs [37].

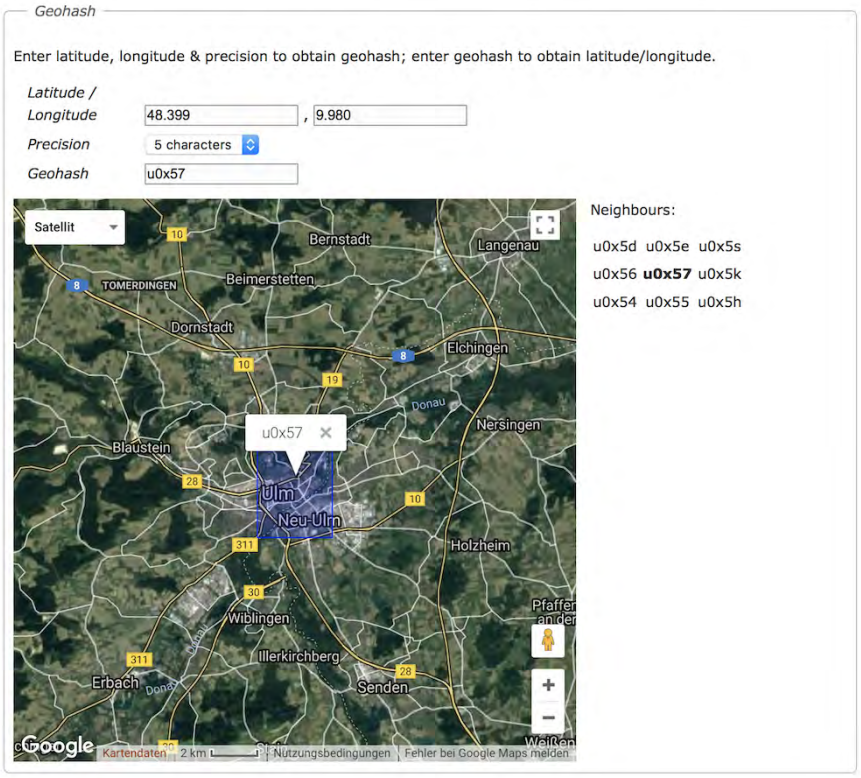

Abbildung 5.16: Geohash am Beispiel Ulm mit der Präzision 5 (vgl. [37])

Die Anwendung verwendet das Geohash-System für den Vergleich zweier Orte, da der Geohash kein genauer Standort sein muss (im Gegensatz zu Koordinaten).

6

Anforderungsabgleich

In diesem Kapitel werden die Anforderungen aus der Anforderungsanalyse (Abschnitt 3.3f.) anhand der prototypischen Umsetzung der mobilen Anwendung verglichen. Durch die Überprüfung kann der Erfüllungsgrad jeder Anforderung bestimmt werden und die Anwendung im Gesamten bewertet werden. Der Erfüllungsgrad (Prozentwert in Klammern) wird zwischen 0 und 100% angegeben.

6.1 Funktionale Anforderungen

Die funktionalen Anforderungen, die an die Benutzeroberfläche gestellt werden (Anforderungen I -VII), sind erfolgreich umgesetzt. Die Anforderungen an die Anwendungslogik werden größtenteils erfüllt, zusätzlich unterstützt die Anwendung verschiedene Sprachen (Deutsch und Englisch).

I **Anmeldung und Registrierung (100%):**
Durch den Einsatz von Firebase ist eine Benutzerverwaltung verfügbar, die dem Benutzer eine Authentifizierung mithilfe der Benutzeroberfläche ermöglicht. Der Zugriff auf persönliche Daten ist nur authentifizierten Benutzern durch die Definition von Sicherheitsregeln möglich.

II **Neue Therapie hinzufügen (80%):**
Neue Behandlungen werden durch die Echtzeit Datenbank automatisch der mobilen Anwendung hinzugefügt. Das manuelle Hinzufügen von Behandlungen ist derzeit nicht möglich. Der Benutzer kann mithilfe der in das moderne mobile Endgerät eingebauten Kamera QR-Codes scannen und interpretieren.

III **Behandlungsübersicht (100%):**
Der Benutzer kann seine Behandlungen jeweils in einer Übersicht (unterteilt in offen und abgeschlossen) einsehen und verwalten. Dabei werden zusätzliche Informationen angezeigt, wie z.B. den behandelnden Therapeuten.

IV **Aufgabenübersicht (100%):**
Der Benutzer kann alle Informationen der Aufgaben einer Behandlung in der mobilen Anwendung auffinden und gemessene Daten in einem Liniendiagramm einsehen.

V **Darstellung der Übungen (100%):**
Der Benutzer findet zu jeder Aufgabe eine Detailansicht vor, die eine Übung multimedial beschreibt. Die Anwendung unterstützt die Medien Text, Bild, Audio und Video, die Übung kann durch eine Download-Funktion offline angesehen werden.

VI **Feedbackformular (100%):**
Nach der letzten Übung zeigt die mobile Anwendung dem Benutzer ein Feedback-

formular. Die Anwendung unterstützt die Formularelemente Textfeld, Einfach- und Mehrfachauswahl und eine Ratingskala.

VII **Kontexterkennung (90%):**
Das System kann Daten sowohl sensorisch erfassen, als auch virtuelle Daten aus Quellen, wie der nativen Kalenderanwendung oder dem Benutzerprofil beziehen. Das System kann jederzeit Kontexte erfassen und erkennen, sofern der Benutzer am System angemeldet ist. Über die Benutzerschaltfläche in der Aufgabenübersicht kann er für jede Aufgabe einzeln die Kontexterfassung aktivieren, bzw. deaktivieren. Über die Schaltflächen in der Notification ist dem Benutzer eine direkte Rückmeldung über die Richtigkeit der Kontexterkennung möglich.

VIII **Unterstützung nativer Funktionen (80%):**
Das System unterstützt einen Großteil der Hardware-Sensoren der mobilen Endgeräte. Dazu gehören die Bewegungssensoren (Accelerometer, Gyroskop) des Gerätes, sowie Sensoren, die verschiedene Umgebungsparameter messen (Barometer, Kompass, Mikrofon) und die physikalische Postion des Gerätes erfassen (GPS). Das System kann auf den Pedometer, der die Schritte des Benutzers in einem gewissen Zeitraum erfasst und auf den nativen Kalender des Betriebssystems zugreifen.

IX **Benachrichtigungen (100%):**
Das System schickt dem Benutzer eine Benachrichtigung, sobald ein Kontext einer Aufgabe erkannt wird. Die Benachrichtigung erscheint dem Benutzer in Abhängigkeit der persönlichen Geräteeinstellungen sowohl im Sperrbildschirm, als auch als Notification mit Warnton.

X **Generische Darstellung (100%):**
Die Anwendung kann in JSON beschriebene inhaltsbezogene Darstellungen interpretieren und anzeigen. Das System beschränkt sich auf die definierten

Eingabe- und Medienelemente. Es bietet eine Kommunikation zwischen Anwendung und Server in Echtzeit, um alle Informationen austauschen zu können.

XI **Download und Offline-Funktion (100%):**
Dem Benutzer kann die Informationen der Übungen vom Server für eine Offline-Nutzung laden.

6.2 Nichtfunktionale Anforderungen

Die nichtfunktionalen Anforderungen beschreiben Qualitäts- und Sicherheitsmerkmale der mobilen hybriden Anwendung und werden größtenteils erfüllt.

I **Funktionalität (100%):**
Die Anwendung unterstützt den Benutzer beim Durchführen von Hausaufgaben, indem das System die Umgebung in einer definierten Frequenz überwacht, einen definierten Kontext erkennt und den Benutzer durch Benachrichtigung motiviert seine Übungen durchzuführen.

II **Benutzerfreundlichkeit (100%):**
Die Anwendung ist übersichtlich mit nativen Bedienelementen gestaltet, so dass die Erwartungen des Benutzers an das Verhalten der Anwendung auf dem jeweiligen Betriebssystem erfüllt werden können. Durch die konsistente Verwendung eines Farbschemas und sechseckiger Formen wird eine einheitliche Benutzeroberfläche von verschiedenen Betriebssystemen erreicht.

III **Zuverlässigkeit (80%):**
Die Anwendung reagiert auf falsche Benutzereingaben tolerant, indem eine Warnung angezeigt wird. Der Benutzer hat darauf hin solange die Möglichkeit seinen

Fehler zu korrigieren, bis das System die Eingabe akzeptiert. Die Verfügbarkeit des Servers ist von der Internetverbindung und von Firebase abhängig.

IV **Effizienz (100%):**
Das System führt alle Aktionen flüssig und performant aus. Die Ladezeiten der Daten sind abhängig von der Internetgeschwindigkeit. Aufgrund der geringen Anzahl von Medienelementen, die zudem im WLAN heruntergeladen werden können, sind die Ladezeiten der JSON-Dateien gering.

V **Portierbarkeit und Erweiterbarkeit (100%):**
Die Anwendung kann aufgrund des hybriden Frameworks Ionic für iOS, Android und weiteren Betriebssystemen, wie Windows oder für den Browser kompiliert und auf den jeweiligen Plattformen installiert werden. Der vollständige Funktionsumfang der Anwendung kann allerdings nur auf den Betriebssystemen iOS und Android gewährleistet werden. Durch Hinzufügen von Plugins von Drittanbietern kann das System leicht auf weitere Sensoren zugreifen, sofern diese für das Betriebssystem zur Verfügung stehen.

VI **Sicherheit und Datenschutz (80%):**
Der Benutzer muss sich am System mit seiner E-Mail-Adresse und Passwort authentifizieren, um seine personenbezogenen Daten seiner Behandlungen zu erhalten.

6.3 Zusammenfassung

Abschließend werden in der Tabelle 6.1 alle funktionalen und nichtfunktionalen Anforderungen zusammengefasst. Dabei wird auch der Erfüllungsgrad in Prozent aufgelistet, der in den Kapitel zuvor ermittelt wurde.

	Funktionale Anforderungen	
I	Anmeldung und Registrierung	100%
II	Neue Therapie hinzufügen	80%
III	Behandlungsübersicht	100%
IV	Aufgabenübersicht	100%
V	Darstellung der Übungen	100%
VI	Feedbackformular	100%
VII	Kontexterkennung	90%
VIII	Unterstützung nativer Funktionen	80%
IX	Benachrichtigungen	100%
X	Generische Content-Darstellung	100%
XI	Download und Offline-Funktion	100%

	Nichtfunktionale Anforderungen	
I	Funktionalität	100%
II	Benutzerfreundlichkeit	100%
III	Zuverlässigkeit	80%
IV	Effizienz	100%
V	Portierbarkeit und Wartbarkeit	100%
VI	Sicherheit und Datenschutz	80%

Tabelle 6.1: Zusammenfassung des Anforderungsabgleichs

7

Zusammenfassung

Durch die Einbindung von Plugins von Drittanbietern ist es der mobilen hybriden Anwendung mithilfe des Ionic-Frameworks möglich, Kontexte zuerkennen, um Patienten bei der Durchführung seiner Hausaufgaben zu motivieren und zu unterstützen. Durch die Anbindung des Webservices können verschiedene Benutzer und Medienelemente mithilfe der Entwicklungs-Plattform Firebase verwaltet und beliebige Situationen durch Kontexte zentral beschrieben werden.

Für die Konzeption und Entwicklung der plattformübergreifenden Anwendung wurden zunächst die unterschiedlichen Hausaufgaben näher betrachtet, die im Rahmen therapeutischer Interventionen als Standardtechnik für einen effizienteren Therapieprozess gelten. Da die Durchführung von Hausaufgaben nicht im Alltag kontrolliert werden kann, bietet sich der Einsatz von Smartphones an, um Patienten über zugewiesene Hausaufgaben zu informieren und zu einer regelmäßigen Durchführung in passender Umgebung zu motivieren. Für die Beschreibung einer Umgebung oder Situation wurden Kontexte definiert. Die Kontexte werden mithilfe von Sensoren gesammelt und für das System

lesbar verarbeitet, damit das System eine Entscheidung treffen und dementsprechend handeln oder sich anpassen kann.

Aufgrund der benötigten Zugriffe auf native Sensoren und Funktionen wurde der hybride Ansatz für die Umsetzung der mobilen Anwendung gewählt.

Das System stellt authentifizierten Benutzern eine Übersicht über Therapien und dessen Aufgaben zur Verfügung, wobei jeder Aufgabe mindestens eine Übung und ein Kontext zugeordnet werden. Ein Kontext verknüpft weitere Kontexte oder mehrere Kontextmerkmale, die einen gemessenen Wert mit einem vordefinierten Schwellenwert vergleicht. Der Schwellenwert ist je nach Typ des Sensors unterschiedlich definiert, wobei das System u.a. auf den Accelerometer, das Gyroskop, den Barometer, den Kompass, das Mikrofon, GPS und den nativen Kalender zugreifen kann.

Die Kontextvalidierung basiert auf Regeln und liefert dem System einen Wahrheitswert zurück. Wird ein Kontext erfolgreich erkannt, wird der Benutzer über eine Notification informiert. Ihm wird die Übung zum Kontext der zugehörigen Aufgabe in der mobilen Anwendung angezeigt. Durch die Verwendung multimedialer Inhalte kann die Anwendung den Benutzer zusätzlich bei der Durchführung der einzelnen Übungen unterstützen.

Integriert wurde eine Statistik-Funktion, die dem Benutzer während einer Übung gemessene Vitalwerte oder die Dauer einer Übung anzeigt. Die Statistik dient zur zusätzlichen Motivation weiterhin Hausaufgaben durchzuführen.

Bei der hybriden Entwicklung der mobilen Anwendung wurde auf eine einheitliche Benutzeroberfläche trotz plattformspezifischer User-Interface-Styleguides geachtet. Der Fokus lag auf einer funktionalen Gestaltung, um den Benutzer bestmöglich bei den therapeutischen Interventionen zu unterstützen.

7.1 Ausblick

Die Anwendung kann durch den Zugriff auf die meisten Sensoren eines modernen mobilen Endgeräts jetzt bereits viele Kontexte, bzw. Situationen erkennen. Zusätzliche Sensoren könnten weitere Situationen beschreiben. Außerdem könnte die Effizienz der

Kontexterkennung durch die Integration von maschinellem Lernen gesteigert werden. Der Funktionsumfang der Anwendung könnte beliebig vergrößert werden.

Sensoren: Weitere Sensoren können für die Kontexterkennung eingesetzt werden, so dass sich weitere Situationen beschreiben lassen. Beispielsweise könnte die Umgebungshelligkeit mittels dem Photometer oder die Umgebungstemperatur und - feuchtigkeit mittels einem in das Endgerät integriertes Thermometer gemessen werden. Die Integration dieser Sensoren setzt die stetige Weiterentwicklung von IonicNative und der Sensoren für moderne mobile Endgeräte voraus.
In der Zukunft könnte die Anwendung die Daten zusätzlich von externen Sensoren beziehen, wie z.B einem Brustgurt für die Messung der Herzfrequenz.

Integration von maschinellem Lernen: Als Teilgebiet der künstlichen Intelligenz können Anwendungen mittels maschinellem Lernen Muster erkennen und eigenständig Lösungen zu Problemen finden. Maschinelles Lernen könnte in der Anwendung dann eingesetzt werden, wenn der Benutzer einen falsch erkannten Kontext zurückmeldet. Durch die Rückmeldung des Benutzers kann die Anwendung eigenständig lernen, um die Kontexterkennung auf Basis erkannter Muster zu optimieren.
Maschinelles Lernen könnte zusätzlich zur Erkennung von komplexeren Benutzeraktivitäten eingesetzt werden, wenn z.B. Bewegungsmuster vorhanden sind.

Datenschutz: Bei Anwendungen mit einem gesundheitlichem Bezug ist die Erhebung, Verarbeitung und Nutzung der personenbezogenen Daten nur unter strengen Anforderungen zulässig. Der Benutzer muss über die Erhebungs- und Verarbeitungszwecke sowie bestehende Risiken informiert werden. Die Anwendung muss die Absichten der Datenspeicherung und den Empfänger der Datenübertragung transparent darstellen. Empfohlen wird eine Datenspeicherung auf Servern innerhalb der EU Es wird empfohlen die Daten bei externen Anbietern unter höchsten Sicherheitsstandards auf Servern innerhalb der EU zu speichern [38, 39].
Als Alternative zur Passworteingabe im Login-Prozess kann die Anwendung in Zukunft den Fingerabdruck verwenden, um die Befugnis des Benutzers bei jedem Öffnen der

Anwendung sicherzustellen. Auch eine Zwei-Faktor-Authentifizierung bietet einen guten Schutz auf personenbezogenen Daten.

Gesamtanwendung: Derzeit liegt der Fokus dieser entwickelten Anwendung auf der Kontexterkennung. Andere Bereiche und Funktionen, wie ein möglicher Marktplatz zur Anzeige freier Behandlungsplätze, wurden bei der Entwicklung vernachlässigt.
Der mobile Prozess der komplexen Übungen wird in dieser Anwendung abstrahiert dargestellt, da er in anderen Abschlussarbeiten bereits thematisiert wurde. Um das Gesamtsystem Albatros zu vervollständigen, müssen alle ausgearbeiteten Teilbereiche in einer Anwendung zusammen geführt werden.

Literaturverzeichnis

[1] Techniker Krankenkasse, Präsentation Klaus Rupp: SmartHealth–Wie smart ist Deutschland? aus TK Pressemappe SmartHealth. https://www.tk.de/centaurus/servlet/contentblob/914416/Datei/3604 (2016)

[2] Mattern, F.: Ubiquitous computing: Szenarien einer informatisierten Welt. In: E-Merging Media. Springer (2004)

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

[4] Fehm, L., Helbig-Lang, S.: Hausaufgaben in der Psychotherapie. Psychotherapeut (2009)

[5] Tang, W., Kreindler, D.: Supporting Homework Compliance in Cognitive Behavioural Therapy: essential Features of Mobile Apps. JMIR mental health (2017)

[6] Schickler, M., Pryss, R., Schlee, W., Probst, T., Langguth, B., Schobel, J., Reichert, M.: Usability Study on Mobile Processes Enabling Remote Therapeutic Interventions. In: 2018 IEEE 31st International Symposium on Computer-Based Medical Systems (CBMS), IEEE (2018)

[7] Depp, C.A., Mausbach, B., Granholm, E., Cardenas, V., Ben-Zeev, D., Patterson, T.L., Lebowitz, B.D., Jeste, D.V.: Mobile Interventions for Severe Mental Illness: Design and Preliminary Data from Three Approaches. The Journal of Nervous and Mental Disease (2010)

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

[9] Institute of Databases and Information Systems: MobileTx. (https://www.uni-ulm.de/en/in/iui-dbis/research/running-projects/mobile-tx/)

[10] Dey, A.K., Abowd, G.D.: Providing Architectural Support for Building Context-Aware Applications. College of Computing, Georgia Institute of Technology (2000)

[11] Knappmeyer, M., Kiani, S.L., Reetz, E.S., Baker, N., Tonjes, R.: Survey of Context Provisioning Middleware. (http://eprints.uwe.ac.uk/18351/1/Survey_of_Context_Provisioning_Middleware.pdf)

[12] Hoseini-Tabatabaei, S.A., Gluhak, A., Tafazolli, R.: A Survey on Smartphone-Based Systems for Opportunistic User Context Recognition. (https://www.researchgate.net/profile/Seyed_Amir_Hoseini_tabatabaei/publication/244476949_A_Survey_on_Smartphone-Based_Systems_for_Opportunistic_User_Context_Recognition)

[13] Ionic Framework: Ionic Native - Calendar. (https://ionicframework.com/docs/native/calendar/)

[14] Baldauf, M., Dustdar, S., Rosenberg, F.: A Survey on Context-Aware Systems. International Journal of Ad Hoc and Ubiquitous Computing (2007)

[15] Gellersen, H.W., Schmidt, A., Beigl, M.: (Multi-Sensor Context-Awareness in Mobile Devices and Smart Artifacts)

[16] Rose, P.D.T.: Technische Projektarbeit - Anwendungen von Sensoren. https://www.fh-muenster.de/pt/labore/forschung/sensortechnik/downloads/Tech_Proj_MscWIIng.pdf (2010)

[17] Strang, T., Linnhoff-Popien, C.: A Context Modeling Survey. In: Workshop on Advanced Context Modelling, Reasoning and Management, UbiComp. (2004)

[18] Ionic Framework: Hybrid vs. Native. (https://ionicframework.com/books/hybrid-vs-native)

[19] Wilken, J., Bradley, A.: Ionic in Action: Hybrid Mobile Apps with Ionic and Angularjs. Manning Publications (2016)

[20] Riippi, J.: Native, Hybrid and HTML5 – Three Years Later. https://www.vincit.fi/blog/native-hybrid-and-html5-three-years-later/ (2016)

[21] Biørn-Hansen, A., Majchrzak, T.A., Grønli, T.M.: Progressive Web Apps: the Possible Web-Native Unifier for Mobile Development. In: Proceedings of the 13th International Conference on Web Information Systems and Technologies (WEBIST). (2017)

[22] Osmani, A.: Getting Started with Progressive Web Apps. (https://addyosmani.com/blog/getting-started-with-progressive-web-apps/)

[23] Krebs, S.: 3 Gründe, Warum Progressive Web Apps Die Besseren Web-Apps Sind! | App Entwicklung München – Biz Factory GmbH. https://www.biz-factory.de/blog/2016/08/04/3-gruende-warum-progressive-web-apps-die-besseren-apps-sind/ (2016)

[24] Griffith, C.: Mobile App Development with Ionic 2: Cross-Platform Apps with Ionic, Angular, and Cordova. O'Reilly Media (2017)

[25] Angular: Angular - Architecture Overview. (https://angular.io/guide/architecture)

[26] Marx, L.: Is Angular 2+ MVVM? https://malcoded.com/posts/angular-2-components-and-mvvmLukasMarx (2016)

[27] Microsoft Corporation: Xamarin-App-Entwicklung mit Visual Studio | Visual Studio. (https://visualstudio.microsoft.com/de/xamarin/)

[28] Sharma, A.: What is Xamarin? What is its need? https://www.quora.com/What-is-Xamarin-What-is-its-need (2018)

[29] AltexSoft: The Good and The Bad of Xamarin Mobile Development. https://www.altexsoft.com/blog/mobile/pros-and-cons-of-xamarin-vs-native/ (2018)

[30] Inguanez, F.: Entity Relationship Modelling. (https://frankieinguanez.files.wordpress.com/2012/01/barkers-erd-notation.pdf)

[31] Wang, P., Zhao, J., Liao, Q.: A Cross-Platform Context-Aware Application Developing Framework for Mobile Terminals. (In: 2012 IEEE 2nd International Conference on Cloud Computing and Intelligence Systems)

[32] Ionic Framework: Ionic Native. (https://ionicframework.com/docs/native/)

[33] Jackson, J.: Google Firebase Trims the Middle Tier for Faster App Dev - The New Stack. https://thenewstack.io/google-firebase-trims-middle-tier-faster-app-dev/ (2016)

[34] Google Developers: Firebase. (https://firebase.google.com/)

[35] East, D.: Github - Angular/Angularfire2: The Official Angular Library for Firebase. (https://github.com/angular/angularfire2)

[36] Wadhams, J.: Jsonlogic. (http://jsonlogic.com/)

[37] Veness, C.: Geohash. (https://www.movable-type.co.uk/scripts/geohash.html)

[38] Die Bundesbeauftragte für den Datenschutz und die Informationsfreiheit: Gesundheits-Apps. https://www.bfdi.bund.de/SharedDocs/Publikationen/DatenschutzKompaktBlaetter/Gesundheits-Apps.pdf (2016)

[39] TK Presse & Politik: Standpunkt: Digitalisierung im Gesundheitswesen. https://www.tk.de/tk/themen/digitale-gesundheit/position-digitale-gesundheit/942942 (2018)

Abbildungsverzeichnis

Tabellenverzeichnis

Listings