ulm university universität
uulm

Universität Ulm | 89069 Ulm | Germany

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

Konzeption und Realisierung eines Rahmenwerks zur Unterstützung von Therapeuten bei der Durchführung von Patientenbehandlungen

Masterarbeit an der Universität Ulm

Vorgelegt von:
Michael Stach
michael.stach@uni-ulm.de

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

Betreuer:
Marc Schickler

2016

Fassung 14. November 2016

Satz: PDF-LaTeX 2$_\varepsilon$

Kurzfassung

Die Verbreitung von Smartphones im Alltag ist längst über alle Altersschichten hinweg erreicht und irreversibel. Dieser Umstand ermöglicht die Etablierung neuer IT-Dienstleistungen in bestehende Märkte, die bereits über einen ausgeprägten Benutzerkreis verfügen. So verzeichnet der Bereich der Fitness und Gesundheits-Apps in den letzten Jahren ein stetiges Wachstum, wobei der Scheitelpunkt dieser Entwicklung noch nicht erreicht ist. Gerade die Untergruppe der elektronischen Gesundheitsdienste, die mobile Endgeräte zur Unterstützung ihrer Dienste verwenden, steht noch am Beginn einer flächenmäßigen Verbreitung, obwohl bereits heute eine Bereitschaft zur Benutzung solcher Dienste vorherrscht. Diese große Bereitschaft beschränkt sich dabei nicht nur auf die Patienten elektronischer Gesundheitsdienste, auch Ärzte, Therapeuten und Forscher zeigen ein besonderes Interesse an den neu entstandenen Möglichkeiten. Da psychische Erkrankungen bereits heute zu der zweithäufigsten Diagnosegruppe gehört, verlangt die Disziplin der Psychotherapie nach neuen effizienten Lösungen, die den Therapeuten bei der Erstellung, Überwachung und ständigen Weiterentwicklung der Behandlung unterstützt. Gleichzeitig ist die Sammlung von Datensätzen zu Forschungszwecken ein stetiges Anliegen von Wissenschaftlern, um neue Erkenntnisse zu gewinnen.

Das Ziel dieser Arbeit ist die Entwicklung und Implementierung eines generischen Rahmenwerks zur Unterstützung von Therapeuten bei der Durchführung ihrer Patientenbehandlungen. Dafür wurde eine Architektur konzipiert, um mittels Mobile Crowdsensing neuartige Patientendaten zu erheben, strukturiert zu speichern und zu verwalten. Dieses Rahmenwerk unterstützt dabei das Therapiemittel der therapeutischen Hausaufgaben und wird am Beispiel der Disziplin Psychotherapie entwickelt. Aufgrund des generischen Aufbaus kann das System jedoch auf weitere Disziplinen, welche Hausaufgaben als Therapiemittel verwenden, übertragen werden. Um den Patienten bei der Durchführung seiner Hausaufgabe zu unterstützten und gleichzeitig zu motivieren, verwaltet das Rahmenwerk alle Behandlungen des Patienten und benachrichtigt ihn, sobald eine Hausaufgabe ausgeführt werden muss. Gleichzeitig kann der Therapeut über die Anwendung den Kontext der Durchführung konfigurieren und somit Einfluss auf die Behandlungssituation außerhalb der Therapiesitzung nehmen. Weitere Informationen

zur Ausführung erhält der Therapeut über die Abgaben des Patienten, welche nach jeder Durchführung der Hausaufgaben erstellt und über ein Monitoring-Tool überwacht und analysiert werden kann. Mithilfe dieser Informationen kann der Therapeut Hausaufgaben fortlaufend an die individuellen Bedürfnisse des Patienten anpassen. Um den emotionalen Zustand nach Beendigung der Hausaufgabe zu erlangen, bietet das Rahmenwerk Feedback-Vorlagen zur Rückkopplung an, über die der Patient Fragen beantworten oder multimediale Inhalte zu seiner Abgabe hinzufügen kann. Gleichzeitig können Patienten ihre Daten für Studien freigeben, um neue Erkenntnisse zu gewinnen und als Rückwirkung effektivere Behandlungen zu erhalten.

Danksagung

An dieser Stelle möchte ich mich bei allen Personen bedanken, die mich während der Anfertigung dieser Arbeit motiviert und unterstützt haben.

Zunächst möchte ich mich bei Herrn *Prof. Dr. Manfred Reichert* für die Begutachtung und die Unterstützung bei der Fertigstellung dieser Arbeit bedanken. Ich möchte mich ebenfalls bei *Marc Schickler* für die Betreuung dieser Arbeit und die interessanten Diskussionen bedanken. Mein Dank gilt auch *Johannes Schobel*, der bei Fragen oder Problemen stets ein offenes Ohr für mich hatte.

Ganz herzlich möchte ich mich bei Herrn *Dr. Rüdiger Pryss* für die Begutachtung dieser Arbeit, sowie für seinen intensiven Einsatz und die Förderung während meines gesamten Masterstudiums, bedanken.

Ich möchte mich außerdem bei *Aliyar Aras* für das Korrekturlesen und die angenehme Zeit im Laufe der Ausarbeitung unserer Abschlussarbeiten bedanken. Ihm und all den Kommilitonen, die nicht namentlich erwähnt wurden, danke ich ferner für die interessante und schöne Studienzeit.

Besonderen Dank schulde ich *meiner Freundin* für ihre Geduld und ihr Verständnis während meines gesamten Studiums.

Abschließend möchte ich mich bei *meinen Eltern* bedanken, die mich stets durch bedingungslosen Rückhalt unterstützt und mir das Studium erst ermöglicht haben.

Inhaltsverzeichnis

„Der Anfang ist die Hälfte des Ganzen.“

Aristoteles

1 Einleitung

Dieses Kapitel motiviert zuerst, dass IT-Lösungen gekoppelt mit mobilen Endgeräten eine große Chance für den Gesundheitssektor bieten. Anschließend wird beschrieben, welche Problemstellung für das Therapiemittel therapeutische Hausaufgaben existieren und welche Ziele mit dieser Arbeit verfolgt werden. Zuletzt folgt der Aufbau dieser Arbeit.

1.1 Motivation

Das von Gordon Moore im Jahr 1965 aufgestellte *Moore's Law* spiegelt die rasante Entwicklung der letzten Jahre in der Informationstechnologie wider. Es besagt, dass sich die Dichte der Transistoren pro Flächeneinheit auf integrierten Schaltkreisen alle 12 bis 24 Monate verdoppelt. Dieses exponentielle Wachstum hat zur Folge, dass immer kleinere, leistungsstärkere und energiesparende Computerchips entwickelt werden können [1]. Am spürbarsten ist dieses anhaltende technologische Wettrennen im Bereich der mobilen Endgeräte, insbesondere bei der Weiterentwicklung von Smartphones. Sie sind längst vollwertige Computer im Taschenformat und drängen, dank performanter

Prozessoren und vielfältigen Sensoren, als aktiver, smarter Begleiter in unseren Alltag. Im Jahr 2016 besitzen in Deutschland bereits 49 Millionen Menschen ein Smartphone [2] und laut einer forsa-Studie im Auftrag der Techniker Krankenkasse (TK) benutzen 89% der 18 bis 29-Jährigen und selbst jeder Zweite der 60 bis 70-Jährigen ein Smartphone [3]. Diese Entwicklung hat einen riesigen Markt für IT-Dienstleistungen geöffnet und ist nicht zuletzt der Masse an mobilen Anwendungen (Apps) geschuldet, die in den App-Stores der verschiedenen Betriebssystemen angeboten werden. Inzwischen gibt es Dienstleistungen und Apps für jeden Lebensbereich: Von *Spielen*, *Business*-, und *Unterhaltungs*-Apps bis zu *Fitness & Gesundheits*-Apps [4].

Letztere sind Bestandteil eines wachsenden Bereichs, zu dem auch elektronische Gesundheitsdienste (eHealth) zählen. Unter eHealth bezeichnet man Anwendungen und Dienstleistungen, die zur Diagnose, Behandlung, Überwachung und Verwaltung von Patienten moderne Informations- und Kommunikationstechnologien (ITK) verwenden [5]. Die Kombination aus ITK und Gesundheitsdiensten ermöglicht neue und innovative Dienste, die den Patienten selbst als integralen Bestandteil der Dienstleistung einsetzen, indem er beispielsweise mit seinem Smartphone Daten aus seiner umliegenden Umgebung erfasst und teilt. Dank des hohen Verbreitungs- und Nutzungsgrad von Smartphones können großflächig Daten erhoben werden, die im Anschluss Forschern die Chance auf neue Erkenntnisse und Behandlungsmethoden eröffnet. Die mobile Unterstützung von Gesundheitsdiensten, oft auch mobile Health (mHealth) genannt, verspricht weiterhin großes Potenzial, da Kosten eingespart, unnötige Sprechstunden vermieden und gleichzeitig der Patient enger in die Behandlung eingebunden wird [6]. So ist mobile Healthcare bereits seit einiger Zeit Gegenstand der Forschung und wurde bereits in einigen Arbeiten [7, 8, 9, 10, 11, 12, 13, 14, 15] behandelt. Laut der *#SmartHealth-Studie* der TK können sich zwei Drittel aller Befragten, die Gesundheitsapps nutzen oder nutzen würden, vorstellen, therapieunterstützende Apps einzusetzen (Abbildung 1.1).

Die Erhöhung der Adhärenz[1] und Integration der Behandlung in den Alltag des Patienten erweckt auch in der Disziplin der Psychotherapie das Interesse an

[1] Übereinstimmung des Ausmaßes des Verhaltens einer Person mit den vereinbarten Empfehlungen des Therapeuten. [16]

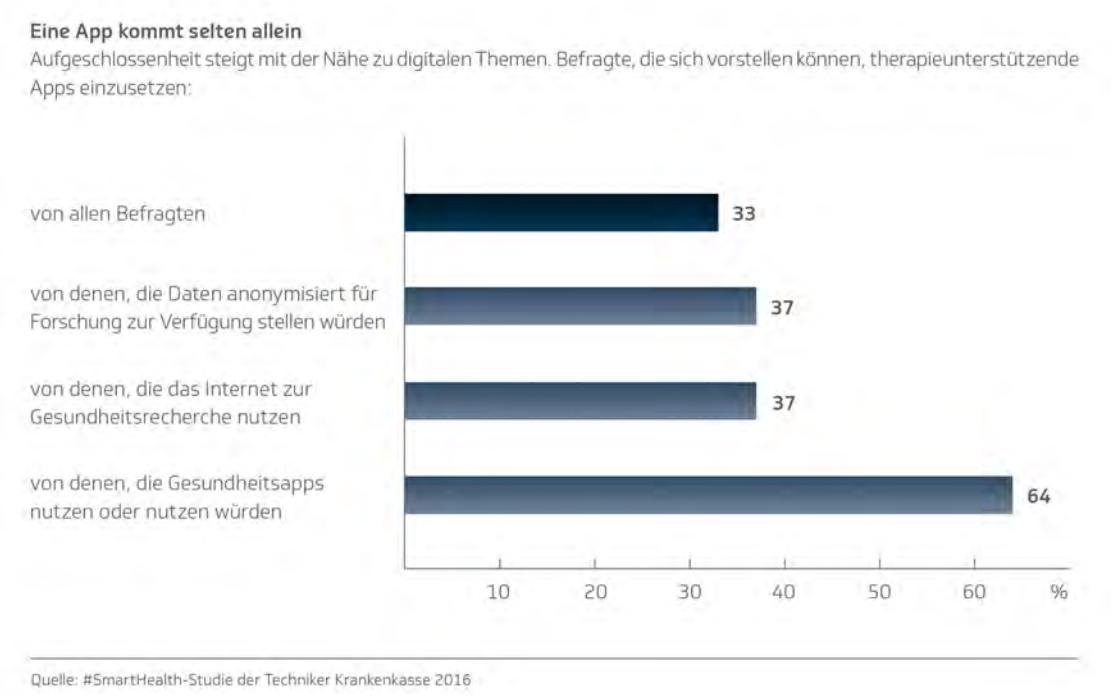


Abbildung 1.1: Bereitschaft zur Verwendung von therapieunterstützender Apps [3]

mHealth. Allein in Deutschland nehmen psychische Erkrankungen, als Ursache für Arbeitsunfähigkeit, seit etwa 15 Jahren kontinuierlich zu [17] und sind 2015 bereits die zweithäufigste Diagnosegruppe bei Krankschreibungen [18]. Dieser anhaltende Trend verlangt effiziente Lösungen, die den Therapeuten bei der Erstellung, Überwachung und ständigen Weiterentwicklung der Behandlung unterstützt. Eine beliebte psychotherapeutische Intervention ist die therapeutische Hausaufgabe, die bereits seit vielen Jahren zum Standardrepertoire [19] und zu den Interventionen mit dem größten Entwicklungspotenzial [20] zählt.

1.2 Problemstellung

Gemäß einer Metaanalyse sind Psychotherapien, bei denen therapeutische Hausaufgaben als Therapiemittel eingesetzt wurden, erfolgreicher als Therapien ohne

Hausaufgaben [21]. Um die Wirksamkeit von Therapien zu erhöhen sind therapeutische Hausaufgaben ein probates Mittel, wobei gleichzeitig einige Herausforderungen für den Therapeuten entstehen. Schon während der Erstellung von Hausaufgaben muss der Therapeut achtgeben, dass Hausaufgaben weder zu belanglos noch zu schwierig sind [22]. Darüber hinaus muss der Therapeut stets die Aufgabendurchführung und die Auswirkung auf den Patienten im Blick behalten, um im Rahmen von Nachbesprechung die Aufgabe an die Bedürfnisse des Patienten anzupassen und die Effektivität der Aufgabe für die Intervention zu gewährleisten. Daneben benötigt der Patient, eventuell gemeinsam erarbeitete Materialien wie Notizen, Schaubilder oder multimediale Aufnahmen, damit er die Aufgabe fehlerfrei durchführen kann. Gleichzeitig kann es notwendig sein, dass der Therapeut dem Patienten eine Situation für die Aufgabendurchführung vorschreibt, die für eine förderliche Wirkung eingehalten werden muss.

Die größte Herausforderung für den Therapeuten ist hierbei herauszufinden, ob der Patient außerhalb der Sitzung die gegeben Vorgaben einhält, da oftmals die Schwierigkeit, der Umfang oder die Aufgabe vom Patienten selbst verändert werden [19]. Der Therapeut kann Abweichung von seiner Aufgabenstellung nur während der Nachbesprechung aus dem subjektiven Bericht des Patienten erfragen, wodurch die Anpassung der Aufgabe und somit die Erhöhung der Wirksamkeit erschwert wird. Aufgrund dessen ist eine objektivere Aufzeichnung und Auswertung der Aufgabendurchführung für eine erfolgreiche Nachbesprechung unerlässlich. Darüber hinaus ist die Erledigung der Hausaufgabe, die der Therapeut zwischen den Sitzungen nicht beeinflussen kann, eines der häufigsten Probleme von Therapeuten [23]. Positive Auswirkung auf die Durchführungsrate können hierbei aktive Aufforderungen und in Einzelfällen die „direkte Vollzugsmeldung“ des Patienten besitzen [19]. Hat der Therapeut, gegebenenfalls mit dem Patienten, eine wirksame Aufgabe entwickelt oder hat er diese aus Fachliteratur bezogen, so muss gewährleistet werden, dass der Arbeitsaufwand für eine Wiederverwendung der Aufgabe so gering wie möglich ist. Um die tatsächliche Wirksamkeit zu überprüfen und die nötigen Anpassungen nachzuvollziehen, muss der Therapeut die verschiedenen Zustände vergleichen und auf weitere Patienten übertragen können.

1.3 Zielsetzung

Ziel dieser Arbeit ist die Entwicklung und Implementierung eines generischen Rahmenwerks für einen mHealth-Dienst mittels Mobile Crowdsensing. Dieses Rahmenwerk soll für das Therapiemittel therapeutische Hausaufgaben in der Psychotherapie umgesetzt, jedoch leicht an weitere Therapieformen, welche Hausaufgaben für die Therapie verwenden, angepasst werden. Das zu entwickelnde System soll Therapeuten eine benutzerfreundliche Plattform zur Erstellung, Durchführung und Auswertung von therapeutischen Hausaufgaben bieten und mithilfe mobiler Endgeräte den Patienten bei der Umsetzung dieser Aufgaben unterstützen und motivieren, sodass die Adhärenz zunimmt und der Behandlungsfortschritt beschleunigt wird.

Der Therapeut soll innerhalb des Systems Aufgaben erstellen und diese mit diversen multimedialen Inhalten anreichern können. Gleichzeitig soll er, getrennt von der Aufgabenstellungen, verschiedene Situationen, Anforderungen an die zu verwendenden Hilfsmittel und Umgebungsbedingungen definieren können, die er als kontextuelle Vorgabe abspeichert. Die Trennung und flexible Kombination aus kontextueller Anforderung und Aufgabenstellen ist besonders notwendig, da eine Aufgabe in verschiedenen Kontexten wiederholt werden muss, um stabile Verhaltensänderungen zu gewährleisten. Sowohl Aufgabenstellung als auch kontextuelle Vorgaben sollen bei jeder Änderung als neue Version zur Verfügung stehen, damit ein Verlauf der Anpassungen für die Analyse des Behandlungsfortschritts und die Hausaufgabenqualität generiert werden kann. Jede Version kann als Vorlage abgespeichert und innerhalb des eigenen Instituts geteilt werden, um die Wiederverwendung von erstellten Inhalten zu vereinfachen. Für angelegte Behandlungen, soll der Therapeut Aufgabenstellung und kontextuelle Vorgabe kombinieren können und somit eine individuelle therapeutische Hausaufgabe generieren. Für jede therapeutische Hausaufgabe kann der Patient Abgaben einreichen, die die aufgezeichneten Daten während der Durchführung beinhaltet. Zusätzlich kann der Therapeut ein Feedback in Form von multimedialen Inhalten oder Antworten auf vordefinierte Fragen verlangen, um den Zustand nach der Durchführung zu erhalten. Der Patient hat hierbei jederzeit die Möglichkeit Änderungen an der Aufgabe anzufordern,

falls er während der Aufgabendurchführung Probleme hatte. Weiterhin soll das System die Möglichkeit bieten, dass Patienten ihre aufgezeichneten Daten für Studien zu Verfügung stellen. Somit bietet sich Forschern die Chance mit diesem System neue Therapieformen zu finden und die Wirksamkeit von therapeutischen Hausaufgaben zu verbessern.

ZIELE DER PROJEKTTEILNEHMER	
Patient	• Erinnerung an die Durchführung der Hausaufgabe • Unterstützung durch multimediale Beschreibung und Anhänge • Integration der Hausaufgabe in den Alltag • zentrale Verwaltung der Hausaufgaben • ort- und zeitunabhängige Verwaltung der Hausaufgaben • Materialien zur Hausaufgabe müssen nicht zusätzlich mitgeführt werden • direkte Rückkopplung mit dem Therapeuten (beispielsweise bei Anpassungen oder Vollzugsmeldungen)
Therapeut	• zentrale Verwaltung von therapeutischen Hausaufgaben • Wiederverwendbarkeit und einfache Anpassung von Hausaufgaben • individuelle und multimediale Beschreibung der Hausaufgabe • multimediales Feedback direkt nach der Durchführung • Unterstützung des Patienten durch Erkennung des Kontextes • Motivierung des Patienten durch Benachrichtigung • Nicht-Ausführung und Modifikation der Hausaufgabe kann vor der nächsten Therapiesitzung erkannt werden • objektivere Sicht auf die Durchführung
Wissenschaftler	• Gewinnung neuer Erkenntnisse • Verbesserung der vorherrschenden Behandlungen • große Datenmengen für Studien • differenzierter Blick auf das Krankheitsbild durch kontextuelle Informationen

Tabelle 1.1: Ziele der Projektteilnehmer

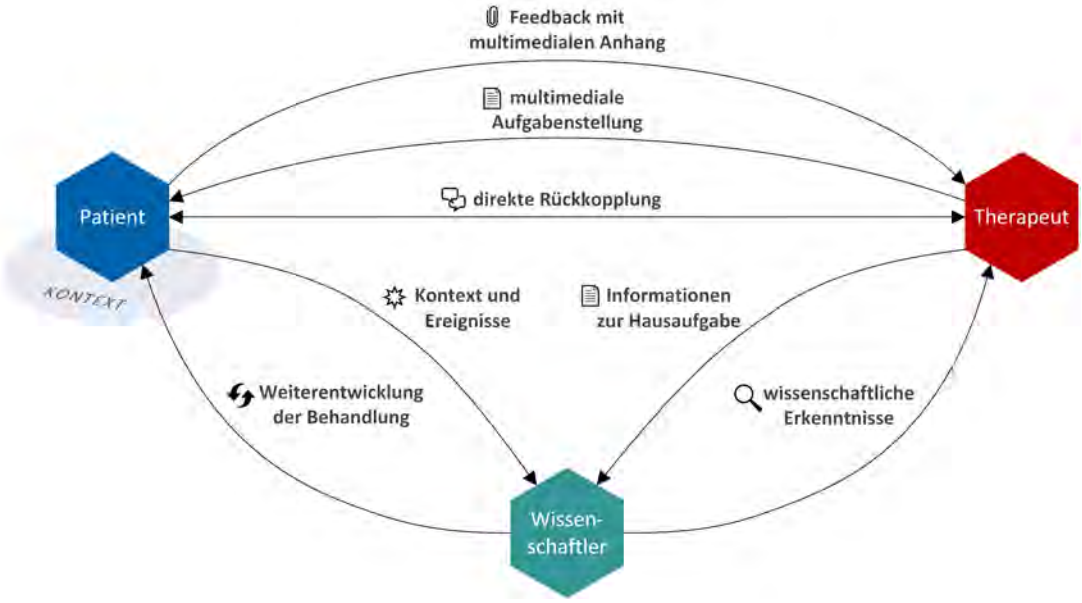


Abbildung 1.2: Austausch der Projektteilnehmer im schematischen Überblick

Innerhalb des Rahmenwerks stehen alle Projektteilnehmer miteinander in Beziehung. In Abbildung 1.2 wird dieser Austausch in einem schematischen Überblick dargestellt. Dabei wird aufgezeigt, dass Patient und Therapeut beim Austausch von Hausaufgabe und Feedback nur unidirektional in Beziehung stehen, durch die direkte Rückkopplung jedoch ein bidirektionaler Kanal entsteht. Weiterhin wird illustriert, dass der Patient stets innerhalb eines Kontextes agiert. Aufgrund dessen muss, für eine differenzierte Analyse des Therapieerfolgs der Hausaufgabe, diese Informationen ebenso erhoben werden. Die folgende Tabelle 1.2 beschreibt die bestehenden Missstände, die durch die Auswertung zusätzlicher Informationen beseitigt werden sollen und zeigt gleichzeitig die dabei entstehenden Chancen auf.

BESTEHENDE MISSSTÄNDE	NEUE CHANCEN
Erfahrungen des Patienten werden getrennt von der eigentlichen Definition der Hausaufgabe verwaltet und können nicht zu einem zeitlichen Ereignis vermerkt werden.	Durch die Benachrichtigung zur Durchführung der Hausaufgabe kann die Wahrscheinlichkeit, dass der Patient die Hausaufgabe erledigt, gesteigert werden.
Hausaufgaben können nur von dem Ort abhängig durchgeführt werden, an dem die benötigten Materialien vorhanden und gegebenenfalls abspielbar sind.	Konfigurationen einer Hausaufgaben können durch die Versionierung miteinander verglichen werden, um somit die effektivste Konfiguration für den Patienten herauszufinden.
Bei Anpassungen der Hausaufgaben müssen entweder neue Materialien erstellt oder die Anpassung mündlich erteilt werden.	Durch die Erkennung des Kontextes, während der Durchführung der Hausaufgabe, kann die Wirkung präziser gesteuert werden.
Modifikationen an der Hausaufgabe können durch den Therapeuten nur schwer nachvollzogen werden, da er diese nur durch den subjektiven Erfahrungsbericht des Patienten erhält.	Patienten können die Wissenschaft bei der Erforschung von Erkrankungen unterstützen und rückwirkend die Weiterentwicklung von Behandlungsmethoden durch neue Erkenntnisse erhalten.
Adjustierung des Schwierigkeitsgrades nur durch subjektive Erfahrungsberichte des Patienten möglich.	Der Therapeut erhält die Möglichkeit sich über einen Marktplatz vorzustellen und sein Behandlungsangebot zu offerieren.
Durch die nicht vorhandene Steuerbarkeit können Hausaufgaben innerhalb eines falschen Kontextes durchgeführt werden und dadurch die Wirksamkeit der Hausaufgabe schwächen, sowie gegebenenfalls unerwünschte Nebeneffekte hervorrufen.	Durch die Verwendung eines innovativen Dienstes und den entstehenden Mehrwert für den Patienten erhält der Therapeut die Möglichkeit zusätzliche Leistungen abzurechnen.
Die Wiederverwendung von Hausaufgaben auf Papier ist nur in erschwerter Form möglich.	Hausaufgaben sind konfigurierbarer und können dadurch feiner auf die Bedürfnisse des Patienten abgestimmt werden.
Der Zustand unmittelbar nach der Ausführung einer Hausaufgabe kann vom Patienten bei der Nachbesprechung nicht mehr eindeutig rekonstruiert werden.	Durch die Möglichkeit des Teilens von Hausaufgaben können Therapeuten auf bereits angewendete Behandlungen ihrer Kollegen zugreifen und diese auf einfache Weise selbst verwenden.
Direkte Vollzugsmeldungen sind für den Patienten mit Aufwand verbunden, obwohl diese das Potential zur Steigerung der Motivation besitzen.	Der Therapeut kann multimediales Feedback von seinem Patienten erhalten und jederzeit mit der Aufgabe verknüpft abrufen und einsehen.
Der Prozess der Hausaufgabenempfehlung ist für den Patienten nicht ausreichend transparent, um ihn zur regelmäßigen Durchführung zu motivieren.	Therapeut kann Hausaufgaben als Dokumentationswerkzeug verwenden, um mit dem Patienten ein gemeinsames Verständnis des Problembewusstseins zu erarbeiten.

Tabelle 1.2: Bestehende Missstände und neue Chancen

1.4 Aufbau der Arbeit

Die vorliegende Arbeit gliedert sich in acht aufeinander aufbauende Kapitel. In **Kapitel 2** wird in die grundlegenden Themen, die für das Verständnis dieser Arbeit benötigt werden, eingeführt. Dabei wird zuerst der Bereich Mobile Crowdsensing vorgestellt und die verschiedenen Formen abgegrenzt. Anschließend wird das Thema therapeutischen Hausaufgaben beschrieben und dabei die Ziele und Arten betrachtet, sowie auf Vorteile von Interventionen mit therapeutischen Hausaufgaben eingegangen.

Im Folgenden **Kapitel 3** werden verwandte Arbeiten recherchiert und ausgewählte Projekte exemplarisch vorgestellt. Dadurch können Erkenntnisse über Anforderungen, Problemstellungen und mögliche Architekturen unter der Verwendung von Mobile Crowdsensing gesammelt werden.

Das **Kapitel 4** widmet sich anschließend der Analyse von Anforderungen des Rahmenwerks dieser Arbeit. Dabei werden die verwendeten Begrifflichkeiten definiert und der aktuelle Prozess der Hausaufgabenempfehlung beschrieben. Mithilfe dieser Informationen werden Soll-Prozesse entwickelt, die die Grundlage der folgenden funktionalen und nicht-funktionalen Anforderungen bilden.

Aus den Erkenntnissen der Anforderungsanalyse wird in **Kapitel 5** eine Architektur für das Rahmenwerk entworfen. Dabei wird das Zusammenwirken der einzelnen Komponenten des Gesamtsystems beschrieben, sowie die Datenstruktur der Server-Anwendung definiert.

Anschließend werden in **Kapitel 6** ausgewählte Aspekte der Implementierung vorgestellt. Dazu werden zu Beginn die verwendeten Technologien eingeführt und der schematischer Aufbau skizziert. Darauf folgend werden die einzelnen Anwendungskomponenten beschrieben, sowieso exemplarische Ausschnitte der Implementierung dargelegt.

Um den aktuellen Stand des Rahmenwerks zu beurteilen, werden in **Kapitel 7** die funktionalen und nicht-funktionalen Anforderungen an das Rahmenwerk mit dem aktuellen Stand verglichen und gleichzeitig bewertet.

Zuletzt schließt das **Kapitel 8** mit einer Zusammenfassung und einem Ausblick auf zukünftige Erweiterungen für das Rahmenwerk diese Arbeit ab.

2

Grundlagen

Dieses Kapitel erläutert grundlegende Begriffe dieser Arbeit, um eine einheitliche Definition zu schaffen. Kapitel 2.1 geht dabei auf das Thema Mobile Crowdsensing ein und versucht die resultierenden Vorteile und die verschiedenen Formen herauszuarbeiten. Kapitel 2.2 führt im Anschluss in die Intervention mit therapeutischen Hausaufgaben ein und stellt die sechs Phasen des Vergabe- und Besprechungsprozesses vor.

2.1 Mobile Crowdsensing

Der Überbegriff Mobile Crowdsensing definiert, basierend auf Sensoren von mobilen Endgeräten wie Smartphone oder Smartwatch, ein neues Paradigma für das großflächige Erfassen und Verarbeiten von Daten. Eine Kamera kann als Sensor für Video- und Ton-Aufnahmen, ein Mikrofon als akustischer Sensor und ein GPS-Empfänger kann für Standort-Informationen verwendet werden. Aus der Kombination von weiteren Sensoren, wie Gyroskop, Beschleunigungssensor und Näherungssensor, können nützliche kontextuelle Informationen berechnet werden. Zusätzlich können weitere Sensoren, beispielsweise über Bluetooth oder WiFi, mit dem Smartphone kommunizieren und ausgewertet werden [24]. Diese lokalen Informationen einzelner Personen werden erhoben und im Anschluss verschiedenen - aufgrund der Verfügbarkeit und der Datenmenge oftmals cloudbasierten - Diensten zur Verfügung gestellt, welche die Messwerte zusammenfassen, um aus vielen lokalen Informationen ein größeres Gesamtbild zu berechnen. Durch die Mobilität der Messgeräte und die Beteiligung des normalen Bürgers, kann auf eine hohe Anzahl Messwerten von ständig wechselnden

Standorten zurückgegriffen und auf eine eigene statische Sensorinfrastruktur verzichtet werden [25]. Die Aggregation vieler unscharfer lokaler Messungen, zurückzuführen auf die unterschiedliche Datenqualität von eingebauten Sensoren in mobilen Endgeräten, ermöglicht dabei die Glättung von Ausreißern[1] und die Berechnung aussagekräftiger Ergebnisse. Abhängig von der Beteiligung der Benutzer, kann Mobile Crowdsensing in zwei Kategorien aufgeteilt werden.

Opportunistic Sensing

Die Messung erfolgt **unbewusst** im Hintergrund und ohne aktives Eingreifen durch den Benutzer. Gleichermaßen werden die gesammelten Daten automatisch, in manchen Fällen ohne das Wissen des Benutzers, im Hintergrund geteilt [26].

Participatory Sensing

Der Teilnehmer muss sich **bewusst** für die Benutzung und Teilnahme der Anwendung entscheiden und kann teilweise bestimmen, welche Sensordaten er mit der Anwendung teilen möchte [26]. Eine typische Architektur für Participatory Sensing Anwendungen wird in Abbildung 2.1 dargestellt.

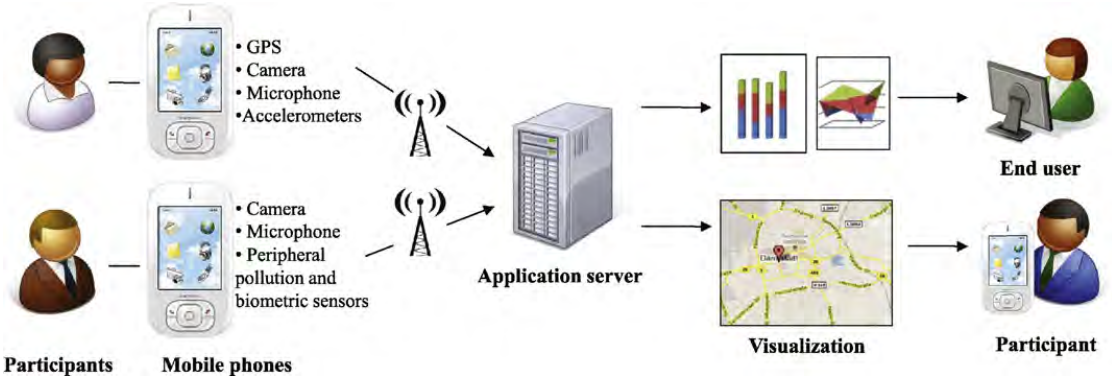


Abbildung 2.1: Architektur einer typischen Participatory Sensing Anwendung [24]

[1] Begriff aus der Statistik: Messwert ist außerhalb des erwarteten Streuungsbereichs.

2.2 Therapeutische Hausaufgaben in der Psychotherapie

In der Psychotherapie gehören therapeutische Hausaufgaben längst zum Standardrepertoire und gelten unter Psychotherapeuten mittlerweile als Methode der Zukunft [19]. Therapeutische Hausaufgaben sind, entweder vom Therapeuten vorgeschlagene oder gemeinsam entwickelte, Aufgaben, die zur Unterstützung der Therapie und Festigung der besprochenen Therapieziele eingesetzt werden. Der Patient soll die entwickelten Aufgaben zwischen den Therapiesitzungen ausführen, um gewünschte Verhaltensweisen des Therapieziels anzueignen und diese auf den eigenen Alltag zu übertragen. Dabei ist die Stabilität der neugewonnen Fertigkeiten und Verhaltensweisen stark von der wiederholten Ausführung in unterschiedlichen Kontexten abhängig [19].

Therapeutische Hausaufgaben können in zwei Kategorien aufgeteilt werden. Die Kategorie der **Kognitiven Aufgaben** enthält dabei Aufgaben, die sich um die gedankliche und emotionale Empfindung des Patienten kümmert, wie beispielsweise das Protokollieren von negativen Ereignissen, die Reflexion von bestimmten Themen oder die Psychoedukation [19]. Die zweite Kategorie der **Verhaltensbezogenen Aufgaben** versucht im Gegensatz dazu das Verhalten der Patienten durch aktive Aufgaben wie Konfrontationsaufgaben, Verhaltensänderungen durch interpersonelle Aufgaben oder kreativen Aufgaben zu beeinflussen [19].

Der Therapeut kann durch die Vergabe von therapeutischen Hausaufgaben von diversen positiven Effekten auf den Patienten profitieren. Schon zu Beginn der Therapie können therapeutische Hausaufgaben als Werkzeug verwendet werden. Der Therapeut kann die Hausaufgaben als Dokumentationswerkzeug für die Diagnostik, aber auch für das Problembewusstsein auf Seite des Patienten verwenden. Damit hat der Therapeut die Möglichkeit, gemeinsam mit dem Patienten einen Problemzustand zu definieren und diesen mit aufbauenden Hausaufgaben zu verbessern. Die für den Patienten erzeugte Transparenz des Prozesses kann somit die Adhärenz erhöhen und den Therapieprozess vereinfachen, indem eine engere Bindung zwischen Therapeut und Patient entsteht. Weiterhin können therapeutische Hausaufgaben folglich, wenn diese innerhalb der Intervention häufig verwendet werden, den Patienten zu dem wünschenswerten

Denkprozess anregen, dass die positiven Auswirkung auf den Therapieverlauf und Verbesserungen des eigenen Zustandes in Ursache mit dem eigenen Handeln und Wirken stehen. Dieser Effekt hat das Potenzial die Therapie des Patient deutlich zu verkürzen, da er aus einer passiven Haltung in eine aktive Rolle wechselt. Durch Aktivierung des Patienten mittels *Gamification* [27], also spielerisches Herantasten an Probleme und positive Bestätigung nach erfolgreicher Durchführung der Hausaufgabe, kann der Patient außerdem für weitere Aufgaben und die Therapie im Allgemeinen motiviert werden.

2.2.1 Vergabe- und Besprechungsprozess

Der Vergabe- und Besprechungsprozess von therapeutischen Hausaufgaben kann nach Scheel et al. [28] in sechs Schritte (Abbildung 2.2) gegliedert werden. Um eine Hausaufgabe zu verbessern und an den Patienten anzupassen, kann der komplette Prozess solange iterieren, bis letztendlich das Therapieziel der Hausaufgabe erreicht ist.

Phase 1: Entwicklung der Aufgabe

Aufgaben werden vom Therapeuten vor der Therapiesitzung, gegebenenfalls durch Zuhilfenahme von Fachliteratur, geplant, gemeinsam mit dem Patienten während der Sitzung entwickelt oder als Kombination beider Fälle vom Therapeuten vorgeschlagen und mit dem Patienten während der Sitzung verfeinert. Aufgaben sollen dabei auf die Fähigkeiten des Patienten aufbauen und gleichzeitig das Therapieziel immer im Auge behalten.

Phase 2: Vergabe der Aufgabe

In Phase 2 vergibt der Therapeut die Hausaufgabe offiziell an den Patienten. Dabei sollte der Therapeut die Hausaufgabe mit dem Patienten besprechen und ihn über die folgenden Übungen und Schwierigkeiten, sowie über die Bedeutung der Aufgabe für die

Therapie, informieren. Wichtig ist hierbei, dass der Therapeut den Grundgedanken der Aufgabe erklärt und diesen an die Problemvorstellung des Patienten anpasst.

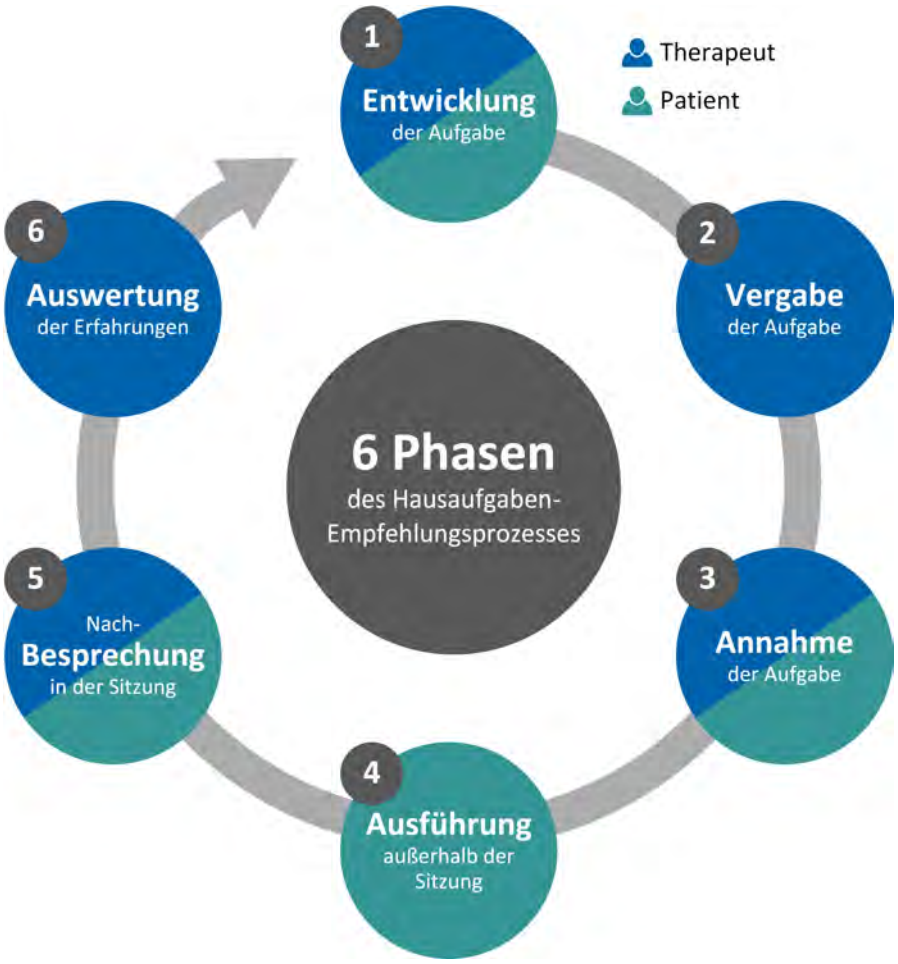


Abbildung 2.2: Sechs Phasen des Hausaufgaben-Empfehlungsprozesses nach Scheel et al. [28]

Phase 3: Annahme der Aufgabe

In der Annahme-Phase soll der Patient versuchen sich mit der Aufgabe und deren Grundgedanken auseinanderzusetzen und dabei dem Therapeuten mitteilen, wie wahrscheinlich eine Erledigung der Hausaufgabe ist und wie der gefühlte Schwierigkeitsgrad der Aufgabe ist. Die Auseinandersetzung mit der Aufgabe soll

dem Therapeuten aufzeigen, ob der Patient die Aufgabe akzeptieren und ausführen, sie komplett ablehnen oder modifizieren wird. Der Therapeut sollte entsprechend der Reaktion des Patienten agieren und auf die Einschätzung des Patienten gegebenenfalls mit Änderungen eingehen.

Phase 4: Ausführung außerhalb der Sitzung

Der Patient führt die Hausaufgabe außerhalb der Sitzung aus und unterliegt äußeren Störfaktoren, die ihn bei der Erledigung seiner Aufgaben abhalten können. Der Therapeut hat während dieser Phase keinen Einfluss auf die Ausführungen.

Phase 5: Nachbesprechung in der Sitzung

In der auf die Ausführung folgenden Sitzung befragt der Therapeut den Patient, ob und wie er die Aufgabe ausgeführt hat. Falls die Aufgabe ausgeführt wurde, so werden die Erlebnisse und Erkenntnisse des Patienten betrachtet. Hat der Patient die Aufgabe jedoch nicht oder modifiziert ausgeführt, so werden die Gründe hierfür diskutiert. In jedem Fall ist eine Nachbesprechung der Hausaufgabe entscheidend, da der Patient ansonsten den Eindruck bekommen könnte, dass die Aufgabe für die Therapie nicht wichtig ist.

Phase 6: Auswertung der Erfahrungen

Zuletzt wird der Therapeut die Erfahrungen des Patienten im Hinblick auf die Therapieziele auswerten. Falls die Aufgabe nicht den gewünschten Erfolg erbracht hat, so wird das Feedback des Patienten die Grundlage einer neuen Aufgabeniteration oder komplett neuen Hausaufgabe sein.

3

Verwandte Arbeiten

In diesem Kapitel werden verwandte Arbeiten gesucht, um Erkenntnisse über etwaige Anforderungen, Problemstellungen und Architekturen zu sammeln. Aufgrund der neuen und vielfältigen Möglichkeiten, die durch die IT-Unterstützung von eHealth-Diensten entstehen, existieren bereits diverse Systeme und Arbeiten unter Verwendung von Mobile Crowdsensing [29, 30, 31, 32, 33, 34, 35], deren Erkenntnisse für diese Arbeit von großen Wert sein können. Die folgenden Abschnitte stellen dabei exemplarisch einige dieser Systeme vor. Dadurch soll die Sensibilität für Probleme und Fallstricke, die bei der Konzeptionierung bedacht werden müssen, angeregt und bereits gewonnene Erkenntnisse auf die Problemstellung dieser Arbeit übertragen werden.

3.1 TrackYourTinnitus

Tinnitus ist eine Erkrankung bei der Töne und Geräusche wahrgenommen werden, die nicht in unmittelbaren Zusammenhang mit einem akustischem Signal in der Umgebung stehen. Obwohl diese Erkrankung stark verbreitet ist, wurde noch keine hinreichende Therapie entwickelt. Dies könnte aus der Heterogenität des Tinnitus resultieren, da Tinnitus eine sehr subjektives Erfahrung ist und hauptsächlich durch Erfragung des Betroffenen gemessen werden kann. Aufgrund dessen existieren sehr viele Formen des Tinnitus, welche gleichzeitig unterschiedliche Ausprägungen besitzen und unterschiedlich Reaktionen auf Behandlungen hervorrufen. Für den Betroffenen ist es zusätzlich schwierig eine zuverlässige Aussage über die Lautstärke und die Belastung durch den Tinnitus zu erteilen, da die Wahrnehmung über den Tag schwankt und vom Kontext der Umgebung abhängig ist.

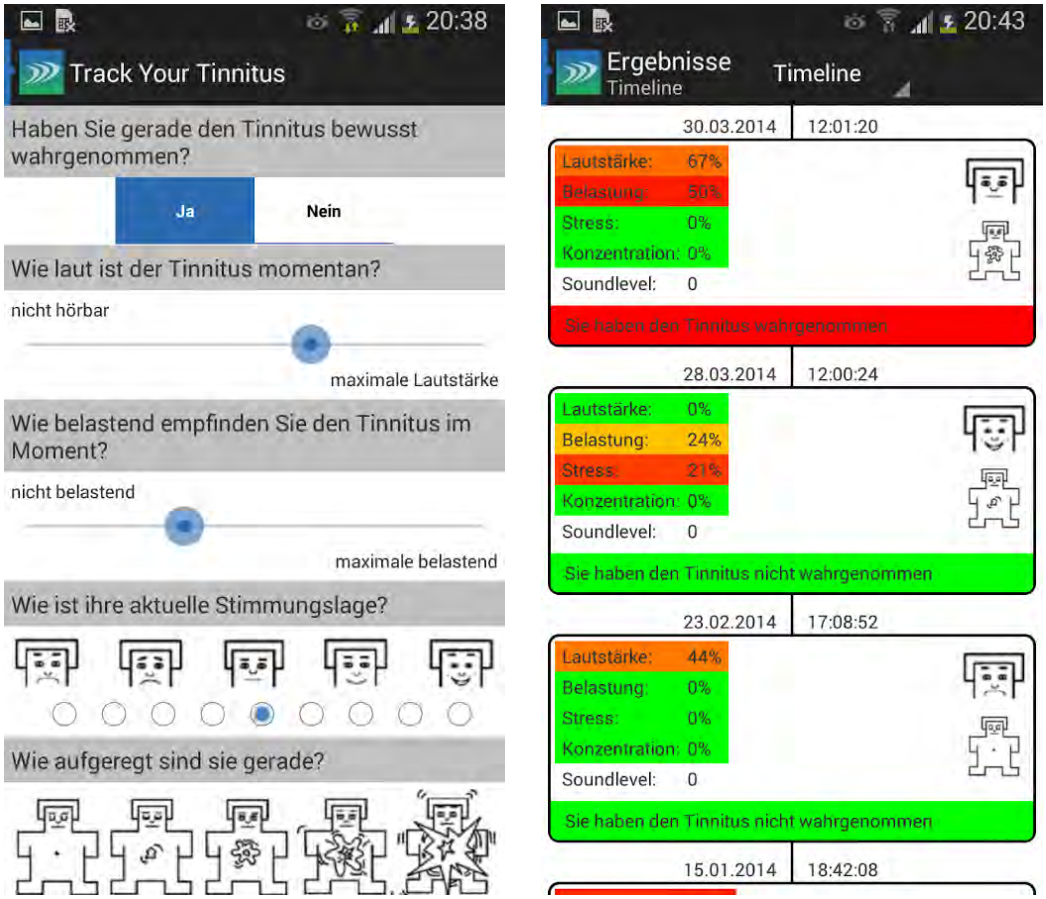


Abbildung 3.1: Ausschnitt aus der Android-App zur *TrackYourTinnitus*-Plattform [36]

Im Kontext eines großen Datenbank-Projekts, zusammengesetzt aus multidisziplinären Forschern-Teams, wurde daraufhin die Mobile Crowdsensing-Plattform *TrackYourTinnitus* als Kooperationsprojekt der *Tinnitus Research Initiative* und des *Instituts für Datenbanken und Informationssysteme* der Universität Ulm entwickelt, um neue Erkenntnisse über die Ursache des Tinnitus zu erlangen. Das Smartphone wird dabei Erhebungsinstrument von Patientendaten, welche aus Befragungen des Betroffenen und Sensordaten durch das Smartphone bestehen. Dadurch wird die individuelle Wahrnehmung des Tinnitus über einen eigens entwickelten Fragebogen an mehreren und zufälligen Zeitpunkten erfragt. Zusätzlich wird während des Ausfüllens des Fragebogens der Geräuschpegel gemessen, um gleichzeitig erweiterte Informationen

über die Situation zum Zeitpunkt des Ausfüllens zu erlangen. Die Zeitpunkte zur Benachrichtigung des Patienten liegen dabei innerhalb eines vom Patienten bestimmten Intervall, wobei die Zeitpunkte einen Mindestabstand zueinander einhalten müssen. Über die Erhebung an unterschiedlichen Zeitpunkten ist die Messung der Fluktuationen in der Wahrnehmung des Patienten möglich und kann gleichzeitig innerhalb einer realen Umgebung und der täglichen Routine des Patienten gemessen werden.

Die gesammelten Informationen werden im Anschluss an eine zentrale Plattform gesendet, die von Forscher-Teams für die Verwaltung der Patienten und den Datenexport verwendet wird. Mithilfe der gesammelten Daten konnten bereits einige neue Erkenntnisse über den Tinnitus gesammelt werden [37, 38, 39, 40, 41, 42, 43, 44]. So konnten beispielsweise Hinweise gefunden werden, dass das Stressniveau des Patienten eine direkte Auswirkung auf die Lautstärke und Belastung des Tinnitus haben.

3.2 myKind

Die Mobile Crowdsensing-Plattform *myKind* [45, 46] ist ein Kooperationsprojekt der Gruppe für *Klinischen Psychologie und Klinischen Neuropsychologie* der Universität Konstanz und des *Instituts für Datenbanken und Informationssysteme* der Universität Ulm. Das Projekt richtet sich an werdende Mütter und dokumentiert, mithilfe von Fragebögen die Belastungen während der Schwangerschaft, die durch psychosoziale Einflüsse ausgelöst werden. Diese Belastungen können die psychische Entwicklung des ungeborenen Kindes beeinflussen und selbst nach der Geburt die Entwicklung des Kindes beeinträchtigen.

Dabei soll der Konstanzer Index (KINDEX), ein strukturiertes Interview zur Ermittlung der Risiken für Kind und Mutter, welches ursprünglich als Instrument in gynäkologischen Praxen und Frauenkliniken zum Einsatz kam, durch die schwangere Frau selbst ausgefüllt und automatisiert über eine App ausgewertet werden. Somit können werdende Mütter selbst die Belastungen und Veränderungen im Verlauf ihrer Schwangerschaft erfassen und verfolgen. Zusätzlich werden den Teilnehmern, basierend auf Antworten der eingegebenen Fragebögen, Ratschläge zur Vorbeugung von Risikofaktoren gegeben.

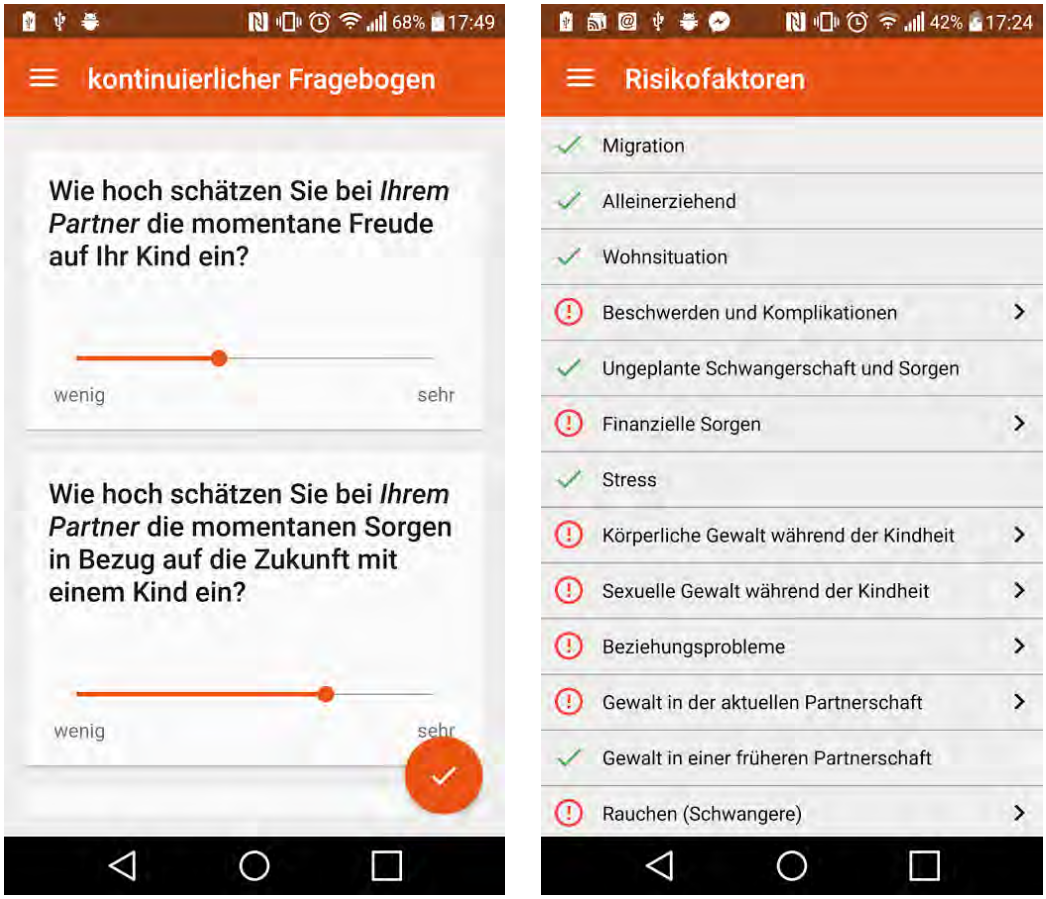


Abbildung 3.2: Ausschnitt aus der Android-App zu *myKind* [47]

3.3 QuestionSys

Wie bereits in den vorangegangenen Abschnitten dargelegt, stellen Fragebögen eine der wichtigsten Instrumente bei psychologische Studien dar. Der Großteil aller Studien im Fachbereich Psychologie verwendet speziell erstellte Fragebögen, deren Aussagekraft im Voraus überprüft und bestätigt wurde. Da die Fragebögen üblicherweise in Papierform vorliegen, entsteht dabei ein massiver Arbeitsaufwand während allen Phasen der Studie. Um diesen Arbeitsaufwand zu verringern, wurde durch das Institut für Datenbanken und Informationssysteme der Universität Ulm das generisches und prozessorientiertes Werkzeug *QuestionSys* [48] entwickelt, um Fragebögen zu Erfassen, Verteilen, Evaluieren, Archivieren und Versionieren. Gleichzeitig werden

über einen Generator Benutzerschnittstellen erstellt, über die der erstellte Prozess auf unterschiedlichen Endgeräten ausgeführt, respektive der Fragebogen ausgefüllt, werden kann.

4

Anforderungsanalyse

In diesem Kapitel werden die Anforderungen an das Rahmenwerk analysiert und definiert. Zuerst werden in Kapitel 4.1 Begriffe definiert, die für das weitere Verständnis der Arbeit notwendig sind und einen homogenen Wissensstand erzeugen. Anschließend wird in Kapitel 4.2 der Ist-Stand des aktuellen Prozesses skizziert und mögliche Schwachstellen herausgearbeitet. Darauf folgend wird in Kapitel 4.3 der zukünftige Soll-Stand des vorher skizzierten Prozesses für ein unterstützendes System definiert und Verbesserungsmöglichkeiten aufgezeigt. Zuletzt werden in Kapitel 4.4 die, aus dem Vergleich beider Zustände, resultierenden funktionalen und nicht-funktionalen Anforderungen beschrieben.

4.1 Fachwissen

Um ein einheitliches Verständnis der folgenden Kapitel zu gewährleisten, werden in diesem Abschnitt Begrifflichkeiten zum Thema therapeutische Hausaufgaben definiert und alphabetisch sortiert aufgelistet. Dabei werden zu jedem vorgestellten Begriff mögliche Synonyme und Varianten angegeben. Dieses Rahmenwerk wird für das Therapiemittel therapeutische Hausaufgaben in der Psychotherapie umgesetzt, soll aber gleichermaßen für andere Therapieformen mit Hausaufgaben als Therapiemittel verwendet werden. Die vorgestellten Begriffe enthalten deshalb gleichzeitig Beispiele aus der Therapieform Psychotherapie und Physiotherapie, um die Übertragbarkeit zu verdeutlichen.

Bezeichner:	**Aufgabe**
Beschreibung:	Eine Aufgabe ist die Kombination aus Aufgabenbeschreibung, Hilfsmittel, Qualität, Quantität, Selbstständigkeit und Zeitrahmen. Aufgaben sind nicht patientenbezogen und können in Repositorien geteilt werden.
Synonym:	Patienten-Aufgabe
Kann sein:	• Körperliche Anstrengung • geistige Anstrengung
Beispiel:	2x wöchentlich, so lange die Aufgabe benötigt wird, unter bewusster Kontrolle des linken Knies, unter Aufsicht einer Pflegeperson, die Hose im Stehen anziehen können, mit Anlehnen am Bettgitter

Bezeichner:	**Aufgabenbeschreibung**
Beschreibung:	Für eine Aufgabe wird immer eine Aufgabenstellung, bzw. ein Aufgabenziel, definiert.
Synonym:	Beschreibung
Kann sein:	Text
Beispiel:	• Die Hose im Stehen anziehen können • Gehen können und selbstständig in eine Liste eintragen • Fremde Personen nach einem komplizierten Weg fragen und Rückfragen stellen • dysfunktionalen Gedanken „Ich bin ein Versager!“ im ABC-Modell umstrukturieren • Entspannungsübung (PMR)

Bezeichner:	**Behandlungsaufgabe**
Beschreibung:	Eine Behandlungsaufgabe ist eine personalisierte Aufgabe und ist immer auf einen einzelnen Patienten zugeschnitten. Dafür wird eine Aufgabe mit einem individualisierten Kontext und optional einem Feedback zur Sicherung der Behandlungsqualität angereichert.
Synonym:	Hausaufgabe

Kann sein:
- Körperliche Anstrengung
- geistige Anstrengung

Beispiel: 2x wöchentlich, so lange die Aufgabe benötigt wird, unter bewusster Kontrolle des linken Knies, unter Aufsicht einer Pflegeperson, die Hose im Stehen anziehen können, mit Anlehnen am Bettgitter; „daheim – abends – im Gang – WLAN“; „Konnte die Übung selbstständig und erfolgreich ausgeführt werden?“

Bezeichner: **Behandlungsvermerk**

Beschreibung: Während der Behandlung durch einen Therapeuten, kann zu einem Patienten ein Behandlungsvermerk hinzugefügt werden, welcher eine textuelle Beschreibung des aktuellen Patienten-Status enthält. Dieser Vermerk wird entweder direkt in Beziehung zu einer Behandlungsaufgabe, wie beispielsweise nach der Beendigung einer Behandlungsaufgabe oder generell zu dem Patienten erstellt. Dadurch kann der Therapeut den Verlauf seiner Behandlung besser nachvollziehen und bei unerwünschten Nebeneffekten die Behandlungsaufgaben nachbessern. Weiterhin werden Behandlungsvermerke in Patienten-Reports angezeigt.

Synonym: Vermerk

Kann sein: Text

Beispiel:
- Patient kann die Hose bis zum Knie anziehen.
- Aufgabe wurde vom Patienten nicht angenommen.

Bezeichner: **Hilfsmittel**

Beschreibung: (Immaterieller) Gegenstand der zur erfolgreichen Absolvierung der Aufgabestellung benötigt wird.

Kann sein:
- Musik
- Gegenstand
- Programm

Beispiel:
- Mit Anlehnen am Bettgitter
- Handlauf

Bezeichner:	**Feedback**
Beschreibung:	Optionale oder verpflichtende Abfrage, die dem Patienten bei Absolvierung der Aufgabe gestellt wird. Eine Feedback-Vorlage kann aus Fragen und Medien-Elementen zum Upload zusammengestellt werden.
Synonym:	• Feedback-Fragebogen • Feedback-Vorlage
Kann sein:	• Ja/Nein-Fragebogen • individueller oder standardisierter Fragebogen • Medien-Upload
Beispiel:	• Haben Sie die Übung vollständig absolviert? • Haben Sie die Übung nur mit Mühe absolvieren können? • Als wie schwer empfanden Sie diese Übung? • Wie würden sie den Erfolg dieser Übung einschätzen?

Bezeichner:	**Kontext**
Beschreibung:	Der Kontext beschreibt die Umgebungsbedingungen, die für eine erfolgreiche Aufgabenausführung benötigt werden und kann als endogener[1] und exogener[2] Kontext spezifiziert werden. Der endogene Kontext ist die Kombination aus den technischen Anforderungen und den Umgebungsbedingungen. Damit kann gewährleistet werden, dass die Aufgabe fehlerfrei und in einer wirksamen und störungsfreien Umgebung, ausgeführt werden kann. Der exogene Kontext kann eine außerhalb spezifizierte und erhobene Quelle sein, deren Wert ausschlaggebend für die Ausführungsumgebung der Aufgabe ist.
Synonym:	Kombinierte Durchführungsparameter
Kann sein:	• endogener Kontext • exogener Kontext
Beispiel:	• daheim, abends, im Gang, WLAN • in der Mittagspause, ruhig, Mikrofon

[1] innerhalb des Systems spezifiziert
[2] Beeinflussung des Systems von extern spezifizierten Bedingungen

Bezeichner:	**Qualität**
Beschreibung:	Zeitliche, kognitive oder physische Belastung während der Ausführung der Aufgabe.
Kann sein:	• Anstrengung • Tonus • Ass. Reaktion • Double Task • Tempo
Beispiel:	• unter bewusster Kontrolle des linken Knies • erhöhter Zeitaufwand ist toleriert

Bezeichner:	**Quantität**
Beschreibung:	Häufigkeit, Dauer oder Anzahl der Aufgabe-Einheiten.
Synonym:	Kombinierte Durchführungsparameter
Kann sein:	• wie weit? • wie oft? • wie lang?
Beispiel:	• 4 x 10 Meter • 30 Minuten • so lange die Aufgabe benötigt wird

Bezeichner:	**Selbstständigkeit**
Beschreibung:	Definiert die Anzahl der unterstützenden Personen für die erfolgreiche Absolvierung der Aufgabe..
Kann sein:	• selbstständig • mit Aufsicht • geringe Hilfe • maximale Hilfe
Beispiel:	• unter Aufsicht einer Pflegeperson • mit einer Person des anderen Geschlechts • selbstständig

Bezeichner:	**Technische Anforderungen**
Beschreibung:	Die Aufgabenstellung wird dem Patienten digital zur Verfügung gestellt und benötigt unter Umständen bestimmte Ressourcen für den Abruf mit dem mobilen Endgerät. Außerdem müssen zur Überprüfung der Umgebungsbedingungen eventuell Sensoren reserviert werden, um beispielsweise den Umgebungsgeräuschpegel zu bestimmen.
Synonym:	Technische Durchführungsparameter
Kann sein:	• Konnektivität • Sensorik • Bildschirmgröße/Auflösung
Beispiel:	• WLAN, GPS, Mikrofon, großer Bildschirm • Internetempfang, Lichtempfindlichkeit-Sensor

Bezeichner:	**Umgebungsbedingungen**
Beschreibung:	Die Umgebungsbedingungen beschreiben die Situation, in welcher eine Aufgabe (ausschließlich) absolviert werden kann. Dabei sollen ortsbezogene Angaben, zeitbezogene Angaben und Sinneswahrnehmungen (ruhig/laut, hell/dunkel, still/in Bewegung) angegeben werden, um die korrekten Durchführungsparameter zu erlangen.
Synonym:	Ortsbezogener Durchführungsparameter
Kann sein:	• Wo? • Wann? • welche Umgebung? • Situation • Geräuschpegel • Zeitpunkt • Intervall
Beispiel:	• morgens nach dem Frühstück • Montag 18 Uhr & Mittwoch 16 Uhr, auf der Straße • zuhause, im Gang, hell • nicht ortsgebunden, in der Mittagspause, ruhig • bei der Arbeit, morgens • daheim, nach dem Waschen • wenn der dysfunktionale Gedanke „Ich bin ein Versager!“ präsent ist

Bezeichner:	**Zeitrahmen**
Beschreibung:	Gibt die Anzahl der Wiederholungen bis zu einem gewissen Datum oder einen zeitlichen Intervall / Rhythmus zur Ausführung der Aufgabenstellung an.
Synonym:	Ausführungsintervall
Kann sein:	• Intervall • Rhythmus • Wiederholungen
Beispiel:	• 15.09.2016 Bis 31.09.2016 • 2x wöchentlich • 5x bis zum 11.04.2016

4.2 Ist-Stand

Eine umfassende Unterstützung des Empfehlungs- und Überwachungsprozesses für therapeutische Hausaufgaben benötigt im Vorfeld eine detaillierte Analyse. Zu Beginn der Analyse wird der bisher implizit definierte Prozess studiert, in einzelne Aktivitäten zerlegt und anschließend strukturiert. Mithilfe der gewonnenen Informationen können nun zwei, aufgrund der Komplexität und Thematik aufgeteilte, grafische Prozessmodelle als Flussdiagramm erstellt werden. Das erste Prozessmodell (Abbildung 4.1) beschreibt die Erstellung und Erteilung von Hausaufgaben während der Therapiesitzung. Darauf folgend bildet das zweite Modell (Abbildung 4.2) das Monitoring von Hausaufgaben ab, welches gleichzeitig die eigentliche Ausführung durch den Patienten enthält. Beide Prozessmodelle bilden fortan die Grundlage für weitere Vergleiche und Analysen.

Notation der verwendeten Flussdiagramme

Flussdiagramme werden für die grafische Darstellung von Prozessen verwendet und sind aufgrund der einfachen Notation eine sehr beliebte Darstellungsform. Die folgende Tabelle definiert die Notation für die folgenden Flussdiagramme dieser Arbeit.

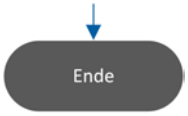
SYMBOL	BEZEICHNUNG	BEDEUTUNG
	Startpunkt	Startpunkt eines Prozesses
	Endpunkt	Endpunkt eines Prozesses
	Aktivität	Eine Aktivität bildet eine einzelne Tätigkeit innerhalb des Prozesses ab.
	Vordefinierter Prozess	Bindet einen vordefinierten Prozess in den Kontrollfluss des aktuellen Prozesses ein.
	Verzweigung	Bei einer Verzweigung wird eine ausgehende Kante für den weiteren Kontrollfluss gewählt. Entscheidend hierfür ist der annotierte Wert an einer Kante.
	Manuelle Eingabe	Symbolisiert die Aktivität einer manuellen Eingabe einer Person (z.B. über Tastatur).
	Manuelle Verarbeitung	Aktivität musst von einer Person (oftmals ohne Unterstützung des Systems) ausgeführt werden.
	Allgemeine Daten *(schreibend)*	Symbolisiert eine Aktivität, die (nicht näher spezifizierte) Daten erhält und speichert.

Tabelle 4.1: Notation von Flussdiagrammen - Teil 1

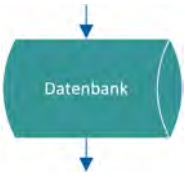
SYMBOL	BEZEICHNUNG	BEDEUTUNG
	Allgemeine Daten *(lesend)*	(Nicht näher spezifizierte) Daten werden von der assoziierten Aktivität gelesen.
	Datenbankzugriff *(schreibend)*	Symbolisiert eine Aktivität, die Daten in eine Datenbank schreibt.
	Datenbankzugriff *(lesend)*	Daten werden von der assoziierten Aktivität aus einer Datenbank gelesen.
	Physisches Dokument *(schreibend)*	Symbolisiert eine Aktivität, die ein physisches Dokument (z.B.: ein Merkblatt) erstellt oder bearbeitet.
	Physisches Dokument *(lesend)*	Informationen eines physischen Dokuments werden von der assoziierten Aktivität benötigt.

Tabelle 4.2: Notation von Flussdiagrammen - Teil 2

4.2.1 Ist-Prozess der Hausaufgabenerteilung

Die Empfehlung und Erteilung von Hausaufgaben findet im Regelfall immer innerhalb einer Therapiesitzung statt und kann somit ausführlich besprochen werden. Dieser Prozess ist dabei ein Subprozess des **Hausaufgaben-Empfehlungsprozesses** nach Scheel et al. [28] (Abbildung 2.2) und beinhaltet die Entwicklung der Aufgabe (*Phase 1*), Vergabe der Aufgabe (*Phase 2*) und Annahme der Aufgabe (*Phase 3*).

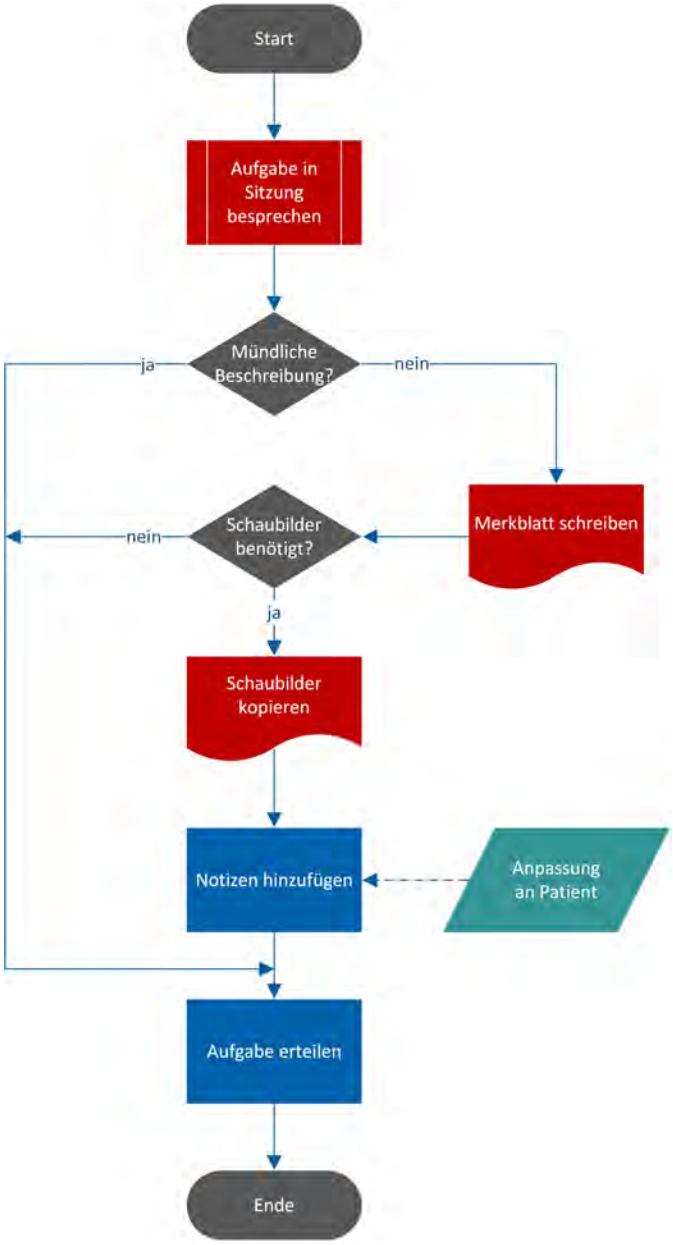


Abbildung 4.1: Ist-Prozess der Hausaufgabenerteilung

Zu Beginn des Prozesses der Hausaufgabenerteilung steht der vordefinierte Prozess Aufgabe in Sitzung besprechen, welcher sowohl den Vorschlag zur Verwendung von therapeutischen Hausaufgaben innerhalb der Therapie, als auch

die Besprechung und Entwicklung möglicher Aufgaben beinhaltet. Dieser Prozess wurde abstrakt definiert und wird im Folgenden nicht detaillierter modelliert, da der Hausaufgaben-Empfehlungsprozess für jede Therapie individuell verläuft und die Abbildung aller möglichen Prozesse, wenn sie die Realität komplett abbilden sollen, zu viele Variationen erzeugt. Zusätzlich ist der detaillierte Ablauf vernachlässigbar, da die ersten beiden Phasen der Hausaufgabenempfehlung weiterhin ein individuelles, persönliches Gespräch enthalten und nicht digital unterstützt werden sollen.

Auf die Besprechung folgt die Entscheidung des Therapeuten, ob er dem Patienten die Hausaufgabe mündlich beschreiben oder Veranschaulichungsmaterial als Erinnerungsstütze mitgeben möchte. Besteht der Therapeut auf eine fehlerfreie Ausführung, so wird er in den meisten Fällen dem Patienten Veranschaulichungsmaterial aushändigen und ein Merkblatt schreiben. Falls der Therapeut weitere Schaubilder oder Artikel aushändigen möchte, so kopiert er diese und fügt dabei, falls der Patient zur Aufgabe abweichende Aufgabenparameter benötigt, die individuellen Anpassungen als Notiz hinzu. Anschließend folgt die Aktivität **Aufgabe erteilen**, in der der Therapeut die besprochene Hausaufgabe offiziell erteilt. Diese Aktivität erfolgt ebenfalls auf die Entscheidung, ob für eine fehlerfreie Ausführung der Hausaufgabe eine mündliche Beschreibung ausreicht und schließt damit den Prozess der Hausaufgabenerteilung ab.

4.2.2 Ist-Prozess des Hausaufgabenmonitoring

Ein häufiges Problem bei der Intervention mit therapeutischen Hausaufgaben ist die Modifizierung oder Nicht-Ausführung durch den Patienten. Modifizierte Hausaufgaben können eine verminderte Auswirkung auf den Patient bedeuten und im schlimmsten Fall kann es zu unerwünschten Nebeneffekten führen. Führt der Patient die Hausaufgabe jedoch erst gar nicht aus, so entfällt nicht nur die gewünschte positive Wirkung, die durch das Einüben erreicht wird, es verzögert durch die resultierende Nachbesprechung, bei der die Ursache besprochen wird, auch den kompletten Therapieverlauf. Übertragen auf den Hausaufgaben-Empfehlungsprozesses nach Scheel et al. [28] (Abbildung 2.2) ist der Prozess des Hausaufgabenmonitoring ein Subprozess, der die Ausführung

außerhalb der Sitzung (*Phase 4*), die Nachbesprechung in der Sitzung (*Phase 5*) und die Auswertung der Erfahrungen (*Phase 6*) modelliert.

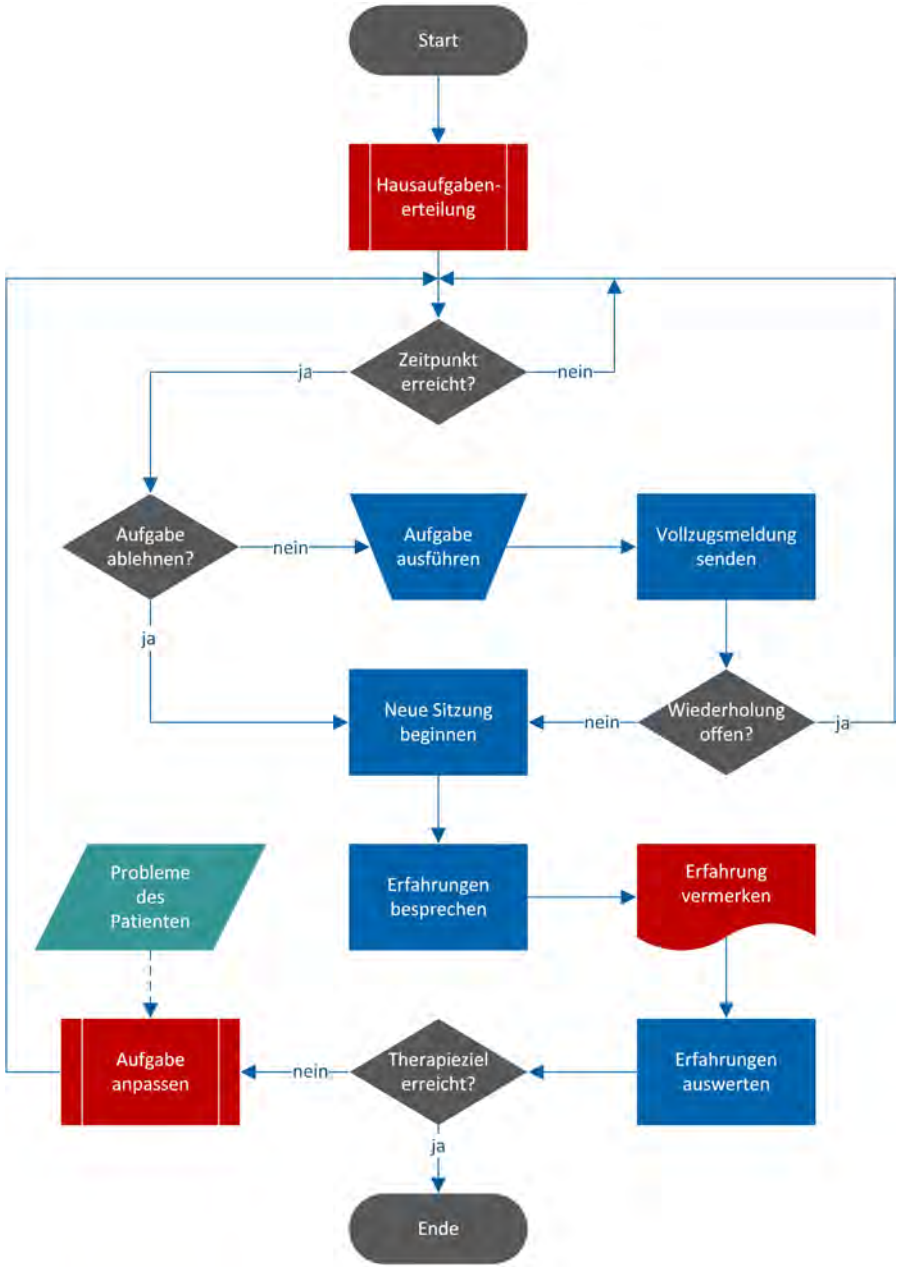


Abbildung 4.2: Ist-Prozess des Hausaufgabenmonitoring

Der Prozess beginnt mit dem in Kapitel 4.2.1 beschriebenen und vordefinierten Prozess der Hausaufgabenerteilung. Dabei wird eine Hausaufgabe besprochen, optional mit Merkzettel und Veranschaulichungsmaterial verdeutlicht und zuletzt wird die Aufgabe offiziell erteilt. Im Anschluss verlässt der Patient die Therapiesitzung und der Prozess wartet, bis das vereinbarte Datum zur Ausführung erreicht wird. Ist der Zeitpunkt erreicht, so muss der Patient entscheiden, ob er die Aufgabe durchführen möchte oder ablehnt. Entscheidet sich der Patient für die Ausführung, so kann er nach Beendigung eine Vollzugsmeldung an den Therapeuten senden. Wurde die Aufgabe mit Wiederholungen geplant, wartet der Patient nun erneut bis das vereinbarte Datum erreicht ist und führt die gleichen Schritte und Entscheidungen nochmals aus. Steht keine Wiederholung aus oder entscheidet sich der Patient die Aufgabe nicht auszuführen, so ist die nächste Aktivität die anschließende Therapiesitzung. Innerhalb der Sitzung werden die Erfahrungen und Gedanken besprochen, die der Patient während der Ausführung erlebt hat. Falls die Aufgabe nicht ausgeführt wurde, so wird in dieser Aktivität die Ursache für die Ablehnung erörtert. Das Ergebnis des Gesprächs, unabhängig ob der Patient die Aufgabe ausgeführt hat oder nicht, wird vom Therapeuten anschließend vermerkt, um die Erfahrungen im Anschluss auszuwerten. Der Therapeut berücksichtigt dabei, ob sich der Zustand des Patienten dem Therapieziel genähert hat und inwiefern Anpassungen an die Aufgabenstellung Verbesserungen bewirken könnten. Mithilfe der Auswertung entscheidet der Therapeut, ob das Therapieziel für diese Hausaufgabe erreicht wurde, respektive ob der Patient für weitere Hausaufgaben bereit ist. Ist er nicht bereit, indem der Patient sich beispielsweise der Ausführung von Hausaufgaben verweigert oder sich ihm die Möglichkeit zur Aufgabendurchführung nicht bietet, so kann der Therapeut den Prozess der Hausaufgabenempfehlung beenden. Erklärt der Patient sich jedoch für eine neue Iteration der Hausaufgabe bereit, da beispielsweise das Therapieziel für diese Aufgabe noch nicht erreicht wurde, so kann der Therapeut den Prozess **Aufgabe anpassen** anstoßen. Dabei fließen die Probleme des Patienten, die während der Ausführungsphase aufgetreten sind, in die Anpassung mit ein. Dieser Prozess ist dem der Hausaufgabenerteilung ähnlich, es kann jedoch auf eine ausführliche Besprechung verzichtet werden, da die Aufgabe dem Patienten bereits bekannt ist. Daraufhin kann eine neue Iteration begonnen werden.

4.2.3 Ansatzpunkte für Erweiterungen

Die in den letzten beiden Kapiteln vorgestellten Prozesse bilden vereinigt den Ist-Stand des Hausaufgaben-Empfehlungsprozesses. Dieser implizit ablaufende Prozess ist zusammengesetzt aus Aktivitäten, die mit modernen IT-Systemen den Prozessablauf zu jeder Zeit kontrollieren und die anfallenden Informationen strukturieren können. Gleichzeit ist eine Modellierung in einem höheren Detailgrad möglich, wodurch der gesamte Prozess transparenter gestaltet wird. Im Folgenden werden Ansatzpunkte herausgearbeitet, welche bei der Konzeption des IT-unterstützten Soll-Standes zu berücksichtigen sind.

Ist-Prozess der Hausaufgabenerteilung

Ansatzpunkt 1 Der Therapeut kann schon einmal eingesetzte Hausaufgaben leicht wiederverwenden und individualisieren. Gleichermaßen sollen Materialien für den Patienten einfach dupliziert werden können, anstatt diese für jeden Patient erneut zu erstellen.

Ansatzpunkt 2 Hausaufgaben können, falls sich Therapeuten in einem Institut zusammenschließen, miteinander geteilt werden, ohne dabei den Schutz von Patientendaten zu schwächen oder geistiges Eigentum zu verletzen.

Ansatzpunkt 3 Therapeuten können Hausaufgaben zu einem beliebigen Zeitpunkt und unabhängig von der Therapiesitzung erstellen, individualisieren und starten.

Ansatzpunkt 4 Der Patient kann Hausaufgaben vor Aufgabendurchführung auf einfachem Wege erhalten und alle Hausaufgaben innerhalb seiner Behandlung zentral organisieren. Somit hat er alle Hausaufgaben jederzeit auf seinem Smartphone und kann die Übung ortsunabhängig durchführen.

Ansatzpunkt 5 Hausaufgaben können durch multimediale Inhalte angereichert werden, um Ausführungsfehler zu minimieren und gleichzeitig eine zentrale Plattform für therapieunterstützende Medien, wie zum Beispiel Entspannungsmusik, zu bieten.

Ansatzpunkt 6 Der Therapeut kann vorher definierte Fragen zur Aufgabe unmittelbar nach der Aufgabenausführung vom Patienten beantworten lassen, die er vor der nächsten Therapiesitzung einsehen kann. Darüber soll eine multimediale Rückkopplung ermöglicht werden, sodass der Patient als Feedback zur Abgabe auch mediale Inhalte anhängen kann.

Ist-Prozess des Hausaufgabenmonitoring

Ansatzpunkt 7 Der Patient kann nach der Durchführung der Hausaufgabe dem Therapeuten auf einfachem Wege eine Vollzugsmeldung senden.

Ansatzpunkt 8 Der Patient kann die Überwachung des Durchführungszeitpunktes delegieren und bekommt eine Erinnerung vom System, falls eine Aufgabe ausgeführt werden soll.

Ansatzpunkt 9 Der Therapeut erfährt vor der nächsten Therapiesitzung, ob der Patient seine Aufgaben ausgeführt, modifiziert oder abgelehnt hat und kann sich damit auf die Nachbesprechung vorbereiten.

Ansatzpunkt 10 Die Erfahrungen des Patienten können innerhalb oder nach der Therapiesitzung zentral abgelegt und jederzeit vom Therapeuten eingesehen werden.

Ansatzpunkt 11 Die Auswertung von ausgeführten Hausaufgaben wird durch das System erleichtert, da es Abweichungen zur Aufgabenstellung und Auffälligkeiten in der Abgabe erkennt und hervorhebt.

Ansatzpunkt 12 Überwachung der Aufgabenausführung, sodass der Patient Aufgaben in der definierten Situation ausführt. Falls nicht, soll der Patient vor der Ausführung und der Therapeut dies bei der Nachbesprechung erkennen können.

4.3 Soll-Stand

Mit den Erkenntnissen aus der Modellierung der Ist-Prozesse in Kapitel 4.2 als Grundlage, insbesondere mit den Erweiterungsmöglichkeiten aus Kapitel 4.2.3, sollen nun die Soll-Prozesse der Hausaufgabenempfehlung konzeptioniert werden. Die Prozessunterstützung durch moderne Informationstechnologie erlaubt es hierbei Prozesse detaillierter zu modellieren, da der Benutzer Aktivitäten innerhalb eines deterministisch Systems ausführen kann und dadurch eine nachvollziehbare Ablaufstruktur vorgeben wird. Im Folgenden werden deshalb zwei ausführlich modellierte Soll-Prozesse vorgestellt, die die Erteilung von Hausaufgaben (Abbildung 4.3) vereinfachen und vor allem den Therapeuten beim Monitoring von Hausaufgaben (Abbildung 4.4) unterstützen sollen.

4.3.1 Soll-Prozess der Hausaufgabenerteilung

Der Soll-Prozess der Hausaufgabenerteilung (Abbildung 4.3) ist, genau wie der aktuelle Ist-Prozess, ein Subprozess des Hausaufgaben-Empfehlungsprozesses nach Scheel et al. [28] (Abbildung 2.2) und beinhaltet die Entwicklung der Aufgabe (*Phase 1*), Vergabe der Aufgabe (*Phase 2*) und Annahme der Aufgabe (*Phase 3*). Eine Besonderheit des Soll-Prozesses, im Vergleich zum Ist-Prozess, besteht jedoch darin, dass die Erstellung von Hausaufgaben unabhängig vom Zeitpunkt der Therapiesitzung erfolgen kann. Das bedeutet, dass der Therapeut nun in der Lage ist die Hausaufgabe vor der Sitzung vorbereiten, während der Sitzung zusammen mit dem Patient erstellen und modifizieren oder im Anschluss an die Sitzung nachbereiten kann.

Zu Beginn des Soll-Prozesses steht in der Regel ebenfalls ein persönliches Gespräch zwischen Therapeut und Patient innerhalb der Therapiesitzung, bei dem die Aufgabe vorgestellt und besprochen wird. Da der Therapeut jedoch den Zeitpunkt der Hausaufgaben-Erstellung und -Erteilung flexibel bestimmen kann, ist ein Überspringen der Aktivität **Aufgabe in Sitzung besprechen** möglich. Mit der nachfolgenden Aktivität **Hausaufgabe erstellen** wird eine neue Hausaufgabe innerhalb des Systems für einen

ausgewählten Patienten erstellt. Der Therapeut kann nun wählen, ob er die Aufgabe komplett neue definieren oder ob er eine bereits erstellte Aufgabe verwenden möchte. Befindet sich keine passende und selbst erstellte Aufgabe in seinem Repositorium, so kann der Therapeut im Anschluss innerhalb eines geschlossenen Instituts-Repositoriums nach geteilten Aufgaben suchen. Hat er eine passende Aufgabe gefunden, so kann er die Aufgabe, inklusive aller multimedialen Inhalte, für den ausgewählten Patienten übernehmen. Um die individuelle Betreuung zu gewährleisten, kann die Aufgabe im Anschluss modifiziert werden, wobei eine neue Version dieser Aufgabe erstellt wird. Möchte der Therapeut eine neue, noch nicht angelegte Aufgabe erstellen, so wird er aufgefordert eine Beschreibung, in der multimediale Inhalte zur Verdeutlichung anhängt werden können, einzutragen und die notwendigen Parameter, wie beispielsweise Hilfsmittel, Selbstständigkeit und Wiederholungen, zu definieren. Hat der Therapeut die benötigten Informationen eingetragen, kann er anschließend die neu erstelle Aufgabe als Vorlage markieren und diese, falls erwünscht, innerhalb seines Instituts zu Verfügung stellen.

Um das IT-unterstütze Monitoring der Hausaufgaben zu ermöglichen, muss der Therapeut nun den Kontext, in dem die Aufgabe durchgeführt werden soll, definieren. Dabei hat er ebenfalls die Möglichkeit einen neuen Kontext zu definieren oder bereits erstelle Kontexte aus seinem eigenen Repositorium oder dem Instituts- Repositorium zu übernehmen. Gleichermaßen kann er den übernommenen Kontext mit den Anpassungen für den aktuellen Patienten modifizieren, wobei eine neue Version erstellt wird. Definiert der Therapeut einen neuen Kontext, so muss er dabei die Situation beschreibende Bedingungen angeben, mithilfe derer das mobile Endgerät den Ausführungskontext erkennen kann. Dabei kann der Therapeut zwischen elementaren, symbolischen und sensorischen Bedingungen wählen und anschließend entscheiden, ob er diesen Kontext als Vorlage markieren möchte.

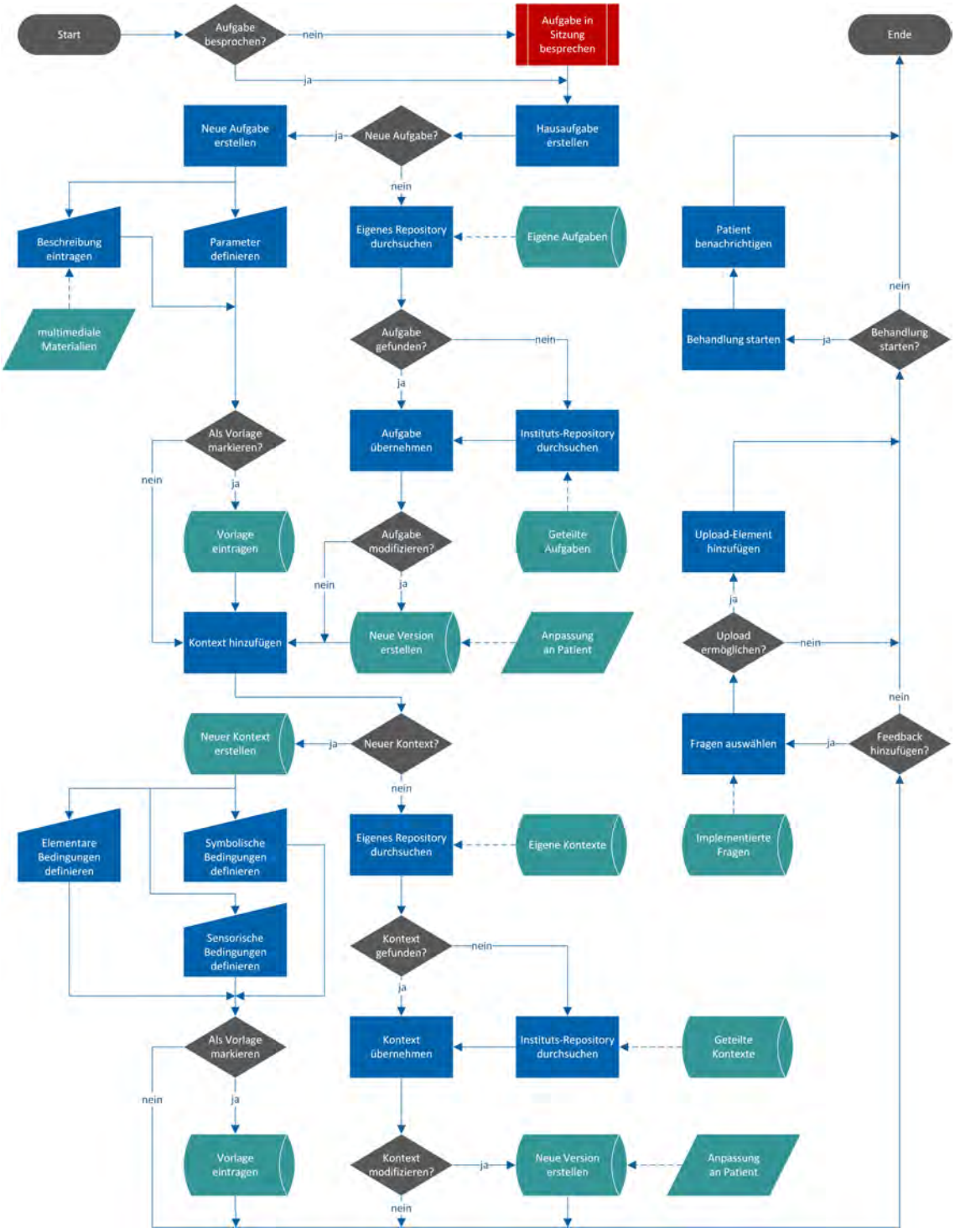


Abbildung 4.3: Soll-Prozess der Hausaufgabenerteilung

Nach der Kontextdefinition hat der Therapeut die Möglichkeit ein Feedback zu definieren, welches ausgefüllt an die Abgabe angehängt werden kann. Dazu wählt der Therapeut Fragen aus, die der Benutzer nach der Durchführung ausfüllen muss. Eine weitere Form der Rückkopplung kann das Upload-Element sein, welches der Therapeut im Anschluss an das Feedback anhängen kann. Innerhalb der Aktivität **Upload-Element hinzufügen** kann der Therapeut dem Patienten vorgeben, dass er als Ergebnis seiner Übung eine Datei, beispielsweise ein Bild, hochladen soll, um die Auswertung der Hausaufgabe zu unterstützen und den Patienten zu motivieren.

Zuletzt kann die Behandlung vom Therapeuten gestartet und der Patient über die neue Aufgabe benachrichtigt werden. Falls der Therapeut sich dagegen entscheidet, kann er die Behandlung mittels der erstellten Aufgabe auch zu einem späteren Zeitpunkt, beispielsweise nach der Therapiesitzung, starten.

4.3.2 Soll-Prozess des Hausaufgabenmonitoring

Hausaufgaben werden oftmals nicht ausgeführt oder durch den Benutzer modifiziert, weshalb der Monitoring-Prozess hauptsächlich den Therapeuten bei der Auswertung der Hausaufgabe unterstützen und den Patienten für die korrekte Ausführung motivieren soll. Auch der Soll-Prozess des Hausaufgabenmonitoring ist, übertragen auf den Hausaufgaben-Empfehlungsprozess nach Scheel et al. [28] (Abbildung 2.2), ein Subprozess für die Ausführung außerhalb der Sitzung (*Phase 4*), die Nachbesprechung in der Sitzung (*Phase 5*) und die Auswertung der Erfahrungen (*Phase 6*).

Der Prozess beginnt mit dem vordefinierten Prozess Hausaufgabenerteilung aus Kapitel 4.2.1, indem der Therapeut eine Hausaufgabe bespricht, erstellt und anschließend erteilt, beziehungsweise die Behandlung startet. Daraufhin wechselt der Akteur von Therapeut zu Patient, da dieser nun die Aufgabe, vor der eigentlichen Ausführung, mittels Smartphone herunterladen und starten muss. Dieses Ereignis kann für die spätere Auswertung von Interesse sein und wird deshalb durch das System erfasst und gespeichert.

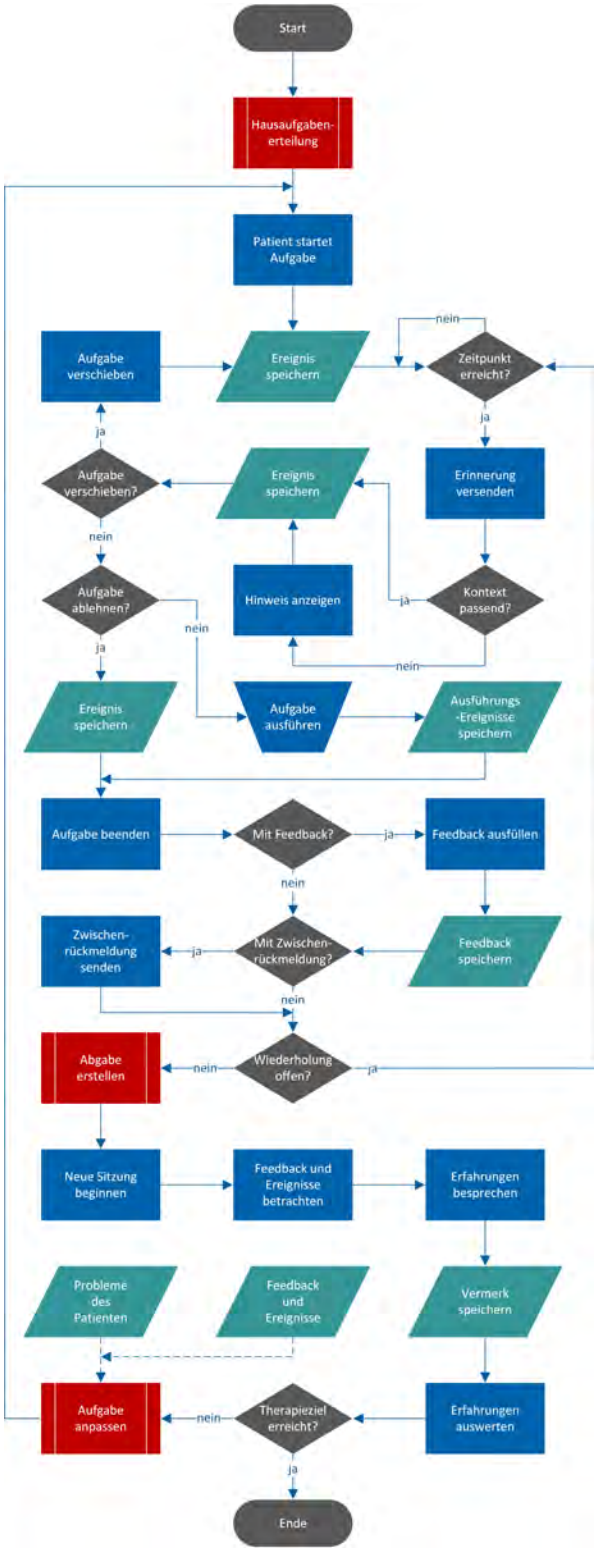


Abbildung 4.4: Soll-Prozess des Hausaufgabenmonitoring

Das System wartet nun bis der vom Therapeuten definierte Zeitpunkt für die Durchführung erreicht ist und zeigt, falls erreicht, den Patienten eine Benachrichtigung zur Ausführung an. Da Hausaufgaben, ausgeführt in verschiedenen Kontexten, auch verschiedene Ergebnisse zur Folge haben, kann das System in Abhängigkeit der Qualität der Kontextdefinition den Kontext ermitteln und den Patient informieren, falls sich dieser zu Beginn der Aufgabendurchführung in einem abweichenden Kontext befindet. Da der Patient Benachrichtigungen durch das System verhindern kann, indem er beispielsweise sein Smartphone ausgeschaltet hat oder sich in einem zur Behandlungsaufgabe abweichenden Kontext befinden kann, sind diese Informationen relevant und werden vermerkt. Hiernach kann der Patient entscheiden, ob er die Hausaufgabe verschieben, durchführen oder ablehnen möchte. Ein Verschieben der Hausaufgabe ist sinnvoll, wenn der Patient beispielsweise nach einem Systemhinweis plant den Kontext zu wechseln. Dafür muss er die Aktivität **Aufgabe verschieben** ausführen und den Zyklus nochmals durchlaufen. Möchte der Patient die Aufgabe durchführen, muss er die Aktivität **Aufgabe ausführen** beginnen, woraufhin die Ereignisse während der Ausführung vom System erfasst und gespeichert werden. Lehnt der Patient die Aufgabe ab, so wird dies vermerkt und die Aufgabe kann, wie auch nach der Ausführung der Aufgabe, offiziell beendet werden.

Abhängig von der Definition der Hausaufgabe, kann nach der Beendigung der Aufgabe ein Feedback ausgefüllt werden, indem der Patient vorher definierte Fragen beantworten oder zusätzlich multimediale Daten hochladen soll. Dies können beispielsweise Videos der Aufgabendurchführung zur gemeinsamen Besprechung innerhalb der nächsten Therapiesitzung oder auch Kameraaufnahmen von dem Ergebnis der Aufgabe sein. Hat der Therapeut eine Zwischenrückmeldung für jede Aufgabenwiederholung verlangt, wird eine Teilabgabe inklusive Feedback zur aktuellen Iteration erzeugt und an den Server versendet. Dadurch ist der Therapeut potenziell in der Lage Aufgaben, anhand des Zwischenergebnisses, während der laufenden Durchführung anzupassen.

Falls weitere Wiederholungen zur Ausführung stehen, wird eine neue Iteration gestartet und auf den Startzeitpunkt gewartet. Andernfalls kann der vordefinierte Prozess **Abgabe erstellen** ausgeführt werden, in dem das System alle Informationen zur Ausführung, insbesondere Änderungen zur Definition des Therapeuten, anreichert und diese an den

Server sendet. Nach der Einreichung einer Abgabe gilt eine Aufgabe als abgeschlossen und kann vom Patienten nicht mehr ausgeführt werden.

In der anschließenden Therapiesitzung kann der Therapeut nun das Feedback und die aufgezeichneten Ereignisse betrachten und mit dem Patienten besprechen. Gleichzeitig verfügt der Therapeut nicht nur über die objektiven Aufzeichnungen für die Bewertung der Übung, sondern kann die subjektiven Erfahrungen des Patienten mit der Aufgabe erfragen und vermerken. Zur Auswertung stehen dem Therapeuten somit zwei Quellen zu Verfügung, aus denen er den aktuellen Zustand des Patienten erfahren und mit dem Therapieziel vergleichen kann. Wurde das angestrebte Ziel erreicht kann der Therapeut die Übung offiziell beenden. Andernfalls kann der Therapeut die Aufgabe, mit Zuhilfenahme des Feedbacks und der Ereignisse, während der laufenden Aufgabe, respektive die berichteten Probleme und Erfahrungen des Patienten, anpassen und nochmals in modifizierter Form erteilen.

4.4 Anforderungen des Rahmenwerks

Dieser Abschnitt gibt einen Überblick über alle funktionalen und nicht-funktionalen Anforderungen des zu entwickelnden Rahmenwerks, welche in den folgenden Unterabschnitten in Module gegliedert, illustriert und beschrieben werden. Jede Anforderung besitzt einen eindeutigen Identifikator (ID) und eine eindeutige Bezeichnung, die für Verweise in anderen Abschnitten verwendet werden. Zusätzlich wird für jede Anforderung eine Priorität mithilfe der **MoSCoW-Priorisierung**[1] vergeben.

[1] Wertebereich: *MUSS*, *SOLL*, *KANN*, *NOCH NICHT* [49]

ID	BEZEICHNUNG	PRIORITÄT	SEITE
GAST-MODUL			
F/01	Marktplatz	*KANN*	49
F/02	API Dokumentation	*MUSS*	50
ACCOUNT-MODUL			
F/03	Autorisierung	*MUSS*	50
F/04	Registrierung	*MUSS*	50
F/05	Authentifizierung	*MUSS*	50
F/06	Benutzerverwaltung	*MUSS*	51
INSTITUTS-MODUL			
F/07	Institut verwalten	*SOLL*	51
F/08	Behandlungsangebot verwalten	*KANN*	51
F/09	Therapeuten einladen	*KANN*	51
KONTEXT-MODUL			
F/10	Kontext-Repositorium	*MUSS*	52
F/11	Kontext definieren	*MUSS*	53
AUFGABEN-MODUL			
F/12	Aufgaben-Repositorium	*MUSS*	53
F/13	Aufgaben definieren	*MUSS*	53
F/14	Medien-Manager	*SOLL*	53
F/15	Disclaimer-Manager	*KANN*	53
F/16	Copyright-Manager	*KANN*	53

Tabelle 4.3: Anforderungen an das Rahmenwerk - Teil 1

ID	BEZEICHNUNG	PRIORITÄT	SEITE
PATIENTEN-MODUL			
F/17	Behandlungsaufgaben abrufen	*MUSS*	55
F/18	Behandlungsaufgaben manipulieren	*MUSS*	55
F/19	Abgabe erstellen	*MUSS*	56
F/20	Studienberechtigung	*SOLL*	56
BEHANDLUNGS-MODUL			
F/21	Behandlungen verwalten	*MUSS*	57
F/22	Behandlungsstatistik berechnen	*NOCH NICHT*	57
F/23	Behandlung erstellen	*MUSS*	57
F/24	Behandlungsvermerke verwalten	*SOLL*	57
F/25	Report erstellen	*NOCH NICHT*	57
THERAPEUTEN-MODUL			
F/26	Behandlungsaufgabe erstellen	*MUSS*	59
F/27	Behandlungsaufgabe bearbeiten	*MUSS*	59
F/28	Aufgabenstatus ändern	*MUSS*	59
F/29	Behandlungsaufgabe überwachen	*SOLL*	59
FEEDBACK-MODUL			
F/30	Feedback verwalten	*SOLL*	59
F/31	Upload-Elemente verwalten	*SOLL*	60
F/32	Fragen verwalten	*MUSS*	60
EINLADUNGS-MODUL			
F/33	Einladungen verwalten	*KANN*	61
F/34	Einladungen prüfen	*KANN*	61

Tabelle 4.4: Anforderungen an das Rahmenwerk - Teil 2

ID	BEZEICHNUNG	PRIORITÄT	SEITE
ADMIN-MODUL			
F/35	Administration	*NOCH NICHT*	62
F/36	Institute konfigurieren	*SOLL*	62
SYSTEM			
NF/01	Zuverlässigkeit	*MUSS*	63
NF/02	Erreichbarkeit	*MUSS*	63
NF/03	Robustheit	*SOLL*	63
NF/04	Sicherheit/Datenschutz	*MUSS*	63
NF/05	Effizienz	*SOLL*	63
NF/06	Wartbarkeit	*KANN*	63
NF/07	Portabilität	*MUSS*	64
NF/08	Benutzerfreundlichkeit	*MUSS*	64

Tabelle 4.5: Anforderungen an das Rahmenwerk - Teil 3

4.4.1 Funktionale Anforderungen

Die funktionalen Anforderungen beschreiben die für das Rahmenwerk dieser Arbeit benötigten Funktionalitäten, insbesondere die Leistungen aus Sicht der Nutzer. Diese Informationen bilden die Grundlage für den Entwurf der Datenstruktur und des Verhaltens des Systems. Aufgrund der Menge an funktionalen Anforderungen, werden diese in Modulen organisiert und als Anwendungsfalldiagramme illustriert. Dabei können die nachfolgenden Akteure beteiligt sein.

Akteur:	**Gast**
Beschreibung:	Gäste sind nicht angemeldete Benutzer, welche nur Zugriff auf öffentliche Inhalte wie das Gäste-Modul und den Login/Registrieren-Bereich besitzen.

Akteur:	**Gast**
Beschreibung:	Gäste sind nicht angemeldete Benutzer, welche nur Zugriff auf öffentliche Inhalte wie das Gäste-Modul und den Login/Registrieren-Bereich besitzen.

Akteur:	**Benutzer**
Beschreibung:	Jeder angemeldete Akteur erbt im System von der abstrakten Rolle Benutzer und hat somit Zugriff auf alle Profilverwaltungsfunktionen, sowie auf die öffentlichen Inhalte. Die Rolle Benutzer kann es im System nicht geben.

Akteur:	**Patient**
Beschreibung:	Patienten sind Benutzer, welche Daten für das System produzieren, um eine verbesserte Lebensqualität durch IT-unterstützte Therapien erhalten zu können. Sie haben Zugriff auf das Patienten-Modul, in dem sie ihre Aufgaben verwalten, Änderungen einsehen, Abgaben einsehen und Statistiken abrufen können.

Akteur:	**Therapeut**
Beschreibung:	Therapeuten sind die zentralen Akteure im System, welche Behandlungen mit dem System durchführen. Dafür können sie Patienten einladen, für welche sie Aufgaben in einem Repositorium erstellen und verwalten, Behandlungsaufgaben hinzufügen und diese individuell an den Kontext anpassen.

Akteur:	**Administrator**
Beschreibung:	Administratoren verwalten alle Vorgänge und Daten im System und können alle nicht patientenbezogen Therapeuten-Funktionen durchführen, um bei Fehlern den Therapeuten zu unterstützen.

Akteur:	**E-Mail-Service**
Beschreibung:	Dienst zur E-Mail-Benachrichtigung von Systembenutzern.

Akteur:	**[Bezeichnung]-Modul**
Beschreibung:	Konnektor zu Anwendungsfällen eines abgegrenzten Systembereich innerhalb des Systems.

GAST-MODUL

Im Folgenden werden alle funktionalen Anforderungen für dieses Modul beschrieben und in Abbildung 4.5 als Anwendungsfalldiagramm illustriert. Funktionen innerhalb des Gast-Moduls können von allen Seitenbesuchern, insbesondere allen angemeldeten Benutzern, aufgerufen werden.

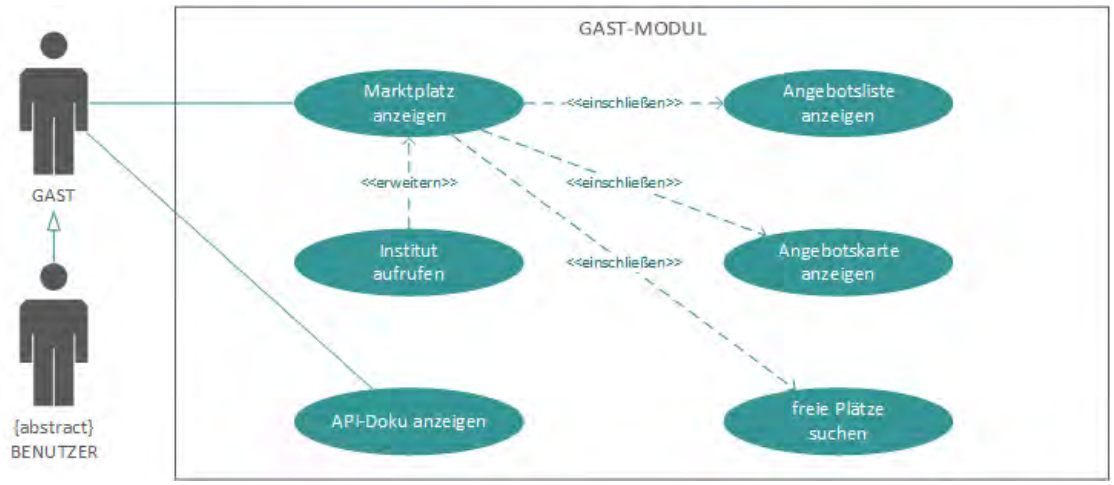


Abbildung 4.5: Anwendungsfalldiagramm für das Gast-Modul

Funktionale Anforderung 1 (Marktplatz) Der Marktplatz gibt alle hinterlegten Institute und deren Angebote aus, welche als Liste oder innerhalb einer Karten-Ansicht angezeigt werden. Gleichzeitig sollen alle Angebote kategorisiert werden und Gäste über den Marktplatz nach freien Behandlungsplätzen suchen können.

Funktionale Anforderung 2 (API Dokumentation) Entwickler können die Schnittstellen-Dokumentation des Systems aufrufen, um Klienten-Anwendungen zu generieren.

ACCOUNT-MODUL

Im Folgenden werden alle funktionalen Anforderungen für dieses Modul beschrieben und in Abbildung 4.6 als Anwendungsfalldiagramm illustriert.

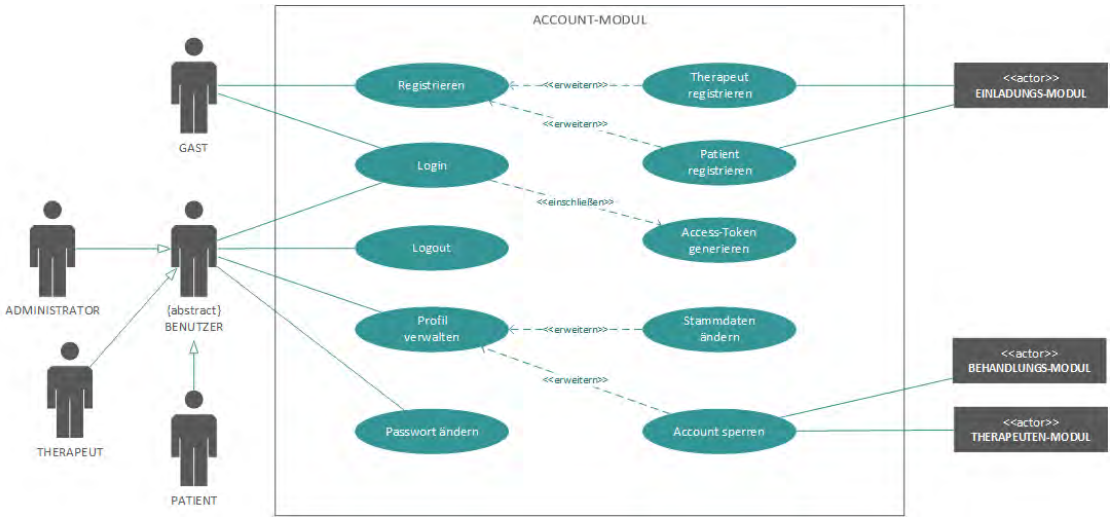


Abbildung 4.6: Anwendungsfalldiagramm für das Account-Modul

Funktionale Anforderung 3 (Autorisierung) Das System ordnet Funktionalitäten verschiedenen Nutzergruppen zu und schränkt den Zugriff entsprechend ein.

Funktionale Anforderung 4 (Registrierung) Gäste mittels Einladung als Therapeut für ein bestimmtes Institut oder als Patient für eine bestimmte Behandlung registrieren.

Funktionale Anforderung 5 (Authentifizierung) Gäste, Patienten, Therapeuten und Administratoren können sich beim System einloggen und ihre Zugriffs-Tokens anfordern, mit welchen sie, innerhalb eines gewissen Zeitraums, Zugriff auf das System bekommen.

Funktionale Anforderung 6 (Benutzerverwaltung) Administratoren, Therapeuten und Patienten können als angemeldete Benutzer zusätzlich ihr Profil verwalten und gegebenenfalls den Account sperren. Account-Sperrungen von Patienten und Therapeuten benachrichtigen die im Verhältnis stehende Gegenseite und stoppen aktuelle Behandlungen.

INSTITUTS-MODUL

Im Folgenden werden alle funktionalen Anforderungen für dieses Modul beschrieben und in Abbildung 4.7 als Anwendungsfalldiagramm illustriert.

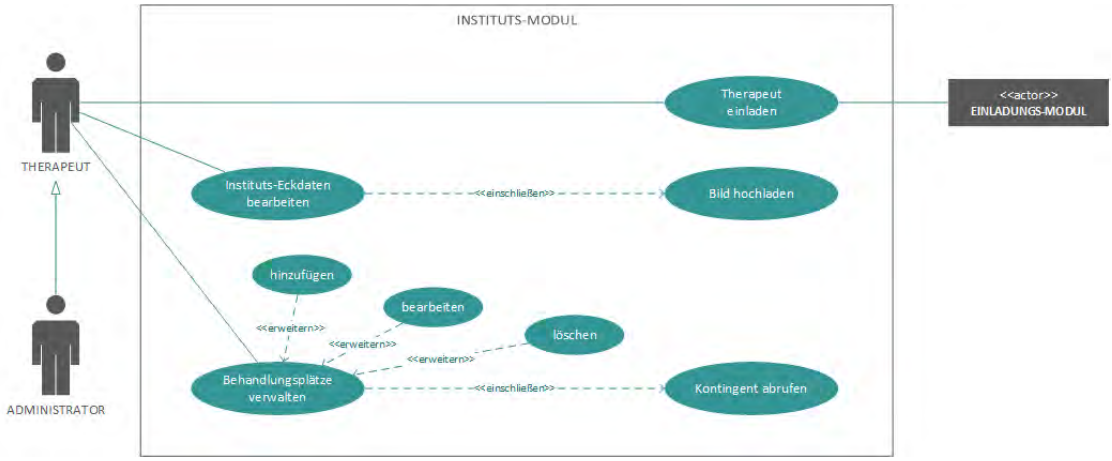


Abbildung 4.7: Anwendungsfalldiagramm für das Instituts-Modul

Funktionale Anforderung 7 (Institut verwalten) Therapeuten werden innerhalb des Systems in Instituten organisiert, welche für den Marktplatz öffentliche Eckdaten wie beispielsweise Adresse, Öffnungszeiten und Logo bereitstellt. Diese können von Administratoren und Therapeuten des zugehörigen Instituts definiert werden.

Funktionale Anforderung 8 (Behandlungsangebot verwalten) Institute können ihr Behandlungsangebot zusammenstellen, beschreiben und kategorisieren.

Funktionale Anforderung 9 (Therapeuten einladen) Ein Therapeut kann eine Einladung für einen Kollegen erstellen, um diesen zu seinem Institut hinzuzufügen.

KONTEXT-MODUL

Im Folgenden werden alle funktionalen Anforderungen für dieses Modul beschrieben und in Abbildung 4.8 als Anwendungsfalldiagramm illustriert.

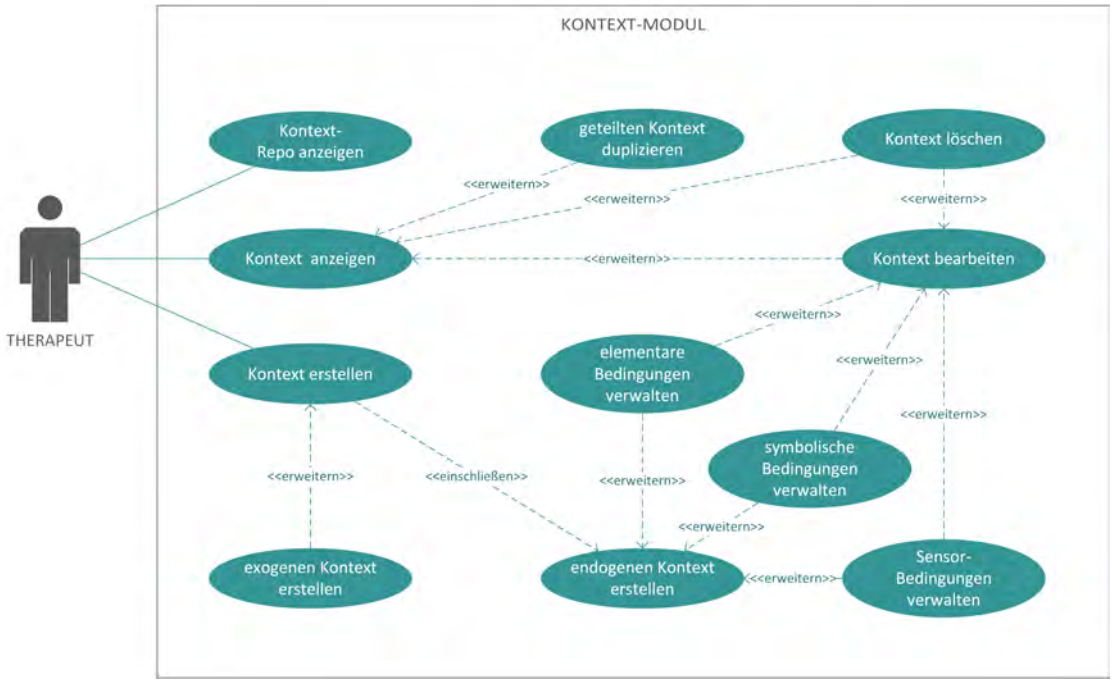


Abbildung 4.8: Anwendungsfalldiagramm für das Kontext-Modul

Funktionale Anforderung 10 (Kontext-Repositorium) Innerhalb dieses Moduls können Therapeuten endogene und exogene Kontexte in einem Repositorium verwalten. Therapeuten können selbst erstellte und geteilte Kontexte anzeigen oder duplizieren und zusätzlich als Vorlage markieren. Kontexte können innerhalb des Instituts geteilt werden oder nur dem Ersteller zu Verfügung stehen.

Funktionale Anforderung 11 (Kontext definieren) Es sollen Kontexte erstellt werden können, welche Information auf zwei unterschiedliche Arten erhalten. Endogene Kontexte können Bedingungen hinzufügen, welche in drei Gruppen aufgeteilt werden können. Die elementare Bedingungen, wie Ort, Zeitpunkt oder Konnektivität, beschreiben sensorisch genau überprüfbare Bedingungen, für die jedoch die Auswahl der Sensorik dem System überlassen wird. Die

symbolischen Bedingungen beschreiben Bedingungen, die vom Menschen interpretiert und bewertet werden müssen. Sensorische Bedingungen betreffen nur einen bestimmten Sensor und besitzen einen Operator mit Schwellwert, um nicht klar definierte Situationen maschinell zu erkennen. Ein exogener Kontext soll in zukünftigen Systemversionen Daten von externen Dienstleistern erfassen und interpretieren, um beispielsweise eine genauere Situationsbestimmung zu erlauben. Jede gespeicherte Änderung soll dabei eine neue Version erzeugen, um Anpassungen nachvollziehen zu können.

AUFGABEN-MODUL

Im Folgenden werden alle funktionalen Anforderungen für dieses Modul beschrieben und in Abbildung 4.9 als Anwendungsfalldiagramm illustriert.

Funktionale Anforderung 12 (Aufgaben-Repositorium) Innerhalb dieses Moduls können Therapeuten Aufgaben in einem Repositorium verwalten. Therapeuten können selbst erstellte und geteilte Aufgaben anzeigen oder duplizieren und zusätzlich als Vorlage markieren. Aufgaben können innerhalb des Instituts geteilt werden oder nur dem Ersteller zu Verfügung stehen.

Funktionale Anforderung 13 (Aufgabe definieren) Es sollen Aufgaben erstellt werden können, welche diverse Inhalte, unter anderem eine Copyright-Angabe, optional diverse Medien-Elemente und einen Disclaimer, beinhalten. Jede gespeicherte Änderung soll dabei eine neue Version erzeugen, um Anpassungen nachvollziehen zu können.

Funktionale Anforderung 14 (Medien-Manager) Um mediale Inhalte über mehrere Aufgaben hinweg zu verwalten, können alle hinzugefügten Medien in einer Manager-Ansicht verwaltet werden.

Funktionale Anforderung 15 (Disclaimer-Manager) Disclaimer-Angaben sollen in einer zentralen Ansicht angelegt und verwaltet werden können.

Funktionale Anforderung 16 (Copyright-Manager) Copyright-Angaben sollen in einer zentralen Ansicht angelegt und verwaltet werden können.

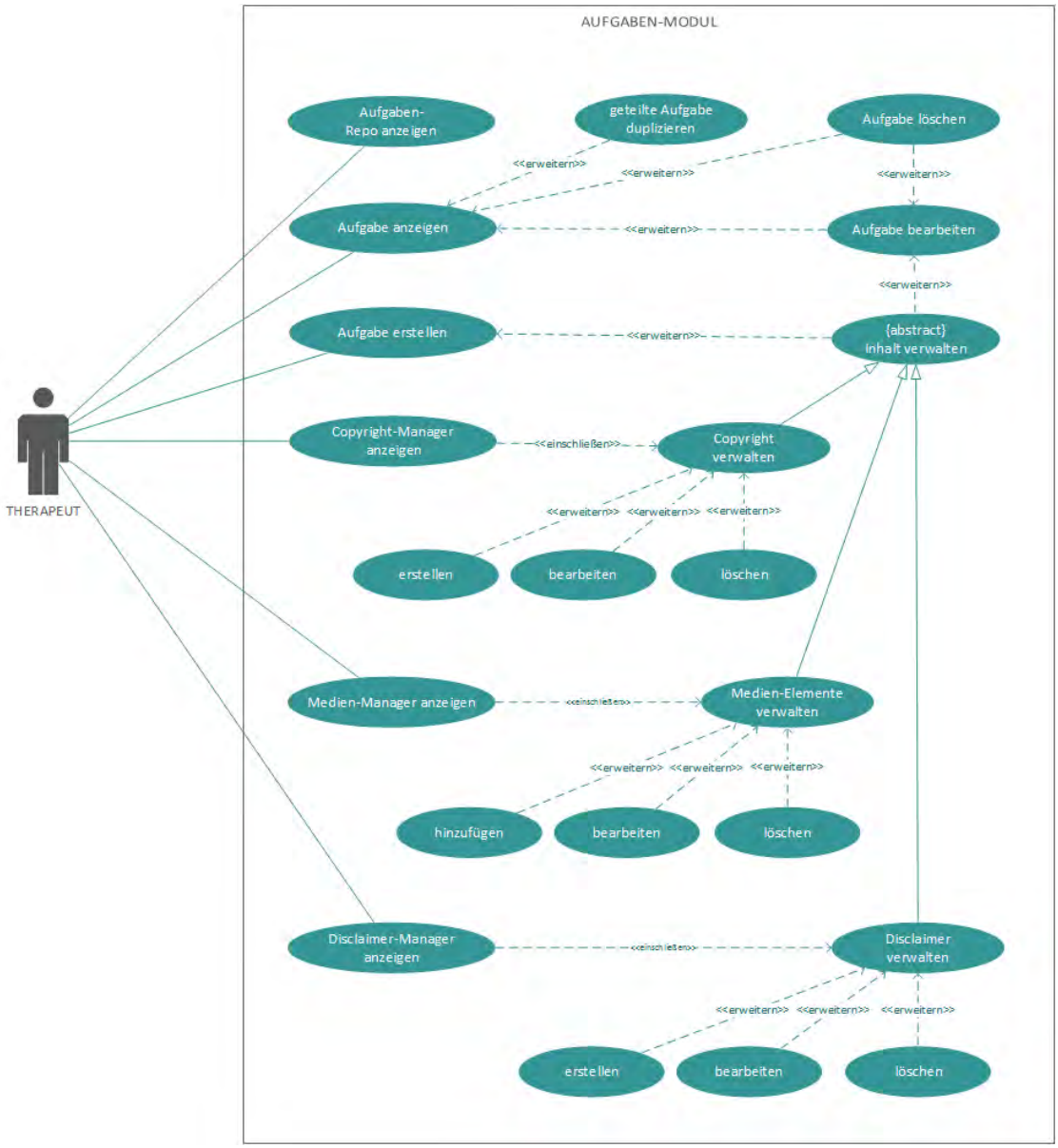


Abbildung 4.9: Anwendungsfalldiagramm für das Aufgaben-Modul

PATIENTEN-MODUL

Im Folgenden werden alle funktionalen Anforderungen für dieses Modul beschrieben und in Abbildung 4.10 als Anwendungsfalldiagramm illustriert.

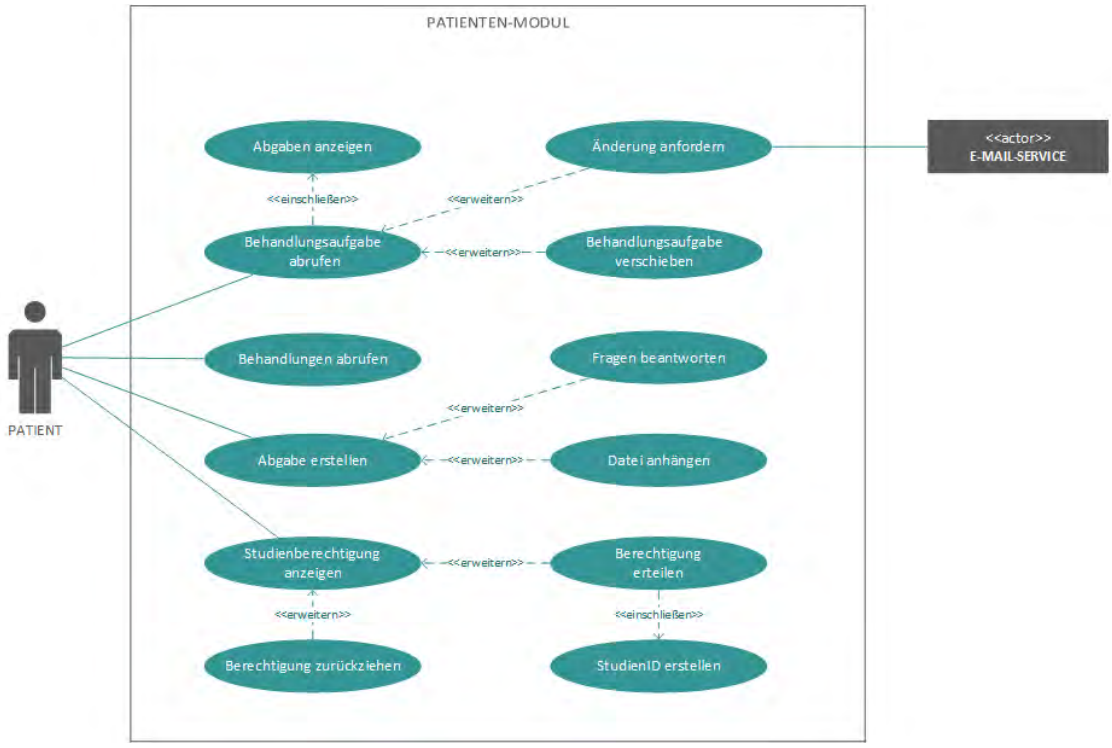


Abbildung 4.10: Anwendungsfalldiagramm für das Patienten-Modul

Funktionale Anforderung 17 (Behandlungsaufgaben abrufen) Patienten können über das Patienten-Modul die Daten zu ihren Behandlungen, sowie die aktuellste Version der Behandlungsaufgaben abrufen. Außerdem können zu jeder abgeschlossenen Behandlungsaufgabe die übermittelten Abgaben angezeigt werden.

Funktionale Anforderung 18 (Behandlungsaufgaben manipulieren) Falls der Patient eine Behandlungsaufgabe zum geplanten Zeitpunkt nicht ausführen kann, so kann er eine Verschiebung vermerken. Der Patient kann Änderungen anfordern, falls Teile der Behandlungsaufgabe angepasst werden müssen, woraufhin der behandelnde Therapeut benachrichtigt wird.

Funktionale Anforderung 19 (Abgabe erstellen) Hat der Patient die Behandlungsaufgabe absolviert, kann er eine Abgabe erstellen und die hinterlegte

Feedback-Vorlage füllen, indem er Fragen beantwortet oder Dateien anhängt.

Funktionale Anforderung 20 (Studienberechtigung) Der Patient kann seine erhobenen Daten jederzeit für die Forschung bereitstellen oder die erteilte Berechtigung entziehen.

BEHANDLUNGS-MODUL

Im Folgenden werden alle funktionalen Anforderungen für dieses Modul beschrieben und in Abbildung 4.11 als Anwendungsfalldiagramm illustriert.

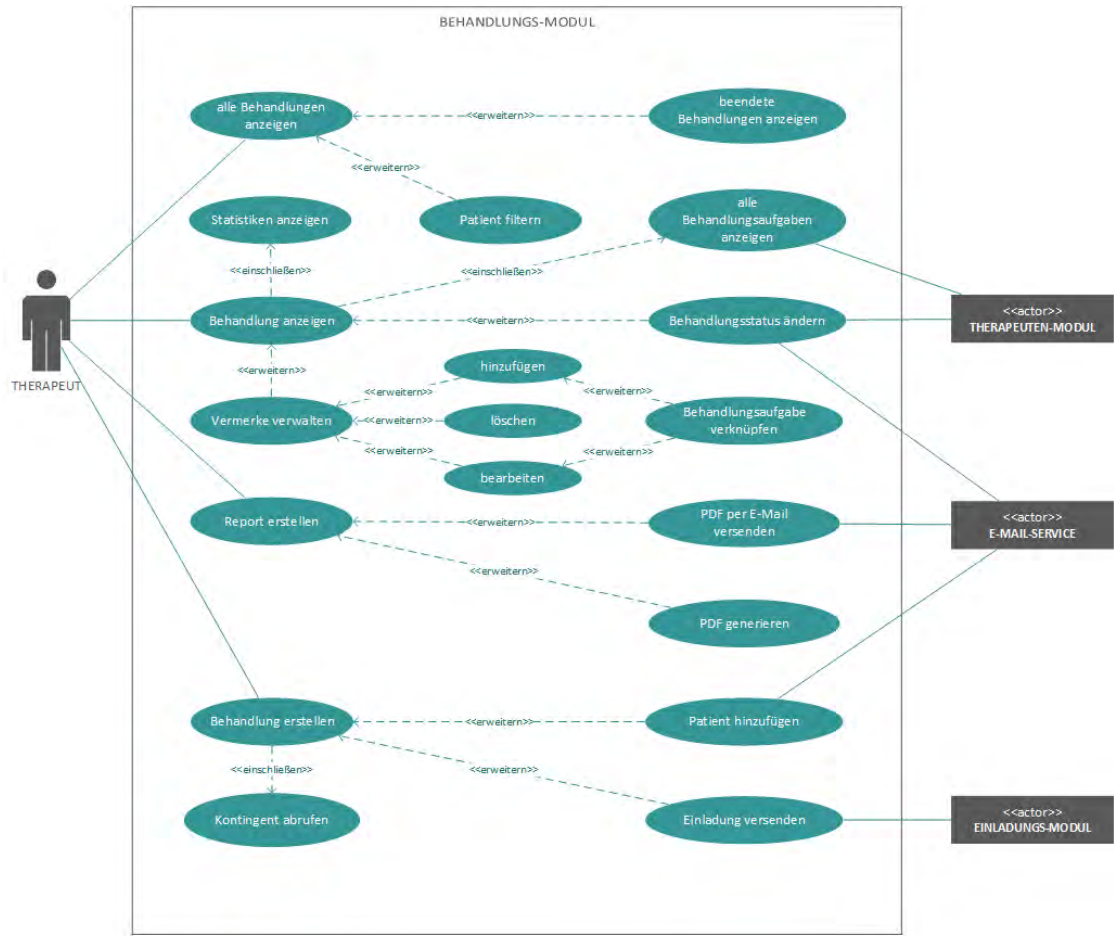


Abbildung 4.11: Anwendungsfalldiagramm für das Behandlungs-Modul

Funktionale Anforderung 21 (Behandlungen verwalten) Therapeuten können über das Behandlungs-Modul alle eigenen aktiven und beendeten Behandlungen einsehen und nach Patienten filtern. In einer Detail-Ansicht können sie einzelne Behandlungen öffnen, um sich Behandlungsaufgaben und Behandlungsstatistiken für diese Behandlung anzeigen zu lassen oder den Behandlungsstatus zu ändern. Damit wird die Behandlung beendet und alle laufenden Behandlungsaufgaben gestoppt oder die Behandlung wird erneut gestartet.

Funktionale Anforderung 22 (Behandlungsstatistik berechnen) Für eine Behandlung werden verschiedene Auswertungen berechnet, um dem Therapeuten einen besseren Überblick über die Behandlung zu geben.

Funktionale Anforderung 23 (Behandlung erstellen) Es wird eine neue Behandlung hinzugefügt, wobei der Patient (registriert oder nicht) entweder mit Angabe der E-Mail-Adresse eingeladen wird oder noch leer bleibt, um sich selbst mittels Einladungs-Code zu registrieren.

Funktionale Anforderung 24 (Behandlungsvermerke verwalten) Es können manuell oder automatisch erstellte Vermerke zur laufenden Behandlungen hinzugefügt, bearbeitet oder gelöscht werden, sowie mit den Behandlungsaufgaben verknüpft werden.

Funktionale Anforderung 25 (Report erstellen) Es können Reports aus den Behandlungsaufgaben und -vermerken für einen Patienten erstellt werden und als PDF angezeigt oder als E-Mail versendet werden.

THERAPEUTEN-MODUL

Im Folgenden werden alle funktionalen Anforderungen für dieses Modul beschrieben und in Abbildung 4.12 als Anwendungsfalldiagramm illustriert.

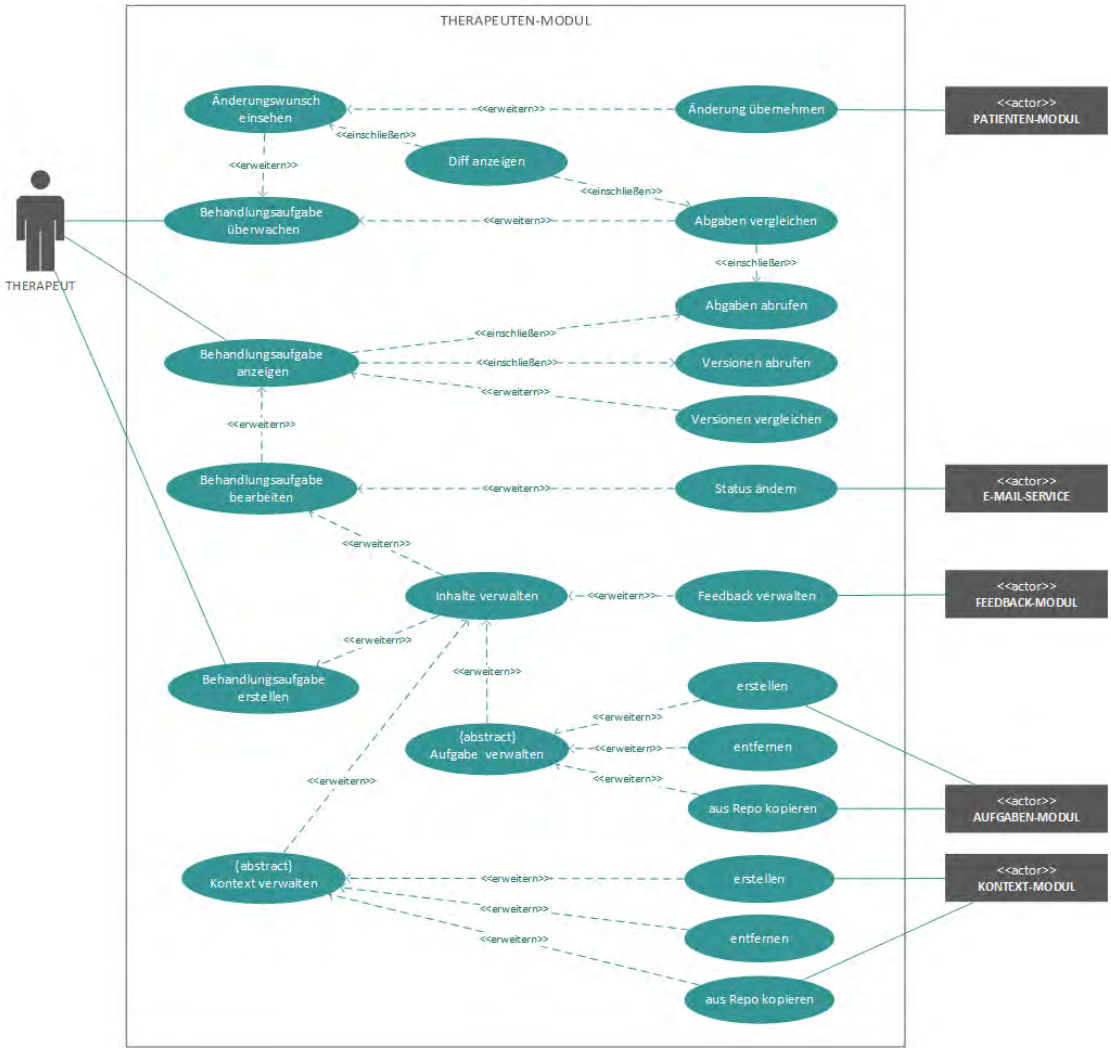


Abbildung 4.12: Anwendungsfalldiagramm für das Therapeuten-Modul

Funktionale Anforderung 26 (Behandlungsaufgabe erstellen) Für eine Behandlung kann eine Behandlungsaufgabe, bestehend aus einer Aufgabe, einem Kontext und einem Feedback, erstellt werden. Dabei kann eine Aufgabe oder ein Kontext direkt erstellt oder aus dem Repositorium kopiert werden. Es ist außerdem möglich komplett neue Vorlagen für Aufgaben oder Kontexte zu erstellen und diese direkt als Vorlage zu teilen. Zusätzlich können vordefinierte Fragen für das Feedback

ausgewählt werden, respektive ein Platzhalter für einen Upload hinzugefügt werden.

Funktionale Anforderung 27 (Behandlungsaufgabe bearbeiten) Behandlungsaufgaben können fortwährend auf den Patienten angepasst werden. Pro Änderung wird eine neue Aufgaben- oder Kontext-Version erstellt. Zusätzlich kann das Feedback verwaltet und Änderungswünsche zur aktuellen Aufgaben-Kontext-Kombination eingesehen werden.

Funktionale Anforderung 28 (Aufgabenstatus ändern) Behandlungsaufgaben besitzen einen definierten Status, der manuell oder automatisch geändert werden kann. Bei jeder Statusänderung muss ein automatischer Behandlungsvermerk eingetragen, respektive die Aufforderung für einen manuellen Behandlungsvermerk gestellt, werden.

Funktionale Anforderung 29 (Behandlungsaufgabe überwachen) Um Behandlungsaufgaben überwachen zu können, werden dem Therapeuten statistische Auswertungen angezeigt. Außerdem können Versionen von Behandlungsaufgaben verglichen werden, sowie Abgabe für eine bestimmte Version eingesehen werden. Änderungswünsche von Patienten können mit der aktuellen Version verglichen und übernommen werden. Weiterhin können Abweichungen von Abgaben zur Aufgabenstellung analysiert und verglichen werden.

FEEDBACK-MODUL

Im Folgenden werden alle funktionalen Anforderungen für dieses Modul beschrieben und in Abbildung 4.13 als Anwendungsfalldiagramm illustriert.

Funktionale Anforderung 30 (Feedback verwalten) Erstellt oder bearbeitet ein Therapeut eine Behandlungsaufgabe, so kann er dafür ein Vorlage für ein Feedback erstellen, die der Patient bei jeder Abgabe ausfüllen muss. In einer Feedback-Vorlage können Fragen oder Upload-Elemente hinzugefügt und verwaltet werden.

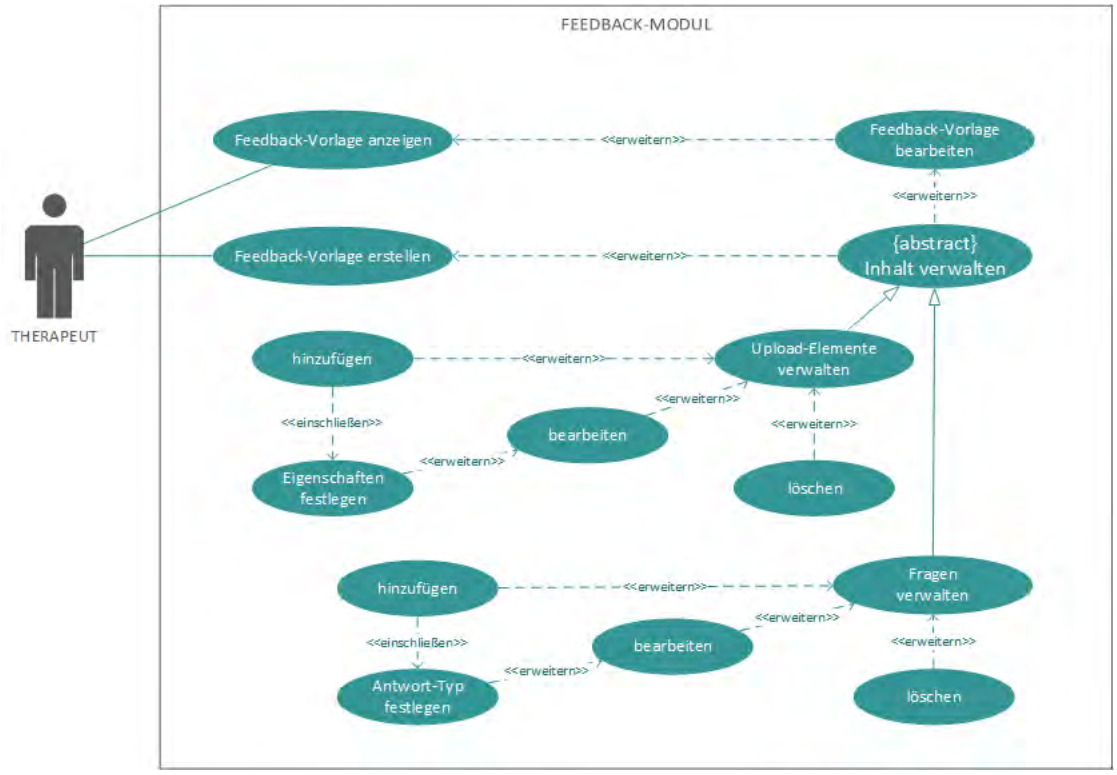


Abbildung 4.13: Anwendungsfalldiagramm für das Feedback-Modul

Funktionale Anforderung 31 (Upload-Elemente verwalten) Es können Medien-Elemente zum Upload hinzugefügt, bearbeitet und gelöscht werden, um Resultate eines vorgegebenen Datei-Typs im System ablegen zu können.

Funktionale Anforderung 32 (Fragen verwalten) Es können zum Feedback hinzugefügt, bearbeitet und gelöscht werden, welche der Patient nach der Aufgabendurchführung beantworten soll.

EINLADUNGS-MODUL

Im Folgenden werden alle funktionalen Anforderungen für dieses Modul beschrieben und in Abbildung 4.14 als Anwendungsfalldiagramm illustriert.

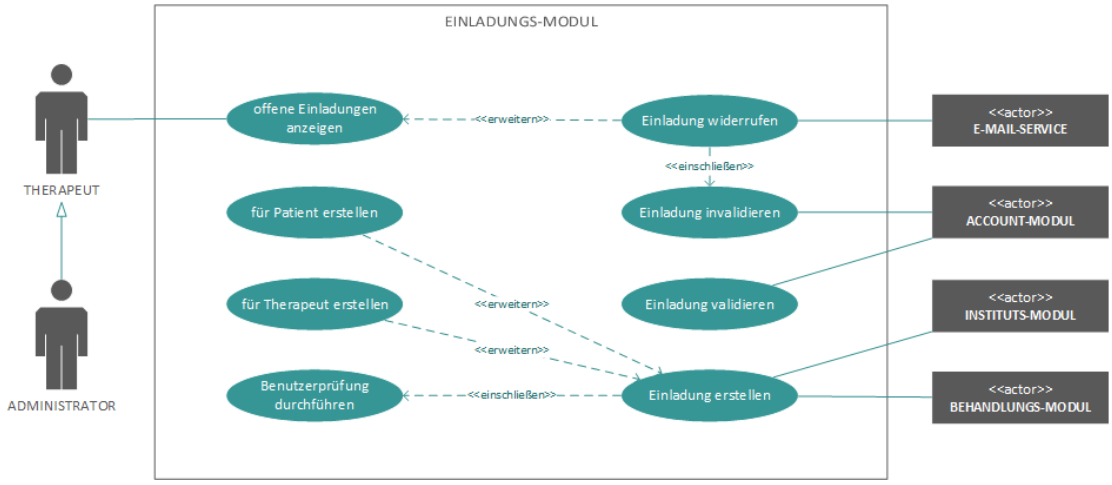


Abbildung 4.14: Anwendungsfalldiagramm für das Einladungs-Modul

Funktionale Anforderung 33 (Einladungen verwalten) Über das Einladungs-Modul können Therapeuten die erstellten Einladungen ihres Instituts und Administratoren alle Einladungen im System anzeigen und widerrufen. Wird eine Einladung widerrufen, so wird der Betreffende benachrichtigt und die Einladung invalidiert.

Funktionale Anforderung 34 (Einladungen prüfen) Das Account-Modul kann Einladungen validieren lassen, um einen Registriervorgang zu bestätigen und anschließend die Einladung invalidieren.

ADMIN-MODUL

Im Folgenden werden alle funktionalen Anforderungen für dieses Modul beschrieben und in Abbildung 4.15 als Anwendungsfalldiagramm illustriert.

Funktionale Anforderung 35 (Administration) In der Administration sollen alle organisatorischen Aufgaben durchgeführt werden können. Der Administrator kann alle Patienten, die eine Studienberechtigung erteilt haben, anzeigen und den kompletten Datensatz aller gelisteter Patienten exportieren. Außerdem kann er den Marktplatz sperren und eine Sperrnachricht zu Wartungszwecken eingeben. Weiterhin kann der Administrator Einstellungen für Repositorien und die Manager

konfigurieren. Über eine Benutzerliste kann er zusätzlich Benutzer anzeigen, sperren und Passwörter zurücksetzen lassen, woraufhin eine E-Mail an den betreffenden gesendet wird.

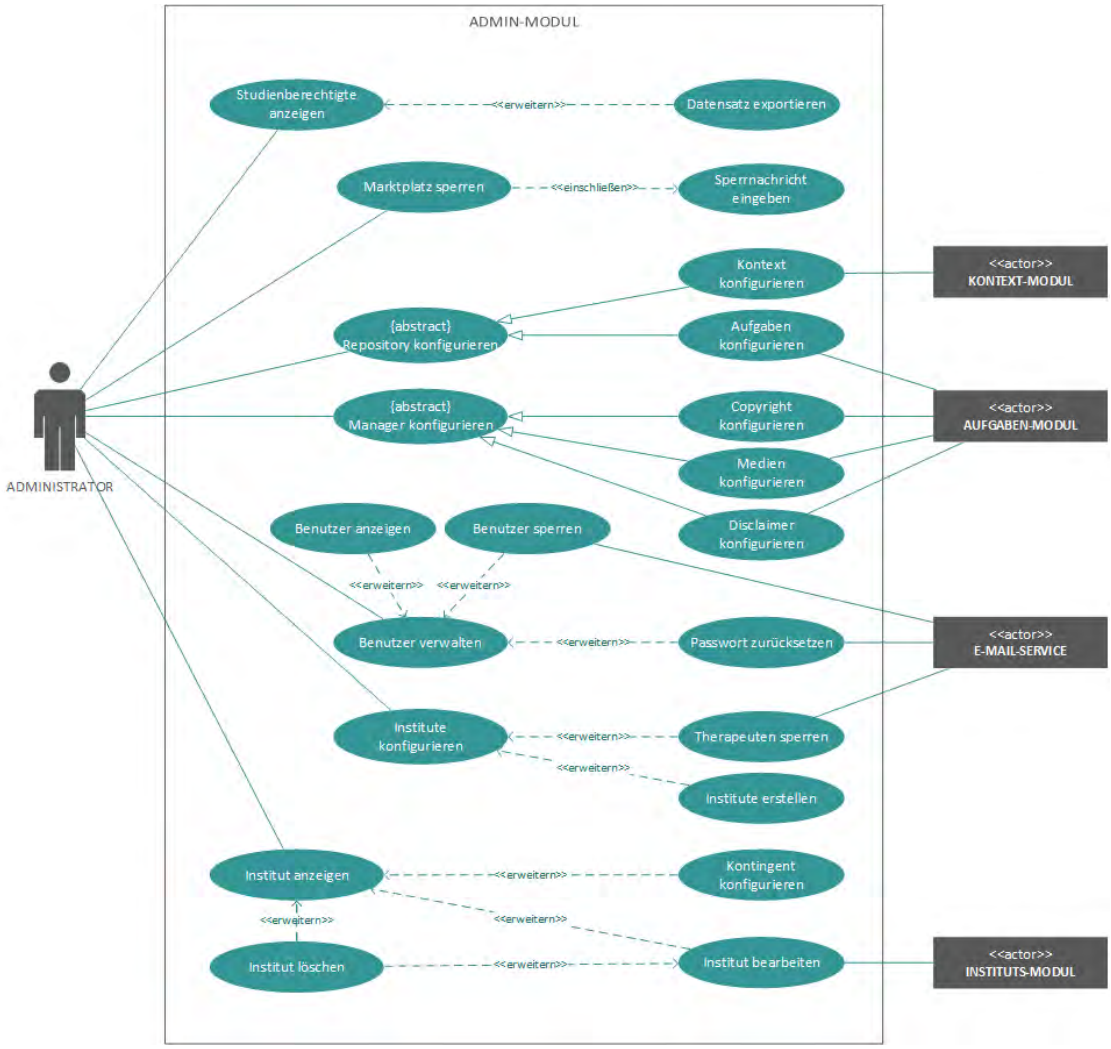


Abbildung 4.15: Anwendungsfalldiagramm für das Admin-Modul

Funktionale Anforderung 36 (Institute konfigurieren) Um neue Benutzer einzuladen, kann der Administrator Institute erstellen. Bestehende Institute und deren Benutzer kann er verwalten oder sperren. Außerdem kann für Institute ein Kontingent für Behandlungen festlegen und deren Eckdaten bearbeiten werden.

ANMERKUNG

Eine detaillierte Beschreibung aller Systemaufgaben der funktionalen Systemanforderungen und deren beteiligten Akteuren befindet sich im angehängten Analysedokument (Anhang A.1) unter Kapitel 2.3.

4.4.2 Nicht-Funktionale Anforderungen

Im Allgemeinen legen nicht-funktionale Anforderungen die Qualitätsansprüche an Funktionen eines System fest. Nachfolgend werden alle nicht-funktionalen Anforderungen an das Rahmenwerk dieser Arbeit definiert und beschrieben.

Nicht-funktionale Anforderung 1 (Zuverlässigkeit) Das System muss für den Benutzer stets zuverlässig, also fehlerfrei, ohne Datenverlust und auf dem aktuellen Datenstand, erreichbar sein.

Nicht-funktionale Anforderung 2 (Erreichbarkeit) Das System darf nur eine minimale „Down-Zeit“ haben und muss den Patienten zuverlässig über die gängigen Tageszeiten bei der Therapie unterstützen.

Nicht-funktionale Anforderung 3 (Robustheit) Das System muss bei fehlerhaften Eingaben in einen definierten Zustand wechseln.

Nicht-funktionale Anforderung 4 (Sicherheit/Datenschutz) Der Schutz von vertraulichen, personenbezogenen Daten spielt bei diesem System eine zentrale Rolle und muss unbedingt gewährleistet werden.

Nicht-funktionale Anforderung 5 (Effizienz) Aufrufe und Zusendungen von Daten, das gilt insbesondere für Behandlungsaufgaben, müssen im System performant bearbeitet werden, um den Benutzer eine flüssige und schnelle Arbeitsumgebung zu schaffen.

Nicht-funktionale Anforderung 6 (Wartbarkeit) Das System muss für Anwender und Entwickler gleichermaßen gut dokumentiert sein und sollte Änderungen und Erweiterungen am Systemkern ohne unnötigen Aufwand ermöglichen.

Nicht-funktionale Anforderung 7 (Portabilität) Das System muss für jede Art von Client oder Betriebssystem erreichbar und interpretierbar sein und darf deshalb nicht auf eine Client-Technologie zugeschnitten entwickelt werden.

Nicht-funktionale Anforderung 8 (Benutzerfreundlichkeit) Die Client-Anwendungen des Systems müssen ohne technische Kenntnisse für Laien selbsterklärend entwickelt werden. Benutzer müssen stets einfach und schnell ihre Aufgabe erfüllen können.

5

Architektur

In diesem Kapitel wird basierend auf der Anforderungsanalyse eine Architektur für das Rahmenwerk entworfen. In Kapitel 5.1 wird die Architektur des gesamten Rahmenwerks, bestehend aus den verschiedenen Klienten- und Serveranwendungen, beschrieben. Kapitel 5.2 skizziert anschließend den generellen Ablauf einer Behandlung und dabei das Zusammenwirken der einzelnen Komponenten aus der Sicht des Therapeuten und Patienten. Zuletzt wird in Kapitel 5.3 die Datenstruktur der Serveranwendung vorgestellt.

5.1 Gesamtsystem

Das Gesamtsystem gliedert sich in vier miteinander kommunizierende Komponenten: die Patienten-Apps (Kapitel 5.1.1), die Web-Anwendungen für Therapeuten und Wissenschaftler(Kapitel 5.1.2), die Server-Anwendung (Kapitel 5.1.3) und die relationale Datenbank (Kapitel 5.1.4). Dabei gliedern sich die Komponenten in fünf unterschiedliche Schichten, die die Aufgabe des jeweiligen Bereichs darlegen. Die Präsentationsschicht enthält alle Benutzerschnittstellen für das System und ist mit Schnittstellenlogik, respektive mit Logik zur Erfassung von Sensor-Daten, ausgestattet. Die Interaktionsschicht regelt die Kommunikation der Anwendungen aus der Präsentationsschicht mit der Server-Anwendung. Die Geschäftslogik, die auf Domänenmodellen operiert, befindet sich in der Logikschicht und kümmert sich um den Datenfluss zwischen Therapeut und Patient. Die Datenhaltungsschicht bildet die Domänenmodelle auf die eigentlichen Persistenzmodellen ab, die vor der Persistierung zuerst validiert und danach mittels ORM-Framework in die relationale Datenbank übertragen werden. Diese relationale Datenbank befindet sich in der untersten Schicht,

der Persistenzschicht. Das Zusammenspiel dieser Schichten, sowie die einzelnen Bereiche der Komponenten, werden in Abbildung 5.1 illustriert.

Um das Rahmenwerk, aufgrund der hoch priorisierten nicht-funktionalen Anforderung **Portabilität** (Kapitel 4.4.2), unabhängig gegenüber Veränderungen der Präsentationstechnologie zu gestalten, werden die Anwendungen der Präsentationsschicht lose an die Server-Anwendung gekoppelt. Ein Webservice ist dabei die Grundlage für den Datenzugriff auf dem Server und wird per HTTP-Protokoll angesprochen. Der Webservice wird dabei nach dem Architekturstil *Representational State Transfer* (REST), beschrieben in der Dissertation von Roy Fielding [50], implementiert. Zusätzlich besitzt das System eine Schnittstelle für Administratoren, um Benutzer zu verwalten und Systemeinstellungen zu konfigurieren. Die Anwendung für diese Nutzergruppe basiert, wie auch die Anwendungen für Therapeuten und Wissenschaftler, auf der Kombination von HTML5 und Javascript. Da die Implementierung der Aufgaben eines Administrators nicht von den in anderen Systemen abweicht, werden diese in den folgenden Abschnitten nicht weiter ausgeführt.

5.1.1 Patienten-Apps

Die mobilen Applikationen für die Patienten des Systems werden für die zwei aktuell meist verbreitetsten mobilen Betriebssysteme *Android* und *iOS* entwickelt [51]. Durch das Entwickeln für die Marktführer, können die Patienten-Apps für die meisten Patienten zu Verfügung gestellt werden. Außerdem kann durch die native Entwicklung sichergestellt werden, dass alle Sensoren direkt angesprochen und Hintergrunddienste initialisiert werden können. Somit besitzen die Patienten-Apps nicht nur Präsentationsfunktionen, sondern zum Teil auch Funktionen der Logik. Ein weiterer positiver Aspekt von nativ implementierten Anwendungen ist, dass die Usability durch den gewohnten *Look & Feel* des Betriebssystems des Patienten gesteigert wird. Die Apps selbst werden innerhalb des MobileTx-Projekts [52] an der Universität Ulm entwickelt und implementieren die Sensor-Ansteuerung der Smartphones, um den Patienten bei der Ausführung seiner Hausaufgabe zu unterstützen und die angefallenen Ausführungsdaten dem System mitzuteilen. Zusätzlich wird ein Interface entwickelt, über das die Patienten

ihre Behandlungen innerhalb dieses Rahmenwerks verwalten und neue Behandlungen mittels QR-Code hinzufügen können. Befindet sich der Patient innerhalb einer Behandlung, kann er seine Hausaufgaben inklusive der multimedialen Inhalte für den Offline-Betrieb abrufen und starten. Dabei startet die Applikation einen Hintergrunddienst, der den Patienten benachrichtigt sobald eine Hausaufgabe ausgeführt werden soll. Hat der Patient eine Aufgabe ausgeführt, so erstellen die Patienten-Apps eine Abgabe für diese Hausaufgabe und integrieren alle relevanten Ausführungsdaten und -ereignisse. Diese Abgabe, sowie die restliche Kommunikation mit dem Server, wird über das HTTP-Protokoll an den Webservice gesendet, der die Daten verarbeitet.

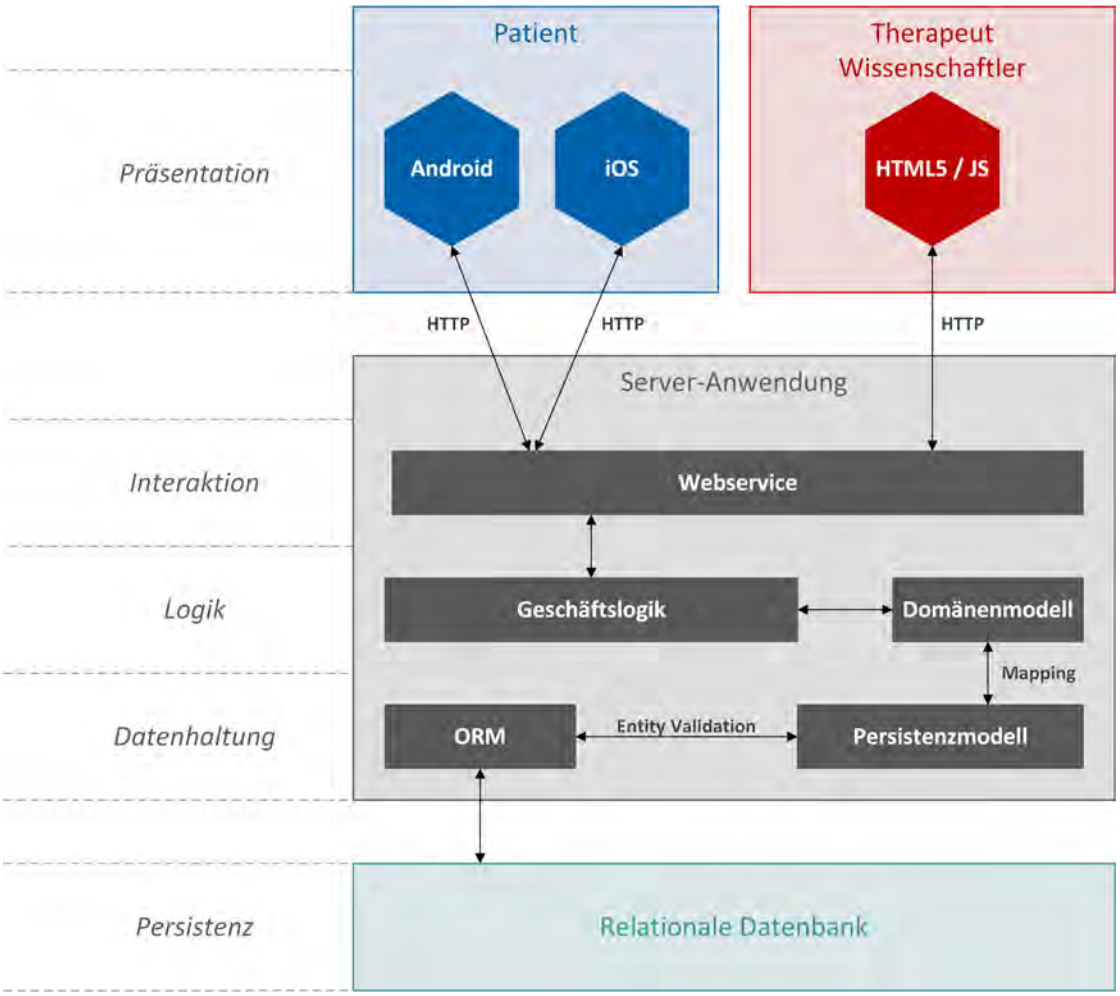


Abbildung 5.1: Architektur des Gesamtsystems

5.1.2 Web-Anwendungen für Therapeuten und Wissenschaftler

Um die Aufgabendurchführung der Patienten zu betreuen und Studiendaten auszuwerten, haben Therapeuten und Wissenschaftler eigene Anwendungen für den Zugriff auf das System. Die Entwicklung der Anwendungen basiert auf der Kombination von HTML5 und Javascript, da diese Anwendungen, im Gegensatz zu den Patienten-Apps, ausschließlich für die Präsentation von Informationen zuständig sind. Für die Aufbereitung von Informationen, insbesondere wenn diese von REST-Webservices bezogen werden, ist diese Kombination zuverlässig und bietet eine große Anzahl an Frameworks zu Applikationsentwicklung. Therapeuten und Wissenschaftler besitzen getrennte Anwendungen, die beide mit Daten aus dem selben Webservice gespeist werden. Die Trennung der Anwendungen besitzt den Vorteil, dass die Wartung einfacher und die Entwicklung weniger anfällig für Fehler ist. Dies ist besonders wichtig, da gerade Fehler bei der Autorisierung und Authentifizierung die nicht-funktionale Anforderung **Sicherheit/Datenschutz** (Kapitel 4.4.2) verletzen.

5.1.3 Server-Anwendung

Der Hauptteil des Systems, der die Geschäftslogik enthält, wird als Server-Anwendung implementiert. Die Anwendung beinhaltet einen Webservice, der die zentrale Schnittstelle darstellt, um mit den Anwendungen aus der Präsentationsschicht zu kommunizieren. Anfragen an den Webservice werden innerhalb der Interaktionsschicht verarbeitet, wobei die Authentizität des Anfragenden, sowie eine ausreichende Autorisierung, geprüft wird. Ist die Anfrage gültig, so wird sie an die Geschäftslogik weitergegeben. Andernfalls wird die Anfrage abgelehnt und nicht bearbeitet. Der Webservice verwendet für die Datenbereitstellung die Domänenmodelle der Anwendung, die er aus dem Repositorium der Geschäftslogik erhält. Gleichzeitig werden die Daten der Anfrage innerhalb der Geschäftslogik validiert, um nur gültige und vollständige Datensätze anzunehmen. Zusätzlich berechnet die Geschäftslogik alle Statistiken und Vergleiche, die der Therapeut für die Behandlungsauswertung benötigt. Gültige Datenmanipulationen leitet er über das Repositorium der Domänenmodelle die Datenhaltungsschicht weiter. Dabei werden die Domänenmodelle

auf die Persistenzmodelle abgebildet, welche die unterste Modell-Ebene ist. Die Persistenzmodelle bilden das Schema für die Relationale Datenbank und werden vor dem eintragen nochmals validiert. Die Validation in der Geschäftslogik und der Validation der Entität unterscheidet sich dahingehend, dass in der Datenhaltung auf Ebene der primitiven Datentypen, in der Geschäftslogik jedoch die Domänenmodelle validiert werden. Ein ORM-Framework kümmert sich nach der Validation um die Persistierung, indem es die Daten in eine relationale Datenbank überträgt.

5.1.4 Relationale Datenbank

Als Mittel zur Persistierung der Systemdaten wird eine relationale Datenbank verwendet, welche getrennt von der Server-Anwendung betrieben und mittel ORM-Framework angesprochen wird. Die lose Kopplung von Server-Anwendung und der Datenbank wurde in Folge der nicht-funktionalen Anforderungen (Kapitel 4.4.2) gewählt, da dadurch die **Portabilität** erhöht und durch eine Trennung der Ressourcenverbrauch aufgeteilt und somit die Effizienz gesteigert werden kann.

5.2 Genereller Ablauf

Innerhalb des Rahmenwerks bildet der Server die zentrale Komponente zur Speicherung und Verteilung der Daten. In Abbildung 5.2 wird dies anhand des Anwendungsfalls Psychotherapie demonstriert. Zu Beginn des Prozesses erstellt der Therapeut in der Therapeuten-Anwendung eine Hausaufgabe, die durch multimediale Inhalte angereichert sein kann. Die erstellte Hausaufgabe übertragt er an den Webservice des Systems, welcher die Daten überprüft und in einer Datenbank persistiert. Jede Hausaufgabe des Therapeuten kann über seine Anwendung verwaltet und statistisch aufbereitet angezeigt werden. Erstellte Hausaufgaben können anschließend vom Patienten abgerufen werden. Mittels einer mobilen Anwendung, die auf unterschiedlichen Endgeräten ausgeführt werden kann, erhält der Patient Unterstützung bei der Aufgabendurchführung, indem er beispielsweise an die Ausführung erinnert wird oder einen Hinweis bekommt, ob er sich im korrekten Ausführungskontext befindet. Während der Ausführungszyklus

der Hausaufgabe läuft werden durch die mobilen Applikationen Daten gesammelt, welche über den Server vom Patienten eingesehen werden können. Dadurch ist es dem Therapeuten möglich außerhalb der Therapiesitzung die Aufgabendurchführung zu überwachen und zu steuern. Zusätzlich können die in der Behandlung anfallenden Daten für wissenschaftliche Zwecke freigegeben werden. Wissenschaftler können zu Verfügung gestellte Datensätze analysieren und durch große Datenmengen statistisch relevante Erkenntnisse gewinnen. Diese neuen Erkenntnisse zur Behandlung von Patienten fließen anschließend, gegebenenfalls als effizientere Methode, in das System zurück und können von Therapeuten für ihre Behandlung verwendet werden. Patienten erhalten somit nicht nur Unterstützung bei der Ausführung, sondern können durch die Verwendung des Systems doppelt profitieren.

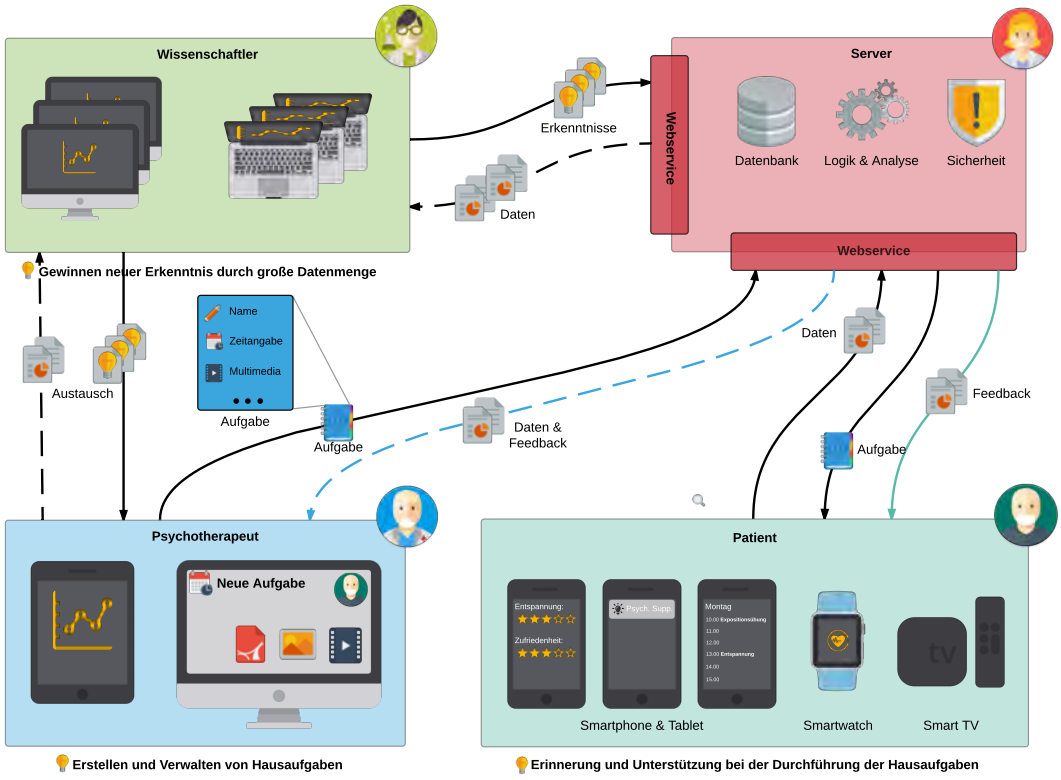


Abbildung 5.2: Überblick über das Gesamtsystem [52]

5.3 Datenstruktur

Dieses Rahmenwerk verwendet eine relationale Datenbank zur Persistierung von Entitäten. Ein ORM-Framework konvertiert dabei die Entitäten aus der relationalen Datenbank in manipulierbare Objekte der Datenhaltungsschicht (Kapitel 5.1.3). In Abbildung 5.3 wird eine Übersicht der Datenstruktur auf Persistenzmodell-Ebene gegeben, welche als UML2-Klassendiagramm notiert wurde. Die Werte der hellgrauen Enumerationen werden nicht in der relationalen Datenbank gespeichert, sondern nur die Position des jeweiligen Wertes in der Enumeration. Die Werte werden zur Laufzeit von der Laufzeitumgebung in die Persistenzmodelle eingefügt. Die folgenden Abschnitte beschreiben von links nach rechts die zur Abgrenzung farblich gekennzeichneten, verschiedenen Bereiche des Klassendiagramms.

5.3.1 Benutzer- und Systemdatenstruktur

Die in Abbildung 5.3 *schwarz* gekennzeichneten Klassen bilden die Benutzergruppen des Systems. Alle Benutzer erben von der abstrakten Klasse **Benutzer**, welche selbst nicht initialisierbar ist. Für jeden Benutzer wird somit ein Vor- und Nachname, die Email-Adresse und das Land, in dem er sich befindet, gespeichert. Die Email-Adresse ist dabei besonders wichtig, da diese als eindeutiger Identifikator eines Benutzers im System dient und für die Einladungsfunktion benötigt wird. Die Angabe des Landes ist aufgrund rechtlicher Aspekte notwendig, da initial ein System für die deutsche Rechtsordnung entwickelt wird. Anhand der Spracheinstellung des Benutzer kann kein Rückschluss gezogen werden, da beispielsweise innerhalb des DACH-Gebiets drei Rechtsordnung existieren. Zusätzlich werden für den Therapeuten Adress- und Geburtsdaten des Patienten erhoben, wobei das Geburtsdatum auch für Studien, falls der Patientendatensatz freigegeben wurde, von Bedeutung ist. Um sich selbst besser im Marktplatz vorstellen zu können, haben Therapeuten die Möglichkeit ein Bild und ein Beschreibung von sich selbst anzugeben, wobei diese Daten eine der angegeben *MimeTypes* entsprechen muss. Über das Patientenkontingent kann der Therapeut angeben, wie viel freie Behandlungsplätze er besitzt.

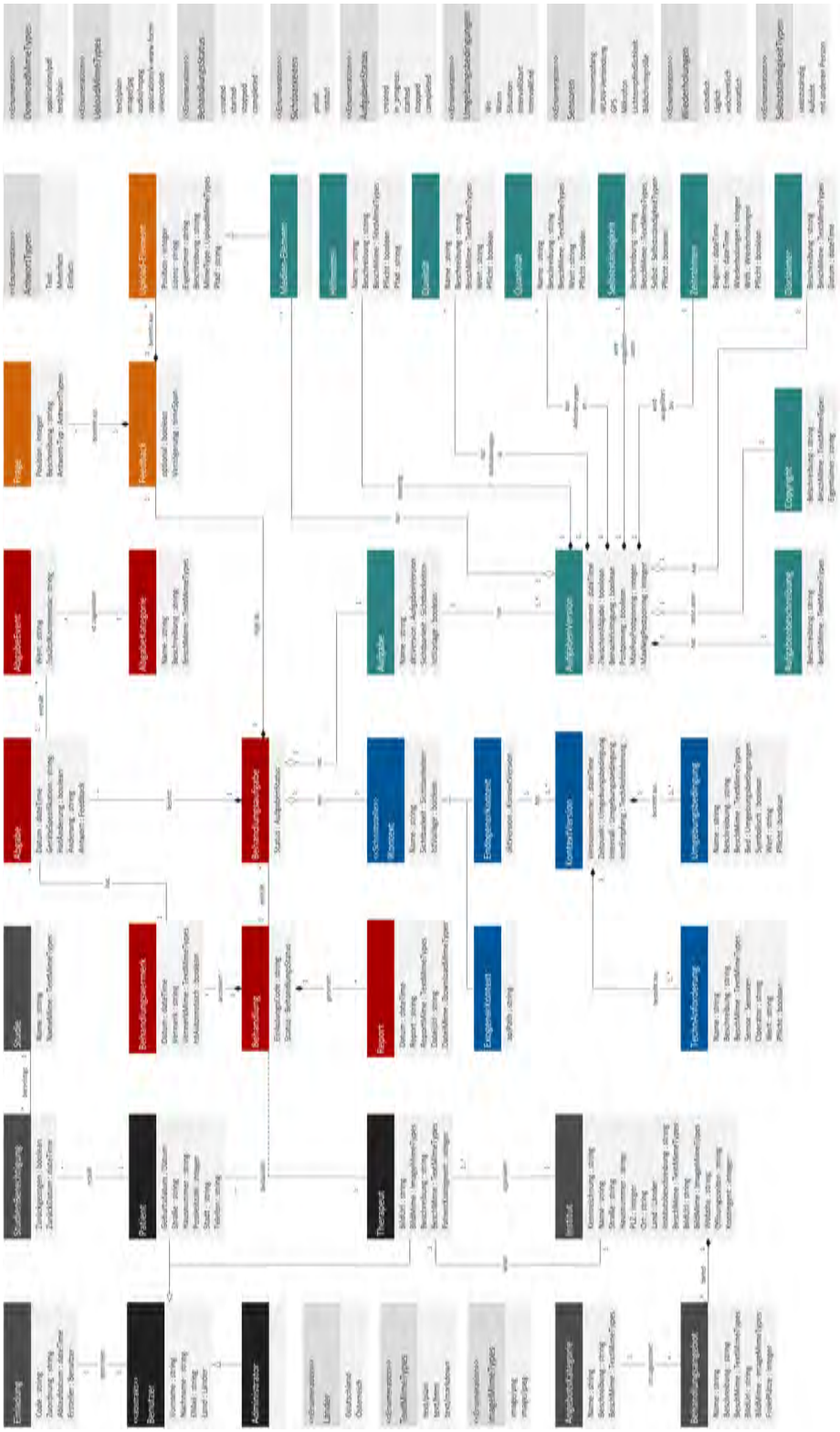


Abbildung 5.3: Datenstruktur der Server-Anwendung

Die in Abbildung 5.3 *grau* hinterlegten Klassen bildet die Struktur für Daten, die zur Verwaltung von Systemdiensten benötigt werden. Dazu zählen die Einladungen, die Patienten zur Registrierung und Behandlung im System mittels eines eindeutigen Codes autorisieren. Eine **Einladung** kann außerdem für einen Therapeut ausgesprochen werden. Um eine eindeutige Zuordnung zu gewährleisten, muss die Zuordnung und der Ersteller der Einladung zusätzlich vermerkt werden. Über ein Ablaufdatum können Einladungen automatisch invalidiert werden, sobald dieses Datum erreicht wurde. Zu den Systemdiensten zählt auch die Erstellung von Studien, für die Patienten ihren gesamten Datensatz zu Verfügung stellen können. Dafür muss eine Studienberechtigung erstellt werden, die jederzeit vom Patienten zurückgezogen werden kann. Für den Betrieb des Marktplatzes ist es notwendig, dass sich Therapeuten innerhalb eines Instituts organisieren und ihr Behandlungsangebot beschreiben können. Aufgrund dessen können Institute angelegt werden, welche das Speichern von Adressdaten, Beschreibung und Bilder eines gültigen *MimeTypes*, Öffnungszeichen und ein eindeutiges Kennzeichen für die Zuordnung von Einladungen ermöglichen. Ein Kontingent für Behandlungen kann vom Administrator vergeben werden. Außerdem können die Behandlungsangebote des Instituts mit ausführlichen Leistungsbeschreibungen angereichert werden. Jedes Behandlungsangebot muss zusätzlich eine Angebotskategorie untergeordnet werden, um den Benutzer bei der Suche innerhalb des Marktplatzes zu unterstützen.

5.3.2 Behandlungsdatenstruktur

Klassen, welche direkt in Beziehung zur Behandlung stehen, sind in Abbildung 5.3 *rot* gekennzeichnet. Mit der Assoziationsklasse **Behandlung** wird eine Beziehung zwischen Patient und Therapeut erstellt. Somit kann ein Patient bei mehreren Therapeuten in Behandlung sein. Um eine Behandlung zu erstellen, muss ein gültiger Einladungscode hinterlegt sein, der vom Patienten zur Zuordnung angegeben wird. Weiterhin besitzt eine Behandlung einen Status, um Behandlungen zu schließen und somit automatisch alle Behandlungsaufgaben zu beenden. Es können beliebig viele Behandlungsaufgaben einer Behandlung hinzugefügt werden, wobei jede Behandlungsaufgabe einen weiteren Status beinhaltet, welche das Abrufen und Speichern von Abgaben steuert.

Zu jeder Behandlungsaufgabe können Abgaben gespeichert werden, welche das Abgabedatum, die Gerätespezifikation des Patienten und optional einen textuell gestellten Änderungswunsch und ein Feedback enthalten kann. Weiterhin können Abgaben mit den Ereignissen der Aufgabenausführung angereichert werden, welche einen Wert und einen Kommentar speichern können. Um das Ereignis zu kategorisieren, werden alle Ereignisse einer Kategorie zugeteilt. Erstellte Abgaben erzeugen einen automatischen Behandlungsvermerk, der der aktuellen Behandlung untergeordnet ist. Behandlungsvermerke besitzen ein Erstelldatum, den Vermerk selbst und dessen *MimeType*, sowie eine Angabe ob der Vermerk automatisch oder manuell erstellt wurde. Für eine Behandlung können Therapeuten einen Report erstellten, der die Daten der Vermerke, Behandlungsaufgaben und Abgaben automatisiert zusammenfasst und entweder als Text oder Datei abspeichert. Ein Report enthält, falls nur als Text verfasst, zusätzlich zum Erstelldatum den Text und dessen *MimeType*. Falls eine Datei erzeugt wurde, wird deren URL und *MimeType* eingefügt.

5.3.3 Kontextdatenstruktur

Therapeuten können über ihre Anwendung **Kontexte** definieren und abspeichern. Die dafür benötigte Datenstruktur ist in Abbildung 5.3 *blau* hinterlegt. Es existieren zwei Arten von Kontexte, wobei beide die gemeinsamen Schnittstelle implementieren, welche die Sichtbarkeit innerhalb des Repositorien regelt. Außerdem muss für jeden Kontext ein Name und ein *Flag*[1] angelegt werden, wobei letzteres den Kontext als Vorlage markiert. Der exogene Kontext bezieht Daten von außerhalb des Systems und speichert deshalb nur den Pfad zur externen Datenquelle. Der endogene Kontext hingegen wird innerhalb des Systems definiert und zeigt auf die aktuelle Version dieses Kontextes. Kontext-Versionen werden mittels Zeitstempels versioniert und enthalten diverse technische Anforderungen und Umgebungsbedingungen. Verpflichtend sind die Angaben zu Zeitpunkt, zum (Ausführungs-) Intervall und zum benötigten Internetempfang, wobei die ersten zwei Angaben als Umgebungsbedingung und die letzte als technische Anforderung kodiert werden. Technische Anforderungen besitzen

[1] Hilfsmittel zur Kennzeichnung von Zuständen

einen Namen und eine Beschreibung mit dem entsprechenden *MimeType*. Außerdem muss ein Sensor angegeben werden, der die aktuellen Werte ausliest und diese Anforderung mittels des angegeben Operators den Wert vergleicht. Außerdem wird ein Pflichtfeld angegeben, dass der Patienten-App signalisieren soll, einen Hinweis anzuzeigen, falls der Sensor-Wert nicht mit den Anforderungen übereinstimmt. Das Pflichtfeld, sowie Name, Beschreibung und *MimeType*, existiert ebenfalls in den Umgebungsbedingungen. Zusätzlich kann eine Bedingung angegeben werden, sowie ein *Flag*, ob diese symbolisch ist oder von der Patient-App überprüft werden soll.

5.3.4 Aufgabendatenstruktur

Die in Abbildung 5.3 *grün* hinterlegten Klassen werden benötigt, um eine komplette Aufgabe mit dem System abzubilden. Die Klasse **Aufgabe** kann durch einen Namen identifiziert werden und enthält, vergleichbar mit dem Kontext, eine aktuelle Version der Aufgabe, eine Sichtbarkeitseinstellung für das Repositorium und ein *Flag*, um die Aufgabe als Vorlage zu kennzeichnen. Für eine Aufgabe kann es verschiedene Versionen geben, die bei Aktualisierungen der Beziehungsklassen erstellt werden. Eine Aufgaben-Version enthält einen Zeitstempel als Versionsnummer, sowie ein *Flag*, ob der Patient zum Ausführungszeitpunkt benachrichtigt werden soll. Zusätzlich wird über ein *Flag* angegeben, ob der Patient bei sich wiederholenden Aufgaben Zwischenabgaben senden soll und ob er die Ausführung verschieben darf. Falls der Patient die Aufgabe verschieben darf, so kann ein Intervall zum Verschieben der Ausführung durch einen maximalen, positiven und negativen Wert definiert werden.

Zusätzlich muss eine Aufgabenbeschreibung mit dem entsprechenden *MimeType* erstellt werden. Zur weiteren Verdeutlichung der Aufgabe kann ein Medien-Elemente angehängt werden, welches von der Klasse Upload-Element erbt und den Pfad, sowie weitere Angaben zu Lizenz und Eigentümer, speichert. Ein Medien-Element muss dabei einen gültigen *MimeType* und optional eine Beschreibung enthalten. Außerdem können Hilfsmittel definiert werden und ähnlich zu den folgenden Klassen mit Name, Beschreibung und den passenden *MimeType* zu Beschreibung versehen werden. Ein Hilfsmittel kann jedoch auch ein nicht-physisches Element sein, welches über

einen Pfad referenziert wird. Das Pflicht-*Flag* gibt zusätzlich an, ob die Zuhilfenahme des beschriebenen Objekts optional oder verpflichtend ist. Ähnlich zur Klasse Hilfsmittel sind die zwei Klassen Qualität und Quantität strukturiert, welche einen Name, eine Beschreibung mit *MimeType*, ein Feld für einen zu interpretierenden Wert und ein Pflicht-*Flag* beinhalten. Außerdem kann die Selbstständigkeit der Aufgabe definiert werden, indem eine Auswahl der *SelbstständigkeitTypen*-Enumeration getroffen wird und optional mittels Beschreibungstext erläutert wird. Das *Flag* zeigt ebenfalls an, ob die Selbstständigkeitsangabe verpflichtend ist. Weiterhin kann über eine Zeitrahmen-Definition mit einem Zeitstempel für Beginn und Ende ein Intervall gespannt werden, für welches die Anzahl und die Spanne zwischen den Wiederholungen angegeben wird. Dadurch kann die Patienten-App einen Zeitplan für die Aufgabendurchführung erstellen. Über ein Pflichtfeld kann wiederum eingetragen werden, ob die Ausführung innerhalb des Zeitrahmens verpflichtend oder optional ist.

Zuletzt muss für eine Aufgaben-Version ein Copyright mitgeliefert werden, welches die Copyright-Angabe selbst und falls dieser angegeben werden muss, den Eigentümer enthält. Auch ein mit Datum versehener Disclaimer muss angehängt werden, um rechtliche Zusatzinformationen geben zu können.

5.3.5 Feedbackdatenstruktur

Die in Abbildung 5.3 *orange* hinterlegten Klassen bilden die Datenstruktur für den Feedback-Mechanismus. Ein **Feedback** kann entweder optional oder verpflichtend abgefragt werden. Um diese Unterscheidung zu treffen, benötigt die Patienten-App das *Flag* optional. Gleichzeitig kann eine Zeitspanne gespeichert werden, welche die Verzögerung zwischen Beendigung der Aufgabe und Benachrichtigung zum Ausfüllen des Feedbacks angibt. Jedes Feedback kann beliebig viele Fragen enthalten, welche über eine Positionsangabe in der Patienten-App strukturiert werden. Die eigentliche Frage wird im Feld *Beschreibung* eingetragen und benötigt keinen *MimeType*, da die Patienten-App die Frage als Plain-Text erwartet. Zusätzlich muss der Antwort-Typ angegeben werden, welche aus der *AntwortTypen*-Enumeration entnommen wird. Zu einem Feedback können auch Upload-Elemente hinzugefügt werden, welche

ebenfalls eine Positionsangabe für die Patienten-App enthalten. Außerdem kann eine Beschreibung mit *MimeType* als Kommentar gespeichert werden, sowie die Lizenz und den Eigentümer dieses Elements.

6

Ausgewählte Implementierungsaspekte

In diesem Kapitel wird auf der Basis der in Kapitel 5 vorgestellten Architektur eine Server-Anwendung als zentrale Schnittstelle aller Klienten-Anwendungen entwickelt. Zuerst werden in Kapitel 6.1 die verwendeten Technologien vorgestellt und erläutert. Anschließend wird in Kapitel 6.2 der schematische Aufbau der Anwendung mit den beschriebenen Technologien skizziert und diskutiert. Die folgenden Abschnitte beschreiben die in Kapitel 6.2 vorgestellten Schichten nach dem Bottom-Up-Prinzip. Begonnen wird in Kapitel 6.3 mit der untersten Schicht: dem Data Access Layer. Dabei wird genauer auf den Aufbau der Persistenz-Datenstruktur und die Implementierung eines generischen Repositorien eingegangen. Anschließend werden in Kapitel 6.4 Aspekte der Business Logic vorgestellt. Zuletzt wird in Kapitel 6.5 der Webservice als oberste Schicht und Schnittstelle zu den Klienten-Anwendungen beschrieben.

6.1 Verwendete Technologien

In diesem Kapitel werden die, zum Aufbau einer Server-Anwendung, verwendeten Technologien vorgestellt und anhand von Beispielen erläutert. Der komplette technologische Stack des Rahmenwerks baut auf dem .NET-Framework von Microsoft auf, mit dem Anwendungen größtenteils auf Basis von Windows-Diensten entwickelt und ausgeführt werden können. Die Entscheidung zum Einsatz von Microsoft-Technologien fußt grundsätzlich auf der ausführlichen Dokumentation und dem reichhaltigen Ökosystem um das .NET-Framework. Eine gute Dokumentation der selbst entwickelten Software, aber im gleichen Maße der verwendeten Software, ist elementar, um die nicht-funktionale Anforderung **Wartbarkeit** (Kapitel 4.4.2) zu erfüllen. Aus den

```
1 GET /repo/exercise/42 HTTP/1.1
2 Host: www.musteradresse.de
3 Accept: application/json; charset=utf-8
4 X-Requested-With: XMLHttpRequest
5 Authorization: Bearer A4xGkbfXnlGryZ6zYRjL5WZ
```

Listing 6.1: Anfrage an einen RESTful-Service

Erfahrungen mit weiteren mHealth-Diensten (Kapitel 3) ist zudem anzumerken, dass die Entwicklung mit einer typsicheren Programmiersprache wie C#, die standardmäßig zur Entwicklung im .NET-Framework verwendet wird, zur Erfüllung der nicht-funktionalen Anforderung **Robustheit** (Kapitel 4.4.2) beiträgt. C# ist eine von Anders Hejlsberg für Microsoft entwickelte objektorientierte, multiparadimatische Programmiersprache [53], die vor allem von der populären Programmiersprache Java beeinflusst wurde.

6.1.1 ASP.NET Web Api

Das Framework ASP.NET von Microsoft ist eine beliebte serverseitige Technologie, um komplexe und gleichzeitig effiziente Web-Anwendungen zu erstellten. Hierzu stellt es Werkzeuge für Aufgaben zu Verfügung, die in der Regel in allen Web-Anwendungen benötigt werden und ist somit selbst keine Programmiersprache. Um Software mit ASP.NET zu programmieren, muss man auf Programmiersprachen des .NET-Frameworks, wie beispielsweise C#, zurückgreifen. ASP.NET ist Teil des .NET-Frameworks und besitzt seit 2015 mit ASP.NET Core eine quelloffene Variante, die sowohl auf Linux, als auch auf Mac OS unterstützt.

Für das ASP.NET-Framework gibt es einige Erweiterungen, wie beispielsweise ASP.NET Web API, womit Webservices auf Basis des HTTP-Protokolls erstellt werden können. Mithilfe des HTTP-Protokolls können zusätzlich sogenannte *RESTful Services* erstellt werden, welche oft nur teilweise nach dem Architekturstil REST aufgebaut sind. Dieser Architekturstil für ressourcen-zentrierte Dienste wurde erstmals in Roy Fieldings Dissertation [50] beschrieben und ist streng genommen nicht auf das HTTP-Protokoll beschränkt. Mit dem REST-Paradigma können Ressourcen über eine eindeutige URL erreicht und über die HTTP-Verben beeinflusst werden. Ein Beispiel für eine gültige URL eines RESTful Service findet sich in Listing 6.1 Zeile 01.

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Server: Microsoft-IIS/8.0
Date: Mon, 17 Oct 2016 13:37:00 GMT
Content-Length: "200"

{
  "Name": "Entspannungsübung mit Musik",
  "IsTemplate": false,
  "Links": [ {
    "Rel": "self",
    "Href": "repo/exercise/42"
  } ]
}
```

Listing 6.2: Antwort eines RESTful-Service mit HATEOAS

Zusätzlich müssen *RESTful Services* zustandslos sein, dürfen also nicht den Sitzungszustandes des Benutzers kennen. Aus dieser Dienst-Eigenschaft bildet sich jedoch der Vorteil heraus, dass *RESTful Services* extrem skalierbar sind und somit mehrere Dienst-Instanzen gleichzeitig laufen können. Aufgrund der Zustandslosigkeit können außerdem Abfrage-Ergebnisse im Cache zwischengespeichert werden, da alle Anfragen über das GET-Verb idempotent sind. Mittels Content Negotiation, illustriert in Listing 6.1 Zeile 03, ist es weiterhin möglich das Format der Antwort, falls der angesprochene Dienst dieses Format anbietet, zu bestimmen.

Durch das Implementieren eines *RESTful-Service* kann darüber hinaus die nicht-funktionale Anforderung **Portabilität** (Kapitel 4.4.2) erfüllt werden, da über diese Architektur die Klienten nur schwach an die Server-Infrastruktur gekoppelt ist. Falls ein *RESTful Service* die sogenannte *HATEOAS*[1] Randbedingung implementiert, kann der Klient innerhalb der Server-Anwendung durch die Datenstruktur navigieren. In Listing 6.2 wird die verkürzte Antwort eines *HATEOAS*-implementierenden Servers zu der Anfrage aus Listing 6.1 demonstriert.

Anwendungen mit ASP.NET Web API müssen außerdem MVC-Implementierung von ASP.NET beinhalten, da der Anwendungscode das MVC-Pattern[2] zur Strukturierung des eigenen Programmcodes verwendet. Einzelne Ressourcen werden als Controller abgebildet und beinhalten alle Operationen auf dieser Ressource. Alle Controller einer

[1] Akronym für Hypermedia as the engine of application state
[2] Akronym für Model-View-Controller

```
1 public class ExerciseController : ApiController
2 {
3   [HttpGet]
4   [Route("repo/exercise/{exerciseId:int:min(100)}")]
5   [AsyncTimeout(3000)]
6   public async Task<IHttpActionResult> FindExerciseById(int exerciseId) { ... }
7 }
```

Listing 6.3: Ausschnitt eines ASP.NET Web API Controllers

Anwendung in ASP.NET Web API müssen von der Klasse **ApiController** erben, die vom Framework vorgegeben wird. Über zusätzliche Annotation der Methodendeklaration können die Parameter des Aufrufs, beispielsweise nur Aufrufe mit dem HTTP-Verb *GET* (Listing 6.3, Zeile 03) und die Eigenschaften der Ausführung, beispielsweise einen Timeout nach 3 Sekunden Nichtbeantwortung zu senden (Listing 6.3, Zeile 05), beschrieben werden. Das Timeout-Attribut verlangt jedoch eine asynchrone Deklaration der Controller-Methode, wofür das Schlüsselwort **async** angegeben und der Rückgabewert ein **Task** mit Ergebnistyp als *Generic* sein muss. Über annotierte Methoden kann ebenfalls das Routing der Anwendung definiert werden. Über die Annotation Route() kann eine Route definiert werden, unter der man diese Methode aufrufen kann. Zusätzlich kann diese Route über Platzhalter eingeschränkt werden, die vor der Ausführung durch das Framework überprüft werden. In Listing 6.3 Zeile 04 wird eine solche Routen-Definition beschrieben.

6.1.2 Entity Framework

Zur Persistierung von Daten wird in diesem Rahmenwerk das Entity Framework verwendet. Das Entity Framework ist ein OR-Mapper (ORM) von Microsoft, das innerhalb von .NET für objektrelationale Abbildungen verwendet wird. Dies erleichtert den Umgang mit Datenbanksystemen, da die meisten Systeme in objektorientierten Programmiersprachen verfasst sind und man durch den Einsatz von ORMs die Datenbank über gewohnte Objektstrukturen manipulieren kann. Dazu bildet ein ORM wie das Entity Framework Klassen, Attribute, Vererbungen und weitere Assoziationen auf entsprechende Konstrukte der relationalen Welt ab, wie Tabellen, Spalten, Diskriminatoren und Fremdschlüssel. Weiterhin können sich wiederholende

```
01 public class ApplicationContext : DbContext
02 {
03   public ApplicationContext() : base("name=DatabaseConnection") { }
04
05   // Treatment
06   public virtual DbSet<Treatment> Treatments { get; set; }
07   public virtual DbSet<TreatmentExercise> TreatmentExercises { get; set; }
08   public virtual DbSet<TreatmentNote> TreatmentsNotes { get; set; }
09   public virtual DbSet<TreatmentReport> TreatmentReports { get; set; }
10
11   // Studies
12   public virtual DbSet<Study> Studies { get; set; }
13   public virtual DbSet<StudyPermission> StudyPermissions { get; set; }
14
15   protected override void OnModelCreating(DbModelBuilder modelBuilder)
16   {
17     base.OnModelCreating(modelBuilder);
18
19     modelBuilder.Ignore<EntityBase>();       // ignore abstract class
20
21     // Override Conventions
22     modelBuilder.Conventions.Remove<OneToManyCascadeDeleteConvention>();
23     modelBuilder.Conventions.Remove<ManyToManyCascadeDeleteConvention>();
24
25     // Override Configuration
26     modelBuilder.Configurations.Add(new TreatmentConfiguration());
27     modelBuilder.Configurations.Add(new PermissionConfiguration());
28   }
29 }
```

Listing 6.4: Ausschnitt des Datenbank Kontextes mit Konventions- und Konfigurationsänderungen

Programmiertätigkeiten, wie das Abfragen, Einfügen, Manipulieren oder Löschen von Daten aus der Datenbank, an das Framework abgegeben werden.

Das Entity Framework verwaltet zusammengehörende Datenstrukturen innerhalb eines **DbContext**, welcher für die Interaktion und Konvertierung mit Daten-Objekten zuständig ist und diese in die Datenbank überführt. Alle Persistenzmodelle, respektive die abbildenden Entitätsklassen, müssen deshalb als **DbSet<TEntity>** mit der Entitätsklasse als *Generic* innerhalb des Kontextes registriert werden (Listing 6.4, Zeile 05-13). Außerdem konvertiert der Kontext Anfragen zu SQL-Queries und registriert Änderungen an Daten-Objekten. Letzteres ermöglicht dem Kontext zusätzlich das Cachen von Daten-Objekten.

Weiterhin kann über den *DbContext* Einfluss auf die Erstellung und Konvertierung von Modellen genommen werden. Diese werden, insbesondere bei Verwendung des Code First Programmiermodells, standardmäßig nach vordefinierten Konventionen erstellt,

welche selbst deklariert, hinzugefügt oder gelöscht (Listing 6.4, Zeile 21-23) werden können.

Zu Beginn eines Projektes mit dem Entity Framework muss man angeben unter welchem Programmiermodell man entwickeln möchte. Es existieren drei verschiedene Programmiermodelle: *Code First*, *Model First* und *Database First*. Für dieses Rahmenwerk wurde das Modell Code First gewählt, welches im folgenden Abschnitt erläutert wird.

Code First

Durch das Code First Programmiermodell ist **Domain Driven Design** möglich, sodass Entwickler nicht mehr im Voraus die Datenbank entwerfen müssen, um im Anschluss ihre Domänenmodelle darauf abzubilden. Außerdem können Entwickler, falls sich Anforderungen ändern, ihr Datenbankschema unkompliziert modifizieren, indem sie die Domänenmodelle erstellen, ändern oder entfernen (Listing 6.5). Im Falle dieses Rahmenwerks werden die Persistenzmodelle abgebildet, welche anschließend in der Geschäftslogik zu den Domänenmodellen zusammengesetzt werden. Um den Entwicklungsprozess zu beschleunigen, verwendet das Entity Framework im Umgang mit der Datenbank standardmäßig Konventionen, welche über Annotationen an den Modellen (Listing 6.5, Zeile 03-05 und Listing 6.7, Zeile 03-36) oder per **FluentAPI** (Listing 6.5, Zeile 27-30) konfiguriert werden können. Konfigurationen per *FluentAPI* können aus dem *DbContext* in Konfigurationsklassen (Listing 6.5, ab Zeile 23), die vom Typ **EntityTypeConfiguration<TEntity>** erben müssen, ausgelagert werden. Anschließend müssen diese zum Datenbank Kontext hinzugefügt werden (Listing 6.4, Zeile 25-27)

Da relationale Datenbanken in der Regel das Konzept der Vererbung nicht kennen, besitzt das Entity Framework drei verschiedene Vererbungsstrategien: **Table per Hierarchy** (TPH), **Table per Type** (TPT) und **Table per Concrete Class** (TPC). Im Regelfall wird die TPH-Strategie verwendet, welche eine einzige Tabelle für die komplette Hierarchie anlegt und über eine Diskriminator-Spalte zwischen den Klassen unterscheidet. Die TPT-Strategie wird im Gegensatz dazu für jede Klasse eine eigene

```
1  public class StudyPermission : EntityBase
2  {
3    [Key]
4    [Column(Order = 0)]
5    [DatabaseGenerated(DatabaseGeneratedOption.None)]
6    public int StudyId { get; set; }
7
8    [ForeignKey("StudyId")]
9    public virtual Study Study { get; set; }
10
11   [Key]
12   [Column(Order = 1)]
13   [DatabaseGenerated(DatabaseGeneratedOption.None)]
14   public int PatientId { get; set; }
15
16   [ForeignKey("PatientId")]
17   public virtual Patient Patient { get; set; }
18
19   [Required]
20   public bool PermissionRevoked { get; set; }
21 }
22
23 public class PermissionConfiguration : EntityTypeConfiguration<StudyPermission>
24 {
25   public PermissionConfiguration()
26   {
27     // ignore inherited key
28     this.Ignore(sp => sp.Id);
29
30     this.HasKey(t => new { t.StudyId, t.PatientId });
31   }
32 }
```

Listing 6.5: Modell-Erstellung und -Konfiguration mit dem Entity Framework

Tabelle erstellen, wobei Attribute aus Superklassen nicht wiederholt werden. Möchte man diese Wiederholung im Datenbankschema, so wählt man die TPC-Strategie. Bei dieser Arbeit wird die standardmäßige TPH-Strategie verwendet, jedoch innerhalb des Datenbank Kontext so modifiziert, dass es die Superklasse **EntityBase** (Abbildung 6.2), zuständig für grundsätzliche Meta-Attribute aller Persistenzmodelle, selbst nicht erstellt (Listing 6.4, Zeile 19). Somit gilt für den Fall *EntityBase* die TPC-Strategie, ansonsten weiterhin die TPH-Strategie.

Migrationen

Bei der Entwicklung mit dem Code First Programmiermodell ist es außerdem möglich das Datenbankschema über automatische oder sogenannte **Code-Based Migrations**

```
public partial class AddPermission : DbMigration
{
  public override void Up()
  {
    CreateTable(
    "dbo.StudyPermissions",
    c => new
    {
      StudyId = c.Int(nullable: false),
      PatientId = c.Int(nullable: false),
      PermissionRevoked = c.Boolean(nullable: false),
      CreatedAt = c.DateTime(nullable: false),
      CreatedBy = c.Int(nullable: false),
      UpdatedAt = c.DateTime(),
      UpdatedBy = c.Int(nullable: false),
      DeletedAt = c.DateTime(),
      DeletedBy = c.Int(nullable: false),
      RowVersion = c.Binary(nullable: false,
      fixedLength: true, timestamp: true, storeType: "rowversion"),
    })
    .PrimaryKey(t => new { t.StudyId, t.PatientId })
    .ForeignKey("dbo.Patients", t => t.PatientId)
    .ForeignKey("dbo.Studies", t => t.StudyId)
    .Index(t => t.StudyId)
    .Index(t => t.PatientId);
  }

  public override void Down()
  {
    DropForeignKey("dbo.StudyPermissions", "StudyId", "dbo.Studies");
    DropForeignKey("dbo.StudyPermissions", "PatientId", "dbo.Patients");
    DropIndex("dbo.StudyPermissions", new[] { "PatientId" });
    DropIndex("dbo.StudyPermissions", new[] { "StudyId" });
    DropTable("dbo.StudyPermissions");
  }
}
```

Listing 6.6: Ausschnitt einer Code-Based Migration

(Listing 6.6) zu steuern. Letztere Methodik wird für die Entwicklung innerhalb dieser Arbeit verwendet, da es dem Entwickler mehr Einfluss auf das Datenbankschema bietet. Eine Migration des Datenbankschemas bedeutet, dass inkrementelle Änderungen am Schema mit einer Versionsnummer versehen und festgehalten werden. Dadurch kann das Datenbankschemas jederzeit erweitert oder auf einen früheren Stand zurückgesetzt werden. Wird eine Migration auf die Datenbank angewendet, so wird die *Seed*-Methode innerhalb der Migration-Konfigurationsdatei aufgerufen, die die Datenbank automatisch mit Daten, beispielsweise für Tests oder initiale Systemeinstellungen befüllt.

6.1.3 ASP.NET Identity

Zur Authentifizierung und Autorisierung von Benutzern wird das ASP.NET Identity Framework von Microsoft verwendet. Dieses Framework bietet zahlreiche Komponenten zur Sicherung von Anwendungen, unter anderem die Unterstützung der OWIN Middleware und um sich mithilfe von Drittanbietern zu authentifizieren. Zur Autorisierung bietet das ASP.NET Identity Framework einen Benutzerrollen-Provider, mit welchen man beispielsweise innerhalb eines ASP.NET Web API Controllers Ressourcen vor unautorisierten Zugriff schützen kann. Standardmäßig erstellt ASP.NET Identity Tabellen für **Anmeldedaten**, **Claims**, **Logins** und **Rollen** erstellt. *Claims* sind Name/Wert-Paare, die den Benutzer näher beschreiben. Somit gewinnt man neben Nutzername und Passwort weitere Identifikatoren, welche zusätzlich als Sicherheitsabfragen verwendet werden können. Unter *Logins* werden Daten zur Authentifizierung über eine Drittanbieter API, wie beispielsweise Google oder Facebook, hinterlegt. Zur besseren Integration in die eigene Programmierung und um Anpassungen zu erleichtern, gibt es zusätzlich eine Variante, die auf das Entity Framework (Kapitel 6.1.2) im Code First Programmiermodell aufbaut.

6.2 Schematischer Aufbau

Die Implementierung des Rahmenwerk kann in acht unabhängige Komponenten aufgeteilt werden, deren Aufbau und Kommunikation untereinander in Abbildung 6.1 illustriert wird. Dieser Aufbau wurde gewählt, um einzelne Dienste bei Bedarf auszutauschen, ohne den Kern der Anwendung (Abbildung 6.1, links) stark zu beeinflussen. So könnte die Speicherung der großen multimedialen Daten von einem Cloud-Dienstanbieter übernommen werden oder der SQL Server auf mehrere Knoten verteilt werden. Eine schwache Kopplung der Komponenten hat nicht nur den Vorteil der Austauschbarkeit der Dienste, zusätzlich werden weitere nicht-funktionale Anforderungen (Kapitel 4.4.2) erfüllt. So kann die **Effizienz** gesteigert werden, indem man einzelne Komponenten, wie beispielsweise die Authentifizierung, auf einem anderen Server ausführt, um die Last zu verteilen. Gleichzeitig ist der Webservice des Systemkerns

selbst eine Komponente, die auf mehreren Knoten lauffähig ist. Systeme, die sich replizieren und skalieren lassen, steigern weiterhin die **Erreichbarkeit** und erfüllen somit die nicht-funktionale Anforderung zwei (Kapitel 4.4.2). Um den Aufbau weiter zu verdeutlichen und die Vorteile dieses Aufbaus hervorzuheben, skizzieren die folgenden Abschnitte den Aufbau und Zusammenhang der einzelnen Komponenten.

Die tiefste Ebene des technologischen Aufbaus persistiert alle anfallenden Daten des Rahmenwerks. Für die Persistierung wird eine relationale Datenbank, genauer der *SQL Server* von Microsoft, verwendet. Dieser wird standardmäßig vom Entity Framework unterstützt und kann somit von verschiedenen Diensten auf die gleiche Weise angesprochen werden, um Daten zu speichern. Dazu zählen unter anderem die Anwendungsdaten aus dem *Data Access Layer* (Kapitel 6.3), welche als Persistenzmodelle in Abbildung 5.3 dargestellt werden.

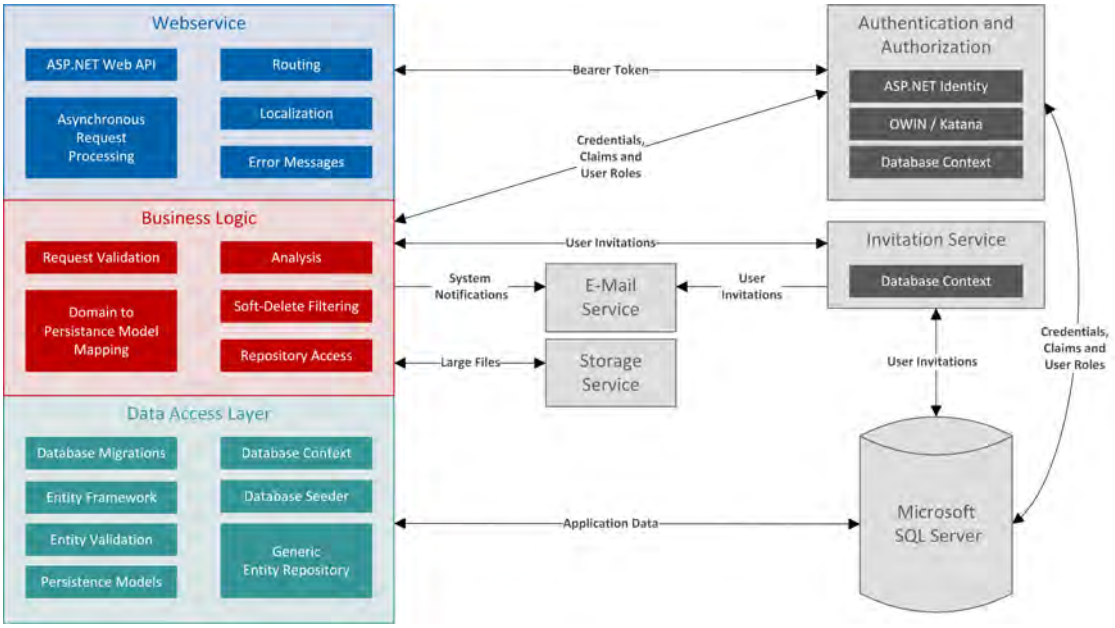


Abbildung 6.1: Schematischer Aufbau der Implementierung

Um die benötigte Struktur für die Modelle anzulegen, kommen innerhalb des **Data Access Layer** (DAL) Migrationen zum Einsatz, welche das Schema der Datenbank erstellen und bei Modelländerungen manipulieren. Weiterhin werden die Anmeldedaten des Administrators über den *Seed*-Mechanismus eingefügt, um initial mit dem System

zu arbeiten. Ist das Datenbankschema erstellt, so können die Persistenzmodelle über den Datenbank Kontext abgerufen und gespeichert werden. Möchten andere Komponenten Daten aus dem DAL abrufen oder manipulieren, so müssen sie das generische Repositorium verwenden, welches anschließend den Datenbank Kontext anspricht. Vor der Weitergabe an die Datenbank werden die Modelle innerhalb des DAL validiert. Scheitert die Validation, so werden die entsprechenden Fehlermeldungen an die darüber liegenden Schichten weitergereicht.

Daten und Fehlermeldungen aus dem DAL kommen über das Repositorium in die **Business Logic**. Da der Benutzer nicht auf Persistenz-, sondern auf Domänenmodellen arbeitet, werden diese zuerst innerhalb der *Business Logic* abgebildet. Bei jedem Zugriff auf das Repositorium wird ein **Soft-Delete**-Filter angewendet, der Entitäten mit einem gesetzten Lösch-Zeitstempel aus der Ergebnismenge entfernen. Wird der *Business Logic* eine valide Anfrage mit multimedialen Daten weitergeleitet, so werden diese vom Rest der Anfrage getrennt und zuerst dem Storage Service übergeben. Dieser speichert die Daten innerhalb eines Dateisystems und übergibt einen Pfad, an dem die Daten zu erreichen sind. Dieser Pfad wird anschließend mit den restlichen Anfragedaten in die Persistenzmodelle geschrieben. Wird eine neue Behandlung für einen Patienten oder ein Therapeut in ein Institut eingeladen, so erstellt die *Business Logic* eine Einladung und teilt diese mit dem **Invitation Service**. Dieser Service hinterlegt alle notwendigen Daten, wie beispielsweise das Ablaufdatum für diese Einladung, über seinen Kontext in der Datenbank. Ist die Einladung valide, so wird dies der *Business Logic* mitgeteilt und gleichzeitig der **E-Mail Service** mit dem Versand einer Einladung an den Patienten beauftragt. Registriert sich ein Benutzer mit einer validen Einladung im System, so werden seine Daten und die vergebene Rolle dem **Authentifizierungs- und Autorisierungsdienst** mitgeteilt, welcher anschließend diese in der Datenbank hinterlegt. Bei einer erfolgreichen Registrierung wird der Benutzer anschließend über den *E-Mail Service* benachrichtigt. Der *E-Mail Service* als zentraler Benachrichtigungsdienst versendet gleichzeitig auch weitere Benachrichtigungen, wie beispielsweise bei neu erstellten Behandlungsaufgaben.

Der **Webservice** ist die zentrale Schnittstelle zu den Klienten-Anwendungen und ist somit bei hoher Auslastung ein möglicher Flaschenhals des Systems. Da der

Webservice auch für die Authentifizierung und Autorisierung zuständig ist, müssen Anfragen asynchron bearbeitet werden können. Bei jeder Anfrage auf eine restriktive Funktion die an den *Webservice* gestellt wird, verlangt dieser ein gültiges **Bearer Token**. Um einen *Bearer Token* (Listing 6.1, Zeile 05) zu erhalten, muss der Anfragende dem *Webservice* seine Anmeldedaten zusenden. Dieser leitet die Anmeldedaten an den *Authentifizierungs- und Autorisierungsdienst* weiter, welcher über in der Datenbank hinterlegte Anmeldedaten und *Claims* den Anfragenden authentifiziert und ein *Bearer Token* erstellt. Dieses Token kann bis zum Ablaufdatum als gültige Authentifizierung beim *Webservice* verwendet werden. Zusätzlich wird die Autorisierung des Anfragenden beim *Authentifizierungs- und Autorisierungsdienst* abgeprüft, indem die entsprechende Rolle für die restriktive Routen mit der hinterlegten verglichen wird. Der *Webservice*, als Komponente der Präsentationsschicht, gibt Ergebnisse und Fehlermeldungen aus und hinterlegt Übersetzungsdateien für den mehrsprachigen Betrieb.

6.3 Data Access Layer

Um den Zugriff auf die Datenstruktur des Systems zu vereinheitlichen, wird ein *Data Access Layer* eingefügt, der den Datenzugriff und das Schema der Datenbank verwaltet. Das bedeutet, dass der DAL die Datenstrukturen für die Persistierung und Validierung auf unterster Ebene bereitstellt und über ein Repositorium für einheitlichen Zugang bietet. Über das Einbinden des Entity Frameworks (Kapitel 6.1.2) nach dem Code First Programmiermodell können die Persistenzmodelle als Datenklassen erstellt werden. Persistenzmodelle können somit wie normale **POCO**[3]-Objekte verwaltet und behandelt werden. Damit das Entity Framework die Datenklassen verwaltet, müssen sie, zusammen mit ihrer Konfiguration, zuvor im Datenbank Kontext registriert werden.

6.3.1 Basisklasse für Entitäten

Um ein einheitliches Datenbankschema zu erstellen, welches für jede Entität vordefinierte Meta-Attribute beinhaltet, wird eine Basisklasse erstellt. Zusätzlich wird

[3] Akronym für Plain-old CLR objects

sichergestellt, dass das generische Repositorium auf alle notwendigen Attribute zugreifen kann. Um dies zu gewährleisten, erbt jede Entität, welche im Datenbank Kontext registriert wird, von der abstrakten Klasse **EntityBase** (Abbildung 6.2). Diese Klasse implementiert alle Attribute der Schnittstelle **IEntity**, indem es die Konfiguration der Entität über Annotation an den Attributen (Listing 6.7, Zeile 03-36) deklariert.

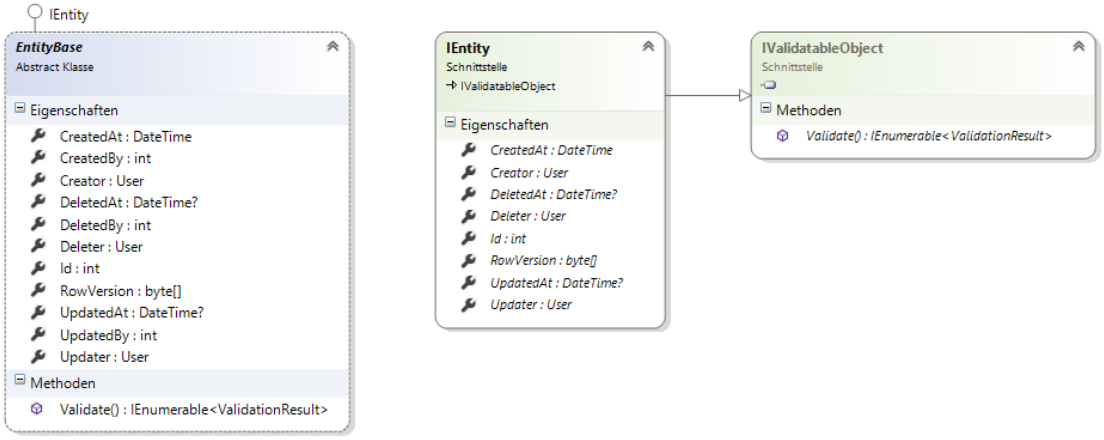


Abbildung 6.2: Abstrakte Basisklasse für Entitäten als Klassendiagramm (Visual Studio Export)

Ein grundsätzliches Attribut der Klasse *EntityBase* ist die *Id*, welche die Bezeichnung des Identifikators einer jeden Entitäten vorgibt und somit vereinheitlicht. Dieses Attribut wird nie selbst, sondern iterativ von der Datenbank generiert (Listing 6.7, Zeile 04). Eigene Identifikatoren werden somit immer mit *Id* gekennzeichnet, währenddessen Fremdschlüssel im Format *<Entitätsname>Id* gekennzeichnet werden. Außerdem werden Attribute zur Identifikation von Operationen auf einem Datensatz bereitgestellt. Wird ein Datensatz erstellt, verändert oder gelöscht, so wird jeweils der Zeitpunkt der Aktion und ein Verweis auf den Benutzer hinterlegt. Dadurch wird einerseits die Filterung, respektive die Zuordnung und Sichtbarkeit, von Daten eines Benutzers vereinfacht, indem auf einfachem Wege nur selbsterstellte Daten abgerufen werden könnten. Andererseits wird durch das Hinzufügen des Attributs *DeletedAt* der *Soft-Delete*-Filter der *Business Logic* (Kapitel 6.4) ermöglicht, da dieser alle Entitäten mit gesetztem Datum aus der Abfrage-Ergebnismenge entfernt. Weiterhin wird das Attribute **RowVersion** zur optimistischen Überprüfung von Nebenläufigkeit gesetzt, welches über die Annotation

timestamp markiert wird. Dieses Attribute wird zusätzlich bei Manipulation der Daten verwendet, um Nebenläufigkeitsfehler zu entdecken. Durch die Annotation **ConcurrencyCheck** wird zusätzlich das Attribut *CreatedAt* zur Überprüfung von optimistischer Nebenläufigkeit verwendet.

Durch die Verwendung der Standard-Vererbungsstrategie *Table per Hierarchy* (Kapitel 6.1.2, Code First) wird eine Tabelle für diese Attribute erstellt und per Diskriminator-Spalte auf die dazugehörenden Klassen verwiesen. Um dieses Verhalten zu unterdrücken und für diesen expliziten Fall die *Table per Concrete Class* Strategie anzuwenden, wird per *FluentAPI* die abstrakte Datenklasse *EntityBase* im Datenbank Kontext dergestalt konfiguriert, dass die Attribute der Klasse in jeder Tabelle erstellt werden. Dies wird über den *Ignore*-Befehl des **DbModelBuilder** (Listing 6.4, Zeile 19) realisiert.

6.3.2 Validierung der Entitätsklassen

Durch das Einbinden des Entity Frameworks ist zusätzlich eine automatische Validation vor jedem Einfügen und Manipulieren möglich. Das Validieren von Datensätzen ist notwendig, um die Datenqualität zu steigern und Daten, die gegen **Business Constraints**[4] verstoßen, nicht zu akzeptieren. Die Datenqualität ist in besonderen Maße wichtig, da alle Statistiken und Analysen für den Therapeuten von den Daten im System abhängen. Sind Daten schlechter Qualität im System, kann auf ihnen keine aussagekräftige Analyse ausgeführt werden. Aus dem selben Grunde ist die Datenqualität weiterhin für Wissenschaftler sehr wertvoll, da diese nur aus Daten guter Qualität Aussagen über Zusammenhänge schließen können und eine vorherige Datenbereinigung teilweise nur von geringen Nutzen sein kann. Gleichzeitig müssen Business Constraints bei der Verwendung des Systems eingehalten werden, um unerwünschtes Verhalten des Systems zu vermeiden. Im Folgenden werden zwei Ansätze erläutert, die dabei vom Entity Framework im Code First Programmiermodell unterstützt werden.

[4] Beschränkungen oder Randbedingungen, die für den Betrieb des System festgelegt wurden.

```
1  public abstract class EntityBase : IEntity
2  {
3    [Key]
4    [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
5    [Column(Order = 0)]
6    public int Id { get; set; }
7
8    [Required]
9    [ConcurrencyCheck]
10   [DataType(DataType.DateTime)]
11   public DateTime CreatedAt { get; set; }
12
13   [ForeignKey("CreatedBy")]
14   [DataType(DataType.EmailAddress)]
15   public virtual User Creator { get; set; }
16   public int CreatedBy { get; set; }
17
18   [DataType(DataType.DateTime)]
19   public DateTime? UpdatedAt { get; set; }
20
21   [ForeignKey("UpdatedBy")]
22   [DataType(DataType.EmailAddress)]
23   public virtual User Updater { get; set; }
24   public int UpdatedBy { get; set; }
25
26   [DataType(DataType.DateTime)]
27   public DateTime? DeletedAt { get; set; }
28
29   [ForeignKey("DeletedBy")]
30   [DataType(DataType.EmailAddress)]
31   public virtual User Deleter { get; set; }
32   public int DeletedBy { get; set; }
33
34   [Timestamp]
35   public byte[] RowVersion { get; set; }
36
37   public IEnumerable<ValidationResult> Validate(ValidationContext vContext)
38   {
39     //throw new NotImplementedException();
40     var result = new List<ValidationResult>();
41
42     // checks the order of the date fields
43     if (CreatedAt >= UpdatedAt || CreatedAt >= DeletedAt)
44     result.Add(new ValidationResult("Erstelldatum muss vor dem Bearbeitungs- oder
45       Löschdatum sein!"));
46
47     if (UpdatedAt > DeletedAt)
48     result.Add(new ValidationResult("Löschdatum kann nicht vor dem
49       Bearbeitungsdatum sein!"));
50
51     return result;
52   }
53 }
```

Listing 6.7: Abstrakte Basisklasse aller Entitäten

Das Entity Framework stellt eine Methode zur einfachen Validation der Entitätsklassen auf Attributsebene bereit. Diese kann entweder über die Annotation von sogenannten **DataAnnotations** konfiguriert werden oder über die *FluentAPI*. In dieser Arbeit wurden *DataAnnotations* verwendet, weshalb diese Funktion in dieser Variante vorgestellt wird. Jede Annotation gilt für ein bestimmtes Attribut der Entitätsklasse und wird eine entsprechende Fehlermeldung mit dem betreffenden Feldnamen erzeugen. Ein Beispiel für solche *DataAnnotations* befindet sich in Listing 6.7 in der Zeile 08 bis 11. Über die Annotation **Required** wird dem Entity Framework signalisiert, dass dieses Attribut bei jedem Speicherversuch gesetzt sein muss. Weiterhin sind Angaben zum Datentyp über die Annotation **DataType()** möglich, wobei innerhalb der Klammer ein vom Framework festgelegter Datentyp ausgewählt werden muss. Diese Annotation erweitert die möglichen primitiven Datentypen innerhalb der Validation um komplexere Datentypen wie E-Mail-Adresse, HTML-Markup oder URL. Über die Annotationen **StringLength()** für Zeichen und **Range()** für numerische Werte kann die Länge und zusätzlich bei numerischen Werten der Bereich festgelegt werden.

Eine weitere Methode zur Validation von Entitätsklassen ist die Validation auf Klassenebene. Dafür erbt die Schnittstelle *IEntity*, welche von der Basisklasse *EntityBase* implementiert wird, von der Schnittstelle **IValidatableObject**. Dadurch muss die abstrakte Klasse *EntityBase* zusätzlich eine *Validate*-Methode implementieren, die bei der Ausführung einen **ValidationContext** erhält und anschließend eine Liste von Objekten des Typs **ValidationResult** zurückgibt. Innerhalb dieser Methode kann auf Klassenebene validiert werden, da im Gegensatz zu den DataAnnotations auf alle Attribute des Objektes zugegriffen werden kann. Dies eignet sich besonders für Operationen zur Steigerung der Datenqualität, da nicht nur der Datentyp sondern auch der Inhalt überprüft und verglichen werden kann. Dies wird in Listing 6.7 ab Zeile 37 verdeutlicht, in dem beispielsweise das Erstell - und Manipulationsdatum miteinander verglichen wird. Ist das Erstelldatum nach dem Löschdatum, so muss ein Fehler vorliegen, weshalb ein *ValidationResult* mit einer passenden Fehlermeldung erzeugt wird.

6.3.3 Generisches Repositorium

Um den Zugriff auf Entitätsklassen innerhalb des Projektes zu vereinheitlichen, wird ein generisches Repositorium für Entitäten (Abbildung 6.3) erstellt. Die Entitäten müssen dabei alle von der Basisklasse *EntityBase* erben und somit die Meta-Attribute beinhalten. Das Repositorium besitzt einen Standardkonstruktor, in dem der Datenbank Kontext erstellt wird, um keine zusätzlichen Verweise innerhalb anderer Komponenten zu benötigen. Die im Repositorium definierten Methoden besitzen einen *Generic*, der den erwarteten Entitätstyp signalisiert. Durch diese Angabe erhält die Methode die Information welche Entität die Abfrage betrifft, respektive von welchem Typ das Ergebnis ist. Um in den Methoden Typ-Sicherheit zu gewährleisten, wird für jede Methode mit *Generic* der Entitätstyp auf die Schnittstelle *IEntity* gesetzt. Durch die Implementierung in der Basisklasse ist sichergestellt, dass jede Entität dieses Datenbank Kontextes mindestens die vordefinierten Meta-Attribute enthält. Dies ist insbesondere für die optionalen Filter entscheidend, da beispielsweise die Konfiguration des *Soft-Delete*-Filters über einen optionalen Parameter **deleted** beeinflussen werden kann.

Da das Repositorium äußerst wichtige verwendete Klasse innerhalb des kompletten Systems darstellt, muss sie häufig von verschiedenen Klassen instanziiert werden. So benötigt jedes Domänenmodell Zugriff auf die Klasse **Repository** und muss sich infolge eine Instanz erstellen, welche an die eigene geringe Lebenszeit gebunden ist. Dies hat bei hoher Server-Auslastung unnötigen Speicherbedarf und Rechenzeit zur Folge, womit das Erfüllen der nicht-funktionalen Anforderung **Effizienz** (Kapitel 4.4.2) erschwert wird. Aus diesem Grund wird das **Singleton**-Entwurfsmuster der *Gang of Four* [54] auf das Repositorium angewendet. Somit können sich alle Klassen, die ein Repositorium benötigen, eine Instanz der Klasse *Repository* teilen und besitzen somit auch einen gemeinsamen Datenbank Kontext. Dies wird über die statische Eigenschaft **Instance** realisiert, welche das gekapselte und nur lesbare Feld **_instance**, das eine Instanz von *Repository* besitzt, zugänglich macht. Dabei markiert das Schlüsselwort **static** die Eigenschaft *Instance* als globale Klasseneigenschaft. Das hat zur Folge, dass somit auf diese Eigenschaft ohne Instanziierung der Klasse zugegriffen werden kann.

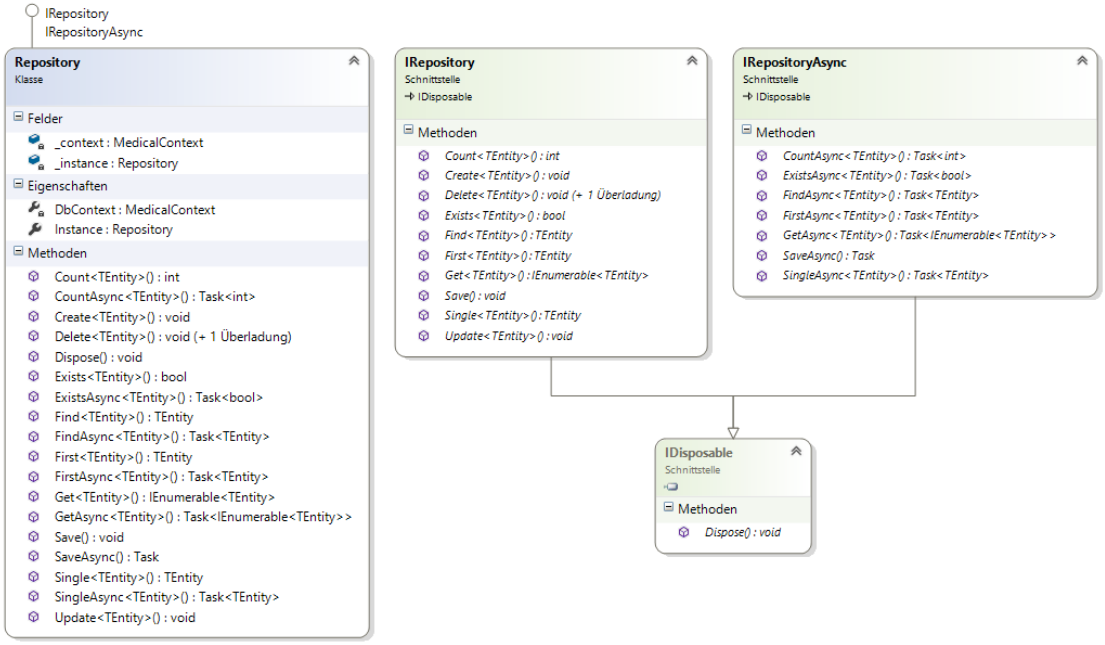


Abbildung 6.3: Generisches Repositorium des DAL als Klassendiagramm (Visual Studio Export)

Weiterhin enthält das Repositorium generische Methoden für alle wichtigen Datenbankfunktionen. So kann mittels **Get<TEntity>()** eine Entitätsmenge als Liste zurückgegeben werden, welche zuvor durch einen **Lambda-Ausdruck**[5] gefiltert oder sortiert werden kann. Über den *Include*-Parameter können dabei verknüpfte Entitäten mit in die Ergebnismenge aufgenommen werden. Dieses vorzeitige Nachladen von weiteren Entitäten wird über die *Include*-Methode des Entity Frameworks realisiert und entspricht dem Lademuster **Eager Loading**. Eine exemplarische Abfrage unter Angabe eines Lambda-Ausdrucks und *Include*-Parameters ist in Listing 6.8 in der Zeile 05 bis 09 dargestellt. Das Ergebnis der Abfrage ist eine Liste von Abgaben des Patienten mit der *Id* drei, abgegeben mit einem Änderungswunsch, angereichert mit den verknüpften Entitäten Abgabeereignis und Abgabekategorie und inklusive aller gelöschten Einträge. Zusätzlich werden mehrere Methoden bereitgestellt, um Elemente die dem Filter entsprechen zu zählen, einzelne zu finden und auf Existenz in der

[5] Anonyme Funktion, mit der unter anderem Ausdrucksbäume erstellt werden können.

```
1  // create new repository instance
2  using (var repo = new Repository<ApplicationContext>(new ApplicationContext()))
3  {
4    // get some submissions
5    var submList = repo.GetAsync<Submission>(
6      filter: s => s.WithChangeRequest == true && s.CreatedBy == 3,
7      include: "SubmissionEvent,SubmissionEventCategory",
8      distinct: false,
9      deleted: true);
10 }  // dispose the DbContext
```

Listing 6.8: Beispiel für die Benutzung des generischen Repositoriums

Datenbank zu überprüfen. Gleichermaßen werden auch generische Methoden zur Erstellung und Manipulation von Entitäten zu Verfügung gestellt.

Zusätzlich enthalten alle Abfrage-Methoden eine asynchrone Implementierung, um bei hoher Serverbelastung die Ressourcen optimaler aufzuteilen. Dazu muss die Klasse *Repository* nicht nur die Schnittstelle **IRepository** implementieren, sondern auch die Schnittstelle **IRepositoryAsync**, welche die asynchronen Methoden deklariert. Asynchrone Methoden erhalten als Rückgabewert einen generischen *Task*, welcher den Typ des Ergebnisses als *Generic* besitzt. Durch die Pausierung von wartenden Datenbankanfragen können mehrere *Tasks* gleichzeitg laufen und somit mehrere Anfragen, ohne dass sich diese gegenseitig blockieren, verarbeitet werden. Methoden zum Erstellen und Manipulieren von Entitäten besitzen keine asynchrone Implementierung und müssen synchron verarbeitet werden. Allerdings können diese über eine asynchrone Variante der *Save*-Methode in die Datenbank übernommen werden.

Einen Verweis auf den Datenbank Kontext, den das Repositorium zum Datenbankzugriff benötigt, wird innerhalb einer gekapselten Eigenschaft erstellt und zurückgegeben (Listing 6.9, Zeile 06-11). Diese Umsetzung ist ohne ansteigenden Ressourcenverbrauch möglich, da für jede Anfrage, welche Operationen auf der Datenbank ausführt, derselbe Datenbank Kontext verwendet wird. Über das Schlüsselwort **using** (Listing 6.8, Zeile 02) ist es weiterhin möglich eine neue *Repository*-Instanz innerhalb eines gewissen Blocks zu instanziieren und nicht die statische Instanz zu verwenden. Damit die verwaisten Einträge im Speicher von der **Garbage Collection** der Laufzeitumgebung erfasst werden, implementiert das Repositorium die Methode **Dispose()** der Schnittstelle **IDisposable** des .NET Frameworks, welche die nicht mehr verwalteten Ressourcen

```
public class Repository : IRepository, IRepositoryAsync
{
  private static readonly Repository _instance = new Repository();
  public static Repository Instance { get { return _instance; } }

  private MedicalContext _context;
  private MedicalContext DbContext
  {
    get { if (_context == null) _context = new MedicalContext(); return _context; }
    set { DbContext = value; }
  }

  // [...]

  public void Update<TEntity>(TEntity entity, User Updater = null)
    where TEntity : class, IEntity
  {
    // if no Updater is set in the Entity
    if (entity.Updater == null && Updater != null)
      entity.Updater = Updater;       // set the Updater

    // attach Entity to Context
    _context.Set<TEntity>().Attach(entity);
    _context.Entry<TEntity>(entity).State = EntityState.Modified;
  }

  public void Save()
  {
    try { _context.SaveChanges(); }
    catch (DbEntityValidationException e)
    {
      // [...]
      foreach (DbEntityValidationResult item in e.EntityValidationErrors)
      {
        DbEntityEntry entry = item.Entry;

        // rollback
        switch (entry.State)
        {
          case EntityState.Added:
            entry.State = EntityState.Detached;
            break;
          case EntityState.Modified:
            entry.CurrentValues.SetValues(entry.OriginalValues);
            entry.State = EntityState.Unchanged;
            break;
          case EntityState.Deleted:
            entry.State = EntityState.Unchanged;
            break;
        }
      }
    }
  }
}
```

Listing 6.9: Ausschnitt des generischen Repositoriums für das Aktualisieren von Entitäten

freigibt. Wird ein *using*-Block geschlossen, so ruft das .NET Framework automatisch die *Dispose*-Methode des Repositoriums auf. Dabei wird, die vom Entity Framework implementierte *Dispose*-Methode der Klasse *DbContext* aufgerufen, welche ebenfalls die Schnittstelle *IDisposable* implementiert und somit den vom Kontext belegten Speicher freigibt. Sollte dadurch der Kontext-Verweis in der *Repository*-Instanz zerstört werden, wird dieser beim nächsten Aufruf neu erstellt und somit wieder zugänglich gemacht.

6.4 Business Logic

Um Business Constaints einzuhalten oder Analysen über mehrere Datensätze zu berechnen, wird eine Logikschicht in Form der *Business Logic* implementiert. Sie kümmert sich zusätzlich um die Validierung von Anfragen aus dem Webservice und um den Zugriff auf die Datenhaltung. Zur Verringerung der Komplexität des Datenmodels und zur Kapselung von Informationen werden Domänenmodelle erstellt. Diese Domänenmodelle verwenden das Repositorium des *Data Access Layer* für den Datenbankzugriff, indem sie ihre Informationen auf mehrere Persistenzmodelle abbilden.

6.4.1 Domänenmodelle

Die Zusammensetzung eines Domänenmodells kann exemplarisch für das Domänenmodell **Aufgabe** in Abbildung 6.4 betrachtet werden. Verweise der einzelnen Persistenzmodelle, aus denen ein Domänenmodell besteht, werden als gekapselte Felder für die Klasse zugänglich gemacht. Zusätzlich erbt jedes Domänenmodell von der abstrakten Basisklasse **DomainModelBase**, die einen initialen Status setzt und das Repositorium für den Datenbankzugriff instanziiert. Um schnell auf die Eigenschaften verknüpfter Persistenzmodelle zugreifen zu können, werden die Eigenschaften alle gekapselten Felder als öffentliche Eigenschaft angeboten (Listing 6.11, Zeile 18-34) und somit ein hierarchisch flaches Datenmodell erstellt.

Um die Änderungen an das Repositorium durchzureichen, implementiert ein Domänenmodell die Schnittstelle **IDomainModel** (Listing 6.10, Zeile 01-08), welche

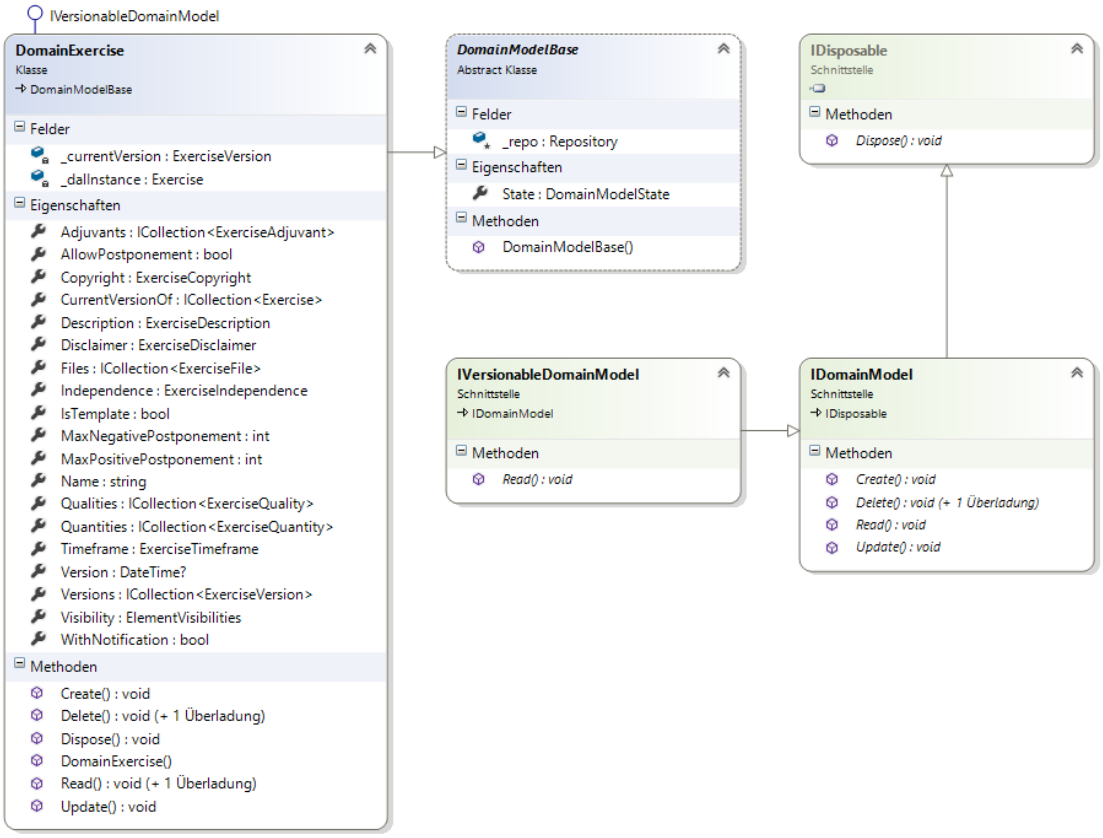


Abbildung 6.4: Domänenmodell für Aufgaben als Klassendiagramm (Visual Studio Export)

die **CRUD**[6]-Operationen als strukturelle Vorgabe beinhaltet. Um ein bereits bestehenden Domänenmodell abzurufen, wird eine *Read*-Methode vorgeben, welche als Parameter die *Id* zu der hierarchisch höchsten Entität beinhaltet. Diese *entityId* wird über die URL einer Anfrage am Webservice zur Geschäftslogik geleitet, welche dazu die passenden Entitäten lädt und als Domänenmodell zusammenfasst. Über die zusätzliche Angabe **deleted** kann der *Soft-Delete*-Filter manipuliert werden und somit auch bereits gelöscht Entitäten in das Ergebnis eingeschlossen werden. Zuletzt setzt die *Read*-Methode den Status des Domänenmodells auf **injected**, um die erfolgreiche Injektion der Persistenzmodelle zu kennzeichnen. Außerdem wird eine *Create*-Methode bereitgestellt, welche das Domänenmodell in einzelne Persistenzmodelle aufteilt und diese über

[6] Akronym für die grundlegenden Datenbankoperationen Create, Read, Update und Delete.

```
1  public interface IDomainModel : IDisposable
2  {
3    void Read(int entityId, bool? deleted = false);
4    void Create();
5    void Update();
6    void Delete();
7    void Delete(int entityId);
8  }
9
10 public interface IVersionableDomainModel : IDomainModel
11 {
12   void Read(int entityId, bool? versions = false, bool? deleted = false);
13 }
```

Listing 6.10: Schnittstellen für Domänenmodelle

das Repositorium in der Datenbank erstellt. Bereits vorhandene Entitäten, welche über die *Read*-Methode abgerufen wurden, können innerhalb des Domänenmodells manipuliert werden und über die *Update*-Methode den alten Datenstand überschreiben. Diese muss dazu dem Repositorium einzeln alle manipulierten Entitäten mitteilen. Weiterhin müssen in der Implementierung der *Update*-Methode des Repositoriums alle modifizierten Entitäten zur Verwaltung durch den Datenbank Kontext registriert und der *EntityState* jeweils auf **Modified** (Listing 6.9, Zeile 15-24) gesetzt werden. Wird der Status nicht für alle Kind-Elemente geändert, so wird das Entity Framework nur die Vater-Elemente updaten. Zusätzlich wird eine *Delete*-Methode angeboten, welche alle im Domänenmodell verbundenen Persistenzmodelle bis auf Kind-Ebene mittels *Soft-Delete*-Filterung als gelöscht markiert. Durch eine Überladung der *Delete*-Methode wird das Löschen der Entität, ohne dass diese vorher angefragt werden muss, ermöglicht. Zuletzt muss stets der Speichervorgang über die *Save*-Methode des Repositoriums (Listing 6.9, Zeile 26-53) angestoßen werden, wobei diese bei Validationsfehlern alle Elemente des Domänenmodells auf den letzten Stand zurücksetzt.

Da die Domänenmodelle *Aufgabe* und *EndogenerKontext* versioniert werden, implementiert diese die erweiterte Schnittstelle **IVersionableDomainModel** (Listing 6.10, Zeile 10-13), die die *Read*-Methode mit einem weiteren Parametern überlädt. Daten, die über das Repositorium abgerufen werden, können mit dem Lademuster *Eager Loading* (Kapitel 6.3.3) um verknüpfte Entitäten erweitert werden. Da die Versionen eines Domänenmodells mehrere verknüpfte Persistenzmodelle, darunter auch *One-To-Many*-Beziehungen, enthalten und bei Erstellung einer Version gegebenenfalls

```
1  public abstract class DomainModelBase
2  {
3    public DomainModelState State;
4    protected Repository _repo;
5
6    public DomainModelBase()
7    {
8      this.State = DomainModelState.empty;
9      _repo = new Repository();
10   }
11 }
12
13 public class DomainExercise : IVersionableDomainModel
14 {
15   private Exercise _dalInstance;
16   private ExerciseVersion _currentVersion;
17
18   public string Name
19   {
20     get { return _dalInstance.Title; }
21     set { _dalInstance.Title = value; }
22   }
23
24   public ElementVisibilities Visibility
25   {
26     get { return _dalInstance.AccessPermission; }
27     set { _dalInstance.AccessPermission = value; }
28   }
29
30   public bool IsTemplate
31   {
32     get { return _dalInstance.IsTemplate; }
33     set { _dalInstance.IsTemplate = value; }
34   }
35
36   // [...]
37
38   public void Update()
39   {
40     _repo.Create<ExerciseVersion>(_currentVersion);
41
42     // [...]
43
44     _repo.Update<Exercise>(_dalInstance);
45     _repo.Save();
46   }
47
48   public void Dispose()
49   {
50     _repo.Dispose();
51   }
52 }
```

Listing 6.11: Ausschnitt des Domänenmodells für Aufgaben

alle verknüpften Elemente ebenfalls neu erstellt werden müssen, kann das Vorhalten aller Versionen beim Abrufen von Domänenmodellen große Datenmengen beinhalten. Durch die Einführung eines Parameters zum optionalen Vorhalten aller Versionen, wird die Datenmenge eines Ladevorgangs, falls nur die aktuelle Version benötigt wird, signifikant verkleinert. Zusätzlich muss bei einem *Update* von versionierten Domänenmodellen darauf geachtet werden, dass im Persistenzmodell die Kind-Elemente neu erstellt (Listing 6.11, Zeile 38-46) werden und mit dem aktuellen Vater-Elemente verknüpft werden. Im Gegensatz dazu wird das Vater-Element über die *Update*-Methode des Repositoriums mit den neuen Daten überschrieben.

Durch die Basisschnittstelle *IDisposable*, von der die Schnittstelle aller Domänenmodelle *IDomainModel* erbt, implementieren alle Domänenmodelle die *Dispose*-Methode. Durch die durchgängige Implementierung dieser Methode, ausgehend vom Domänenmodell über Repositorium bis zum Datenbank Kontext, können die benötigten Ressourcen nach jeder bearbeiteten Anfrage freigegeben und der Speicher bereinigt werden.

6.4.2 Erbauerklasse für Domänenmodelle

Jedes Domänenmodell muss durch eine eigene Klassen definiert und implementiert werden, da Domänenmodelle für den Benutzer ihren Unterbau, respektive die Zusammensetzung durch Persistenzmodelle, verbergen sollen. Durch Implementierung der Schnittstellen *IDomainModel* und *IVersionableDomainModel* besitzen alle Domänenmodelle die gleiche Struktur für *CRUD*-Operationen mit der Datenbank und können somit nach außen identisch adressiert werden. Zusätzlich wird vorgegeben, dass jedes Domänenmodell durch den Standard-Konstruktur erstellt werden kann, um anschließend über die *Read*-Methode die benötigten Daten zu injizieren. Um den Zugriff auf Domänenmodelle zu vereinheitlichen und die Komplexität vor der Präsentationsschicht zu verbergen, wird die generische Klasse **DomainBuilder** (Abbildung 6.5) implementiert, die Domänenmodelle, respektive Listen von Domänenmodellen, erbauen kann. Das Erbauungsprinzip basiert auf dem **Erbauer**-Entwurfsmuster der *Gang of Four* [54], welches die Wiederverwendbarkeit von Konstruktionsprozessen durch Abstraktion der Erstellung komplexer Objekte erhöht.

Somit kann die Kopplung zwischen der Logikschicht und der Präsentationsschicht verringert werden, da außerhalb der Logikschicht kein Wissen über die Erstellung und Implementierung der Domänenmodelle benötigt wird. Zusätzlich kann eine direkte Verbindung zwischen Präsentationsschicht und Persistenzschicht vermieden werden. Durch die Verwendung von *Generics* und Einschränkungen auf deren Typparameter (Listing 6.12), sowie mithilfe der strukturellen Vorgaben von Domänenmodell-Operationen, kann bei der Erbauung von Domänenmodellen auf unnötige Dopplung von Programmiercode verzichtet werden. Stattdessen werden alle Operationen auf dem abstrakten Typ des Parameters ausgeführt (Listing 6.12, Zeile 27-28).

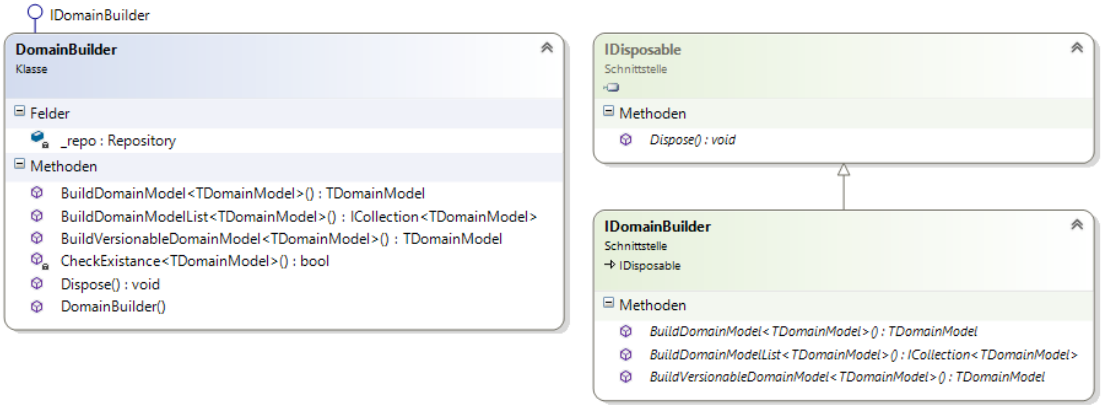


Abbildung 6.5: Erbauerklasse für Domänenmodelle als Klassendiagramm (Visual Studio Export)

Die *BuildDomainModel*- und *BuildVersionableDomainModel*-Methode liefern Domänenmodelle mit unterschiedlichen Status. Wird kein Vater-Persistenzmodell eines Domänenmodells des angegebenen Typs mit der übergebenen *id* gefunden, so wird ein neues Domänenmodell erstellt und mit dem Status *empty* versehen. Das erstellte Objekt kann nun mit Benutzerdaten gefüllt und anschließend über die *Create*-Methode des Domänenmodells persistiert werden. Wird jedoch ein Vater-Persistenzmodell gefunden, werden die Daten der Persistenzmodelle injiziert und der Status des Domänenmodells auf *injected* gesetzt. Die Überprüfung der Vater-Elemente wird über eine private Methode innerhalb der Klasse *DomainBuilder* bewerkstelligt. Alle *Build*-Methoden

```
public interface IDomainBuilder : IDisposable
{
  TDomainModel BuildDomainModel<TDomainModel>(
    int id, bool? deleted = false)
  where TDomainModel : class, IDomainModel, new();

  TDomainModel BuildVersionableDomainModel<TDomainModel>(
    int id, bool? versions = false, bool? deleted = false)
  where TDomainModel : class, IVersionableDomainModel, new();

  ICollection<TDomainModel> BuildDomainModelList<TDomainModel>(
    bool? deleted = false)
  where TDomainModel : class, IVersionableDomainModel, new();
}


public class DomainBuilder : IDomainBuilder
{
  private Repository _repo;

  public DomainBuilder()
  {
    this._repo = new Repository();
  }

  public TDomainModel BuildDomainModel<TDomainModel>(
    int id, bool? deleted = false)
  where TDomainModel : class, IDomainModel, new()
  {
    var domainModel = new TDomainModel();

    if (CheckExistance<TDomainModel>(id, deleted))
    domainModel.Read(id, deleted);

    return domainModel;
  }

  public TDomainModel BuildVersionableDomainModel<TDomainModel>(
    int id, bool? versions, bool? deleted)
  where TDomainModel : class, IVersionableDomainModel, new()
  {
    var domainModel = new TDomainModel();

    if (CheckExistance<TDomainModel>(id, deleted))
      domainModel.Read(id, versions, deleted);

    return domainModel;
  }

  // [...]

  public void Dispose()
  {
    this._repo.Dispose();
  }
}
```

Listing 6.12: Interface und Implementierung der Erbauerklasse für Domänenmodelle

bieten die Manipulation des *Soft-Delete*-Filters an und im Fall von versionierten Domänenmodellen einen Filter, um zusätzlich alle Versionen dieses Domänenmodells zu erhalten. Da die Klasse *DomainBuilder* ebenfalls die Schnittstelle *IDisposable* implementiert, kann der durch das Repositorium und dessen Felder benötigt Speicher freigegeben werden.

6.5 Webservice

Um Anwendungsdaten zwischen Klienten- und Server-Anwendung austauschen zu können, wird ein Webservice, genauer ein HTTP-Service nach dem REST-Architekturstil, implementiert. Dafür wird die ASP.NET Web Api Erweiterung (Kapitel 6.1.1) verwendet, die in ein ASP.NET Web-Projekt nach dem MVC-Pattern eingebunden wird. Da nach dem REST-Architekturstil ein ressourcen-zentrierter Dienst erstellt werden soll, wird jede Ressource von einem eigenen Controller verwaltet. Aufgrund dessen besitzt der Controller die zu verwaltende Ressource in seinem Klassennamen, wodurch das Framework bei Aufruf einer Ressource die Zuständigkeit zuordnen kann und somit das Erstellen einer Routing-Tabelle entfällt. Gleichzeitig implementiert ein Controller für jedes HTTP-Verb eine eigene Methode, die die *Id* der Ressource und die Parameter, sowie den Inhalt des *Request-Body*, als Methoden-Parameter erhält. Die Annotation **FromUri**, respektive die Annotation **FromBody**, muss dabei vor den Methoden-Parameter gesetzt werden, um dem Framework den Deklarationsort mitzuteilen. Da die URL-Parameter und der *Request-Body* im Gegensatz zur *Id* keine Pflichtfelder darstellen, müssen diese mit einem **?** nach dem Datentyp und der Angabe eines Standardwerts als optionale Parameter gekennzeichnet werden. Dies wird exemplarisch für das Domänenmodell, respektive für die Ressource *DomainExercise* (Abbildung 6.4) in Listing 6.13 demonstriert.

Zusätzlich können über Annotationen dem Framework weitere Anweisungen gegeben werden. Durch die Annotation oberhalb des Controllers, werden die Anweisungen auf alle Methoden übertragen. Über die Annotation **RoutePrefix()** wird dem Framework mitgeteilt, dass dieser Controller für alle Routen, die mit dem angegeben *String* beginnen,

```
01 [RoutePrefix("repo/exercise")]
02 [Authorize(Roles = "Therapist, Admin")]
03 public class ExerciseController : ApiController
04 {
05   private DomainBuilder _builder;
06
07   [Route("{id:int}")]
08   [ResponseType(typeof(DomainExercise))]
09   public IHttpActionResult Get(
10   int id,
11   [FromUri] bool? versions = false,
12   [FromUri] bool? deleted = false)
13   {
14     using (var _builder = new DomainBuilder())
15     {
16       var exercise = _builder.BuildVersionableDomainModel<DomainExercise>(
17         id, versions, deleted);
18       // check persistence model injection
19       if (exercise.State == DomainModelState.empty)
20         return NotFound();
21       else
22         return Ok(exercise);
23     }
24   }
25   // [...]
26 }
```

Listing 6.13: Ausschnitt des API-Controllers für Aufgaben

zuständig ist. Die explizite Angabe der Route ergänzt dabei die Standard-Konfiguration der API. Im Gegensatz zur Annotation **Route()** aus Kapitel 6.1.1 wird keine Aussage über den restlichen Teil der URL getroffen, weshalb die Einschränkung eines Platzhalters, dargestellt in Listing 6.3, nicht möglich ist. Es ist jedoch möglich, dass beide Annotationen ergänzend verwendet werden. So kann in Listing 6.13 Zeile 07 die Route-Annotation an einer Methode verwendet werden, um diese als zuständig für alle Routen mit dem *Prefix* des Controllers und dem angegebenen *Subfix* der Methode zu kennzeichnen. Über die Annotation **Authorize()** kann weiterhin angegeben werden, dass diese restriktive Ressource nur für authentifizierte Benutzer zugänglich ist, die beispielsweise die Rolle eines Therapeuten oder Admins besitzen müssen. Die Authentifizierung wird durch den **AccountController** realisiert, der mithilfe des in Kapitel 6.1.3 beschriebenen ASP.NET Identity Frameworks Zugriffsberechtigungen in Form eines *Bearer Tokens* (Listing 6.1, Zeile 05) erteilen kann. Um die funktionale Anforderung API Dokumentation (Kapitel 4.4.1) und die nicht-funktionale Anforderung **Wartbarkeit** (Kapitel 4.4.2) zu realisieren, wird die Annotation **ResponseType()** oberhalb der Controller-Methoden

definiert, um den Rückgabetyp zu kennzeichnen. Diese Information fließt in die Schnittstellen-Dokumentation der Server-Anwendung ein und kann somit dem Entwickler mitteilen, was er beim Aufruf dieser URL als Antwort des Systems erhält.

Innerhalb einer Controller-Methode wird auf die Ressource entsprechend des HTTP-Verbs zugegriffen. In Listing 6.13 Zeile 07 bis 23 wird dies für das Domänenmodell *DomainExercise* und das HTTP-Verb GET demonstriert. Dabei wird über das *using*-Schlüsselwort eine Instanz der Erbauerklasse für Domänenmodelle erstellt, die nur innerhalb des aufgespannten Blocks gültig ist. Die *Id* der Ressource, sowie die optionalen URL-Parameter, werden zur Erstellung eines Objektes der Klasse *DomainExercise* der *Build*-Methode übergeben. Kann ein Objekt mit der übergebenen *Id* erstellt werden, so wird es mit dem Status *injected* zurückgegeben. Falls es jedoch nicht gefunden wurde, so wird ein leeres Objekt zurückgegeben, welches den Status *empty* besitzt. Dies wird im Anschluss an den Erbauungsprozess durch den Controller geprüft und anhand des Ergebnisses eine entsprechende Antwort generiert. Der Antwort-Typ ist die Schnittstelle **IHttpActionResult**, da somit Methoden der Basisklasse **ApiController** zur Generation von Antworten mit den korrespondierenden Statuscodes des HTTP-Protokolls verwendet werden können. Ist das Ergebnis vom Status *empty*, so wird dem Aufrufer über die Methode *NotFound()* eine Antwort mit dem Statuscode *404* gesendet. Andernfalls wird über die Methode *Ok()* eine Antwort mit dem Statuscode *200* gesendet, welche das angefragte Objekt im *Body* beinhaltet. Das Format, indem das Objekt beantwortet wird, kann über den *Accept*-Header der Anfrage (Listing 6.1, Zeile 03) bestimmt werden und wird vom Framework, falls dieses das Format unterstützt, automatisch konvertiert.

6.5.1 Interaktion der Komponenten

In diesem Abschnitt wird die Anfrage einer Ressource über den Webservice, respektive die Interaktion der beteiligten Komponenten, detaillierter betrachtet, um das Zusammenspiel der vorherigen Kapitel der Implementierung zu verdeutlichen. Anhand eines UML2-Kommunikationsdiagramms (Abbildung 6.6) wird die Interaktion der Klassen für die Anfrage einer spezifischen Ressource und gleichzeitig die Anfrage einer Liste aller Ressourcen desselben Typs illustriert. Da der Erstellungsprozess eines

Domänenmodells durch den Webservice im Vordergrund der Illustration stehen soll, wurde auf die Darstellung des Authentifizierung und Autorisierung, sowie auf alternative Abläufe durch Ausnahmebehandlungen, verzichtet.

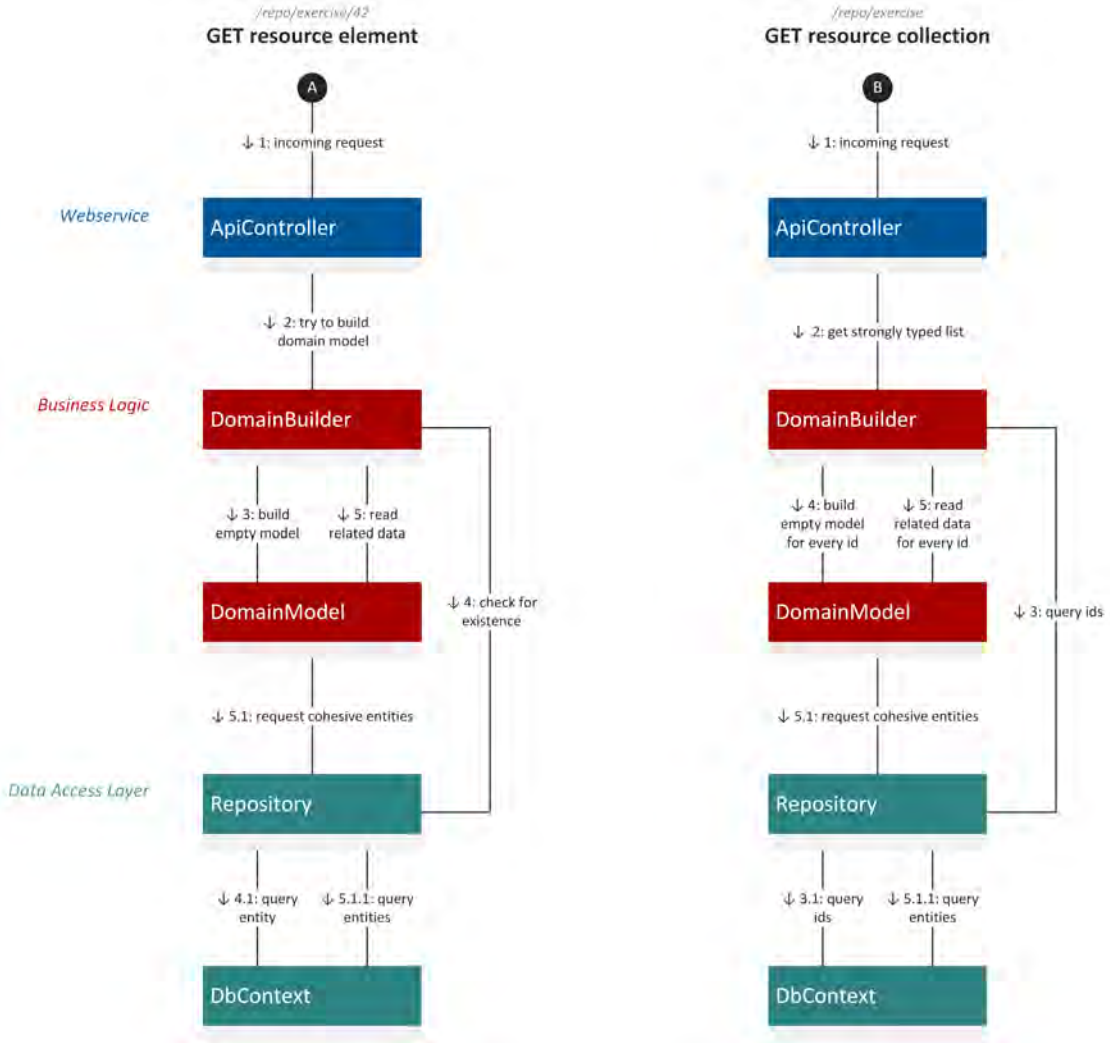


Abbildung 6.6: Kommunikation der Komponenten bei Anfrage einer Ressource über den Webservice

Um eine einzelne, spezifische Ressource anzufordern, muss eine GET-Anfrage mit einer *Id* am Ende der Route (Abbildung 6.6, Beispiel A) beim Server eingehen. Das Framework leitet die Anfrage dabei an den zugeordneten *ApiController*, respektive einen abgeleiteten Controller, weiter, welcher die entsprechende Methode

implementiert. Anschließend versucht der Controller ein Objekt über die Erbauerklasse für Domänenmodelle zu erstellen, welches über die übergebene *Id* verfügt. Zuerst erstellt der *DomainBuilder* dazu eine neue, leere Instanz des angeforderten Datentyps. Anschließend prüft er über das Repositorium, ob für die angegeben *Id* eine Entität in der Datenbank gefunden werden kann. Das Repositorium formt diese Anfrage um und stellt sie darauf folgend dem Datenbank Kontext, respektive dessen Ableitung. Die Antwort auf diese Anfrage wird nun vom Repositorium interpretiert und an den *DomainBuilder* zurückgeleitet. Anhand des Ergebnisses trifft dieser nun die Entscheidung, ob ein leeres Modell an den Webservice zurückgegeben wird oder verfügbare Daten injiziert werden. Für den Fall, dass in der Datenbank die Entität gefunden werden konnte, wird nun die *Read*-Methode des Domänenmodells vom *DomainBuilder* aufgerufen, welche das nötige Wissen um die Zusammenstellung von Persistenzmodellen besitzt. Für jedes einzelne Persistenzmodelle wird nun eine Anfrage an das Repositorium gesendet, welches diese wiederum für den Datenbank Kontext übersetzt und weiterleitet. Wurden durch die *Read*-Methode alle Persistenzmodelle gesammelt und injiziert, setzt diese den Status des Domänenmodells auf *injected*. Der *DomainBuilder* übergibt nun das Objekt, welches entweder leer ist oder Daten enthält, als Antwort auf den Aufruf an den Controller zurück. Zuletzt überprüft dieser den Status des Objektes und erstellt entweder eine Antwort mit Fehler-Statuscode oder eine Antwort mit dem Objekt selbst.

Die Interaktion durch eine Anfrage für eine Liste von Ressourcen (Abbildung 6.6, Beispiel B) ist vergleichbar mit der Anfrage einer einzelnen Ressource. Im Folgenden wird deshalb auf die Unterschiede zwischen den Interaktionen eingegangen. Die erste Abweichung befindet sich bereits im Controller, da dieser kein Domänenmodell über den *DomainBuilder* erstellt, sondern eine stark typisierte Liste von Objekten desselben Typs. Aufgrund dessen instanziiert der *DomainBuilder* eine leere Liste des angegeben Typs und stellt beim Repositorium eine Anfrage, um alle Identifikationsnummern für die betroffenen Objekte zu erhalten. Diese Anfrage wird vom Repositorium ebenfalls umgeformt und an den Datenbank Kontext weitergeleitet. Erhält der *DomainBuilder* die Identifikationsnummern, so wird für jede *Id* das entsprechende Domänenmodelle erstellt und über die *Read*-Methode mit den Persistenzmodellen befüllt. Das bedeutet, dass für jede gefundene *Id* der Teil-Prozess zur Zusammenstellung der Persistenzmodelle

aus Beispiel A ausgeführt wird. Im Anschluss daran wird die Liste an den Controller übergeben und kodiert als *HTTP-Response* an den Aufrufer geleitet.

7

Anforderungsabgleich

In diesem Kapitel werden die Anforderungen an das Rahmenwerk aus Kapitel 4.4 mit der Umsetzung, bestehend aus der Konzeption der Architektur und der Realisierung durch Implementierung, verglichen. Wie bereits in Kapitel 4.4, werden die Anforderungen dabei in funktional und nicht-funktional aufgeteilt.

7.1 Funktionale Anforderungen

In diesem Abschnitt werden die in Kapitel 4 analysierten und in Kapitel 4.4.1 definierten funktionalen Anforderungen an das Rahmenwerk mit der Umsetzung abgeglichen. Hierbei wird das Hauptaugenmerk auf die Implementierung gerichtet, da dieser Teil der Umsetzung die Funktionen realisiert. Um den Erfüllungsgrad der einzelnen Anforderungen zu dokumentieren, wird dabei die folgende Skala verwendet.

(5) Die Anforderung wurde vollständig erfüllt.

(4) Die Anforderung wurde zufriedenstellend erfüllt.

(3) Die Anforderung wurde nicht zufriedenstellend erfüllt.

(2) Die Anforderung ist noch nicht fertiggestellt.

(1) Die Anforderung wurde vorbereitet.

(0) Die Anforderung wurde nicht erfüllt.

ID	BEZEICHNUNG	PRIORITÄT	ERFÜLLUNGSGRAD
GAST-MODUL			
F/01	Marktplatz	*KANN*	**(1)**
F/02	API Dokumentation	*MUSS*	**(4)**
ACCOUNT-MODUL			
F/03	Autorisierung	*MUSS*	**(4)**
F/04	Registrierung	*MUSS*	**(3)**
F/05	Authentifizierung	*MUSS*	**(4)**
F/06	Benutzerverwaltung	*MUSS*	**(3)**
INSTITUTS-MODUL			
F/07	Institut verwalten	*SOLL*	**(4)**
F/08	Behandlungsangebot verwalten	*KANN*	**(5)**
F/09	Therapeuten einladen	*KANN*	**(1)**
KONTEXT-MODUL			
F/10	Kontext-Repositorium	*MUSS*	**(4)**
F/11	Kontext definieren	*MUSS*	**(4)**
AUFGABEN-MODUL			
F/12	Aufgaben-Repositorium	*MUSS*	**(5)**
F/13	Aufgaben definieren	*MUSS*	**(5)**
F/14	Medien-Manager	*SOLL*	**(1)**
F/15	Disclaimer-Manager	*KANN*	**(4)**
F/16	Copyright-Manager	*KANN*	**(4)**

Tabelle 7.1: Abgleich funktionaler Anforderungen - Teil 1

ID	BEZEICHNUNG	PRIORITÄT	ERFÜLLUNGSGRAD
PATIENTEN-MODUL			
F/17	Behandlungsaufgaben abrufen	*MUSS*	**(4)**
F/18	Behandlungsaufgaben manipulieren	*MUSS*	**(3)**
F/19	Abgabe erstellen	*MUSS*	**(3)**
F/20	Studienberechtigung	*SOLL*	**(3)**
BEHANDLUNGS-MODUL			
F/21	Behandlungen verwalten	*MUSS*	**(5)**
F/22	Behandlungsstatistik berechnen	*NOCH NICHT*	**(0)**
F/23	Behandlung erstellen	*MUSS*	**(5)**
F/24	Behandlungsvermerke verwalten	*SOLL*	**(5)**
F/25	Report erstellen	*NOCH NICHT*	**(1)**
THERAPEUTEN-MODUL			
F/26	Behandlungsaufgabe erstellen	*MUSS*	**(5)**
F/27	Behandlungsaufgabe bearbeiten	*MUSS*	**(5)**
F/28	Aufgabenstatus ändern	*MUSS*	**(5)**
F/29	Behandlungsaufgabe überwachen	*SOLL*	**(1)**
FEEDBACK-MODUL			
F/30	Feedback verwalten	*SOLL*	**(1)**
F/31	Upload-Elemente verwalten	*SOLL*	**(1)**
F/32	Fragen verwalten	*MUSS*	**(1)**
EINLADUNGS-MODUL			
F/33	Einladungen verwalten	*KANN*	**(0)**
F/34	Einladungen prüfen	*KANN*	**(0)**

Tabelle 7.2: Abgleich funktionaler Anforderungen - Teil 2

ID	BEZEICHNUNG	PRIORITÄT	ERFÜLLUNGSGRAD
ADMIN-MODUL			
F/35	Administration	*NOCH NICHT*	**(1)**
F/36	Institute konfigurieren	*SOLL*	**(2)**

Tabelle 7.3: Abgleich funktionaler Anforderungen - Teil 3

7.2 Nicht-Funktionale Anforderungen

In diesem Abschnitt werden die in Kapitel 4 analysierten und in Kapitel 4.4.2 definierten nicht-funktionalen Anforderungen an das Rahmenwerk mit der Umsetzung abgeglichen. Hierbei werden Architektur und Implementierung gleichwertig gewichtet, da Fehlentscheidungen in Konzeption der Architektur bedeutende Auswirkungen auf die nicht-funktionalen Anforderungen haben können. Um den Erfüllungsgrad der einzelnen Anforderungen zu dokumentieren, wird dabei ebenfalls die folgende Skala verwendet.

(5) Die Anforderung wurde vollständig erfüllt.

(4) Die Anforderung wurde zufriedenstellend erfüllt.

(3) Die Anforderung wurde nicht zufriedenstellend erfüllt.

(2) Die Anforderung ist noch nicht fertiggestellt.

(1) Die Anforderung wurde vorbereitet.

(0) Die Anforderung wurde nicht erfüllt.

ID	BEZEICHNUNG	PRIORITÄT	ERFÜLLUNGSGRAD
NF/01	Zuverlässigkeit	*MUSS*	**(4)**
NF/02	Erreichbarkeit	*MUSS*	**(5)**
NF/03	Robustheit	*SOLL*	**(4)**
NF/04	Sicherheit/Datenschutz	*MUSS*	**(4)**
NF/05	Effizienz	*SOLL*	**(4)**
NF/06	Wartbarkeit	*KANN*	**(3)**
NF/07	Portabilität	*MUSS*	**(5)**
NF/08	Benutzerfreundlichkeit	*MUSS*	**(4)**

Tabelle 7.4: Abgleich nicht-funktionaler Anforderungen

„Das Ende eines Dinges ist der Anfang eines anderen.“

Leonardo da Vinci

8

Zusammenfassung und Ausblick

Dieses Kapitel rekapituliert abschließend die Zielsetzung des Rahmenwerks und Ergebnisse dieser Arbeit. Unter Einbeziehung der Problemstellung, widmet sich Kapitel 8.1 den erbrachten Leistungen und potenziellen Verbesserungen durch diese Arbeit. Anschließend folgt in Kapitel 8.2 ein Ausblick auf zukünftige Erweiterungen für das Rahmenwerk.

8.1 Zusammenfassung

Diese Arbeit hat sich das Ziel gesetzt, ein Rahmenwerk für die Verwaltung und Ausführung von Hausaufgaben im therapeutischen Kontext zu entwickeln. Dazu musste zuerst ein Verständnis für den Prozess der Hausaufgabenempfehlung geschaffen und die positiven Effekte von therapeutischen Hausaufgaben herausgearbeitet werden. Es wurde beschrieben, dass Hausaufgaben ein enormes Wirkungspotenzial entfalten können, wenn sie zur Festigung positiver Verhaltensweisen in unterschiedlichen Kontexten wiederholt werden und der Patient in den Empfehlungsprozess

eingebunden wird. Zur Sammlung von Erkenntnissen für den Aufbau eines Mobile Crowdsensing-Dienstes, wurden anschließend verwandte Arbeiten recherchiert. Auf Basis des Hausaufgaben-Empfehlungsprozesses und dieser Erkenntnisse, mussten Anforderungen für unterschiedliche Projektteilnehmer analysiert und im Anschluss erhoben werden. Dabei wurde der aktuelle IST-Stand der Hausaufgabenerteilung und der Hausaufgabenüberwachung verwendet, um Ansatzpunkte für Erweiterungen und IT-gestützte Aktivitäten zu erarbeiten. Es wurde daraufhin ein SOLL-Stand entwickelt, bei dem diese Ansatzpunkte berücksichtigt wurden. Die IT-Unterstützung erweitert dabei die Möglichkeiten beider Prozesse, insbesondere jedoch den Prozess der Hausaufgabenüberwachung. Durch die Etablierung eines Kontextes ist der Therapeut in der Lage die Präzision der positiven Effekte von therapeutischen Hausaufgaben zu steigern. Gleichzeitig erhält er detailliertere Informationen über die Durchführung vor der Nachbesprechung der Hausaufgabe und ist somit in der Lage individueller auf den Patienten einzugehen, ihn zu motivieren und die Aufgabe an dessen Bedürfnisse anzupassen. Aus den gewonnen Informationen konnten darüber hinaus Anwendungsfälle kreiert werden, die in funktionale und nicht-funktionale Anforderungen resultierten.

Diese Anforderungen bildeten die Grundlage für eine mehrschichtige Architektur für das Gesamtsystem. Dabei wurde berücksichtigt, dass jeder Projektteilnehmer unterschiedliche Anforderungen an das System und Vorstellung für die Umsetzung besitzt. Aufgrund dessen wurde für eine Trennung der Klienten-Anwendungen entschieden, welche lose gekoppelt mit der Server-Anwendung interagieren. Durch eine feingliedrige Definition wurde gleichzeitig die Erweiterungsmöglichkeit der Datenstruktur gewährleistet, sowie die Versionierung von Aufgaben und Kontexten ermöglicht. Dadurch sind Therapeuten in der Lage Hausaufgaben stetig weiterzuentwickeln und Änderungen der Abgaben, respektive Änderungen des Zustandes eines Patienten, mit den unterschiedlichen Version zu vergleichen.

Zuletzt wurde die Realisierung der ausgearbeiteten Anforderungen mithilfe der entwickelten Architektur vorgestellt. Dabei wurde die Entscheidung für eine Entwicklung mit Microsoft-Technologien begründet und in die verwendeten Technologien anhand von Beispielen eingeführt. Die Zusammensetzung der verwendeten Technologien

wurde durch einen schematischen Aufbau der Server-Anwendung illustriert. Über die Aufteilung der Server-Anwendungen in einzelne, unabhängige Komponenten wurde dabei die Möglichkeit aufgezeigt, bei Änderungen oder Erweiterungen, beispielsweise aus Gründen der Effizienz, komplette Komponenten auszutauschen. Durch die lose Kopplung der Komponenten können diese nicht nur ausgetauscht, sondern auch auf verschiedenen Servern, beziehungsweise über Cloud-Dienstanbieter, ausgeliefert werden. Weiterhin wurde die zentralen Komponenten der Server-Anwendung vorgestellt und die Funktionsweise anhand von Beispielen erklärt. So wurde für den *Data Access Layer* die Struktur der Persistenzmodelle und der Aufbau eines generischen Repositoriums erläutert. Es wurde weiterhin dargestellt, wie der Zugriff auf verschiedene Entitäten durch den Einsatz von Schnittstellen, Basisklassen und *Type-Constraints* über das Repositorium verallgemeinert werden kann. Darauf aufbauen wurde die Struktur und Zusammenstellung von Domänenmodellen innerhalb der *Business Logic* beschrieben, sowie die *CRUD*-Operationen zur Manipulation der untergeordneten Persistenzmodelle. Um Domänenmodelle ohne Kenntnis über die Struktur zu erstellen und somit die Schichtenarchitektur nicht zu verletzen, wurde eine Erbauerklasse entwickelt, die Domänenmodelle jedes Typs erstellt. Zuletzt wurde die Implementierung des *Webservice* verdeutlicht, in dem ein exemplarischen Controller dargestellt wurde. Dadurch konnte letztendlich das Zusammenwirken und die Kommunikation der Server-Komponenten untereinander anhand eines exemplarischen Ressourcen-Aufrufs verdeutlicht werden.

8.2 Ausblick

Bereits während der Anforderungsanalyse wurde deutlich, dass dieses Rahmenwerk einen Umfang besitzt, die durch eine Arbeit allein nicht vollständig gelöst werden kann. Aufgrund dessen war das primäre Ziel dieser Arbeit, aus einer umfassenden Anforderungsanalyse eine erweiterbare Architektur zu entwickeln. Gleichzeitig sollten Grundfunktionalitäten implementiert werden, um das Zusammenwirken mit den Klienten-Anwendungen, welche getrennt entwickelt werden, zu testen. Wie bereits aus der Anforderungsanalyse entnommen werden kann, ist die Implementierung zusätzlicher Komponenten und Dienste geplant.

So soll der *Invitation Service* als eigenständiger Dienst implementiert werden, der für jede Einladungen einen vordefinierten Prozess startet und durchläuft. Der *E-Mail Service* soll in diesem Zuge ebenfalls realisiert werden, über den automatisiert Nachrichten versendet werden können, um Einladungen oder Benachrichtigungen des Systems zu verteilen. Auch der *Storage Service* ist eine noch zu entwickelnde Komponente, um große Dateien, beispielsweise hochauflösende Bilder oder große Audio-Dateien, referenziert abzulegen. Dabei ist angedacht, dass der *Storage Service* eine Schnittstelle zu Cloud-Dienstanbieter implementiert, um große Datenmengen performant und zuverlässig zugänglich zu machen.

Zusätzlich soll der Feedback-Mechanismus durch die Definition von eigenen Fragen erweitert werden, da die erste Version dieser Plattform nur Fragen aus einer vordefinierten Auswahl anbieten soll. Es ist angedacht, dass nach der ersten Version der Klienten-Anwendungen die Struktur der Fragebögen verfeinert und anschließend in sowohl Server-, als auch Klienten-Anwendung, implementiert wird. Gleichzeitig soll in diesem Zuge die Integration des multimedialen Feedbacks verbessert werden.

Um kontextuelle Informationen von außerhalb des Systems zu erlangen, soll die Integration des exogenen Kontext ausgearbeitet werden. Dabei sollen verschiedene Dienstanbieter des Bereichs *Social-Media* angesprochen werden, um über diese beispielsweise Ortsinformationen zu erlangen. Weiterhin sollen verschiedene kleinere Erweiterungen eingebracht werden. In Abstimmung mit den Klienten-Anwendung soll das Erfassen zusätzlicher Ereignistypen während der Aufgabendurchführung ermöglicht werden. Außerdem soll der Therapeut während des Überwachungsprozesses automatisierte Vorschläge zur Anpassungen von Hausaufgaben, basierend auf den Daten der Abgabe des Patienten, unterbreitet werden.

Die in diesem Abschnitt beschriebenen Erweiterungsmöglichkeiten stellen nur einen kleinen Teil der möglichen Erweiterungen für dieses Rahmenwerk dar. Um die Anforderungen für zukünftige Erweiterungen zu priorisieren, soll das System, sobald alle Komponenten in einer lauffähigen Version vorliegen, in einem realen Umfeld getestet werden. Dabei sollen speziell die Erfahrungen der Benutzer einfließen, die das System

im Alltag verwenden, um somit den Mehrwert für Benutzer des Systems möglichst zielgerichtet zu erhöhen.

Literaturverzeichnis

[1] Häberlein, T.: Technische Informatik: Ein Tutorium der Maschinenprogrammierung und Rechnertechnik. Springer-Verlag (2011)

[2] comScore: Anzahl der Smartphone-Nutzer in Deutschland in den Jahren 2009 bis 2016 (in Millionen). (https://de.statista.com/statistik/daten/studie/198959/umfrage/anzahl-der-smartphonenutzer-in-deutschland-seit-2010) Online; zuletzt besucht am 10.11.2016.

[3] TK-Pressestelle: #SmartHealth-Studie 2016. (https://www.tk.de/tk/themen/studien-und-auswertungen/smart-health-2016/914412) Online; zuletzt besucht am 05.10.2016.

[4] PocketGamer.biz: Ranking der Top-20-Kategorien im App Store im Oktober 2016. (https://de.statista.com/statistik/daten/studie/166976/umfrage/beliebteste-kategorien-im-app-store) Online; zuletzt besucht am 10.11.2016.

[5] Europäische Kommission: Strategie zur eGesundheit. (http://ec.europa.eu/health/ehealth/policy/index_de.htm) Online; zuletzt besucht am 10.11.2016.

[6] Robert Istepanian, Swamy Laxminarayan, C.S.: M-Health - Emerging Mobile Health Systems. Springer-Verlag (2006)

[7] Pryss, R., Langer, D., Reichert, M., Hallerbach, A.: Mobile Task Management for Medical Ward Rounds - The MEDo Approach. In: 1st Int'l Workshop on Adaptive Case Management (ACM'12), BPM'12 Workshops. Number 132 in LNBIP, Springer (2012) 43–54

[8] Schobel, J., Ruf-Leuschner, M., Pryss, R., Reichert, M., Schickler, M., Schauer, M., Weierstall, R., Isele, D., Nandi, C., Elbert, T.: A generic questionnaire framework supporting psychological studies with smartphone technologies. In: XIII Congress of European Society of Traumatic Stress Studies (ESTSS) Conference. (2013) 69–69

[9] Schobel, J., Schickler, M., Pryss, R., Nienhaus, H., Reichert, M.: Using Vital Sensors in Mobile Healthcare Business Applications: Challenges, Examples, Lessons Learned. In: 9th Int'l Conference on Web Information Systems and Technologies (WEBIST 2013), Special Session on Business Apps. (2013) 509–518

[10] Isele, D., Ruf-Leuschner, M., Pryss, R., Schauer, M., Reichert, M., Schobel, J., Schindler, A., Elbert, T.: Detecting adverse childhood experiences with a little help from tablet computers. In: XIII Congress of European Society of Traumatic Stress Studies (ESTSS) Conference. (2013) 69–70

[11] Crombach, A., Nandi, C., Bambonye, M., Liebrecht, M., Pryss, R., Reichert, M., Elbert, T., Weierstall, R.: Screening for mental disorders in post-conflict regions using computer apps - a feasibility study from Burundi. In: XIII Congress of European Society of Traumatic Stress Studies (ESTSS) Conference. (2013) 70–70

[12] Schobel, J., Schickler, M., Pryss, R., Maier, F., Reichert, M.: Towards Process-Driven Mobile Data Collection Applications: Requirements, Challenges, Lessons Learned. In: 10th Int'l Conference on Web Information Systems and Technologies (WEBIST 2014), Special Session on Business Apps. (2014) 371–382

[13] Schobel, J., Schickler, M., Pryss, R., Reichert, M., Elbert, T.: A Domain-Specific Framework for Collecting Data in Trials with Smart Mobile Devices. In: XIV Conference of European Society for Traumatic Stress Studies (ESTSS) Conference. (2015)

[14] Schobel, J., Schickler, M., Pryss, R., Reichert, M.: Process-Driven Data Collection with Smart Mobile Devices. In: 10th International Conference on Web Information Systems and Technologies (Revised Selected Papers). Number 226 in LNBIP. Springer (2015) 347–362

[15] Schobel, J., Pryss, R., Reichert, M.: Using Smart Mobile Devices for Collecting Structured Data in Clinical Trials: Results From a Large-Scale Case Study. In: 28th IEEE International Symposium on Computer-Based Medical Systems (CBMS 2015), IEEE Computer Society Press (2015) 13–18

[16] Wikipedia: Adhärenz - Wikipedia, Die freie Enzyklopädie. (`https://de.wikipedia.org/w/index.php?title=Adh%C3%A4renz&oldid=15647356`) Online; zuletzt besucht am 11.10.2016.

[17] DAK-Gesundheit: DAK Psychoreport 2015. (`https://www.dak.de/dak/bundes-themen/DAK-Psychoreport_2015-1718178.html`) Online; zuletzt besucht am 05.10.2016.

[18] BKK Dachverband: BKK Gesundheitsreport 2015. (`http://www.bkk-dachverband.de/publikationen/bkk-gesundheitsreport/`) Online; zuletzt besucht am 05.10.2016.

[19] Fehm, L., Helbig-Lang, S.: Hausaufgaben in der Psychotherapie - Standardtechnik mit hohem Potenzial. CME.springer.de - Zertifizierte Fortbildung für alle Psychotherapeuten (2009)

[20] Norcross, J.C., Hedges, M., Prochaska, J.O.: The face of 2010: A Delphi poll on the future of psychotherapy. **3** (2002) 316–322

[21] Kazantzis, N., Whittington, C., Dattilio, F.: Meta-analysis of homework effects in cognitive and behavioral therapy: A replication and extension. Clinical Psychology: Science and Practice **17** (2010) 144–156

[22] Sonnenmoser, M.: Hausaufgaben in der Psychotherapie - Noch unentdecktes Potenzial. Deutsches Ärzteblatt **1** (Januar 2010) 16–17

[23] Fehm, L., Fehm-Wolfsdorf, G.: Hausaufgaben in der Psychotherapie - Hausaufgaben als therapeutische Intervention: Ausnahme oder Alltag? Psychotherapeut **6** (2001) 386–390

[24] Christin, D., Reinhardt, A., Kanhere, S.S., Hollick, M.: A survey on privacy in mobile participatory sensing applications. Journal of Systems and Software **84** (2011) 1928–1946

[25] Guo, B., Wang, Z., Yu, Z., Wang, Y., Yen, N.Y., Huang, R., Zhou, X.: Mobile crowdsensing and computing: The review of an emerging human-powered sensing paradigm. ACM Computing Surveys (CSUR) **48** (2015) 7

[26] Ma, H., Zhao, D., Yuan, P.: Opportunities in Mobile Crowd Sensing. IEEE Communications Magazine **52** (2014) 29–35

[27] Wikipedia: Gamification - Wikipedia, Die freie Enzyklopädie. (`https://de.wikipedia.org/w/index.php?title=Gamification&oldid=157971413`) Online; zuletzt besucht am 11.11.2016.

[28] Scheel, M.J., Hanson, W.E., Razzhavaikina, T.I.: The Process of Recommending Homework in Psychotherapy: A Review of Therapist Delivery Methods, Client Acceptability, and Factors That Affect Compliance. Psychotherapy: Theory, Research, Practice, Training **41** (2004) 38–55

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

[30] Pryss, R., Mundbrod, N., Langer, D., Reichert, M.: Supporting medical ward rounds through mobile task and process management. Information Systems and e-Business Management **13** (2015) 107–146

[31] Schobel, J., Pryss, R., Schickler, M., Reichert, M.: A Lightweight Process Engine for Enabling Advanced Mobile Applications. In: 24th International Conference on Cooperative Information Systems (CoopIS 2016). Number 10033 in LNCS, Springer (2016) 552–569

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

[33] Schobel, J., Pryss, R., Wipp, W., Schickler, M., Reichert, M.: A Mobile Service Engine Enabling Complex Data Collection Applications. In: 14th International Conference on Service Oriented Computing (ICSOC 2016). Number 9936 in LNCS (2016) 626–633

[34] Schobel, J., Pryss, R., Schickler, M., Ruf-Leuschner, M., Elbert, T., Reichert, M.: End-User Programming of Mobile Services: Empowering Domain Experts to Implement Mobile Data Collection Applications. In: 5th IEEE International Conference on Mobile Services (MS 2016), IEEE Computer Society Press (2016) 1–8

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

[36] Winfried Schlee: Track Your Tinnitus – Android-Apps auf Google Play. (`https://play.google.com/store/apps/details?id=com.jochenherrmann.trackyourtinnitus`) Online; zuletzt besucht am 11.11.2016.

[37] Probst, T., Pryss, R., Langguth, B., Schlee, W.: Emotion dynamics and tinnitus: Daily life data from the “TrackYourTinnitus” application. Scientific Reports **6** (2016)

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

[39] Schickler, M., Pryss, R., Reichert, M., Schobel, J., Langguth, B., Schlee, W.: Using Mobile Serious Games in the Context of Chronic Disorders - A Mobile Game Concept for the Treatment of Tinnitus. In: 29th IEEE Int’l Symposium on Computer-Based Medical Systems (CBMS 2016). (2016) 343–348

[40] Pryss, R., Reichert, M., Herrmann, J., Langguth, B., Schlee, W.: Mobile Crowd Sensing in Clinical and Psychological Trials - A Case Study. In: 28th IEEE Int’l Symposium on Computer-Based Medical Systems, IEEE Computer Society Press (2015) 23–24

[41] Pryss, R., Reichert, M., Langguth, B., Schlee, W.: Mobile Crowd Sensing Services for Tinnitus Assessment, Therapy and Research. In: IEEE 4th International Conference on Mobile Services (MS 2015), IEEE Computer Society Press (2015) 352–359

[42] Schickler, M., Schobel, J., Pryss, R., Reichert, M.: Mobile Crowd Sensing - A New way of collecting data from trauma samples? In: XIV Conference of European Society for Traumatic Stress Studies (ESTSS) Conference. (2015) 244

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

[44] Serquera, J., Schlee, W., Pryss, R., Neff, P., Langguth, B.: Music Technology for Tinnitus Treatment Within Tinnet. In: Audio Engineering Society Conference: 58th International Conference: Music Induced Hearing Disorders. (2015)

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

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

[47] mykind: mykind – Android-Apps auf Google Play. (`https://play.google.com/store/apps/details?id=de.uni_ulm.bachmeier.mykind`) Online; zuletzt besucht am 11.11.2016.

[48] Institut für Datenbanken und Informationssysteme: QuestionSys - Universität Ulm. (`http://www.uni-ulm.de/in/iui-dbis/forschung/laufende-projekte/questionsys/`) Online; zuletzt besucht am 11.11.2016.

[49] Wikipedia: MoSCoW-Priorisierung - Wikipedia, Die freie Enzyklopädie. (https://de.wikipedia.org/w/index.php?title=MoSCoW-Priorisierung&oldid=141959307) Online; zuletzt besucht am 11.11.2016.

[50] Fielding, R.T.: Architectural styles and the design of network-based software architectures. PhD thesis, University of California, Irvine (2000)

[51] Kantar: Marktanteile der mobilen Betriebssysteme am Absatz von Smartphones in Deutschland von Juli bis September in den Jahren 2015 und 2016. (https://de.statista.com/statistik/daten/studie/198435/umfrage/marktanteile-der-smartphone-betriebssysteme-am-absatz-in-deutschland) Online; zuletzt besucht am 11.11.2016.

[52] Institut für Datenbanken und Informationssysteme: MobileTx - Universität Ulm. (http://www.uni-ulm.de/in/iui-dbis/forschung/laufende-projekte/mobiletx/) Online; zuletzt besucht am 11.11.2016.

[53] Wikipedia: C-Sharp - Wikipedia, Die freie Enzyklopädie. (https://de.wikipedia.org/w/index.php?title=C-Sharp&oldid=159001321) Online; zuletzt besucht am 11.11.2016.

[54] Gamma, E., Johnson, R., Helm, R., Vlissides, J.: Entwurfsmuster: Elemente wiederverwendbarer objektorientierter Software. Professionelle Softwareentwicklung. Addison-Wesley (2004)

„Zeit ist eine dreckige Scheiße.“

Rüdiger Pryss

A

Anhang

A.1 Analysedokument

Analyse

Konzeption und Realisierung eines Rahmenwerks
zur Unterstützung von Therapeuten bei der
Durchführung von Patientenbehandlungen

Michael Stach
michael.stach@uni-ulm.de

Versionsnummer
2016-10-30

INHALTSVERZEICHNIS

1. ÜBERBLICK

Diese Kapitel liefert einen groben Überblick, in dem es unter anderem die Motivation und Vision hinter dem Projekt beschreibt.

1.1 Einleitung

Dieses Dokument beschreibt ein System zur Unterstützung von Therapeuten bei der Betreuung und Durchführung von Patientenbehandlungen. Um die Vorstellungen der einzelnen Projektteilnehmer[1] zu konkretisieren und später in einen Entwurf zu überführen, werden die Ideen zur Realisierung eines solchen Systems in diesem Dokument festgehalten.

Das System soll Aufgaben zentral verwalten und Therapeuten die Möglichkeit bieten, diese Aufgaben kontextbezogen für ihre Behandlung zu verwenden. Um die erfolgreiche Erfüllung von Aufgaben zu gewährleisten, wird das System bei Beendigung der Aufgabe Feedback und gegebenenfalls Änderungen der Behandlungsaufgabe von dem Patienten empfangen. Dadurch wird der Therapeut ein objektiveres Bild der Behandlungsdurchführung und des aktuellen Behandlungsstatus erhalten und kann den Patienten individueller betreuen.

1.2 Motivation

Bei der Behandlung von Patienten stehen Therapeuten vor dem Problem, dass Aufgaben oftmals in einem gewissen Kontext durchgeführt werden müssen, um einen Erfolg der Behandlung zu gewährleisten. Die Parameter der Durchführung kann der Therapeut jedoch nur vom Patienten erfragen, welcher eine sehr subjektive Sicht auf die Durchführung besitzt. Fehlerhafte Durchführung der Behandlungsaufgabe und die Durchführung während ungünstiger Umgebungsbedingungen[2] sind die häufigsten Fehlerquellen, die den zukünftigen Behandlungserfolg verzögern können. Um eine fehlerhafte Durchführung zu erkennen, soll das System die Durchführungsparameter festlegen, überwachen, mit fortwährenden Anpassungen der Aufgabe vergleichen und die Beendigung der Aufgabe mittels eines Feedbacks zu bewerten. Basierend auf dieser Quelle kann der Therapeut den aktuellen Behandlungsstatus besser einschätzen und gegebenenfalls auf häufige Änderungen der Aufgabenparameter durch den Patienten reagieren.

Dieses Projekt umfasst einen Teil der geschilderten Aufgabenstellung und wird prototypisch umgesetzt. Es soll ein Backend zur zentralen Verwaltung von Benutzern, Aufgaben, Umgebungsbedingungen und Feedback erstellt werden. Die eigentliche Überwachung der Durchführungsparameter übernimmt eine Client-Anwendung, welche über eine Schnittstelle in Verbindung mit dem zentralen Backend steht.

1.3 Vision

Durch dieses System sollen Therapeuten eine einfache und benutzerfreundliche Plattform zur Erstellung und Durchführungsauswertung von Behandlungsaufgaben erhalten. Dadurch wird die subjektive Verzerrung in der Patient-Therapeut-Kommunikation überwindet und der Behandlungsfortschritt beschleunigt.

Das System soll Therapeuten ermöglichen Aufgaben innerhalb eines privaten oder globalen Archivs zu erstellen und verwalten, welche den zugeordneten Patienten zur Behandlung verordnet werden können. Der Therapeut soll die Möglichkeit haben mit dem Patienten eine individuell zugeschnittene Aufgabe zu erstellen und diese mittels Einsatz verschiedener Medientypen[3] dem Patienten genauer zu beschreiben. Diese Aufgaben werden immer in einem gewissen Kontext[4] erledigt, welcher zusätzlich erstellt und verwaltet werden muss. Zusätzlich soll es die Möglichkeit einer Validierung der Aufgabenerfüllung mittels einheitlichen,

1 Universität Ulm, Universität Regensburg
2 fortab „Kontext" genannt
3 Text, Bild, Audio und Video
4 bspw. <daheim, ruhige Umgebung, WLAN-Empfang>

systemverwalteten Feedback-Fragebögen geben. Die Kombination aus Aufgabe, Kontext und Feedback soll dem Patienten über eine Web-Schnittstelle zur Verfügung stehen, um die Aufgabenerfüllung mittels clientseitiger Applikationen zu unterstützen. Metadaten zur Ausführung, Anpassungen und Feedback sollen bei Aufgabenerfüllung von der Schnittstelle angenommen werden und innerhalb eines Dashboards dem Therapeuten dargestellt werden, um häufige Anpassungen der Aufgabenstellung oder des Kontextes zu identifizieren und gegebenenfalls in zukünftig mit einzubeziehen.

1.4 Projektkontext

Das Projekt soll im Rahmen mehrerer Bachelor- und Masterarbeiten entwickelt werden und entsteht in einer Kooperation der Universität Ulm mit der Universität Regensburg. Die Beteiligten der Universität Ulm sind hierbei zuständig für die technische Ausarbeitung der Projektidee, wobei die Beteiligten der Universität Regensburg den medizinisch-psychologischen Aspekt übernehmen.

Das in diesem Dokument beschriebene Teilsystem des Projektes wird in Bezug zur konzeptionellen Masterarbeit von Aliyar Aras und konzeptionellen Bachelorarbeit von Johannes Schmid entwickelt und ist Basis für weitere Ausbaustufen. Eine zukünftige Ausbaustufe ist die Entwicklung clientseitiger Anwendungen, die mit diesem Backend kommunizieren und die Überwachung der Aufgabenparameter übernehmen.

2. ANFORDERUNGSANALYSE

Das folgende Kapitel definiert die funktionalen Anforderungen aus Sicht der Anwender, in dem Kontext und Abläufe des Systems mittels UML-Notation dargestellt werden. Leistungen, die das neue System erbringen muss, werden hier als Anwendungsfälle geschildert und teilweise sequenziell entschlüsselt.

2.1 Fachwissen

Bezeichner: **Aufgabe**

Beschreibung: *Eine Aufgabe ist die Kombination aus Aufgabenbeschreibung, Hilfsmittel, Qualität, Quantität, Selbstständigkeit und Zeitrahmen. Aufgaben sind nicht patientenbezogen und können in Repositories geteilt werden.*

Synonym: *Patienten-Aufgabe*

Kann sein:
- *Körperliche Anstrengung*
- *geistige Anstrengung*

Beispiel: *2x wöchentlich, so lange die Aufgabe benötigt wird, unter bewusster Kontrolle des linken Knies, unter Aufsicht einer Pflegeperson, die Hose im Stehen anziehen können, mit Anlehnen am Bettgitter*

Bezeichner: **Aufgabenbeschreibung**

Beschreibung: *Für eine Aufgabe wird immer eine Aufgabenstellung, bzw. ein Aufgabenziel, definiert.*

Synonym: *Beschreibung*

Kann sein: *Text*

Beispiel:
- *Die Hose im Stehen anziehen können*
- *Gehen können und selbstständig in eine Liste eintragen*
- *Fremde Personen nach einem komplizierten Weg fragen und Rückfragen stellen*
- *dysfunktionalen Gedanken „Ich bin ein Versager!" im ABC-Modell umstrukturieren*
- *Entspannungsübung (PMR)*

Bezeichner: **Behandlungsaufgabe**

Beschreibung: *Eine Behandlungsaufgabe ist eine personalisierte Aufgabe und ist immer auf einen einzelnen Patienten zugeschnitten. Dafür wird eine Aufgabe mit einem individualisierten Kontext und einem Feedback zur Sicherung der Behandlungsqualität angereichert.*

Synonym: *Hausaufgabe*

Kann sein:
- *Körperliche Anstrengung*
- *geistige Anstrengung*

Beispiel: *2x wöchentlich, so lange die Aufgabe benötigt wird, unter bewusster Kontrolle des linken Knies, unter Aufsicht einer Pflegeperson, die Hose im Stehen anziehen können, mit Anlehnen am Bettgitter; „daheim – abends – im Gang – WLAN"; „Konnte die Übung selbstständig und erfolgreich ausgeführt werden?"*

Bezeichner: **Behandlungsvermerk**

Beschreibung: *Während der Behandlung durch einen Therapeuten, kann zu einem Patienten ein Behandlungsvermerk hinzugefügt werden, welcher eine textuelle Beschreibung des aktuellen Patienten-Status enthält. Dieser Vermerk wird entweder direkt in Beziehung zu*

einer Behandlungsaufgabe, wie beispielsweise nach der Beendigung einer Behandlungsaufgabe, oder generell zu dem Patienten erstellt. Dadurch kann der Therapeut den Verlauf seiner Behandlung besser nachvollziehen und bei unerwünschten Nebeneffekten die Behandlungsaufgaben nachbessern. Weiterhin werden Behandlungsvermerke in Patienten-Reports angezeigt.

Synonym: *Vermerk*

Kann sein: *Text*

Beispiel:
- *Patient kann die Hose bis zum Knie anziehen.*
- *Aufgabe wurde vom Patienten nicht angenommen.*

Bezeichner: **Hilfsmittel**

Beschreibung: *(Immaterieller) Gegenstand der zur erfolgreichen Absolvierung der Aufgabestellung benötigt wird.*

Kann sein:
- *Musik*
- *Gegenstand*
- *Programm*

Beispiel:
- *Mit Anlehnen am Bettgitter*
- *Handlauf*

Bezeichner: **Feedback**

Beschreibung: *Optionale oder verpflichtende Abfrage, die dem Patienten bei Absolvierung der Aufgabe gestellt wird. Eine Feedback-Vorlage kann aus Fragen und Medien-Elementen zum Upload zusammengestellt werden.*

Synonym:
- *Feedback-Fragebogen*
- *Feedback-Vorlage*

Kann sein:
- *Ja/Nein-Fragebogen*
- *individueller oder standardisierter Fragebogen*
- *Medien-Upload*

Beispiel:
- *Haben Sie die Übung vollständig absolviert?*
- *Haben Sie die Übung nur mit Mühe absolvieren können?*
- *Als wie schwer empfanden Sie diese Übung?*
- *Wie würden sie den Erfolg dieser Übung einschätzen?*

Bezeichner: **Kontext**

Beschreibung: *Der Kontext beschreibt die Umgebungsbedingungen, die für eine erfolgreiche Aufgabenausführung benötigt werden und kann als endogener[5] und exogener[6] Kontext spezifiziert werden. Der endogene Kontext ist die Kombination aus den technischen Anforderungen und den Umgebungsbedingungen. Damit kann gewährleistet werden, dass die Aufgabe überhaupt, respektive korrekt und in einer wirksamen und störungsfreien Umgebung, ausgeführt werden kann. Der exogene Kontext kann eine außerhalb spezifizierte und erhobene Quelle sein, deren Wert ausschlaggebend für die Ausführungsumgebung der Aufgabe ist.*

Synonym: *Kombinierte Durchführungsparameter*

Kann sein:
- *endogener Kontext*

5 innerhalb des Systems spezifiziert
6 Beeinflussung des Systems von extern spezifizierten Bedingungen

- *exogener Kontext*

Beispiel:
- *daheim, abends, im Gang, WLAN*
- *in der Mittagspause, ruhig, Mikrofon*

Bezeichner: **Medien-Element**

Beschreibung: *Daten eines vorgegebenen Typs, beziehungsweise eines vorgegebenen MIME-Types, die im System abgelegt werden können. Diese stehen im Bezug zu einer Aufgabe oder zu einem Feedback in einer Abgabe. Weiterhin sind META-Daten wie „Lizenz" und „Medien-Beschreibung" erforderlich.*

Kann sein: *Upload-Element*

Beispiel:
- *image/jpg*
- *audio/mpeg*
- *text/plain*

Bezeichner: **Qualität**

Beschreibung: *Zeitliche, kognitive oder physische Belastung während der Ausführung der Aufgabe.*

Kann sein:
- *Anstrengung*
- *Tonus*
- *Ass. Reaktion*
- *Double Task*
- *Tempo*

Beispiel:
- *unter bewusster Kontrolle des linken Knies*
- *erhöhter Zeitaufwand ist toleriert*

Bezeichner: **Quantität**

Beschreibung: *Häufigkeit, Dauer oder Anzahl der Aufgabe-Einheiten.*

Kann sein:
- *Wie weit?*
- *wie oft?*
- *wie lang?*

Beispiel:
- *4 x 10 Meter*
- *30 Minuten*
- *so lange die Aufgabe benötigt wird*

Bezeichner: **Selbstständigkeit**

Beschreibung: *Definiert die Anzahl der unterstützenden Personen für die erfolgreiche Absolvierung der Aufgabe.*

Kann sein:
- *Selbstständig*
- *mit Aufsicht*
- *geringe Hilfe*
- *max. Hilfe*

Beispiel:
- *unter Aufsicht einer Pflegeperson*
- *mit einer Person des anderen Geschlechts*
- *selbstständig*

Bezeichner: **Technische Anforderungen**

Beschreibung: *Die Aufgabenstellung wird dem Patienten digital zur Verfügung gestellt und benötigt unter Umständen bestimmte Ressourcen für den Abruf mit dem mobilen Endgerät. Außerdem müssen zur Überprüfung der Umgebungsbedingungen eventuell Sensoren reserviert werden, um beispielsweise den Umgebungsgeräuschpegel zu bestimmen.*

Synonym: *Technische Durchführungsparameter*

Kann sein:
- *Konnektivität*
- *Sensorik*
- *Bildschirmgröße/Auflösung*

Beispiel:
- *WLAN, GPS, Mikrofon, großer Bildschirm*
- *Internetempfang, Lichtempfindlichkeits-Sensor*

Bezeichner: **Umgebungsbedingungen**

Beschreibung: *Die Umgebungsbedingungen beschreiben die Situation, in welcher eine Aufgabe (ausschließlich) absolviert werden kann. Dabei sollen ortsbezogene Angaben, zeitbezogene Angaben und Sinneswahrnehmungen (ruhig/laut, hell/dunkel, still/in Bewegung) angegeben werden, um die korrekten Durchführungsparameter zu erlangen.*

Synonym: *Ortsbezogener Durchführungsparameter*

Kann sein:
- *Wo?*
- *Wann?*
- *welche Umgebung?*
- *Situation*
- *Geräuschpegel*
- *Zeitpunkt*
- *Intervall*

Beispiel:
- *morgens nach dem Frühstück*
- *Montag 18 Uhr & Mittwoch 16 Uhr, auf der Straße*
- *zuhause, im Gang, hell*
- *nicht ortsgebunden, in der Mittagspause, ruhig*
- *bei der Arbeit, morgens*
- *daheim, nach dem Waschen*
- *wenn der dysfunktionale Gedanke „Ich bin ein Versager!" präsent ist*

Bezeichner: **Zeitrahmen**

Beschreibung: *Gibt die Anzahl der Wiederholungen bis zu einem gewissen Datum oder einen zeitlichen Intervall / Rhythmus zur Ausführung der Aufgabenstellung an.*

Synonym: *Ausführungsintervall*

Kann sein: *Intervall, Rhythmus, Wiederholungen*

Beispiel:
- *15.09.2016 Bis 31.09.2016*
- *2x wöchentlich*
- *5x bis zum 11.04.2016*

2.2 Anwendungskontext

Um einen Überblick über die funktionalen Anforderungen aus Anwendersicht zu erhalten, werden in diesem Kapitel die geforderten Anwendungsfälle beschrieben und im Kontext des Gesamtsystems als Systemprofil strukturiert dargestellt. Für komplexe Anwendungsfälle werden zum Verständnis die Handlungsabläufe zusätzlich aufgeschlüsselt.

2.3.1 Systemprofil

Die folgenden Anwendungsfalldiagramme beschreiben das Systemprofil, welches alle Subsysteme, Anwendungsfälle aus Anwendersicht und alle beteiligten (menschlichen und technischen) Akteure definiert und sämtliche Verbindungen und Abhängigkeiten darstellt.

AKTEURE IM SYSTEM

Akteur: **Gast**

Beschreibung: *Gäste sind nicht angemeldete Benutzer, welche nur Zugriff auf öffentliche Inhalte wie das Gäste-Modul und den Login/Registrieren-Bereich besitzen.*

Akteur: **Benutzer**

Beschreibung: *Jeder angemeldete Akteur erbt im System von der abstrakten Rolle Benutzer und hat somit Zugriff auf alle Profilverwaltungsfunktionen, sowie auf die öffentlichen Inhalte. Die Rolle Benutzer kann es im System nicht geben.*

Akteur: **Patient**

Beschreibung: *Patienten sind Benutzer, welche Daten für das System produzieren, um eine verbesserte Lebensqualität durch computerunterstützte Therapien erhalten zu können. Sie haben Zugriff auf das Patienten-Modul, in dem sie ihre zu erledigenden Aufgaben verwalten, Änderungen einsehen, schon abgegebene Aufgaben einsehen können und Statistiken abrufen können.*

Akteur: **Therapeut**

Beschreibung: *Therapeuten sind die zentralen Akteure im System, welche Behandlungen mit dem System durchführen. Dafür können sie Patienten einladen, für welche sie Aufgaben in einem Repository erstellen und verwalten, Behandlungsaufgaben hinzufügen und diese individuell an den Kontext anpassen. Therapeuten werden einem Institut zugeordnet und durch Administratoren freigeschaltet, nachdem sie sich registriert haben.*

Akteur: **Administrator**

Beschreibung: *Administratoren verwalten alle Vorgänge und Daten im System und können alle nicht patientenbezogen Therapeuten-Funktionen durchführen, um bei Fehlern den Therapeuten zu unterstützen.*

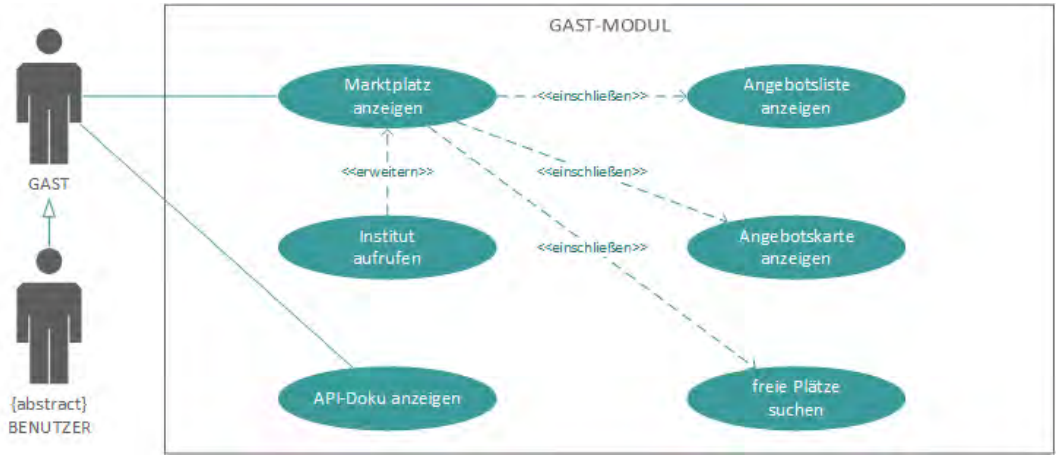
Akteur: **E-Mail-Service**

Beschreibung: *Dienst zur E-Mail-Benachrichtigung von Systembenutzern.*

Akteur:	**[Bezeichnung]-Modul**
Beschreibung:	*Konnektor zu Anwendungsfällen eines abgegrenzten Systembereich innerhalb des Systems.*

GAST-MODUL

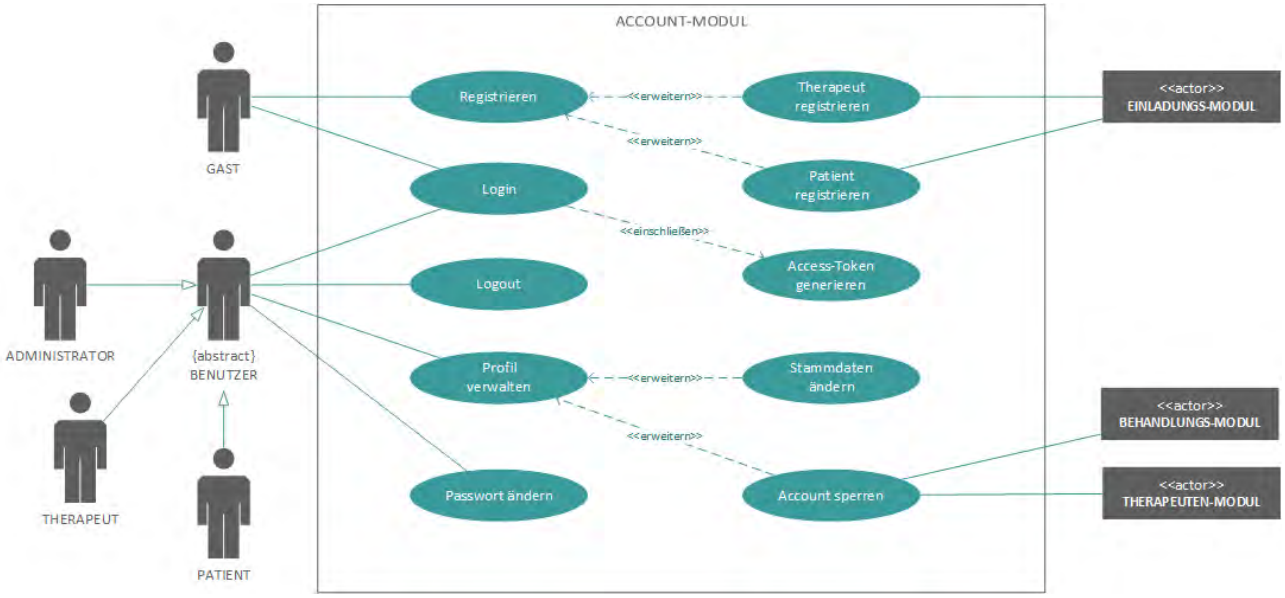
Funktionen innerhalb des Gast-Moduls können von allen Seitenbesuchern, insbesondere allen angemeldeten Benutzern, aufgerufen werden. Ein Marktplatz listet alle hinterlegten Institute und deren Angebote, welche unterschiedlich durchsucht werden können. Außerdem können Entwickler die API-Dokumentation aufrufen, um mobile Anwendungen bereitzustellen.



Anwendungsfalldiagramm: Gast-Modul

ACCOUNT-MODUL

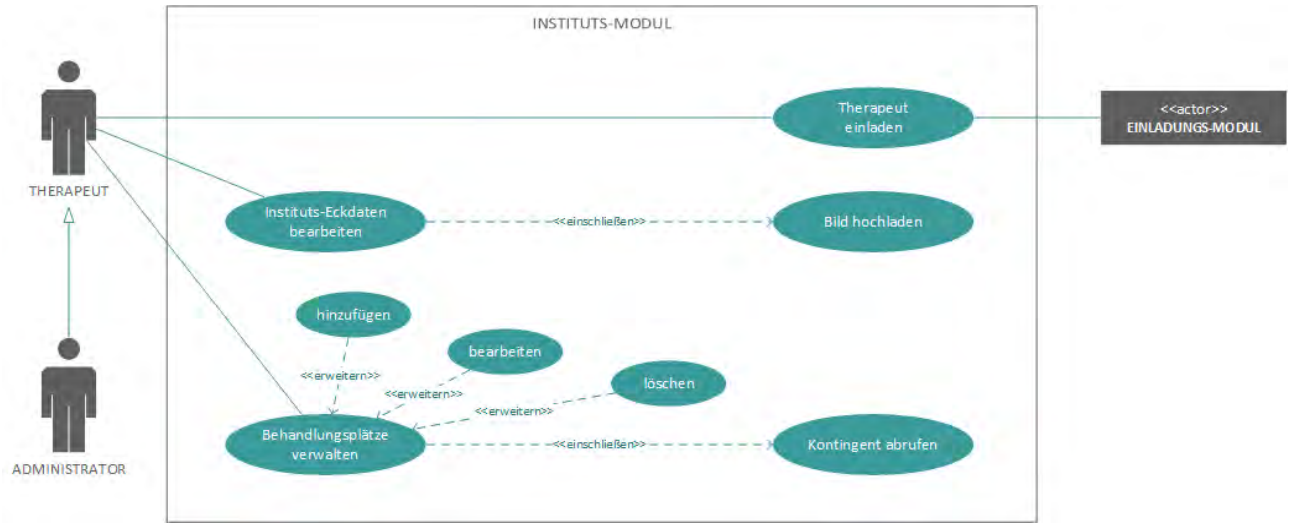
Innerhalb des Account-Moduls können sich Gäste, Patienten, Therapeuten und Administratoren beim System einloggen und ihre Zugriffs-Tokens anfordern, mit welchen sie, innerhalb eines gewissen Zeitraums, Zugriff auf das System bekommen. Des Weiteren können sich Gäste mittels Einladung beim System als Therapeut für ein bestimmtes Institut registrieren oder als Patient für eine bestimmte Behandlung. Administratoren, Therapeuten und Patienten können als angemeldete Benutzer zusätzlich ihr Profil verwalten und gegebenenfalls den Account sperren. Account-Sperrungen von Patienten und Therapeuten benachrichtigen die im Verhältnis stehende Gegenseite und stoppen aktuelle Behandlungen.



Anwendungsfalldiagramm 2: Account-Modul

INSTITUTS-MODUL

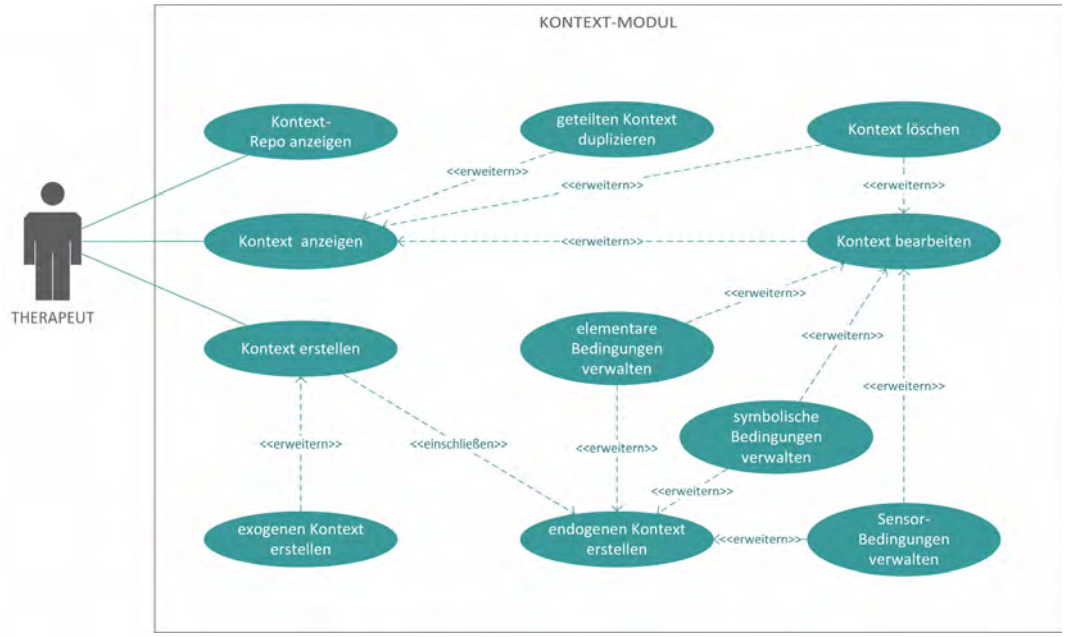
Therapeuten werden innerhalb des Systems in Instituten organisiert, welches für die Marktplatz-Funktion öffentliche Eckdaten inklusive Logo und ihr Angebot an (freien) Behandlungsplätzen bereitstellt. Diese können von Administratoren und Therapeuten des zugehörigen Instituts definiert werden. Außerdem kann ein Therapeut eine Einladung für einen Kollegen erstellen, um diesen zu seinem Institut hinzuzufügen.



Anwendungsfalldiagramm: Instituts-Modul

KONTEXT-MODUL

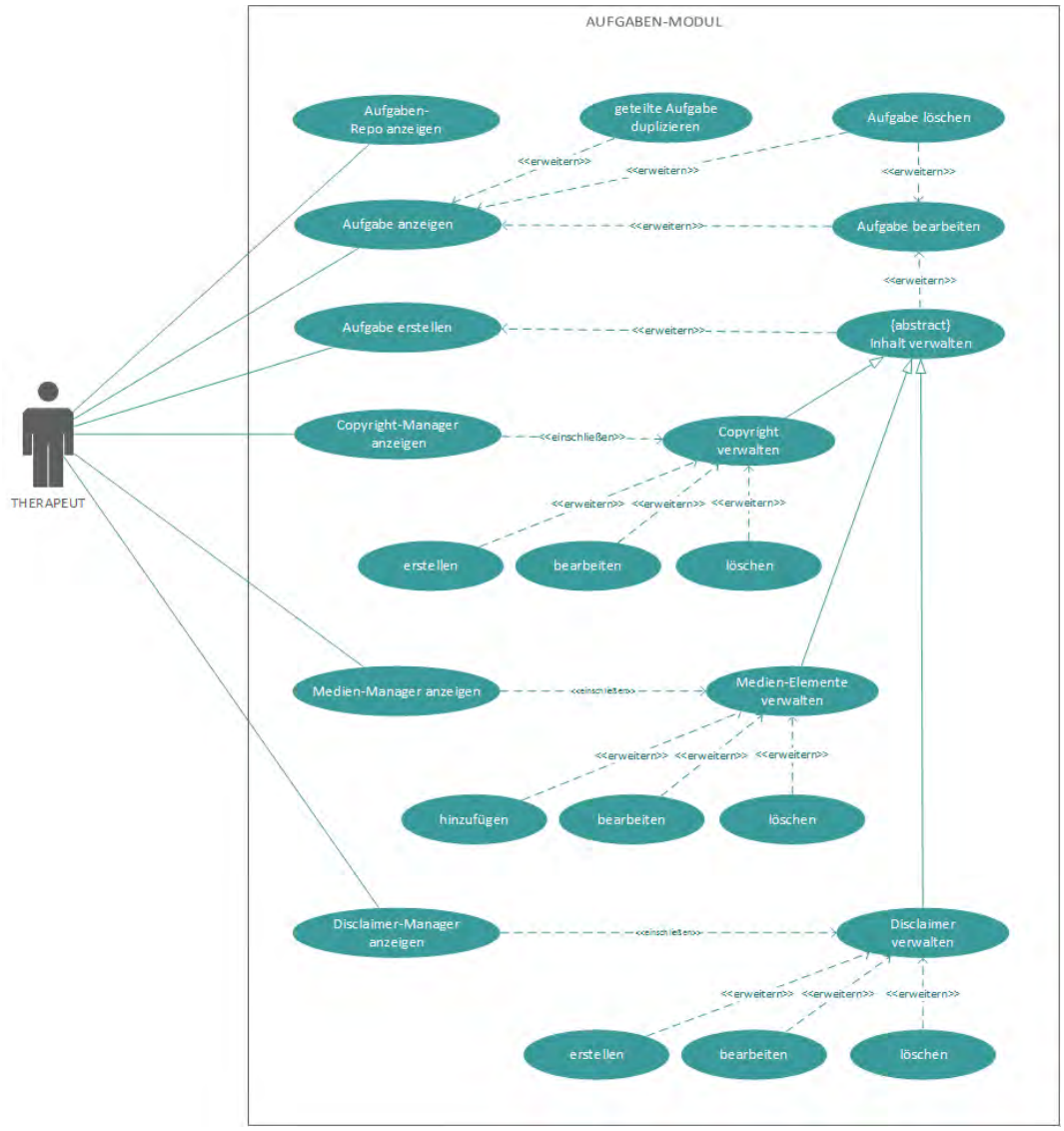
Innerhalb dieses Moduls können Therapeuten endogene und exogene Kontexte (siehe Kapitel 2.1 Fachwissen) in einem Repository verwalten. Endogene Kontexte können dabei Bedingungen hinzufügen, welche in drei Gruppen aufgeteilt werden können. Die elementare Bedingungen, wie Ort, Zeitpunkt oder Konnektivität, beschreiben sensorisch genau überprüfbare Bedingungen, für die jedoch die Auswahl der Sensorik dem System überlassen wird. Die symbolischen Bedingungen beschreiben Bedingungen, die vom Menschen interpretiert und bewertet werden müssen. Sensorische Bedingungen betreffen nur einen bestimmten Sensor und besitzen einen Operator mit Schwellwert, um nicht klar definierte Situationen maschinell zu erkennen. Ein exogener Kontext wird in zukünftigen Systemversionen Daten von externen Dienstleistern erfassen und interpretieren, um eine genauere Situationsbestimmung zu erlauben. Kontexte können innerhalb des Instituts geteilt werden oder nur dem Ersteller zu Verfügung stehen. Bearbeitet oder gelöscht können diese Kontexte nur vom Ersteller. Allerdings können Therapeuten des gleichen Instituts diese Kontexte duplizieren und somit erweitern.



Anwendungsfalldiagramm: Kontext-Modul

AUFGABEN-MODUL

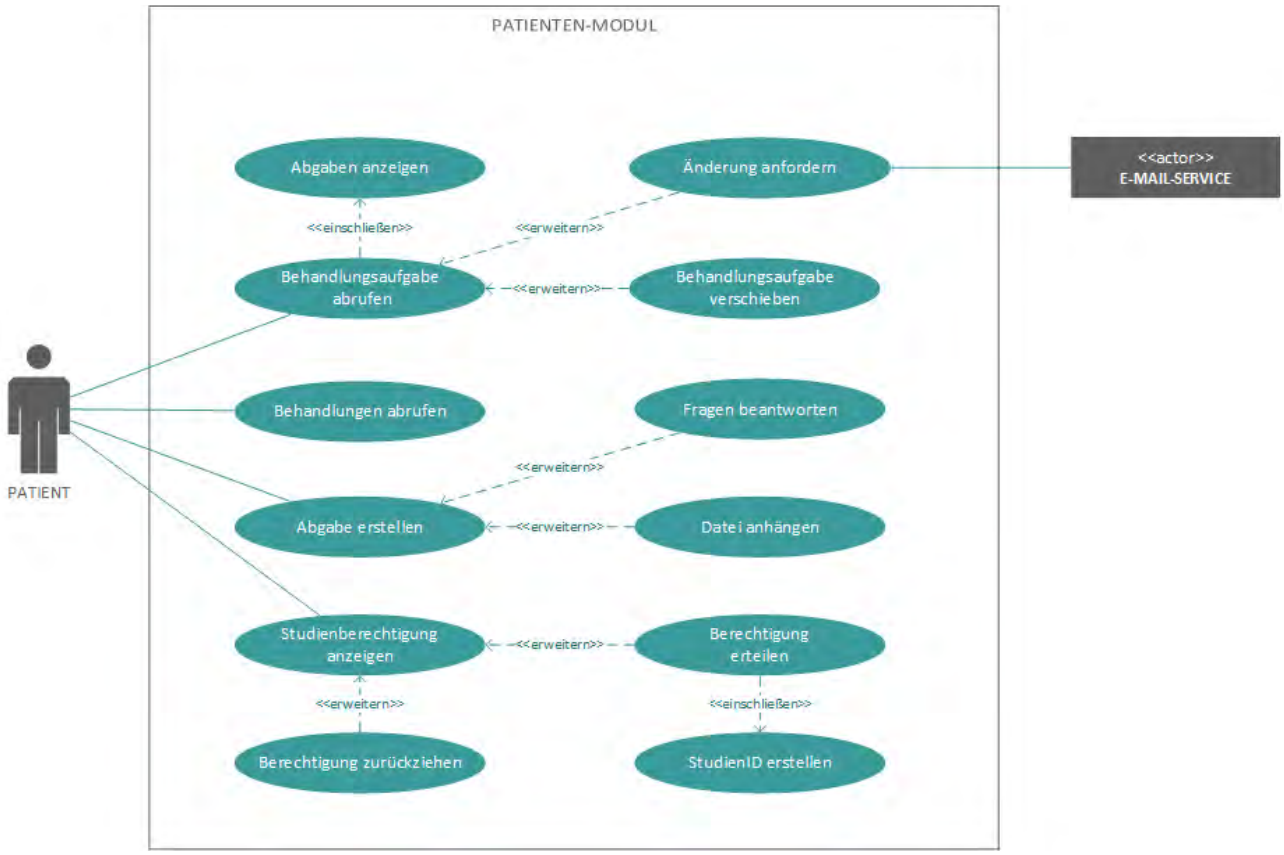
Therapeuten können in einem Repository Aufgaben erstellen und verwalten. Aufgaben beinhalten diverse Inhalte, unter anderem eine Copyright-Angabe, optional diverse Medien-Elemente und einen Disclaimer. Um Inhalte über mehrere Aufgaben hinweg zu verwalten, können alle erstellen Inhalte in jeweils einer Manager-Ansicht verwaltet werden. Aufgaben können innerhalb des Instituts geteilt werden oder nur dem Ersteller zu Verfügung stehen. Bearbeitet oder gelöscht können Aufgaben nur vom Ersteller. Allerdings können Therapeuten des gleichen Instituts diese Aufgaben duplizieren und somit erweitern.



Anwendungsfalldiagramm: Aufgaben-Modul

PATIENTEN-MODUL

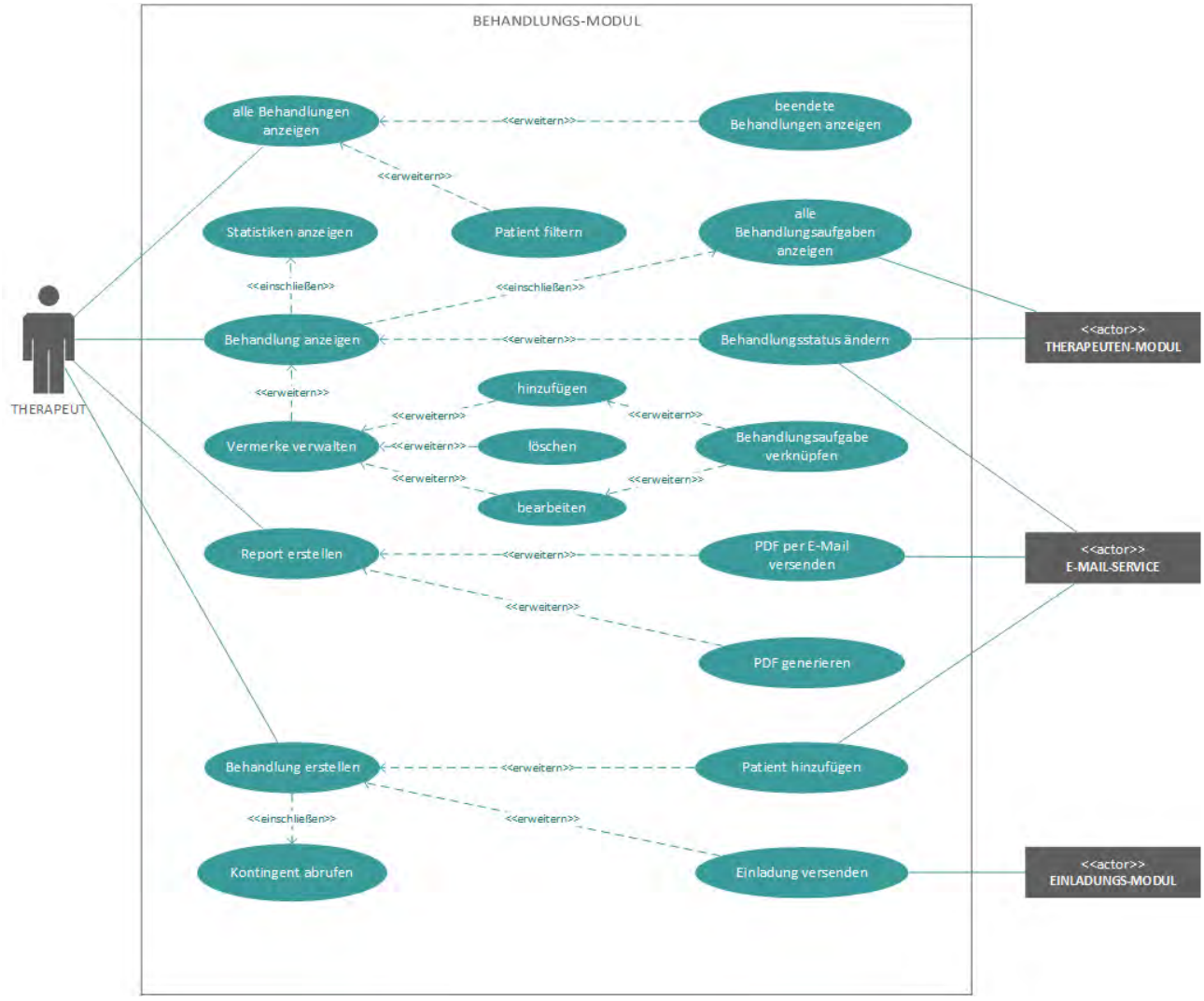
Patienten können über das Patienten-Modul die Daten zu ihren Behandlungen, sowie die aktuellste Version der Behandlungsaufgaben abrufen. Außerdem können zu jeder abgeschlossenen Behandlungsaufgabe die übermittelten Abgaben angezeigt werden. Falls der Patient eine Behandlungsaufgabe zum geplanten Zeitpunkt nicht ausführen kann, so kann er eine Verschiebung vermerken. Der Patient kann Änderungen anfordern, falls Teile der Behandlungsaufgabe angepasst werden müssen, woraufhin der behandelnde Therapeut benachrichtigt wird. Hat der Patient die Behandlungsaufgabe absolviert, kann er eine Abgabe erstellen und die hinterlegte Feedback-Vorlage füllen, indem er Fragen beantwortet oder Dateien anhängt. Der Patient kann seine erhobenen Daten jederzeit für die Forschung bereitstellen oder die erteilte Berechtigung entziehen.



Anwendungsfalldiagramm: Patienten-Modul

BEHANDLUNGS-MODUL

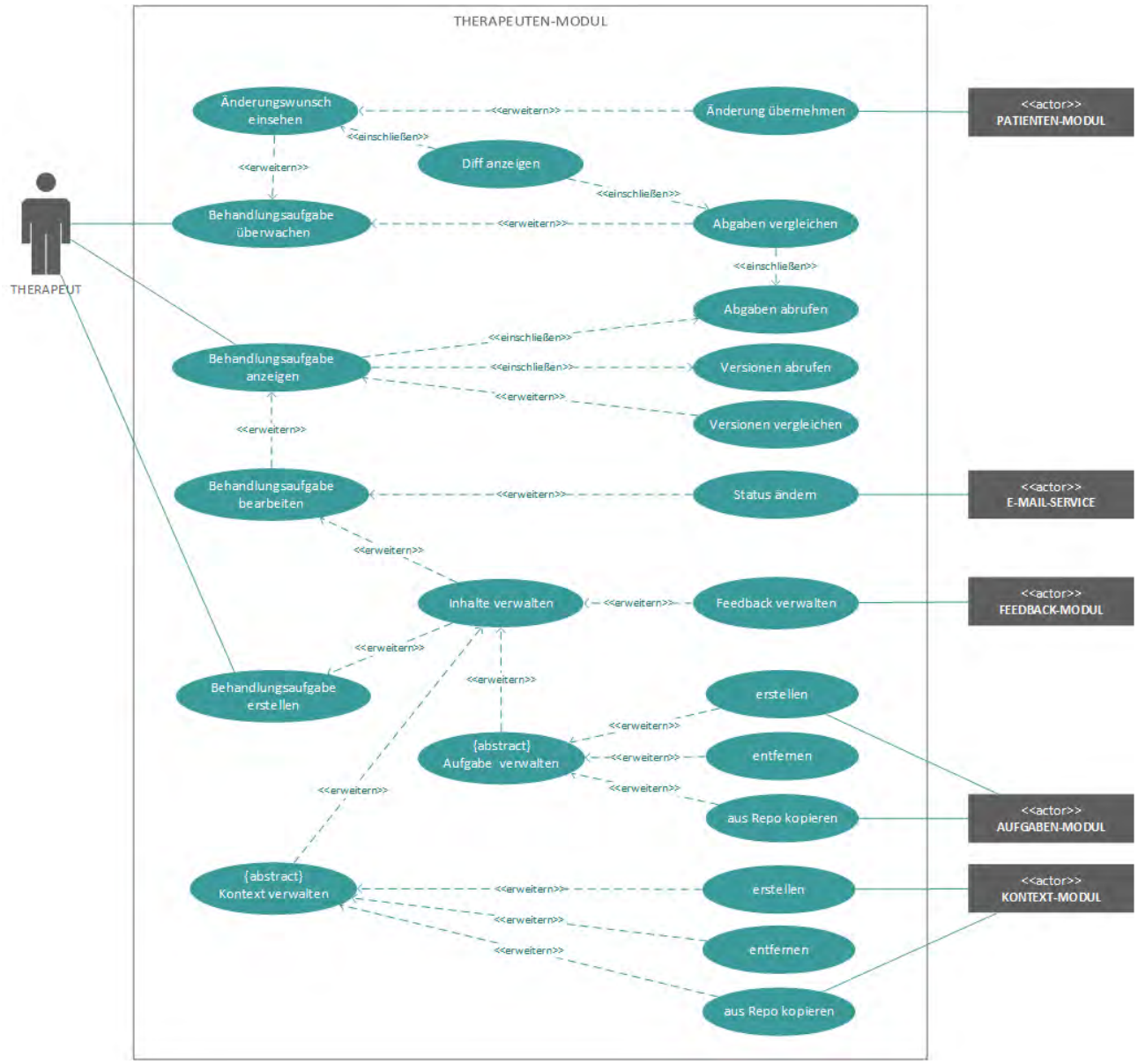
Therapeuten können über das Behandlungs-Modul alle eigenen aktiven und beendeten Behandlungen einsehen und nach Patienten filtern. In einer Detail-Ansicht können sie einzelne Behandlungen öffnen, um sich Behandlungsaufgaben und Behandlungsstatistiken für diese Behandlung anzeigen zu lassen oder den Behandlungsstatus zu ändern. Damit wird die Behandlung beendet und alle laufenden Behandlungsaufgaben gestoppt oder die Behandlung wird erneut gestartet. Außerdem können manuell erstellte oder automatisch erzeugte Vermerke zur laufenden Behandlungen hinzugefügt, bearbeitet oder gelöscht werden, sowie mit Behandlungsaufgaben verknüpft werden. Zusätzlich können Reports aus den Behandlungsaufgaben und -vermerken für einen Patienten erstellt werden und als PDF ausgedruckt oder als E-Mail versendet werden. Über das Einladungs-Modul können außerdem, falls das Institut über ausreichend Kontingent verfügt, Patienten eingeladen werden, um Benutzer des Systems zu werden.



Anwendungsfalldiagramm: Behandlungs-Modul

THERAPEUTEN-MODUL

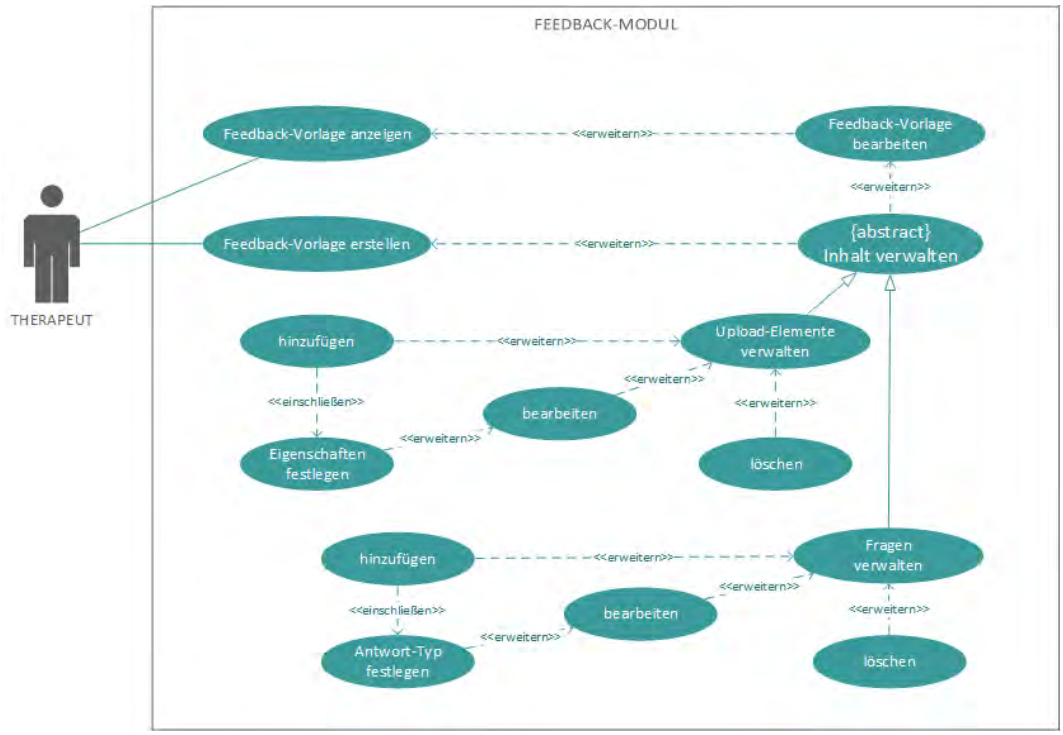
Um Behandlungsaufgaben verwalten und überwachen zu können, können diese Funktionen von einem Therapeuten innerhalb einer Behandlung aufgerufen werden. Außerdem können Versionen von Behandlungsaufgaben verglichen werden, sowie Abgabe für eine bestimmte Version eingesehen werden. Änderungswünsche von Patienten können mit der aktuellen Version verglichen und übernommen werden. Außerdem können Abweichungen von Abgaben zur Aufgabenstellung analysiert und verglichen werden. Muss eine Behandlungsaufgabe bearbeitet oder erstellt werden, so können diverse Inhalte, wie Aufgabe, Kontext oder Feedback, verwaltet werden. Um Inhalte wieder zu verwenden, können Aufgaben und Kontexte aus dem Repository kopiert werden. Es ist außerdem möglich komplett neue Vorlagen für Aufgaben oder Kontexte zu erstellen und diese direkt als Vorlage zu teilen. Behandlungsaufgaben besitzen weiterhin einen Status, um unter anderem die Behandlungsaufgabe zu starten oder zu beenden.



Anwendungsfalldiagramm: Therapeuten-Modul

FEEDBACK-MODUL

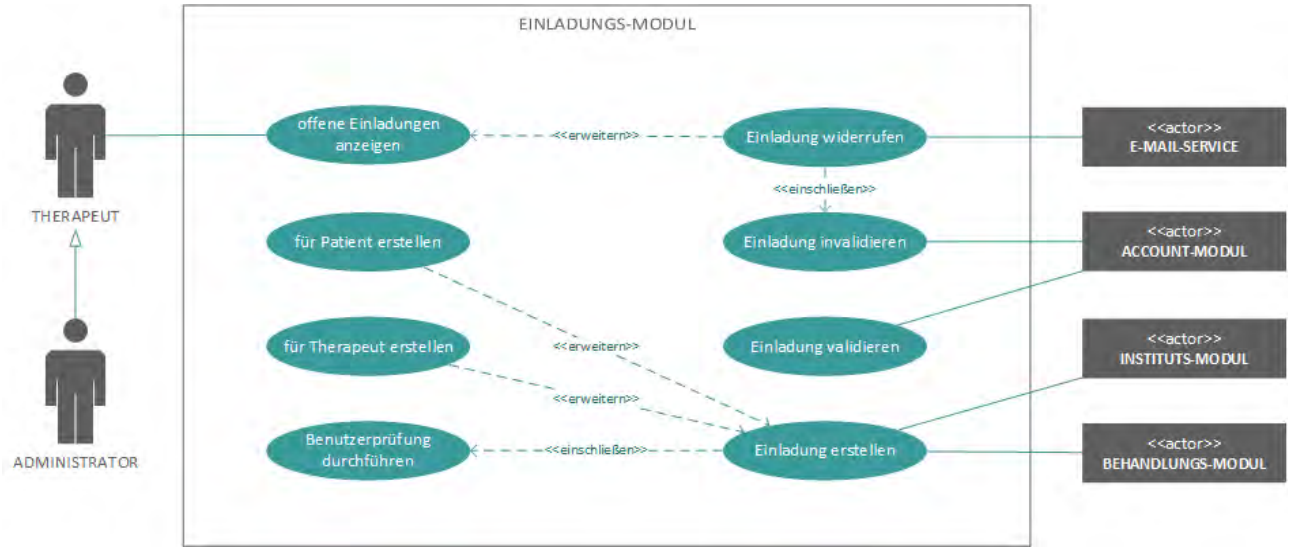
Erstellt oder bearbeitet ein Therapeut eine Behandlungsaufgabe, so kann er dafür eine Feedback-Vorlage erstellt, die der Patient bei jeder Abgabe ausfüllen muss. In einer Feedback-Vorlage können Fragen hinzugefügt, bearbeitet oder gelöscht werden, welche alle Antworten eines gewissen Typs bereitstellen. Außerdem können Medien-Elemente zum Upload hinzugefügt, bearbeitet und gelöscht werden, um Resultate eines vorgegebenen Datei-Typs im System ablegen zu können.



Anwendungsfalldiagramm: Feedback-Modul

EINLADUNGS-MODUL

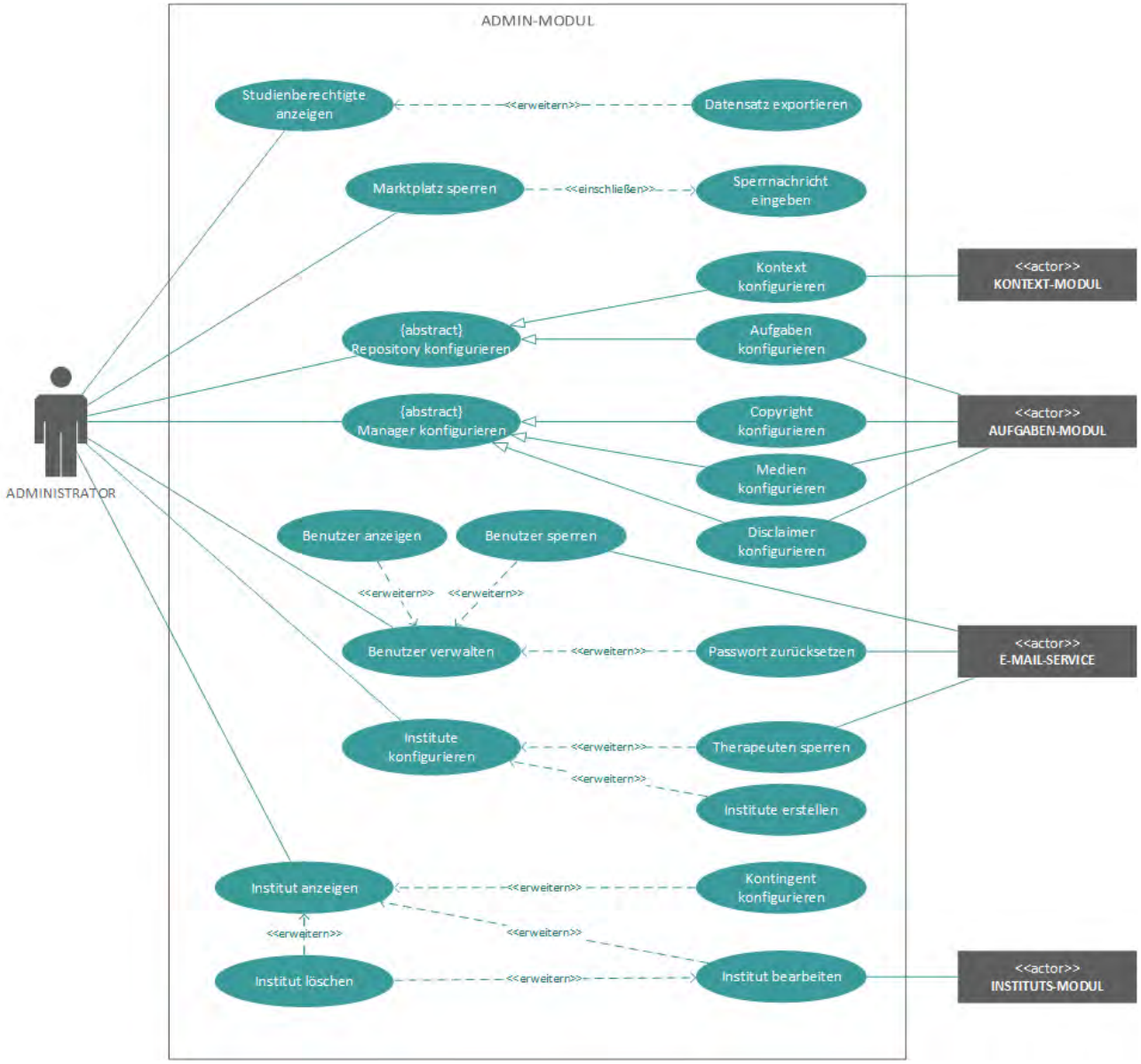
Über das Einladungs-Modul können Therapeuten die erstellten Einladungen ihres Instituts und Administratoren alle Einladungen im System anzeigen und widerrufen. Wird eine Einladung widerrufen, so wird der Betreffende benachrichtigt und die Einladung invalidiert. Außerdem kann das Account-Modul Einladungen validieren lassen, um einen Registriervorgang zu bestätigen und anschließend die Einladung invalidieren. Um Einladungen zentral zu erstellen, können weitere Module Einladungen für Patienten und Therapeuten erstellen und überprüfen, ob ein Benutzer schon im System registriert ist.



Anwendungsfalldiagramm: Einladungs-Modul

ADMIN-MODUL

Der Administrator kann alle Patienten, die eine Studienberechtigung erteilt haben, anzeigen und den kompletten Datensatz aller gelisteter Patienten exportieren. Außerdem kann er den Marktplatz sperren und eine Sperrnachricht zu Wartungszwecken eingeben. Weiterhin kann der Administrator Einstellungen für Repositories und die Manager konfigurieren. Über eine Benutzerliste kann er zusätzlich Benutzer anzeigen, sperren und Passwörter zurücksetzen lassen, woraufhin eine E-Mail an den betreffenden gesendet wird. Um neue Benutzer einzuladen, kann der Administrator Institute erstellen. Außerdem kann für Institute ein Kontingent für Behandlungen festlegen und deren Eckdaten bearbeiten.



Anwendungsfalldiagramm: Administration

2.3 Funktionale Systemanforderungen

Die folgenden Abschnitte definieren die vollständigen funktionalen Anforderungen an das System aus Sicht der Anwender, entnommen aus den Anwendungsfällen des Systemprofils. Für jede Funktion wird eine Priorität (KANN, SOLL, MUSS) angegeben.

ACCOUNT-MODUL

Aufgabe:	**RegistriereTherapeut**
Beschreibung:	*Ein neuer Therapeut muss einen Einladungscode eingeben, um im System registriert zu werden. Dadurch kann er einem bestehenden Institut hinzugefügt werden. Einladungen zu einem Institut können Administratoren und andere Therapeuten aussprechen.*
Beteiligt:	*Gast, Einladungs-Modul*
Priorität:	*MUSS*
Anwendungsfall:	***Therapeut registrieren***

Aufgabe:	**RegistrierePatient**
Beschreibung:	*Ein neuer Patient wird zu einer Behandlung bei seinem Therapeuten hinzugefügt und über einen Einladungscode zugewiesen.*
Beteiligt:	*Gast, Einladungs-Modul*
Priorität:	*MUSS*
Anwendungsfall:	***Patient registrieren***

Aufgabe:	**GeneriereAccessToken**
Beschreibung:	*Eindeutigen Token erstellen, um den Nutzer über diesen Token zu authentifizieren.*
Beteiligt:	*Gast / Benutzer*
Priorität:	*MUSS*
Anwendungsfall:	***Login***

Aufgabe:	**InvalidiereAccessToken**
Beschreibung:	*Eindeutigen Token zerstören.*
Beteiligt:	*Benutzer*
Priorität:	*MUSS*
Anwendungsfall:	***Logout***

Aufgabe:	**ZeigeEigeneStammdaten**
Beschreibung:	*Stammdaten des Benutzers werden ausgegeben.*
Beteiligt:	*Benutzer*
Priorität:	*KANN*
Anwendungsfall:	***Profil verwalten***

Aufgabe:	**ÄndereEigeneStammdaten**
Beschreibung:	*Stammdaten des Benutzers werden eingegeben und überschrieben.*
Beteiligt:	*Benutzer*
Priorität:	*KANN*
Anwendungsfall:	***Stammdaten ändern***

Aufgabe:	**ÄnderePasswort**
Beschreibung:	*Neues Passwort des Benutzers wird eingegeben und überschrieben. Über die Passwort-Ändern-Aufforderung des Administrators kann ein Passwort-Wechsel erzwungen werden.*
Beteiligt:	*Benutzer*
Priorität:	*SOLL*
Anwendungsfall:	***Passwort ändern***

Aufgabe:	**SperreAccount**
Beschreibung:	*Benutzerkonto zu einem gewissen Zeitpunkt mit Eingabe des Passwortes sperren. Patienten oder Therapeuten werden vom System benachrichtigt, falls sich die Gegenseite sperrt. Außerdem wird die aktuelle Behandlung beendet und alle laufenden Aufgaben gesperrt.*
Beteiligt:	*Benutzer, Behandlungs-Modul, Therapeuten-Modul*
Priorität:	*KANN*
Anwendungsfall:	***Account sperren***

GAST-MODUL

Aufgabe:	**HoleAlleAngebote**
Beschreibung:	*Es werden die Behandlungsangebote aller Institute gesucht und zurückgegeben.*
Beteiligt:	*Gast / Benutzer*
Priorität:	*KANN*
Anwendungsfall:	***Marktplatz anzeigen***

Aufgabe:	**ListeAlleAngebote**
Beschreibung:	*Es werden die Behandlungsangebote aller Institute in einer Liste ausgegeben.*
Beteiligt:	*Gast / Benutzer*
Priorität:	*KANN*
Anwendungsfall:	***Angebotsliste anzeigen***

Aufgabe:	**ZeigeAlleAngeboteAufKarte**
Beschreibung:	*Es werden die Behandlungsangebote aller Institute auf einer Karte an der Position der Institute angezeigt.*
Beteiligt:	*Gast / Benutzer*
Priorität:	*KANN*
Anwendungsfall:	***Angebotsliste anzeigen***

Aufgabe:	**SucheFreiePlätze**
Beschreibung:	*Es werden alle Institute nach freien Plätzen für ein Angebot gesucht und zurückgegeben.*
Beteiligt:	*Gast / Benutzer*
Priorität:	*KANN*
Anwendungsfall:	***Freie Plätze suchen***

Aufgabe:	**ZeigeInstitutsDetails**
Beschreibung:	*Gibt die Instituts-Eckdaten und jeweiligen Behandlungsangebote des Instituts zurück.*
Beteiligt:	*Gast / Benutzer*
Priorität:	*KANN*
Anwendungsfall:	***Detail-Ansicht aufrufen***

Aufgabe:	**ZeigeApiDoku**
Beschreibung:	*Gibt die Instituts-Eckdaten zurück.*
Beteiligt:	*Gast / Benutzer*
Priorität:	*SOLL*
Anwendungsfall:	***API-Doku anzeigen***

INSTITUTS-MODUL

Aufgabe:	**ZeigeInstitutsDaten**
Beschreibung:	*Holt alle Informationen zu diesem Institut, sowie die Information für freie Behandlungsplätze, aus der Datenbank und gib diese aus.*
Beteiligt:	*Therapeut / Administrator*
Priorität:	*KANN*
Anwendungsfall:	***Instituts-Eckdaten bearbeiten***

Aufgabe:	**BearbeiteInstitutsDaten**
Beschreibung:	*Überschreibt alle Informationen zu diesem Institut aus der Datenbank. Dazu gehören auch freie Behandlungsplätze in diesem Institut.*
Beteiligt:	*Therapeut / Administrator*
Priorität:	*KANN*
Anwendungsfall:	***Instituts-Eckdaten bearbeiten***

Aufgabe:	**LadeInstitutBildHoch**
Beschreibung:	*Lädt ein Logo oder ein Bild des Instituts hoch, um es im Marktplatz anzuzeigen.*
Beteiligt:	*Therapeut / Administrator*
Priorität:	*KANN*
Anwendungsfall:	***Bild hochladen***

Aufgabe:	**FügeBehandlungsplatzHinzu**
Beschreibung:	*Fügt ein Behandlungsangebot mit Angabe von freien Plätzen hinzu.*
Beteiligt:	*Therapeut / Administrator*
Priorität:	*KANN*
Anwendungsfall:	***Behandlungsplätze verwalten***

Aufgabe:	**BearbeiteBehandlungsplatz**
Beschreibung:	*Aktuallisiere ein Behandlungsangebot mit Angabe von freien Plätzen.*
Beteiligt:	*Therapeut / Administrator*
Priorität:	*KANN*
Anwendungsfall:	***Behandlungsplätze verwalten***

Aufgabe:	**LöscheBehandlungsplatz**
Beschreibung:	*Löscht ein Behandlungsangebot aus diesem Institut.*
Beteiligt:	*Therapeut / Administrator*
Priorität:	*KANN*
Anwendungsfall:	***Behandlungsplätze verwalten***

Aufgabe:	**HoleVerfügbaresKontingent**
Beschreibung:	*Hole das verfügbare Kontingent dieses Instituts und gebe die Differenz mit den aktuell belegten Plätzen zurück.*
Beteiligt:	*Therapeut / Administrator*
Priorität:	*KANN*
Anwendungsfall:	***Kontingent abrufen***

Aufgabe:	**LadeTherapeutEin**
Beschreibung:	*Ein Therapeut des Instituts kann eine Einladung an seinen Kollegen erstellen, wobei er eine E-Mail-Adresse und ein Ablaufdatum angeben muss. Mittels des generierten Einladungscode kann sich der Therapeut zu diesem Institut hinzufügen.*
Beteiligt:	*Therapeut, Einladungs-Modul*
Priorität:	*KANN*
Anwendungsfall:	***Therapeut einladen***

KONTEXT-MODUL

Aufgabe:	**ZeigeKontextRepository**
Beschreibung:	*Es werden alle eigenen Kontexte des Therapeuten, sowie die Kontexte aller Therapeuten des Instituts, in der Datenbank gesucht und zurückgegeben.*
Beteiligt:	*Therapeut*
Priorität:	*MUSS*
Anwendungsfall:	***Kontext-Repo anzeigen***

Aufgabe:	**ZeigeKontext**
Beschreibung:	*Einen eigenen Kontext des Therapeuten oder geteilten Kontext aller Therapeuten des Instituts, inklusive aller definierten Bedingungen, wird in der Datenbank gesucht und inklusive aller Versionen zurückgegeben.*
Beteiligt:	*Therapeut*
Priorität:	*MUSS*
Anwendungsfall:	***Kontext anzeigen***

Aufgabe:	**DupliziereKontext**
Beschreibung:	*Ein Therapeut kann Kontexte, die innerhalb des Instituts geteilt werden, duplizieren und somit erweitern oder anpassen. Gleichzeitig wird für den neuen Kontext eine neue Version erstellt.*
Beteiligt:	*Therapeut*
Priorität:	*MUSS*
Anwendungsfall:	***Geteilten Kontext duplizieren***

Aufgabe:	**LöscheKontext**
Beschreibung:	*Ein Therapeut löscht seine eigenen Kontext, inklusive aller dazugehörenden Bedingungen, aus der Datenbank. Geteilte Kontexte sind nun nicht mehr im Repository sichtbar.*
Beteiligt:	*Therapeut*
Priorität:	*MUSS*
Anwendungsfall:	***Kontext löschen***

Aufgabe:	**BearbeiteKontext**
Beschreibung:	*Ein Therapeut bearbeitet seinen eigenen Kontext und kann die elementaren, symbolischen oder sensorischen Bedingungen für diesen Kontext verwalten. Änderungen werden als neue Version gespeichert, um Anpassungen abzubilden.*
Beteiligt:	*Therapeut*
Priorität:	*MUSS*
Anwendungsfall:	***Kontext bearbeiten***

Aufgabe:	**ErstelleKontextVersion**
Beschreibung:	*Änderung an eines Kontextes werden als neue Version in der Datenbank abgelegt und den Zeiger auf die aktuelle Version erneuert.*
Beteiligt:	*Therapeut*
Priorität:	*MUSS*
Anwendungsfall:	***Kontext erstellen, Kontext bearbeiten***

Aufgabe:	**ErstelleEndogenenKontext**
Beschreibung:	*Kontexte mit den dazugehörenden elementaren, symbolischen oder sensorischen Bedingungen werden, wie in Kapitel 2.1 Fachwissen definiert, erstellt. Gleichzeitig wird die initiale Version erstellt.*
Beteiligt:	*Therapeut*
Priorität:	*MUSS*
Anwendungsfall:	***Endogenen Kontext erstellen***

Aufgabe:	**ErstelleExogenenKontext**
Beschreibung:	*Es wird eine Quelle für den exogenen Kontext ausgewählt und die passende Bedingung definiert.*
Beteiligt:	*Therapeut*
Priorität:	*KANN*
Anwendungsfall:	***Exogenen Kontext erstellen***

AUFGABEN-MODUL

Aufgabe:	**ZeigeAufgabenRepository**
Beschreibung:	*Es werden alle Aufgaben des Therapeuten, sowie alle geteilten Aufgaben des gleichen Instituts, jedoch exklusive der dazugehörenden Medien-Elemente und rechtlichen Angaben, in der Datenbank gesucht und zurückgegeben.*
Beteiligt:	*Therapeut*
Priorität:	*MUSS*
Anwendungsfall:	***Aufgaben-Repo anzeigen***

Aufgabe:	**ZeigeAufgabe**
Beschreibung:	*Eine eigene Aufgabe oder geteilte Aufgabe, inklusive Medien-Elementen und rechtlichen Angaben, wird in der Datenbank gesucht und für jede Versionen zurückgegeben.*
Beteiligt:	*Therapeut*
Priorität:	*MUSS*
Anwendungsfall:	***Aufgabe anzeigen***

Aufgabe:	**DupliziereAufgabe**
Beschreibung:	*Aufgaben, die innerhalb des Instituts geteilt werden, duplizieren und somit erweitern oder anpassen. Gleichzeitig wird für die neue Aufgabe eine neue Version erstellt.*
Beteiligt:	*Therapeut*
Priorität:	*MUSS*
Anwendungsfall:	***Geteilte Aufgabe duplizieren***

Aufgabe:	**LöscheAufgabe**
Beschreibung:	*Ein Therapeut löscht seine eigene Aufgabe, inklusive aller dazugehörenden Medien-Elementen und rechtlichen Angaben, aus der Datenbank. Geteilte Aufgaben sind nun nicht mehr im Repository sichtbar.*
Beteiligt:	*Therapeut*
Priorität:	*MUSS*
Anwendungsfall:	***Aufgabe löschen***

Aufgabe:	**BearbeiteAufgabe**
Beschreibung:	*Ein Therapeut bearbeitet seine eigene Aufgabe und kann gleichzeitig auch alle angehängten Inhalte wie Medien-Elementen und rechtlichen Angaben verwalten. Änderungen werden als neue Version gespeichert, um Anpassungen abzubilden.*
Beteiligt:	*Therapeut*
Priorität:	*MUSS*
Anwendungsfall:	***Aufgabe bearbeiten***

Aufgabe:	**ErstelleAufgabe**
Beschreibung:	*Aufgaben können mit den dazugehörenden Parametern, wie in Kapitel 2.1 Fachwissen definiert, erstellt werden. Zusätzlich können Inhalte aus dem Manager hinzugefügt oder direkt erstellt und bearbeitet werden. Außerdem kann man auswählen, ob diese Aufgabe innerhalb des Instituts geteilt werden soll. Gleichzeitig wird eine initiale Version erstellt.*
Beteiligt:	*Therapeut*
Priorität:	*MUSS*
Anwendungsfall:	***Aufgabe erstellen***

Aufgabe:	**ErstelleAufgabenVersion**
Beschreibung:	*Änderung an einer Aufgabe werden als neue Version in der Datenbank abgelegt und den Zeiger auf die aktuelle Version erneuert.*
Beteiligt:	*Therapeut*
Priorität:	*MUSS*
Anwendungsfall:	***Aufgabe erstellen, Aufgabe bearbeiten***

Aufgabe:	**VerknüpfeInhaltMitAufgabe**
Beschreibung:	*Verknüpfte ein Inhalts-Elemente mit der gewählten Aufgabe (URL oder Kopie).*
Beteiligt:	*Therapeut*
Priorität:	*MUSS*
Anwendungsfall:	***Inhalt verwalten***

Aufgabe:	**ZeigeErstellteCopyrights**
Beschreibung:	*Alle vom Benutzer erstellten Copyrights werden gesucht und zurückgegeben.*
Beteiligt:	*Therapeut*
Priorität:	*SOLL*
Anwendungsfall:	***Copyright-Manager anzeigen***

Aufgabe:	**ZeigeCopyright**
Beschreibung:	*Ein bestimmtes Copyright wird in der Datenbank gesucht und zurückgegeben.*
Beteiligt:	*Therapeut*
Priorität:	*MUSS*
Anwendungsfall:	***Copyright verwalten***

Aufgabe:	**ErstelleCopyright**
Beschreibung:	*Benutzer erstellt ein Copyright und kann es direkt mit einer Aufgabe verknüpfen.*
Beteiligt:	*Therapeut*
Priorität:	*MUSS*
Anwendungsfall:	***Copyright verwalten***

Aufgabe:	**BearbeiteCopyright**
Beschreibung:	*Benutzer bearbeitet ein bereits erstelltes Copyright.*
Beteiligt:	*Therapeut*
Priorität:	*MUSS*
Anwendungsfall:	***Copyright verwalten***

Aufgabe:	**LöscheCopyright**
Beschreibung:	*Benutzer löscht ein bestehendes Copyright aus der Aufgabe. Die Vorlagen im Manager sind davon nicht betroffen und müssen separat gelöscht werden.*
Beteiligt:	*Therapeut*
Priorität:	*MUSS*
Anwendungsfall:	***Copyright verwalten***

Aufgabe:	**ZeigeErstellteMedienElemente**
Beschreibung:	*Alle vom Benutzer erstellten Medien-Elemente werden gesucht und zurückgegeben.*
Beteiligt:	*Therapeut*
Priorität:	*SOLL*
Anwendungsfall:	***Medien-Manager anzeigen***

Aufgabe:	**ZeigeMedienElement**
Beschreibung:	*Ein bestimmtes Medien-Element wird in der Datenbank gesucht und zurückgegeben.*
Beteiligt:	*Therapeut*
Priorität:	*MUSS*
Anwendungsfall:	***Medien-Elemente verwalten***

Aufgabe:	**FügeMedienElementHinzu**
Beschreibung:	*Benutzer fügt ein Element hinzu und kann es direkt mit einer Aufgabe verknüpfen.*
Beteiligt:	*Therapeut*
Priorität:	*MUSS*
Anwendungsfall:	***Medien-Elemente verwalten***

Aufgabe:	**BearbeiteMedienElement**
Beschreibung:	*Benutzer bearbeitet ein bereits erstelltes Medien-Element.*
Beteiligt:	*Therapeut*
Priorität:	*MUSS*
Anwendungsfall:	***Medien-Elemente verwalten***

Aufgabe:	**LöscheMedienElement**
Beschreibung:	*Benutzer löscht ein bestehendes Medien-Element aus dem System.*
Beteiligt:	*Therapeut*
Priorität:	*MUSS*
Anwendungsfall:	***Medien-Elemente verwalten***

Aufgabe:	**ZeigeErstellteDisclaimer**
Beschreibung:	*Alle vom Benutzer erstellten Disclaimer werden in der Datenbank gesucht und zurückgegeben.*
Beteiligt:	*Therapeut*
Priorität:	*SOLL*
Anwendungsfall:	***Disclaimer-Manager anzeigen***

Aufgabe:	**ZeigeDisclaimer**
Beschreibung:	*Ein bestimmter Disclaimer wird in der Datenbank gesucht und zurückgegeben.*
Beteiligt:	*Therapeut*
Priorität:	*MUSS*
Anwendungsfall:	***Disclaimer verwalten***

Aufgabe:	**ErstelleDisclaimer**
Beschreibung:	*Benutzer erstellt ein Disclaimer und kann es direkt mit einer Aufgabe verknüpfen.*
Beteiligt:	*Therapeut*
Priorität:	*MUSS*
Anwendungsfall:	***Disclaimer verwalten***

Aufgabe:	**BearbeiteDisclaimer**
Beschreibung:	*Benutzer bearbeitet einen bereits erstellten Disclaimer.*
Beteiligt:	*Therapeut*
Priorität:	*MUSS*
Anwendungsfall:	***Disclaimer verwalten***

Aufgabe:	**LöscheDisclaimer**
Beschreibung:	*Benutzer löscht einen bestehenden Disclaimer. Die Vorlagen im Manager sind davon nicht betroffen und müssen separat gelöscht werden.*
Beteiligt:	*Therapeut*
Priorität:	*MUSS*
Anwendungsfall:	***Disclaimer verwalten***

PATIENTEN-MODUL

Aufgabe:	**HoleBehandlungen**
Beschreibung:	*Daten zu allen Behandlungen eines Patienten, wie beispielsweise laufende Behandlungsaufgaben, behandelnde Therapeuten und Metadaten zur Behandlung, werden abgerufen.*
Beteiligt:	*Patient*
Priorität:	*MUSS*
Anwendungsfall:	***Behandlungen abrufen***

Aufgabe:	**HoleBehandlungsaufgabe**
Beschreibung:	*Die aktuellste Version einer bestimmten Behandlungsaufgabe, inklusive Medien-Elementen, Copyright-Angabe und Disclaimer, wird in der Datenbank gesucht und zurückgegeben.*
Beteiligt:	*Patient*
Priorität:	*MUSS*
Anwendungsfall:	***Behandlungsaufgabe abrufen***

Aufgabe:	**ZeigeAbgaben**
Beschreibung:	*Alle Abgaben zu einer Version der Behandlungsaufgabe wird inklusive Feedback abgerufen und zurückgegeben.*
Beteiligt:	*Patient*
Priorität:	*MUSS*
Anwendungsfall:	***Abgaben anzeigen***

Aufgabe:	**FordereBehandlungsÄnderung**
Beschreibung:	*Änderungen der Aufgabenparameter oder des Kontextes einer Behandlungsaufgabe werden vom Patienten eingegeben und als Vorschlag dem Therapeuten zugesendet.*
Beteiligt:	*Patient, E-Mail-Service*
Priorität:	*MUSS*
Anwendungsfall:	***Änderung anfordern***

Aufgabe:	**VerschiebeBehandlungsaufgabe**
Beschreibung:	*Nach der Erinnerung des Systems nimmt der Patient eine zeitliche Änderungen der Behandlungsaufgaben-Erinnerung vor, welche zuerst lokal auf seinem gesichert und nach der Behandlungsaufgaben-Absolvierung dem System mit der Abgabe zugeschickt wird.*
Beteiligt:	*Patient*
Priorität:	*MUSS*
Anwendungsfall:	***Behandlungsaufgabe verschieben***

Aufgabe:	**ErstelleAbgabe**
Beschreibung:	*Nach der Aufgabenabsolvierung sendet der Patient die erhobenen Daten an das System.*
Beteiligt:	*Patient*
Priorität:	*MUSS*
Anwendungsfall:	***Abgabe erstellen***

Aufgabe:	**BeantworteFeedbackFragen**
Beschreibung:	*Die beantworteten Fragen aus der Feedback-Vorlage werden zu der Abgabe hinzugefügt.*
Beteiligt:	*Patient*
Priorität:	*MUSS*
Anwendungsfall:	***Fragen beantworten***

Aufgabe:	**HängeDateiAn**
Beschreibung:	*Dateien werden hochgeladen und zu der Abgabe hinzugefügt.*
Beteiligt:	*Patient*
Priorität:	*MUSS*
Anwendungsfall:	***Datei anhängen***

Aufgabe:	**ZeigeBetroffeneDatensätze**
Beschreibung:	*Falls der Patient eine Studienberechtigung erteilt hat, so werden alle betroffenen Daten ausgelesen und angezeigt.*
Beteiligt:	*Patient*
Priorität:	*KANN*
Anwendungsfall:	***Studienberechtigung anzeigen***

Aufgabe:	**ErteileBerechtigung**
Beschreibung:	*Dem Patienten wird eine einmalige StudienID zugeteilt und somit die Berechtigung zur Studienteilnahme erteilt.*
Beteiligt:	*Patient*
Priorität:	*KANN*
Anwendungsfall:	***Berechtigung erteilen***

Aufgabe:	**EntzieheBerechtigung**
Beschreibung:	*Der Patient entzieht die Berechtigung zur Verwendung seiner erhobenen Daten. Die erteilte StudienID wird daraufhin entzogen.*
Beteiligt:	*Patient*
Priorität:	*KANN*
Anwendungsfall:	***Berechtigung zurückziehen***

BEHANDLUNGS-MODUL

Aufgabe:	**HoleAlleAktivenBehandlungen**
Beschreibung:	*Es werden die laufenden Behandlungen des Therapeuten gesucht und zurückgegeben.*
Beteiligt:	*Therapeut*
Priorität:	*MUSS*
Anwendungsfall:	***alle Behandlungen anzeigen***

Aufgabe:	**HoleAlleBeendetenBehandlungen**
Beschreibung:	*Es werden die beendeten Behandlungen des Therapeuten gesucht und zurückgegeben.*
Beteiligt:	*Therapeut*
Priorität:	*MUSS*
Anwendungsfall:	***beendete Behandlungen anzeigen***

Aufgabe:	**FilterBehandlungenNachPatient**
Beschreibung:	*Es werden nur noch Behandlungen für einen bestimmten Patienten angezeigt*
Beteiligt:	*Therapeut*
Priorität:	*SOLL*
Anwendungsfall:	***Patient filtern***

Aufgabe:	**HoleBehandlung**
Beschreibung:	*Es werden die Daten, inklusive aller Behandlungsaufgaben, für eine bestimmte Behandlung des Therapeuten gesucht und angezeigt.*
Beteiligt:	*Therapeut*
Priorität:	*MUSS*
Anwendungsfall:	***Behandlung anzeigen***

Aufgabe:	**ZeigeAlleBehandlungsaufgaben**
Beschreibung:	*Zu einem ausgewählten Patienten werden alle Versionen der Behandlungsaufgaben, inklusive der verknüpften Medien-Elementen und rechtlichen Angaben, sowie die zugehörigen Abgaben in der Datenbank gesucht und zurückgegeben.*
Beteiligt:	*Therapeut, Therapeuten-Modul*
Priorität:	*MUSS*
Anwendungsfall:	***Alle Behandlungsaufgaben anzeigen***

Aufgabe:	**BerechneStatistik**
Beschreibung:	*Für die Behandlung werden, anhand aller Behandlungsaufgaben, Statistiken berechnet, welche dem Therapeuten einen besseren Überblick über den Behandlungsstand und -erfolg geben.*

Beteiligt: *Therapeut*

Priorität: *MUSS*

Anwendungsfall: ***beendete Behandlungen anzeigen***

Aufgabe: **ÄndereBehandlungsstatus**

Beschreibung: *Für eine Behandlung kann der Status auf „created", „started", „stopped" oder „completed" gestellt werden. Wird eine Behandlung beendet, so werden alle laufenden Aufgaben gestoppt und es kann keine weitere Abgabe erfolgen. In jedem Fall wird der Patient über die Änderung benachrichtigt.*

Beteiligt: *Therapeut*

Priorität: *MUSS*

Anwendungsfall: ***Behandlung anzeigen***

Aufgabe: **ErstelleBehandlungsvermerk**

Beschreibung: *Für einen Patienten wird während der Behandlung ein Vermerk, der mit einer Behandlungsaufgabe des aktuellen Patienten verknüpft werden kann, erstellt.*

Beteiligt: *Therapeut, Patienten-Modul*

Priorität: *KANN*

Anwendungsfall: ***Vermerke verwalten***

Aufgabe: **BearbeiteBehandlungsvermerk**

Beschreibung: *Ein bestimmter Behandlungsvermerk wird überschrieben oder mit einer anderen Behandlungsaufgabe verknüpft.*

Beteiligt: *Therapeut, Patienten-Modul*

Priorität: *KANN*

Anwendungsfall: ***Vermerke verwalten***

Aufgabe: **LöscheBehandlungsvermerk**

Beschreibung: *Manuell oder automatisch erstellter Behandlungsvermerk wird aus dem System gelöscht.*

Beteiligt: *Therapeut, Patienten-Modul*

Priorität: *KANN*

Anwendungsfall: ***Vermerke verwalten***

Aufgabe: **ErstelleReport**

Beschreibung: *Für alle vom Benutzer ausgewählten Behandlungsaufgaben wird ein Report mit den Behandlungsvermerken über den Behandlungsverlauf erstellt.*

Beteiligt: *Therapeut*

Priorität: *MUSS*

Anwendungsfall: ***Report erstellen***

Aufgabe:	**GenerierePdfAusReport**
Beschreibung:	*Aus einem erstellten Report wird ein PDF generiert.*
Beteiligt:	*Therapeut*
Priorität:	*KANN*
Anwendungsfall:	***PDF generieren***

Aufgabe:	**VersendePdfAusReport**
Beschreibung:	*Aus einem erstellten Report wird ein PDF generiert und mit einem variablen Text zu einer angegebenen E-Mail-Adresse gesendet.*
Beteiligt:	*Therapeut, E-Mail-Service*
Priorität:	*KANN*
Anwendungsfall:	***PDF per E-Mail versenden***

Aufgabe:	**ErstelleBehandlung**
Beschreibung:	*Falls das Institut des Therapeuten über genügend Kontingent verfügt, so kann eine neue Behandlung erstellt werden. Die Zuordnung zu einem Patienten erfolgt über die E-Mail-Adresse. Entweder ist der Patient schon registriert, dann werden beide Parteien über ein erfolgreiches hinzufügen benachrichtigt oder es kann eine Einladung über das Einladungs-Modul versendet werden. Diese Entscheidung erfolgt automatisch.*
Beteiligt:	*Therapeut, Einladungs-Modul*
Priorität:	*MUSS*
Anwendungsfall:	***Behandlung erstellen***

Aufgabe:	**HoleVerfügbaresKontingent**
Beschreibung:	*Hole das verfügbare Kontingent dieses Instituts und gebe die Differenz mit den aktuell belegten Plätzen zurück.*
Beteiligt:	*Therapeut*
Priorität:	*KANN*
Anwendungsfall:	***Kontingent abrufen***

THERAPEUTEN-MODUL

Aufgabe:	**HoleBehandlungsaufgabe**
Beschreibung:	*Die aktuellste Version und alle Abgaben einer bestimmten Behandlungsaufgabe, inklusive Medien-Elementen und rechtlichen Angaben, werden gesucht und zurückgegeben.*
Beteiligt:	*Therapeut*
Priorität:	*MUSS*
Anwendungsfall:	***Behandlungsaufgabe anzeigen***

Aufgabe:	**ÄndereBehandlungsaufgabe**
Beschreibung:	*Behandlungsaufgabe wird entweder im Status geändert (created, in progress, started, stopped, completed) und behält dabei die aktuelle Version oder es werden ihre Inhalte verwaltet. Bei letzterem wird eine neue Version für diese Behandlungsaufgabe erstellt und der Zeiger aktualisiert.* *Zusätzlich wird eine Feedback-Vorlage verwaltet, um die Qualität der Behandlung zu bemessen (siehe Feedback-Modul).*
Beteiligt:	*Therapeut, Feedback-Modul, Aufgaben-Modul, Kontext-Modul*
Priorität:	*MUSS*
Anwendungsfall:	***Behandlungsaufgabe bearbeiten, Status ändern***

Aufgabe:	**BeendeBehandlungsaufgabe**
Beschreibung:	*Es wird eine Nachricht an den Patienten gesendet. Anschließend kann ein Behandlungsvermerk für die Behandlungsaufgabe eingetragen werden.*
Beteiligt:	*Therapeut, E-Mail-Service*
Priorität:	*MUSS*
Anwendungsfall:	***Status ändern***

Aufgabe:	**HoleAlleBehandlungsaufgabenVersionen**
Beschreibung:	*Alle Versionen der Behandlungsaufgabe, inklusive aller Inhalte, werden zurückgegeben.*
Beteiligt:	*Therapeut*
Priorität:	*SOLL*
Anwendungsfall:	***Versionen abrufen***

Aufgabe:	**HoleAlleAbgaben**
Beschreibung:	*Alle Abgaben der Behandlungsaufgabe, inklusive aller Uploads, werden geladen und zurückgegeben.*
Beteiligt:	*Therapeut*
Priorität:	*SOLL*
Anwendungsfall:	***Abgaben abrufen***

Aufgabe:	**VergleicheBehandlungsaufgabenVersionen**
Beschreibung:	*Es wird ein Diff aus allen benachbarten Versionen erstellt und die Änderungen ähnlich wie in Git angezeigt.*
Beteiligt:	*Therapeut*
Priorität:	*KANN*
Anwendungsfall:	***Versionen vergleichen***

Aufgabe:	**VergleicheBehandlungsaufgabeMitWunsch**
Beschreibung:	*Der Änderungswunsch des Patienten wird mit der betreffenden Behandlungsaufgabe verglichen und Unterschiede angezeigt.*
Beteiligt:	*Therapeut*
Priorität:	*MUSS*
Anwendungsfall:	***Behandlungsaufgabe überwachen, Änderungswunsch einsehen***

Aufgabe:	**ÄndereBehandlungsaufgabeAusWunsch**
Beschreibung:	*Daten zur Behandlungsaufgabe werden aus einem Änderungswunsch des Patienten entnommen und als neue, aktuelle Version dieser Behandlungsaufgabe im System gespeichert.*
Beteiligt:	*Therapeut, Patienten-Modul*
Priorität:	*MUSS*
Anwendungsfall:	***Behandlungsaufgabe überwachen, Änderung übernehmen***

Aufgabe:	**VergleicheBehandlungsaufgabeMitAbgabe**
Beschreibung:	*Eine Behandlungsaufgabe wird mit der Abgabe zur passenden Version verglichen, um Unterschiede, beziehungsweise Anpassungen durch den Patienten, zu erkennen und gegebenenfalls die Aufgabe zu bearbeiten.*
Beteiligt:	*Therapeut*
Priorität:	*MUSS*
Anwendungsfall:	***Behandlungsaufgabe überwachen, Abgaben vergleichen***

Aufgabe:	**ErstelleBehandlungsaufgabe**
Beschreibung:	*Zu einem ausgewählten Patienten wird eine neue Behandlungsaufgabe hinzugefügt. Dabei wird entweder eine Aufgabe neu erstellt (siehe Seite 26ff) oder eine bereits vorhandene aus dem Aufgaben-Repository kopiert. Gleiches gilt für den Kontext, welcher entweder komplett neu erstellt (siehe Seite 24ff) oder aus dem Kontext-Repository kopiert werden kann.* *Zusätzlich wird eine Feedback-Vorlage verwaltet, um die Qualität der Behandlung zu bemessen (siehe Feedback-Modul).*
Beteiligt:	*Therapeut, Patienten-Modul, Aufgaben-Modul, Kontext-Modul*
Priorität:	*MUSS*
Anwendungsfall:	***Behandlungsaufgabe erstellen***

Aufgabe:	**ErstelleAufgabe**
Beschreibung:	*Siehe Systemfunktion im Aufgaben-Modul auf Seite 26ff.*
Beteiligt:	*Therapeut*
Priorität:	*MUSS*
Anwendungsfall:	***Aufgabe verwalten, erstellen***

Aufgabe:	**EntferneAufgabe**
Beschreibung:	*Eine Version der Aufgabe wird entfernt und durch eine andere aus dem Repository ersetzt.*
Beteiligt:	*Therapeut*
Priorität:	*MUSS*
Anwendungsfall:	***Aufgabe verwalten, entfernen***

Aufgabe:	**ÜbernehmeAufgabeAusRepository**
Beschreibung:	*Eine gewählte Aufgabe aus dem Repository wird konvertiert und in die Behandlungsaufgabe eingefügt.*
Beteiligt:	*Therapeut, Aufgaben-Modul*
Priorität:	*SOLL*
Anwendungsfall:	***Aufgabe verwalten, aus Repo kopieren***

Aufgabe:	**ErstelleKontext**
Beschreibung:	*Siehe Systemfunktion im Kontext-Modul auf Seite 24ff.*
Beteiligt:	*Therapeut*
Priorität:	*MUSS*
Anwendungsfall:	***Kontext verwalten, erstellen***

Aufgabe:	**EntferneKontext**
Beschreibung:	*Eine Version des Kontextes wird entfernt und durch eine andere aus dem Repository ersetzt.*
Beteiligt:	*Therapeut*
Priorität:	*MUSS*
Anwendungsfall:	***Kontext verwalten, entfernen***

Aufgabe:	**ÜbernehmeKontextAusRepository**
Beschreibung:	*Ein gewählter Kontext aus dem Repository wird konvertiert und in die Behandlungsaufgabe eingefügt.*
Beteiligt:	*Therapeut, Kontext-Modul*
Priorität:	*SOLL*
Anwendungsfall:	***Kontext verwalten, aus Repo kopieren***

FEEDBACK-MODUL

Aufgabe:	**ZeigeFeedbackVorlage**
Beschreibung:	*Holt alle Fragen und Upload-Elemente, die mit dieser Feedback-Vorlage, beziehungsweise dieser Behandlungsaufgabe, verknüpft sind und gibt diese aus.*
Beteiligt:	*Therapeut*
Priorität:	*MUSS*
Anwendungsfall:	***Feedback-Vorlage anzeigen***

Aufgabe:	**ÄndereFeedbackParameter**
Beschreibung:	*Parameter des Feedbacks, die beispielsweise den Patienten zur Beantwortung nach jeder Absolvierung der Aufgabe verpflichten, werden im System überschrieben.*
Beteiligt:	*Therapeut*
Priorität:	*MUSS*
Anwendungsfall:	***Feedback-Vorlage erstellen, Feedback-Vorlage bearbeiten***

Aufgabe:	**ÄndereFeedbackInhalte**
Beschreibung:	*Vom Benutzer definierte Fragen oder Upload-Elemente werden für eine Behandlungsaufgabe geändert. Es können beliebig viele Fragen mit bereits definierten oder leeren Antwortfeldern verwaltet werden, für welche man den gewünschten Antwort-Typ vorher festlegen muss. Gleiches gilt für Upload-Elemente, welche Dateien mit den gewünschten Eigenschaften (z.B. Typ und maximale Größe) zum Upload vorsehen und somit das Ergebnis einer Aufgabe dokumentieren können.*
Beteiligt:	*Therapeut*
Priorität:	*MUSS*
Anwendungsfall:	***Feedback-Vorlage erstellen, Feedback-Vorlage bearbeiten, Inhalte verwalten***

Aufgabe:	**ErstelleFeedbackVorlage**
Beschreibung:	*Für eine Behandlungsaufgabe wird eine Feedback-Vorlage mit standardmäßiger Parameter-Auswahl erstellt.*
Beteiligt:	*Therapeut*
Priorität:	*MUSS*
Anwendungsfall:	***Feedback-Vorlage erstellen***

EINLADUNGS-MODUL

Aufgabe:	**HoleOffeneEinladungen**
Beschreibung:	*Therapeuten bekommen alle offenen Einladungen des Instituts zurück und Administratoren alle im System erstellen Einladungen.*
Beteiligt:	*Therapeut, Administrator*
Priorität:	*SOLL*
Anwendungsfall:	***offene Einladungen anzeigen***

Aufgabe:	**WiderrufeEinladung**
Beschreibung:	*Einladung wird invalidiert und der betreffende Benutzer wird via E-Mail benachrichtigt.*
Beteiligt:	*Therapeut, Administrator, E-Mail-Service*
Priorität:	*SOLL*
Anwendungsfall:	***Einladung widerrufen***

Aufgabe:	**InvalidiereEinladung**
Beschreibung:	*Einladung wird invalidiert und kann nicht mehr eingelöst werden.*
Beteiligt:	*Therapeut, Administrator, Account-Modul*
Priorität:	*SOLL*
Anwendungsfall:	***Einladung invalidieren***

Aufgabe:	**ValidiereEinladung**
Beschreibung:	*Einladung wird validiert, ob die Kombination aus Einladungscode, Behandlungs-/Institutscode, Ablaufdatum und optional Behandlungsplatz stimmt.*
Beteiligt:	*Account-Modul*
Priorität:	*SOLL*
Anwendungsfall:	***Einladung validieren***

Aufgabe:	**ErstelleEinladung**
Beschreibung:	*Es wird eine Benutzerprüfung durchgeführt, ob er Benutzer schon existiert. Ansonsten wird ein neuer Einladungscode mit Ablaufdatum erstellt und je nach Einladungsart ein Behandlungs- oder Institutscode angegeben.*
Beteiligt:	*Instituts-Modul, Behandlungs-Modul*
Priorität:	*SOLL*
Anwendungsfall:	***Einladung erstellen***

Aufgabe:	**FührеBenutzerprüfungDurch**
Beschreibung:	*Es wird überprüft, ob eine E-Mail-Adresse im System schon vorhanden ist und ob dieser Account noch benutzt wird. Ist dies der Fall, so wird das aufrufende Modul benachrichtigt und keine Einladung erstellt.*
Beteiligt:	*Instituts-Modul, Behandlungs-Modul*
Priorität:	*SOLL*
Anwendungsfall:	***Benutzerprüfung durchführen***

ADMIN-MODUL

Aufgabe:	**SuchePatientenMitStudienberechtigung**
Beschreibung:	*Alle Patienten, die eine Studienberechtigung für ihre Datensätze erteilt haben und somit eine StudienID besitzen, werden samt ihrer Datensätze gesucht und zurückgegeben.*
Beteiligt:	*Administrator*
Priorität:	*SOLL*
Anwendungsfall:	***Patienten mit Studienberechtigung anzeigen***

Aufgabe:	**ExportiereDatensatz**
Beschreibung:	*Die Daten zu allen Patienten mit erteilter Studienberechtigung werden gesammelt und zum Export angeboten.*
Beteiligt:	*Administrator*
Priorität:	*SOLL*
Anwendungsfall:	***Datensatz exportieren***

Aufgabe:	**SperreMarktplatz**
Beschreibung:	*Der Zugang zum Marktplatz wird global oder für ausgewählte Institute gesperrt. Wird er global gesperrt, so kann eine Sperrnachricht hinterlegt werden.*
Beteiligt:	*Administrator*
Priorität:	*KANN*
Anwendungsfall:	***Markplatz sperren***

Aufgabe:	**KonfiguriereAnzahlUmgebungsbedingungen**
Beschreibung:	*Die maximale Anzahl an Umgebungsbedingungen für ein neues Kontext-Objekt wird festgelegt.*
Beteiligt:	*Administrator, Kontext-Modul*
Priorität:	*KANN*
Anwendungsfall:	***Kontext konfigurieren***

Aufgabe:	**KonfiguriereAnzahlTechAnforderungen**
Beschreibung:	*Die maximale Anzahl an technischen Anforderungen für ein neues Kontext-Objekt wird festgelegt.*
Beteiligt:	*Administrator, Kontext-Modul*
Priorität:	*KANN*
Anwendungsfall:	***Kontext konfigurieren***

Aufgabe:	**KonfiguriereAnzahlPrivaterKontexte**
Beschreibung:	*Die maximale Anzahl an privaten Kontexte, innerhalb des Instituts eines Therapeuten, wird festgelegt.*
Beteiligt:	*Administrator, Kontext-Modul*
Priorität:	*KANN*
Anwendungsfall:	***Kontext konfigurieren***

Aufgabe:	**KonfiguriereAnzahlGlobalerKontexte**
Beschreibung:	*Die maximale Anzahl an globaler Kontexte, innerhalb des Instituts eines Therapeuten, wird festgelegt.*
Beteiligt:	*Administrator, Kontext-Modul*
Priorität:	*KANN*
Anwendungsfall:	***Kontext konfigurieren***

Aufgabe:	**KonfiguriereAnzahlMedienElemente**
Beschreibung:	*Die maximale Anzahl an Medien-Elemente für ein neues Aufgaben-Objekt wird festgelegt.*
Beteiligt:	*Administrator, Aufgaben-Modul*
Priorität:	*KANN*
Anwendungsfall:	***Aufgaben konfigurieren***

Aufgabe:	**KonfiguriereCopyrightAngabe**
Beschreibung:	*Die Parameter zur Copyright-Angabe bei globalen Aufgaben wird gesetzt.*
Beteiligt:	*Administrator, Aufgaben-Modul*
Priorität:	*KANN*
Anwendungsfall:	***Aufgaben konfigurieren***

Aufgabe:	**KonfiguriereDisclaimerAngabe**
Beschreibung:	*Die Parameter zur Disclaimer-Angabe bei privaten und globalen Aufgaben wird gesetzt.*
Beteiligt:	*Administrator, Aufgaben-Modul*
Priorität:	*KANN*
Anwendungsfall:	***Aufgaben konfigurieren***

Aufgabe:	**KonfiguriereAnzahlPrivaterAufgaben**
Beschreibung:	*Die maximale Anzahl an privaten Aufgaben, innerhalb des Instituts eines Therapeuten, wird festgelegt.*
Beteiligt:	*Administrator, Aufgaben-Modul*
Priorität:	*KANN*
Anwendungsfall:	***Aufgaben konfigurieren***

Aufgabe:	**KonfiguriereAnzahlGlobalerAufgaben**
Beschreibung:	*Die maximale Anzahl an globaler Aufgaben, innerhalb des Instituts eines Therapeuten, wird festgelegt.*
Beteiligt:	*Administrator, Aufgaben-Modul*
Priorität:	*KANN*
Anwendungsfall:	***Aufgaben konfigurieren***

Aufgabe:	**KonfiguriereMaxDateigröße**
Beschreibung:	*Die maximale Dateigröße für Copyrights, Disclaimer und Medien-Elemente setzen.*
Beteiligt:	*Administrator, Aufgaben-Modul*
Priorität:	*KANN*
Anwendungsfall:	***Manager konfigurieren***

Aufgabe:	**InvalidiereBenutzerToken**
Beschreibung:	*Hole alle angemeldeten Benutzer im System und invalidiere die gewählten Tokens. Damit wird der Systemzugriff unterbunden.*
Beteiligt:	*Administrator*
Priorität:	*KANN*
Anwendungsfall:	***API-Token anzeigen, API-Token sperren***

Aufgabe:	**ZeigeBenutzerverwaltung**
Beschreibung:	*Holt alle Benutzer jeder Rolle aus der Datenbank und gibt diese zurück.*
Beteiligt:	*Administrator*
Priorität:	*MUSS*
Anwendungsfall:	***Benutzer verwalten***

Aufgabe:	**ZeigeBenutzer**
Beschreibung:	*Holt einen spezifischen Benutzer aus der Datenbank und gibt diesen zurück.*
Beteiligt:	*Administrator*
Priorität:	*MUSS*
Anwendungsfall:	***Benutzer anzeigen***

Aufgabe:	**SperreBenutzer**
Beschreibung:	*Sperrt einen spezifischen Benutzer mittels Soft-Delete. Dieser wird darüber benachrichtigt.*
Beteiligt:	*Administrator, Email-Service*
Priorität:	*MUSS*
Anwendungsfall:	***Benutzer löschen***

Aufgabe:	**VergebeNeuesPasswort**
Beschreibung:	*Sendet dem ausgewählten Benutzer eine E-Mail mit einem Link auf die Passwort-Ändern-Funktion. Wurde diese Funktion ausgeführt, so muss der Benutzer sein Passwort wechseln, bevor er sich wieder in das System einloggen kann.*
Beteiligt:	*Administrator*
Priorität:	*MUSS*
Anwendungsfall:	***Passwort zurücksetzen***

Aufgabe:	**ErstelleInstitut**
Beschreibung:	*Ein Administrator erstellt ein neues Institut und kann anschließend Therapeuten für dieses Institut einladen.*
Beteiligt:	*Administrator, E-Mail-Service*
Priorität:	*MUSS*
Anwendungsfall:	***Therapeut erstellen***

Aufgabe:	**ZeigeAlleInstitute**
Beschreibung:	*Alle Institute mit den dazugehörenden Eckdaten in der Datenbank suchen und zurückgeben.*
Beteiligt:	*Administrator*
Priorität:	*MUSS*
Anwendungsfall:	***Institute konfigurieren***

Aufgabe:	**ZeigeInstitut**
Beschreibung:	*Ein Institut mit den dazugehörenden Eckdaten, freien Behandlungsplätzen und den verknüpften Therapeuten, sowie den freizuschaltenden Therapeuten für dieses Institut, in der Datenbank suchen und zurückgeben.*
Beteiligt:	*Administrator*
Priorität:	*MUSS*
Anwendungsfall:	***Institut anzeigen***

Aufgabe:	**BearbeiteInstitut**
Beschreibung:	*Ein Institut mit den dazugehörenden Eckdaten, freien Behandlungsplätzen und den verknüpften Therapeuten, sowie den freizuschaltenden Therapeuten für dieses Institut, in der Datenbank suchen und mit den Änderungen des Nutzers überschrieben..*
Beteiligt:	*Administrator, Instituts-Modul*
Priorität:	*MUSS*
Anwendungsfall:	***Institut bearbeiten***

Aufgabe:	**LöscheInstitut**
Beschreibung:	*Ein Institut wird mittels Soft-Delete gesperrt, wenn alle Patienten des Instituts gesperrt wurden oder keine Patienten zu dem jeweiligen Therapeuten existieren.*
Beteiligt:	*Administrator*
Priorität:	*KANN*
Anwendungsfall:	***Institut löschen***

Aufgabe:	**KonfiguriereInstitutsKontingente**
Beschreibung:	*Das maximale Kontingent an Behandlungen und Therapeuten, die einem Institut zugeordnet sind, wird konfiguriert.*
Beteiligt:	*Administrator*
Priorität:	*KANN*
Anwendungsfall:	***Kontingente konfigurieren***

Aufgabe:	**SperreTherapeut**
Beschreibung:	*Der ausgewählte Therapeut kann das System nicht länger verwenden, wird jedoch nicht aus der Datenbank gelöscht. Das System informiert ihn darüber über E-Mail.*
Beteiligt:	*Administrator, E-Mail-Service*
Priorität:	*MUSS*
Anwendungsfall:	***Therapeuten sperren***

3. SOFTWARESPEZIFIKATION

In diesem Kapitel wird das Datenmodell des Systems definiert und beschrieben. Aus Gründen der Aufgaben-Aufteilung innerhalb des Projektes, werden die Systemschnittstellen und das dazugehörende Nutzungskonzept, sowie die Systemfunktionen, hier nicht definiert.

3.1 Systemschnittstellen

Die Beschreibung der Systemschnittstellen für dieses Softwaresystem wird von *Aliyar Aras* in seiner konzeptionellen Masterarbeit und *Johannes Schmid* in seiner konzeptionellen Bachelorarbeit erarbeitet.

3.2 Nutzungskonzept

Die Beschreibung des Nutzungskonzeptes für die Systemschnittstellen für dieses Softwaresystem wird von *Aliyar Aras* in seiner konzeptionellen Masterarbeit und *Johannes Schmid* in seiner konzeptionellen Bachelorarbeit erarbeitet.

3.3 Datenmodell

Das System wird in diesem Abschnitt als Domänenmodell dargestellt, um alle Daten und deren Beziehungen als UML2-Diagramm dazustellen. Außerdem werden Invarianten des Modells beschrieben.

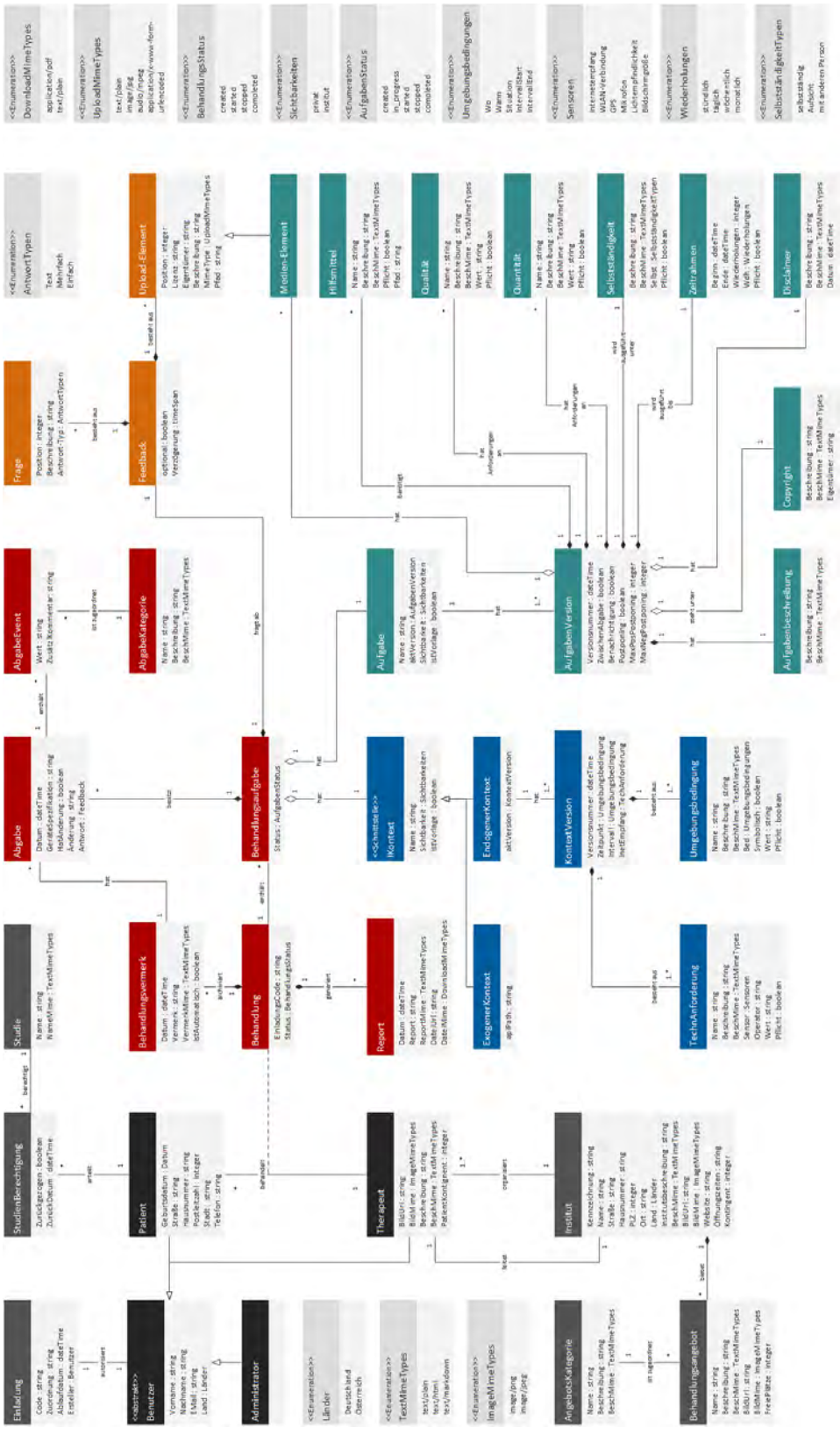
INVARIANTEN

Bezeichner:	**GlobaleAdministratorAufgaben**
Beschreibung:	*Administratoren können nur globale Aufgaben, jedoch keine privaten Aufgaben, besitzen.*
Beteiligte Klassen:	*Administrator, Aufgabe*

Bezeichner:	**AufgabenVersionDerAbgabe**
Beschreibung:	*Bei Änderungen der AufgabenVersion werden in der Abgabe nur folgende Klassen berücksichtigt: Hilfsmittel, Qualität, Quantität, Selbstständigkeit, Zeitrahmen*
Beteiligte Klassen:	*Abgabe, AufgabenVersion*

Bezeichner:	**VersionsnummerDerAufgabenVersion**
Beschreibung:	*Die Versionsnummer der AufgabenVersion und die Eigenschaft „aktuelleVersion" der Aufgabe können nicht in der Zukunft liegen.*
Beteiligte Klassen:	*Aufgabe, AufgabenVersion*

Bezeichner:	**BehandlungsTypKontingent**
Beschreibung:	*Es können nur so viel Behandlungstypen erstellt werden, wie im Kontingent hinterlegt.*
Beteiligte Klassen:	*Institut, Behandlungstyp*



Datenmodell: Domänenmodell des Systems

3.4 Funktionen

Aufgrund der Aufgabenteilung und fortwährenden Änderungen in diesem Projekt, entfallen die zu beschreibenden Systemfunktionen aus Entwicklersicht. Eine Auflistung aller Funktionen aus Anwendersicht wird in Kapitel 2.3 Funktionale Systemanforderungen beschrieben.

4. RANDBEDINGUNGEN

Dieses Kapitel beschreibt die Rahmenbedingungen des Systems, das heißt unter welchen Bedingungen es entwickelt und anschließend eingesetzt wird.

4.1 Qualität

Dieser Abschnitt beschreibt die nicht-funktionalen Anforderungen des (Gesamt-) Systems. Für die Client-Anwendungen gelten, insbesondere für den Bereich Benutzerfreundlichkeit, mindestens die angegebenen Prioritäten, bestenfalls noch höher. Priorisiert wird in aufsteigender Reihenfolge mit dem folgenden Wertebereich: 0, +, ++

Anforderung:	**Zuverlässigkeit**
Beschreibung:	*Das System muss für den Benutzer stets zuverlässig, also fehlerfrei, ohne Datenverlust und auf dem aktuellen Datenstand, erreichbar sein.*
Aspekte:	*Erreichbarkeit*
Priorität:	*++*

Anforderung:	**Erreichbarkeit**
Beschreibung:	*Das System darf nur eine minimale „Down-Zeit" haben und muss den Patienten zuverlässig über die gängigen Tageszeiten bei der Therapie unterstützen.*
Aspekte:	*Erreichbarkeit*
Priorität:	*++*

Anforderung:	**Robustheit**
Beschreibung:	*Das System muss bei fehlerhaften Eingaben in einen definierten Zustand wechseln.*
Priorität:	*+*

Anforderung:	**Sicherheit/Datenschutz**
Beschreibung:	*Der Schutz von vertraulichen, personenbezogenen Daten spielt bei diesem System eine zentrale Rolle und muss unbedingt gewährleistet werden.*
Priorität:	*++*

Anforderung:	**Effizienz**
Beschreibung:	*Aufrufe und Zusendungen von Daten, das gilt insbesondere für Behandlungsaufgaben, müssen im System performant bearbeitet werden, um den Benutzer eine flüssige und schnelle Arbeitsumgebung zu schaffen.*
Priorität:	*+*

Anforderung:	**Wartbarkeit**
Beschreibung:	*Das System muss für Anwender und Entwickler gleichermaßen gut dokumentiert sein und sollte Änderungen und Erweiterungen am Systemkern ohne unnötigen Aufwand ermöglichen.*

Aspekte:	*Portabilität*
Priorität:	*0*

Anforderung:	**Portabilität**
Beschreibung:	*Das System muss für jede Art von Client oder Betriebssystem erreichbar und interpretierbar sein und darf deshalb nicht auf eine Client-Technologie zugeschnitten entwickelt werden.*
Priorität:	*++*

Anforderung:	**Benutzerfreundlichkeit**
Beschreibung:	*Die Client-Anwendungen des Systems müssen ohne technische Kenntnisse für Laien selbsterklärend entwickelt werden. Benutzer müssen stets einfach und schnell ihre Aufgabe erfüllen können.*
Priorität:	*++*

4.2 Betriebskonzept

Das System wird als Webservice nach dem REST-Paradigma von Roy Fielding[7] entwickelt, um verschiedene Client-Technologien gleichermaßen anzubinden. Es wird fortlaufend weiterentwickelt und muss deshalb über eine Schnittstellen-Versionierung verfügen, um veraltete Systeme gleichermaßen bedienen zu können.

4.3 Entwicklungsvorgaben

Die Systemschnittstelle wird innerhalb der „ASP.NET Application Framework" von Microsoft mittels „ASP.Net WebApi" entwickelt. Das System wird mit der Entwicklungsumgebung „Visual Studio 2015" und der Programmiersprache „C#" umgesetzt. Als Datenbank kommt „SQL Server" von Microsoft zum Einsatz.

4.4 Abnahmekriterien

Das System wird als Prototyp mit Mindestfunktionsumfang (*Benutzerverwaltung, Aufgabenverwaltung, Behandlungsaufgabenverwalten, Patientenverwaltung*) entwickelt und ist nach erfolgreicher Funktionsprüfung abnahmefähig.

7 http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

A.2 Anforderungsdokument

Funktionale Anforderungen

	ID	TITEL	BESCHREIBUNG
GAST-MODUL	F/1	**API Dokumentation**	Dokumentation der API für Entwickler
	F/2	**Marktplatz**	Zeigt freie Behandlungsplätze aller Institute an - Angebote als Liste (Patient sucht nach Angebot) - Angebote als Karte (Patient sucht in seiner Nähe) - freie Behandlungsplätze suchen - Detail-Ansicht eines Instituts
ACCOUNT-MODUL	F/3	**Autorisation**	Verwalten von folgenden Benutzergruppen: **Admin, Therapeut, Patient und Benutzer**
	F/4	**Registrierung**	Registrierung von Benutzern - Therapeuten - mit Instituskennzeichnung - Patienten
	F/5	**Authentifizierung**	- Login - Access-Token generieren - Logout / Invalidieren des Token
	F/6	**Benutzerverwaltung**	Bearbeiten von Profildaten - Stammdaten - Passwort ändern - Account sperren
INSTITUTS-MODUL	F/7	**Institut verwalten**	Therapeuten sind einem bestimmten Institut zugeordnet, welches Eckdaten (Adresse,...) und freie Behandlungsplätze hält. - Eckdaten bearbeiten - Name - Leiter - Instituts-Typ - Adresse - Öffnungszeiten - Bild / Logo - Über uns
	F/8	**Behandlungsangebot verwalten**	- freie Behandlungsplätze bearbeiten - Kontingent abrufen
	F/9	**Therapeuten einladen**	- Therapeuten für dieses Institut einladen - via Email **(**E-Mail schicken**)**
KONTEXT-MODUL	F/10	**Kontext-Repository**	Benutzer sieht alle bisher erstellten Kontexte und kann diese Vorlagen bearbeiten (neue Version erstellen) oder löschen. Außerdem kann er Kontexte definieren. - Interface für Kontext erstellen - geteilten Kontext duplizieren - **Endogen** => im System spezifiziert - alle (bisher erstellten) Kontexte anzeigen - neue Kontext-Vorlage definieren **- Exogen** => Kontext von außerhalb (z.B. Twitter, Google Now,..) - alle (bisher erstellten) Kontexte anzeigen - neue Kontext-Vorlage definieren
	F/11	**Kontext definieren**	Kombination aus Umgebungsbedingungen + techn. Anforderungen definieren - Kontext nur innerhalb des Instituts - jedes Item sollte als **PFLICHT** markiert werden können **- nicht überprüfbar (grundsätzlich)** - symbolischer Ort - Situation - symbolischer Zeitpunkt - Gerätetyp **- überprüfbar (grundsätzlich)** - physicher Ort - Konnektivität **(**Internet-Empfang/Netz**)** - genauer Zeitpunkt - Gerätetyp/Bildschirmgröße - (Ausführungs-) Intervall

			- **überprüfbar (optional)** - Mikrofon **(**Geräuschpegel**)** - Kamera **(**Lichtpegel**)** - GPS **(**Koordinaten**)** - Accelorator **(**Vibrationspegel**)**
	F/11a	**Kontext versionieren**	Es wird eine neue Version der Kontextvorlage erstellt und der Zeiger aktuallisiert!
AUFGABEN-MODUL	F/12	**Aufgaben-Repository**	Aufgaben-Vorlagen sollen in einem Repository verwaltet werden können - Sichtbarkeit innerhalb eines Instituts - geteilte Aufgaben duplizieren
	F/13	**Aufgaben definieren**	Therapeuten können Aufgaben (ohne/mit Patientenzuordnung) definieren - Name/Titel - Beschreibungstext als Medien-Element (HTML/Markdown) - Zwischenrückmeldung **(**ja / nein**)** - Zeitrahmen auswählen - Benachrichtigung (aktiv / passiv) - Feedback-Vorlage (optional) - Verschieben / Postponen zulassen - erlaubtes Zeitfenster definieren - Multimedia-Upload (optional) - Sichtbarkeit - Copyright (vordefiniertes auswählen oder neu) - Disclaimer setzen (vordefiniertes auswählen oder neu)
	F/13a	**Aufgaben versionieren**	Es wird eine neue Version der Aufgabenvorlage erstellt und der Zeiger aktuallisiert!
	F/14a	**Medien-Elemente hinzufügen**	Upload von Medien-Elementen auf den Server **(max. 3 pro Aufgabe, max. 5 MB)** - LIZENZ und EIGENTÜMER beachten! - HTML - Markdown - Plain-TEXT - Bild (JPG, PNG) - Audio (mp3) - Video (URL)
	F/14b	**Medien-Manager**	Schnittstelle zum Abrufen und Verwalten von Multimedia-Dateien.
	F/15	**Disclaimer-Manager**	Disclaimer sollen verwaltet werden können - vordefinierte Disclaimer verwalten (nur Admin) - Vorlage erstellen - Vorlage bearbeiten - Vorlage löschen
	F/16	**Copyright-Manager**	Copyrights sollen verwaltet werden können - vordefinierte Copyright verwalten (nur Admin) - Vorlage erstellen - Vorlage bearbeiten - Vorlage löschen
PATIENTEN-MODUL	F/17	**Behandlungsaufgaben abrufen**	Therapeuten & Patienten können über eine mobile Anwendung mit der Plattform interagieren - Aufgabe ansehen
	F/18	**Behandlungsaufgaben manipulieren**	Patient fordert Änderung der Aufgabe beim Therapeuten an. - verschieben / postponen **(**Email schicken**)** - inhaltliche Änderung **(**Email schicken**)**
	F/19	**Abgabe erstellen**	Patient quittiert die Abgabe zu einer gewissen Aufgaben- und Kontext-Version!!! - automatischer Behandlungsvermerk beim Abgeben einer Aufgabe - komplettes JSON-File zur Abgabe speichern - Medien-Uploads speichern

Funktionale Anforderungen

	F/20	**Studienberechtigung**	Patienten verfügen über eine **zusätzliche Studien-ID müssen für die Datenerhebung (Datenschutzerklärung!) einwilligen**
BEHANDLUNGS-MODUL	F/21	**Behandlungen verwalten**	Behandlungen verknüpfen den Patienten mit dem Therapeute und haben eine eindeutige Behandlungskennzeichnung (HASH), welcher bei Einladungen zur Identifikation zusätzlich angeben wird. - alle Behandlungen des Therapeuten anzeigen - alle Behandlungen zu einem Patienten anzeigen - Behandlung beenden **(**E-Mail schicken**)** - Behandlung wiederherstellen **(**E-Mail schicken**)**
	F/22	**Behandlungsstatistik berechnen**	Für eine Behandlung werden verschiedene Auswertungen berechnet, um dem Therapeuten einen bessern Überblick über die Behandlung zu geben. - Statistik über alle Behandlungsaufgaben inklusive ihrer Abgaben
	F/23	**Behandlungen erstellen**	Es wird eine neue Behandlung hinzugefügt, wobei der Patient (registriert oder nicht) entweder mit Angabe der E-Mail-Adresse eingeladen wird ODER noch leer bleibt, um sich selbst mittels QR-Code zu registrieren. - Einladung mit Ablaufdatum erstellen - per Mail senden **(**E-Mail schicken**)** - QR-Code erstellen
	F/24	**Behandlungsvermerke verwalten**	Behandlungsvermerke sollen wie eine Patientenakte geführt werden und werden automatisch oder manuell erstellt. - manuell erstellen - mit einer Behandlungsaufgabe verknüpfen - bearbeiten - löschen
	F/25	**Report erstellen**	Ein Report über die Behandlung sollen als Handout dargestellt werden. - als PDF - E-Mails versenden mit PDF-Anhang
THERAPEUTEN-MODUL	F/26a	**Versionierung von Behandlungsaufgaben**	Behandlungsaufgaben müssen versioniert werden können!!!
	F/26	**Behandlungsaufgaben erstellen**	Aufgabe wird zu einem Patienten hinzugefügt und individualisiert - Aufgabe erstellen ODER kopieren - Kontext erstellen ODER kopieren - Feedback verwalten
	F/27	**Behandlungsaufgabe bearbeiten**	Behandlungsaufgaben werden fortwährend auf den Patienten angepasst, wobei pro Änderung eine neue Aufgaben-/Kontext-Version erzeugt wird. - Änderungswunsch übernehmen - Diffs ausgeben - Kontext bearbeiten - Aufgabe bearbeiten - Feedback verwalten
	F/28	**Aufgabenstatus ändern**	Behandlungsaufgaben müssen manuell gestartet werden. Ab diesem Zeitpunkt kann der Patient unterstützt werden (und Daten erhoben werden). Behandlungsstatus: **created, in progress, started, stopped, completed** - erstellen einer Behandlungsaufgabe: **created** - bearbeiten einer Behandlungsaufgabe, die noch nicht läuft: **in progress** - starten einer Behandlungsaufgabe: **started** - stoppen, aber nicht beenden einer Behandlungsaufgabe: **stopped** - beenden einer Behandlungsaufgabe: **completed**
	F/29	**Behandlungsaufgabe überwachen**	Behandlungsaufgabe soll (statistische) Werte bereitstellen, um die Aufgabenerfüllung zu überwachen - Änderungen (Diffs) ausgeben - Aufgabenstellung VS. Abgabe - Aufgabenstellung VS. mehrere Abgaben - Aufgabenstellung VS. Änderungswunsch - Änderungswünsche einsehen
FEEDBACK-MODUL	F/30	**Feedback verwalten**	Feedback-Vorlagen können mit Fragen und Upload-Elementen gefüllt werden und sind einer Behandlungsaufgabe zugewiesen. - Vorlage anzeigen - Vorlage erstellen - Vorlage bearbeiten
	F/31	**Upload-Elemente verwalten**	Upload-Elemente sind Platzhalten für Datei-Uploads - hinzufügen - bearbeiten - löschen
	F/32	**Fragen verwalten**	Fragebogen haben verschiedene Fragen-Elemente (Forms)

			- hinzufügen - bearbeiten - löschen
EINLADUNGS-MODUL	F/33	**Einladungen verwalten**	Einladungen können an Patienten und Therapeuten ausgesprochen werden, weshalb diese extra verwaltet werden müssen. - alle offenen Einladungen anzeigen - Einladungen erstellen - Patient - Therapeut für vorhandenes Institut - Einladungen widerrufen **(**Email schicken**)**
	F/34	**Einladungen prüfen**	Einladungen müssen auf ihre Validität überprüft werden - Benutzerprüfung
ADMIN-MODUL	F/35	**Administration**	Administration des Systems - Patienten mit Studienberechtigung anzeigen - Export-Funktion - Repository-Einstellungen konfigurieren - Aufgaben - alle Aufgaben anzeigen - Aufgaben löschen - Kontext - alle Kontexte anzeigen - Kontexte löschen - Manager konfigurieren - Medien - Copyright - Disclaimer - Marktplatz sperren - Sperrnachricht eingeben - Benutzer verwalten - anzeigen - PW zurücksetzen - sperren
	F/36	**Institute konfigurieren**	Admin kann Institute anzeigen, erstellen und verwalten - Institut anzeigen - Institut erstellen - Institut bearbeiten - Institut löschen - Institute konfigurieren - Kontingent definieren - Therapeuten sperren (Email schicken)

Nicht funktionale Anforderungen

	ID	TITEL	BESCHREIBUNG
ALLGEMEIN	nF/1	**Zuverlässigkeit**	Das System muss für den Benutzer stets zuverlässig, also fehlerfrei, ohne Datenverlust und auf dem aktuellen Datenstand, erreichbar sein.
	nF/2	**Erreichbarkeit**	Das System darf nur eine minimale „Down-Zeit“ haben und muss den Patienten zuverlässig über die gängigen Tageszeiten bei der Therapie unterstützen.
	nF/3	**Robustheit**	Das System muss bei fehlerhaften Eingaben in einen definierten Zustand wechseln.
	nF/4	**Sicherheit/Datenschutz**	Der Schutz von vertraulichen, personenbezogenen Daten spielt bei diesem System eine zentrale Rolle und muss unbedingt gewährleistet werden.
	nF/5	**Effizienz**	Aufrufe und Zusendungen von Daten, das gilt insbesondere für Behandlungsaufgaben, müssen im System performant bearbeitet werden, um den Benutzer eine flüssige und schnelle Arbeitsumgebung zu schaffen.
	nF/6	**Wartbarkeit**	Das System muss für Anwender und Entwickler gleichermaßen gut dokumentiert sein und sollte Änderungen und Erweiterungen am Systemkern ohne unnötigen Aufwand ermöglichen.
	nF/7	**Portabilität**	Das System muss für jede Art von Client oder Betriebssystem erreichbar und interpretierbar sein und darf deshalb nicht auf eine Client-Technologie zugeschnitten entwickelt werden.
	nF/8	**Benutzerfreundlichkeit**	Die Client-Anwendungen des Systems müssen ohne technische Kenntnisse für Laien selbsterklärend entwickelt werden. Benutzer müssen stets einfach und schnell ihre Aufgabe erfüllen können.

Technische Anforderungen

	ID	TITEL	BESCHREIBUNG
API	T/1	**API Doku mit Swagger**	Die Dokumentation der API sollte mittels dem Swagger NuGet-Package erweitert werden. Dazu sollten alle Kommentare (inkl. Remarks) erstellt und als XML ausgegeben werden.
	T/2	**API Versionieren**	Versionierung der APIs nach SOC Vortrag
ACCOUNT / AUTH	T/3	**Rechteverwaltung**	Implementierung mittels der Vorlage des ASP.NET Identity Frameworks.
	T/4	**OWIN**	Autorisierung (Token) mittels des OWIN-Frameworks von Microsoft.
	T/6	**Benutzerverwaltung**	Identity Framework (über API)
ALLGEMEIN	T/7	**Mehrsprachigkeit**	Multilingualität über Cultures und Resource-Files.

Abbildungsverzeichnis

Tabellenverzeichnis

Name: Michael Stach Matrikelnummer: 724231

Erklärung

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

Ulm, den .....

Michael Stach