Fachhochschule Dortmund

University of Applied Sciences and Arts

Team 6: Facharzt-Termin-Verwaltungstool

28.07.2017

Projektbericht zum Modul „Fortgeschrittene Softwaretechnologie“

Fachhochschule Dortmund

4. Semester

Sommersemester 2017

Prüfer: Prof. Dr.-Ing. Stefan Bente

Inhaltsverzeichnis

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

Abkürzung	Bedeutung
AHP	Analytical Hierarchy Process
App	Applikation / Application
DB	Datenbank
GUI	Graphical User Interface
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
ISO	Internationale Organisation für Normung
JDK	Java Development Kit
JPA	Java Persistence API
MS	Microsoft
MVC	Model View Controller
SQL	Structured Query Language
UC	Use Case (Anwendungsfall)
UML	Unified Modeling Language
WAR	Web Application Archive

1 Einleitung

Im ersten Abschnitt dieser Projektdokumentation werden einführend einige grundlegende Worte zum Projekt selbst und dem Durchlauf gegeben.

1.1 Teilnehmer

Folgende Teilnehmer haben bei der Durchführung des Projektes und der Erstellung des Projektberichtes mitgewirkt:

Name	Vorname	Matrikelnummer	Hochschule	E-Mail
Janßen	Daniel	11111264	Köln	janssen_daniel@web.de
Nguyen	Felix	7097608	Dortmund	felix.ngu@gmx.de
Scherer	Tim	11111276	Köln	t.scherer@outlook.com
Schmidt	Daniel	7097643	Dortmund	daniel@schmidt-ct.de

Tabelle 1: Teilnehmer

1.2 Dokumente

Die Dokumente inklusive der Implementierung befinden sich bei Github. Eine Installationsanleitung inklusive dem Anlegen der Testdaten befindet sich im nächsten Kapitel. Die Dokumentationen mit den überarbeiteten Dokumenten für die verschiedenen Meilensteine sind unter dem Ordner *Dokumentation* zu finden. Dort ist auch der Portfolio-Überblick, der angibt, welcher Teilnehmer welche Arbeit innerhalb des Teams geleistet hat. Unter folgendem Github Link sind alle erwähnten Dokumente zu finden:

https://github.com/DanielJanssen/doctorcalendar.git

1.3 Installationsanleitung

Für das Projekt wird eine relationale Datenbank benötigt, empfohlen wird XAMPP mit MySQL. Zudem wird ein Eclipse benötigt, welches mit einem JDK ausgestattet und gestartet wird. Des Weiteren wird ein Git-Plugin zum Lesen aus dem Repository, sowie ein Maven-Plugin benötigt, um das Projekt korrekt zu bauen und benötigten Code zu generieren. Nach dem „Pull“ von Git kann die Installationsanleitung verwendet werden, die unter „/documentation/installation/installation.txt“ zu finden ist.

2 Vision

Im zweiten Abschnitt des Projektberichts wird die Vision für das Projekt dargestellt. Diese beinhaltet die Idee hinter der geplanten Applikation, in welchem Kontext sie eingesetzt werden soll, welchen Mehrwert die Applikation dem Anwender bringt und mit welchen Tools und Werkzeugen entwickelt werden soll.

2.1 Produktidee

Es soll eine webbasierte Applikation entwickelt werden, die von Arztpraxen und ihren Patienten zur Terminverwaltung genutzt wird. Vor allem Patienten sollen dazu in der Lage sein auch kurzfristig einen Termin bei einem Facharzt zu bekommen, um gegen akute Probleme zielstrebig vorzugehen. Dazu bieten Ärzte ihre freien und kurzfristig freigewordenen Termine an, die dann von Patienten reserviert werden können. Nach der Reservierung muss der Arzt den Termin bestätigen oder aber ablehnen, sofern es Probleme zum Beispiel mit der Zeitdauer gibt. Für die Patienten und Ärzte gibt es dabei unterschiedliche Ansichten, sodass sie nur ihre eigenen Daten einsehen können. Patienten können ferner nach bestimmten Ärzten oder nach Ärzten in ihrer Umgebung suchen.

Dabei müssen die gesetzlichen Rahmenbedingungen der sensiblen personenbezogenen Daten bei Patienten eingehalten werden.

2.2 Einsatzkontext

Grundsätzlich bestehen verschiedene Möglichkeiten, wie mit dieser Applikation Geld für den Betreiber verdient werden kann. So kann zum einen auf der Webseite Werbung geschaltet werden, sodass mit jedem Klick Geld verdient wird. Zum anderen kann Geld von den Ärzten für ihre Teilnahme am System verlangt werden, dass kann dann je nach Anzahl der vergebenen Termine oder aber über eine monatliche Grundgebühr geregelt werden. Patienten werden vermutlich nicht für eine solche Applikation oder Vermittlung bezahlen, da zwar ein Mehrwert besteht, aber grundsätzlich durch ein Gesetz innerhalb von vier Wochen ein Termin vergeben werden muss. Auch ist die Zahlungsbereitschaft der Patienten insgesamt für Vermittlungsleistungen nicht sonderlich hoch.

Die Applikation soll nur innerhalb Deutschlands betrieben werden, um sich nur auf deutsche Gesetze zum Datenschutz vor allem bei sensiblen Patientendaten, sowie eine deutsche Sprache auf der Oberfläche konzentrieren zu können.

2.3 Ist-Analyse

Eine einheitliche Applikation in diesem Bereich ist momentan nicht vorhanden. Es gibt zwar Branchensoftware für die Terminverwaltung, diese wird aus verschiedenen Gründen allerdings nur von wenigen Praxen eingesetzt. Die Gründe sind unteranderem die fehlende Möglichkeit von schnellen evtl. notwendigen Anpassungen und die fehlende Integration der Patienten. Viele Ärzte arbeiten noch nach dem „alten“ Weg und schreiben Termine in einen nicht elektronischen Kalender. Dieser Prozess ist zwar bewährt, besitzt in der heutigen schnelllebigen Zeit allerdings einige Nachteile.

Ein Kontakt zwischen Patient und Praxis besteht ausschließlich persönlich oder über das Telefon. Die Verwendung von moderneren Technologien, u.a. Applikationen auf Smartphones oder Zugänge / Prozesse über das Internet, sind kaum bis gar nicht vorhanden. Umfragen unter Patienten, Praxen und Ärzte haben ergeben, dass diese nicht vorhandenen Funktionen gewünscht werden. Denn so erhoffen Patienten einen direkteren Kontakt zu den Ärzten und eine effizientere Terminverwaltung. Bei akuten Problemen müssen derzeit viele Arztpraxen angerufen werden, um schnell einen Termin ohne viel Wartezeit zu bekommen oder in die offene Sprechstunde zu müssen. Meistens werden spontan frei gewordene Termine nicht anderweitig vergeben, da diese nicht bekannt gemacht werden können.

2.4 Projektorganisation

Innerhalb des Projektteams findet die Kommunikation überwiegend über drei verschiedene Medien statt. Hauptsächlich über eine WhatsApp-Gruppe, da so schnell Informationen an allen Teilnehmer verteilt werden können. Sofern es nötig ist, werden Skype-Konferenzen abgehalten um einen noch direkteren Kontakt mit allen Teilnehmern herstellen zu können inkl. der Möglichkeit seinen eigenen Desktop zu teilen. Während der Vorlesungstermine kann sich auch persönlich abgesprochen und komplexe Sachverhalte geklärt werden. Für die Organisation wurde bei Trello ein Kanban-Board eingerichtet, um die Meilensteine und die dazugehörigen Aufgaben verteilen und koordinieren zu können.

Innerhalb des Projektteams gab es eine Rollenverteilung. Da Daniel Janßen als Programmierer das einzige Teammitglied mit Projekterfahrung im Bereich der Softwareentwicklung ist, wurde ihm die Rolle des Projektleiters zugeteilt, um das Projekt möglichst effizient und zielführend koordinieren zu können.

2.5 Vorgehensmodell

Als Vorgehensmodell wurde das erweiterte Wasserfallmodell gewählt, da es ein sehr einfaches Modell mit klar vorgegebenen Strukturen ist. Zudem gab es zeitlich und inhaltlich fix vorgegebene Meilensteine, die ebenfalls ein elementarer Bestandteil des erweiterten Wasserfallmodells sind. Auf Grund der geringen Erfahrung in Softwareprojekten der meisten Teilnehmer kam ein durchaus komplexeres agiles System, wie zum Beispiel Scrum, nicht in Frage, da der Aufwand möglichst geringgehalten werden sollte und die Einhaltung der Meilensteine im Fokus stand.

2.6 Werkzeuge

Für die Durchführung des Projektes wurden verschiedene Werkzeuge mit unterschiedlichen Verwendungszwecken verwendet. Für die Kommunikation werden folgende zwei bereits erwähnte Tools benutzt:

- WhatsApp
- Skype

Für die gesamte Projektorganisation wurde sich auf Trello geeinigt. Das Arbeiten über der Onlineoberfläche und einer eigenen Applikation auf dem Smartphone vereinfacht schnelle Absprachen innerhalb des Teams.

Auf Grund der Erfahrungen des Projektleiters wurden zudem Eclipse und XAMPP mit MySQL zur Entwicklung gewählt. Die Dokumentenverwaltung findet durch GIT bei Github statt und die Dokumentation wurde mit der Microsoft Office Suite erstellt:

- MS Word für Texte
- MS Excel für Tabellen
- MS Visio für Grafiken und Diagramme

3 Anforderungsspezifikation

In dem dritten Kapitel des Projektberichtes werden die Anforderungsspezifikationen (Lastenheft) dargelegt. Das Lastenheft beschreibt alle Anforderungen des Auftraggebers mit allen notwendigen Leistungen, die das Produkt erfüllen soll. Es beinhaltet außerdem eine Priorisierung aller notwendigen Funktionen.

3.1 Stakeholder

Nachfolgend sind alle Stakeholder des Facharzt-Termin-Verwaltungstool aufgelistet und gruppiert. Dazu gehören auf der einen Seite die späteren Anwender, die die Applikation später benutzen werden:

- Facharztpraxen
    - Ärzte
    - Verwaltung (Assistenten und Sprechstundenhilfe)
- Patienten

Auf der anderen Seite ist es der Auftragnehmer, welches das Projekt in seiner Gesamtheit durchführt:

- Management
- Entwickler
- Tester
- Administrator (Hardware, DB)

Eine dritte Gruppe sind sonstige Interessierte, die ein berechtigtes Interesse an der entwickelten Applikation besitzen:

- Datenschutzbeauftragter
- Ärztekammer
- Hersteller der Branchensoftware

3.2 Funktionen

Die Applikation muss verschiedene Funktionen beinhalten und erfüllen. Diese lassen sich in zwei Bereiche unterteilen. Die funktionalen Anforderungen legen fest, was die Applikation tun muss und die nicht funktionalen Anforderungen legen die Randbedingungen und Qualität der Funktionen fest.

3.2.1 Funktionalen Anforderungen

Nachfolgend sind alle funktionalen Anforderungen der Applikation dargestellt. Diese lassen sich in den genannten drei Kategorien unterteilen.

3.2.1.1 In Bezug auf die Applikation selbst

1. Die Applikation sollte eine Registrierungsfunktionalität bieten
2. Die Applikation sollte eine Anmeldefunktionalität bieten
3. Verschiedene Sichten mit unterschiedlichen Berechtigungen müssen verfügbar sein
    a. Patienten
        i. Die Applikation sollte eine Übersicht über vergangene Termine bieten
        ii. Die Applikation sollte eine Übersicht über zukünftige Termine bieten
        iii. Die Applikation sollte eine Übersicht aller freien Termine beinhalten
    b. Ärzte / Verwaltung
4. Eine Verbindung bereits vorhandener Applikationen (z.B. Kalender) soll vorhanden sein inkl. Export- und Import-Funktionen
5. Die Applikation sollte einen Austausch der Termindaten mit einer existierenden Branchensoftware durchführen können
6. Die Applikation sollte auf der Oberfläche in deutscher Sprache gehalten werden

3.2.1.2 In Bezug auf die Fachärzte und ihre Verwaltung

7. Arztpraxen sollten neue Termine anbieten können
    a. Manuell
    b. Elektronische Schnittstelle
8. Arztpraxen sollten gewählte Termine von Patienten bestätigen können
9. Arztpraxen sollten gewählte Termine von Patienten begründet stornieren können
10. Arztpraxen sollten eine detaillierte Übersicht aller Termine eines Datums bekommen können (inkl. Patienteninformationen)
11. Arztpraxen sollten bereits reservierte Termine mit Daten erweitern können
12. Arztpraxen sollten eine Übersicht über die Termine (vergangen und in der Zukunft) eines bestimmten Patienten in ihrer Praxis bekommen können

3.2.1.3 In Bezug auf die Patienten

13. Ein Patient sollte Termine reservieren können
14. Ein Patient sollte reservierte Termine wieder stornieren können
15. Ein Patient sollte nach Fachärzten in der gewünschten Umgebung suchen können
16. Ein Patient sollte nach allen vorhandenen Fachärzten einer bestimmten Fachrichtung suchen können
17. Ein Patient sollte nach dem nächsten freien Termin eines Arztes suchen können
18. Ein Patient sollte nach dem nächsten freien Termin aller Ärzte einer Fachrichtung suchen können
19. Ein Patient sollte bei einem Arzt nach allen freien Terminen suchen können
20. Ein Patient sollte eine Übersicht aller freien und belegten Terminen eines Arztes bekommen (ohne Patienteninformationen)
21. Ein Patient sollte eine Anfrage stellen können, um bei einem freigewordenen Termin automatisch informiert zu werden
22. Ein Patient sollte bei der Bestätigung eines Termins vom System automatisch informiert werden
23. Ein Patient sollte bei der Stornierung eines Termins durch den Arzt vom System automatisch informiert werden

3.2.2 Nicht-funktionale Anforderungen

Auf Basis von EN ISO 25010 soll die Anwendung nicht-funktionale Anforderungen erfüllen, die in der nachfolgenden Tabelle aufgelistet sind. Dabei sind die jeweiligen Anforderungen in Kategorien eingeteilt.

Kategorien	Unterkategorie	Nicht-funktionale Anforderungen
Kompatibilität	Interoperabilität	Importieren von Daten aus gängigen Kalendern
Portabilität	Installierbarkeit	Anwendung soll webbasiert sein
	Anpassungsfähigkeit	Kompatibilität zu den gängigen Browsern soll gegeben sein
Wartbarkeit	Modifizierbarkeit	Umsetzen von Änderungen an der Anwendung sollen mit minimalen Systemausfall durchführbar sein
Leistung und Effizienz	Zeitverhalten	Kurze Antwortzeiten (<0,5s) bei Interaktion mit der Anwendung bis die erste Reaktion dem Anwender sichtbar gemacht wird
Zuverlässigkeit	Wiederherstellbarkeit	Datenwiederherstellung der letzten 14 Tage muss gegeben sein
Benutzbarkeit	Lernförderlichkeit	Intuitive und übersichtliche Bedienoberfläche ohne großen Einarbeitungsaufwand, welches mit durchschnittlichen Softwareerfahrungen zu bewältigen ist
	Fehlererkennung	Fehlerhafte Eingaben sollen abgefangen und eine Alternative angezeigt werden
	Zugänglichkeit	Anwendung soll über ein Browser zugänglich sein
Sicherheitsanforderungen	Vertraulichkeit	Zugriff auf Patientendaten darf nur durch autorisiertes Personal erfolgen. Die Daten sollen verschlüsselt gespeichert werden.
	Integrität	Änderungen von Terminen darf nur durch autorisiertem Personal erfolgen
	Nachweisbarkeit	Aufzeichnen jeder Tätigkeit innerhalb des Systems
	Zurechenbarkeit	Verwenden von eindeutigen und zuweisbaren Benutzernamen
	Authentizität	Überprüfung des Authentizität des Anwenders beim Aufruf

Tabelle 2: Nicht-funktionale Anforderungen

Der Einsatzzweck der Anwendung erfordert den Fokus auf die nicht-funktionalen Anforderungen in den zwei Bereichen Sicherheit und Benutzbarkeit. Schließlich sind die meisten Nutzer der späteren Applikation Personen, die größtenteils über keine tiefgreifenden informationstechnischen Kenntnisse verfügen. Zudem sollen die Anwender der Softwarelö-

sung ohne Schulungen und mit geringen Einarbeitungszeiten in der Lage sein, die Anwendung zu bedienen. Dabei werden mit Patientendaten gearbeitet, die höchst sensible personenbezogene Daten darstellen. Daher ist es wichtig, dass nur autorisiertes Ärztepersonal der Arztpraxis, zu dem die Daten gehören, auf diese zugreifen und bearbeiten kann. Ein Patient darf zudem auch nur seine eigenen Daten sehen und kein Zugriff auf Daten anderer besitzen.

3.2.3 Charakterisierung nach Kano

Nachdem alle Wünsche des Auftraggebers bekannt sind, werden anschließend alle funktionalen Anforderungen charakterisiert. Für diese Charakterisierung wird das Kano-Modell verwendet. Dieses Modell beschreibt den Zusammenhang zwischen den Anforderungen der Applikation und der erwarteten Zufriedenheit des Auftraggebers (Kunde). Die Ergebnisse werden während des gesamten Projektzyklus verwendet.

Nachfolgend befindet sich die Priorisierungsmatrix, welche für eine anschließende Priorisierung der funktionalen Anforderungen verwendet wird:

Antwort auf "Merkmal vorhanden" (funktionale Frage)	Antwort auf "Merkmal fehlt" (dysfunktionale Frage): Gewünscht (würde mich sehr freuen)	Muss (setze ich voraus)	Egal	Noch OK (nehme ich gerade noch hin)	Unerwünscht (würde mich sehr stören)
Gewünscht	I	A	A	A	L
Muss	I	?	?	?	B
Egal	N	?	U	U	B
Noch OK	N	?	U	?	B
Unerwünscht	N	N	?	?	I

Legende
B = Basisfaktor
L = Leistungsfaktor
A = Begeisterungsfaktor
N = Not to do
U = Unerheblicher Faktor
? = Antwort indifferent
I = Antwort inkonsistent

Abbildung 1: Schablone für die Priorisierung der Funktionen

Die Schablone beschreibt den Zusammenhang von den Gedanken und das Verhalten der Kunden, wenn Merkmale vorhanden oder nicht vorhanden sind. Beispiele und Beschreibungen relevanter Faktoren befinden sich auf der nächsten Seite.

Mithilfe der abgebildeten Priorisierungsmatrix in Abbildung 1 werden alle gewünschten Funktionen der Applikation bewertet:

Kategorisierung	Anforderung/Merkmal	Vorhanden	Fehlt	Kano
Generell	Registrierungsfunktionalität	Muss	Unerwünscht	B
	Anmeldefunktionalität	Muss	Unerwünscht	B
	Sichten mit unterschiedlichen Berechtigungen	Muss	Unerwünscht	B
	Verbindung mit anderen Applikationen für Import und Export	Gewünscht	Noch OK	A
	Datenaustausch mit existierenden Branchensoftware	Gewünscht	Noch OK	A
Fachärzte und ihre Verwaltung	Anbieten von neuen Terminen (manuell)	Muss	Unerwünscht	B
	Anbieten von neuen Terminen (elektronisch)	Muss	Unerwünscht	B
	Bestätigen von gewählten Terminen von Patienten	Muss	Unerwünscht	B
	Begründete Stornierung von gewählten Terminen von Patienten	Muss	Unerwünscht	B
	Detaillierte Übersicht aller Temine eines Datums anzeigen lassen	Muss	Unerwünscht	B
	Erweitern von bereits reservierten Terminen mit Daten	Egal	Egal	U
Patienten	Reservieren von Terminen	Muss	Unerwünscht	B
	Stornieren von reservierten Terminen	Muss	Unerwünscht	B
	Suche nach Terminen / Ärzten	Muss	Unerwünscht	B
	Anzeige einer Übersicht aller freien und belegten Terminen eines Arztes	Egal	Egal	U
	Stellen einer Anfrage, um bei freigewordene Termine informiert zu werden	Muss	Unerwünscht	B

Abbildung 2: Priorisierung der gewünschten Funktionen

Wie in Abbildung 2 zu entnehmen, werden alle funktionalen Anforderungen für die Kano Charakterisierung aufgelistet. Nun muss bewertet werden, ob diese Funktionen obligatorisch sind, nur vom Kunden gewünscht oder ob es egal ist, ob diese verfügbar sind oder nicht. Anschließend muss außerdem bewertet werden, wie zufrieden / unzufrieden der Kunde ist, wenn diese Funktionen nicht implementiert sind.

Anhand der erstellten Schablone für die Priorisierung (Abbildung 1) wird nun die Kano Bewertung vorgenommen. So werden Funktionen mir einem „B“ als Basisfaktor bewertet, diese werden als notwendig angesehen. Eine Begeisterung beim Kunden findet bei Verfügbarkeit nicht statt, fehlen diese allerdings, sinkt die Zufriedenheit beim Kunden enorm. In unserer Applikation sind dies beispielsweise die Registrierungs- und Anmeldefunktionen, das Anbieten von neuen Terminen und die Reservierung und Stornierung dieser.

Ein „A“ charakterisiert die Funktion als mehr oder weniger optional. Beim Fehlen dieser sinkt die Zufriedenheit eher mäßig, beim Erfüllen steigt diese, da diese vom Kunden nicht erwartet werden. In dem Facharzt-Termin-Verwaltungstool sind dies zwei Funktionen. Einmal die Verbindung mit anderen Applikationen für einen Im- und Export und der Datenaustausch mit existierender Branchensoftware. Als dritte Kategorie findet sich ein „U“ wieder, welche unnötige Funktion aufzeigt. Hier sind dies die Erweiterung von bereits reservierten Terminen und die Übersicht aller freien und belegten Termine eines Arztes.

3.2.4 Priorisierung nach AHP

Nachdem die funktionalen Anforderungen charakterisiert sind, müssen diese anschließend priorisiert werden. Für diese Priorisierung wird der analytische Hierarchieprozess (AHP) angewendet. Initial entwickelt um Entscheidungsprozesse zu unterstützen werden somit die Funktionen priorisiert, um so spätere Entscheidungsprozesse während des Projektes zu unterstützen.

Für die Priorisierung der Anforderungen des Facharzt-Termin-Verwaltungstools werden folgende Funktionen verwendet:

ID	Funktionen
1	Registrierungsfunktionalität
2	Anmeldefunktionalität
3	Sichten mit unterschiedlichen Berechtigungen
4	Verbindung mit anderen Applikationen für Import und Export
5	Datenaustausch mit existierenden Branchensoftware
6	Anbieten von neuen Terminen (manuell)
7	Anbieten von neuen Terminen (elektronisch)
8	Bestätigen von gewählten Terminen von Patienten
9	Begründete Stornierung von gewählten Terminen von Patienten
10	Detaillierte Übersicht aller Termine eines Datums anzeigen lassen
11	Erweitern von bereits reservierten Terminen mit Daten
12	Reservieren von Terminen
13	Stornieren von reservierten Terminen
14	Suche nach Ärzten / Terminen
15	Anzeige einer Übersicht aller freien und belegten Terminen eines Arztes
16	Stellen einer Anfrage, um bei freigewordene Termine informiert zu werden

Tabelle 3: Verwendete Funktionen für AHP inkl. verwendeter ID

Im Anhang[1] befindet sich der gesamte Analytical Hierarchy Process aller genannten Funktionen aus Tabelle 3. Ergebnis der AHP Analyse ist die Priorisierung der Funktionen absteigend der hier genannten:

- Suche nach Terminen / Ärzten
- Detaillierte Übersicht aller Termine eines Datums anzeigen lassen
- Reservieren von Terminen
- Bestätigen von gewählten Terminen von Patienten
- Stornieren von reservierten Terminen

Die genannten fünf Funktionen haben die höchste Priorität der Applikation und sollten entsprechend so behandelt werden.

[1] Siehe Anhang 1, S. X

4 Softwarespezifikation

Nachdem das Lastenheft erstellt und die Anforderungen des Auftraggebers bekannt sind, ist der nächste Schritt die Softwarespezifikation. Es ist eine Art Pflichtenheft, welches vom Auftragnehmer erstellt wird, das die Lösung für die Erfüllung aller Anforderungen beschreibt. In diesem Projektbericht ist es nur ein kleiner Ausschnitt des Pflichtenheftes, da sonst der vorgegebene Projektrahmen gesprengt werden würde.

4.1 Anwendungsfälle

Ein großer und wichtiger Bereich in der Softwarespezifikation sind die Anwendungsfälle. Diese beschreiben alle möglichen Szenarien innerhalb der Applikation die eintreten können, wenn ein Akteur (ein Benutzer der Applikation) das System verwendet, um ein bestimmtes Ziel zu erreichen.

4.1.1 Liste der Anwendungsfälle

In folgender Auflistung befinden sich die fünf wichtigsten Anwendungsfälle der Facharzt-Termin-Verwaltungstools. Sie stammen aus den Ergebnissen des analytischen Hierarchieprozesses (AHP):

ID	Hauptakteur	Anwendungsfall
UC1	Patient, Arztpraxis	Suchen innerhalb des Systems
UC2	Patient, Arztpraxis	Anfordern einer Übersicht innerhalb des Systems
UC3	Patient	Termin reservieren
UC4	Patient	Termin stornieren
UC5	Arztpraxis	Termin bestätigen

Tabelle 4: Übersicht der fünf wichtigsten Anwendungsfälle

Um den Projektrahmen nicht zu sprengen, wird nachfolgend nur ein einzelner relevanter Anwendungsfall näher betrachtet: die Terminreservierung.

4.1.2 Anwendungsfall: Termin reservieren

Der Anwendungsfall der Terminreservierung ist einer der am häufig verwendeten innerhalb der Applikation. Dieser beschreibt den Vorgang, wenn ein Patient einen freien Termin bei einem gewünschten Arzt reservieren möchte. In den folgenden drei Unterkapiteln wird dieser Vorgang grafisch dargestellt, detailliert beschrieben und die grafische Oberfläche wird gezeigt.

4.1.2.1 Diagramm

In der nachfolgenden Abbildung befindet sich die grafische Darstellung des Anwendungsfalldiagrams für die Terminreservierung:

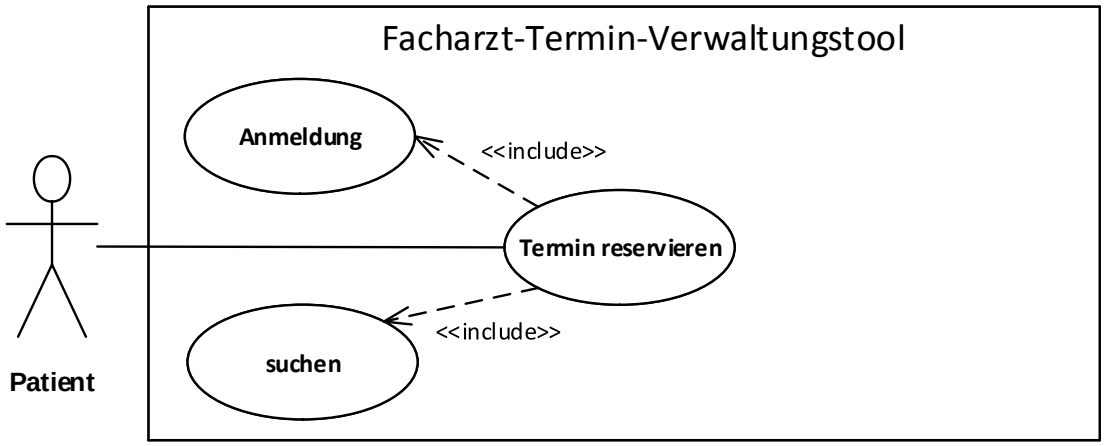

Abbildung 3: Grafische Darstellung des Anwendungsfall einer Terminreservierung

Als einziger Akteur dieses Vorganges ist der Patient, welcher den Vorgang einer Reservierung startet. Dieser Vorgang inkludiert zwei weitere Vorgänge, die vor oder während des Vorganges benötigt werden. Dies ist einmal eine vorherige Anmeldung an der Applikation und ein Suchen innerhalb der Datenbank, um den Status aller Termine zu bekommen (frei oder belegt).

4.1.2.2 Beschreibung

Neben der grafischen Darstellung des Anwendungsfalles befindet sich nachfolgend eine detaillierte textuelle Beschreibung dessen:

Use Case ID:	UC3
Use Case Name:	Termin reservieren
Akteure:	Patient
Beschreibung:	Der Patient möchte bei einem Facharzt seiner Wahl einen Termin reservieren.
Auslöser:	Der Patient betätigt im Anschluss an die erfolgreiche Terminsuche über einen Button die Funktion „Termin reservieren“
Vorbedingungen:	1. Der Patient besitzt einen gültigen Account auf der Website “Terminbuchung” 2. Der Patient hat sich erfolgreich eingeloggt 3. Der zu reservierende Termin muss verfügbar sein
Ergebnisse und Nachbedingungen:	1. Der Patient reserviert erfolgreich einen Termin 2. Der Patient erhält eine Bestätigung zur erfolgreichen Terminreservierung
Normaler Ablauf:	1. Der Patient betätigt die Funktion „Termin reservieren“ 2. System zeigt dem Patienten die aus dem Suchprozess gewählten Daten (Tag, Uhrzeit, Terminart) 3. Der Patient gibt eine Terminbeschreibung ein 4. Patient bestätigt die Reservierung

Normaler Ablauf:	5. System bucht die Reservierung 6. System gibt eine Meldung über die erfolgreiche Reservierung und der Info über eine Reservierungsbestätigung per E-Mail
Alternative Abläufe:	5a. In Schritt 5 wird die Reservierung vom Endkunden nicht bestätigt 1. Endkunde bricht die Reservierung ab 2. System fragt Endkunden nach Änderung der Termindaten 3. Endkunde bestätigt den Wunsch nach Änderung der Termindaten 4. Use Case UC1 (Termin suchen) wird aufgerufen 5b. In Schritt 5 wird die Reservierung vom Endkunden nicht bestätigt 1. Endkunde bricht die Reservierung ab 2. System fragt Endkunden nach Änderung der Termindaten 3. Endkunde lehnt eine Änderung ab 4. System bricht den Reservierungsprozess ab
Ausnahmen:	1. Datenbank ist nicht verfügbar und die Reservierung schlägt fehl ➔ Schritt 5 ist zu wiederholen. Meldung an den Anwender. Wenn es länger nicht möglich ist, soll der Anwender informiert werden
Includes / excludes:	Anmeldung und Suchen
Anwendungshäufigkeit:	häufig
Spezielle Anforderungen:	1. Dauer zwischen Bestätigung der Reservierung und dem darauf folgenden Informationsscreen darf nicht länger als 5 Sekunden dauern.
Rahmenbedingung:	1. Patient muss deutsch sprechen können 2. System (Frontend und Backend) steht zur Verfügung
Mockups	07 terminübersicht_termin_reservieren_patient.png
Bemerkungen:	-

Tabelle 5: Beschreibung des Anwendungsfalls einer Terminreservierung

4.1.2.3 Gui-Mockup

Einen ersten Einblick der Benutzeroberfläche (GUI Mockup) für die Terminreservierung zeigt folgende Abbildung:

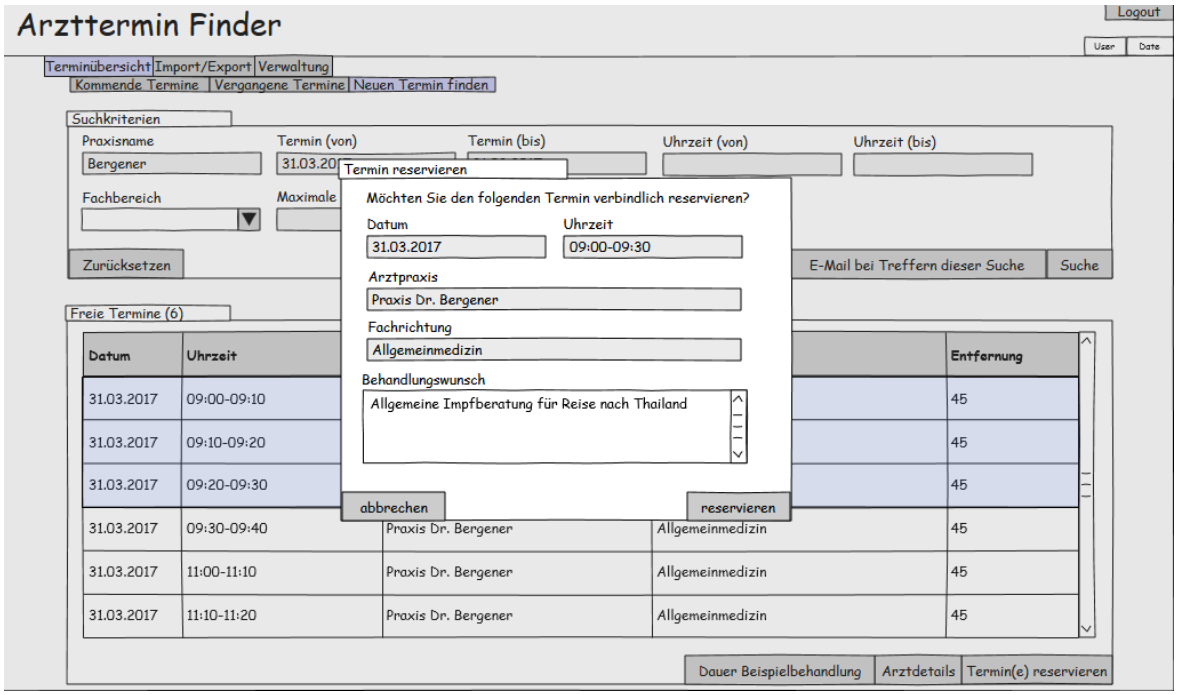

Abbildung 4: Gui-Mockup für die Reservierung eines Termins

Wie in der Abbildung 4 zu entnehmen befindet sich mehr im Hintergrund die Übersicht aller Termine eines Arztes. Insgesamt gibt es drei verschiedene Übersichten:

- Kommende Termine
- Vergangene Termine
- Neuen Termin finden

Innerhalb jeder Übersicht können detaillierte Suchkriterien verwendet werden. Dies beinhalten beispielsweise das Suchen eines Termins anhand des Datums und Uhrzeit, aber auch eine Filterung mittels eines Arztes oder Fachbereiches ist möglich.

Findet der Patient nun einen möglichen freien Termin, muss er diesen markieren und über den Button „Termin reservieren“ öffnet sich ein weiteres Fenster. Dort sieht er eine Auflistung weiterer Details des markierten Termins (Datum, Uhrzeit, Arztpraxis und Fachrichtung) und der Patient muss in dem Textfeld „Behandlungswunsch“ beschreiben, warum er den Arzt besuchen möchte. Beim bestätigen des Buttons „reservieren“ endet der Anwendungsfall der Terminreservierung.

4.2 Domänenklassendiagramm

Nachdem die Anwendungsfälle der Applikation ausführlich analysiert und beschrieben sind, müssen sich erste Gedanken um die Modellierung der Applikation gemacht werden. Das Domänenklassendiagramm liefert einen ersten Überblick über benötigte Datenklassen und wie diese in Beziehung zueinanderstehen. Nachfolgend befindet sich das Domänenklassendiagram des Facharzt-Termin-Verwaltungstools:

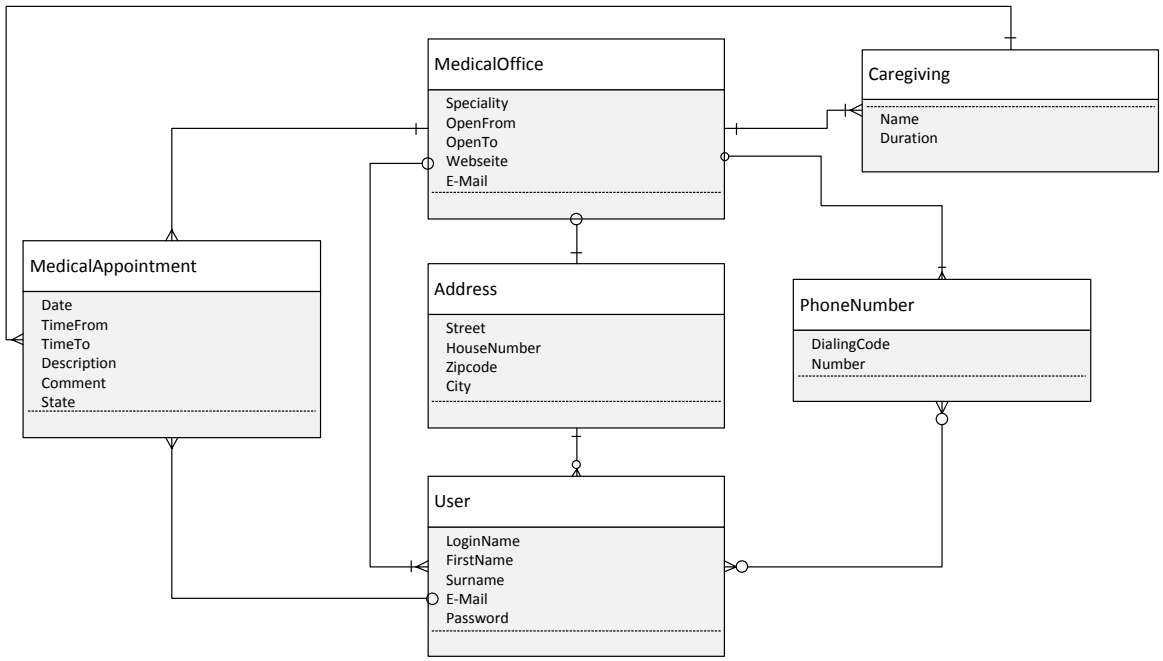

Abbildung 5: Domänenklassendiagramm

Insgesamt gibt es sechs verschiedene Datenklassen:

- MedicalAppointment -> Für die Behandlung selbst
- MedicalOffice -> Für die Arztpraxen
- Address -> Adressen für Patienten und Arztpraxen
- User -> Informationen der Patienten
- Caregiving -> Für die verschiedenen Arten von Behandlungen
- PhoneNumber -> Für Telefonnummern der Patienten und Ärzte

Eine ausführlichere Beschreibung der Klassen und deren Beziehungen zueinander befinden sich beim Grobentwurf später im Projektbericht. Die Entitäten und Attribute sind zudem im Glossar (Kapitel 8) näher erläutert[2].

[2] Siehe Kapitel 8: Glossar, S. 34

5 Architekturkonzeption

Nach dem die Softwarespezifikation abgeschlossen und eine Idee der Applikation vorhanden ist, kann mit dem nächsten Schritt, der Architekturkonzeption, begonnen werden. Im weiteren Verlauf wird die grundlegende Architektur skizziert, das Konzept der Applikation dargelegt und die verwendeten Frameworks beschrieben.

5.1 MVC

Model View Controller (MVC) ist ein Architektur- und Entwurfsmuster in der Softwareentwicklung, um die drei Komponenten Datenmodell (model), Präsentation (view) und Programsteuerung (controller) zu trennen. Dabei verfolgen die drei Komponenten unterschiedliche Aufgaben:

- **Model:** Enthält die Daten für die GUI und GUI-Geschäftslogik. Es ist unabhängig von View und Controller.
- **View:** Stellt das Modell grafisch auf einer Oberfläche dar und kann auf das Modell direkt zugreifen. Die Anfragen des Benutzers werden an den Controller weitergeleitet.
- **Controller:** Steuert dagegen die Applikation. Er verwaltet die verschiedenen GUIs, leitet die Benutzeranfragen an die anderen Schichten weiter und nimmt deren Ergebnisse entgegen.

Eine grafische Darstellung der Elemente und ihre Beziehung zueinander befinden sich in nachfolgender Abbildung:

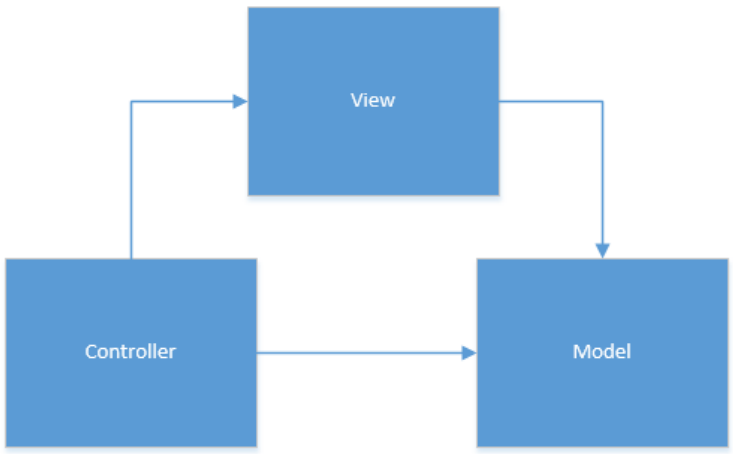

Abbildung 6: MVC – Pattern

5.2 Architekturkonzept

Das Facharzt-Terminverwaltungstool wird als eine mehrschichtige, objektorientierte Java-Applikation aufgebaut. Dabei übernehmen und kapseln die vorhandenen Schichten die verschiedenen Teilbereiche der Software. Um die Abhängigkeiten möglichst gering zu halten, ist nur der Zugriff der höheren in die niedrigeren Schichten erlaubt.

Es gibt vier Schichten innerhalb der Applikation:

- **Präsentationsebene / GUI-Schicht:** In der Präsentationsebene werden fachlich die verschiedenen GUIs und ihre Interaktionen untereinander gekapselt. Dabei findet der Aufbau in dieser Schicht nach dem MVC (Model-View-Controller) Design Pattern statt[3]. Durch den einheitlichen Aufbau, sowie die Verwendung des sehr bekannten Patterns, wird die Einarbeitungszeit verkürzt, sowie Wartbarkeit und Lesbarkeit des Codes erhöht.
- **Serviceebene:** In der Serviceebene werden die Anfragen aus der GUI zentral entgegengenommen und weiterverarbeitet. Eine entitätsübergreifende Geschäftslogik wird in dieser Schicht gehalten.
- **Applikationsebene:** In der Applikationsebene werden die Daten gehalten und Geschäftslogik innerhalb einzelner Entitäten realisiert.
- **Persistenzebene:** Die Persistenzebene regelt den Zugriff auf die relationale MySQL - Datenbank und sorgt für das Mapping zwischen den Tabellen und den Java-Objekten.

Alle Quellcode-Dateien werden in einem WAR (Web Application Archive) liegen, die Trennung der Schichten, sowie innerhalb der Schichten werden über Java-Packages, also Ordnerstrukturen realisiert.

Die Daten werden dabei in einer relationalen Datenbank dauerhaft und zentral persistiert und stehen somit den verschiedenen Benutzern jederzeit zur Verfügung.

[3] Siehe Kapitel 5.1: MVC, S. 17

5.3 Architekturskizze

Nachdem das Konzept der Applikation feststeht, muss noch die Architektur im gesamten skizziert werden, die einen groben Überblick der verschiedenen Ebenen gibt. Daher zeigt die nachfolgende Abbildung die Architekturskizze:

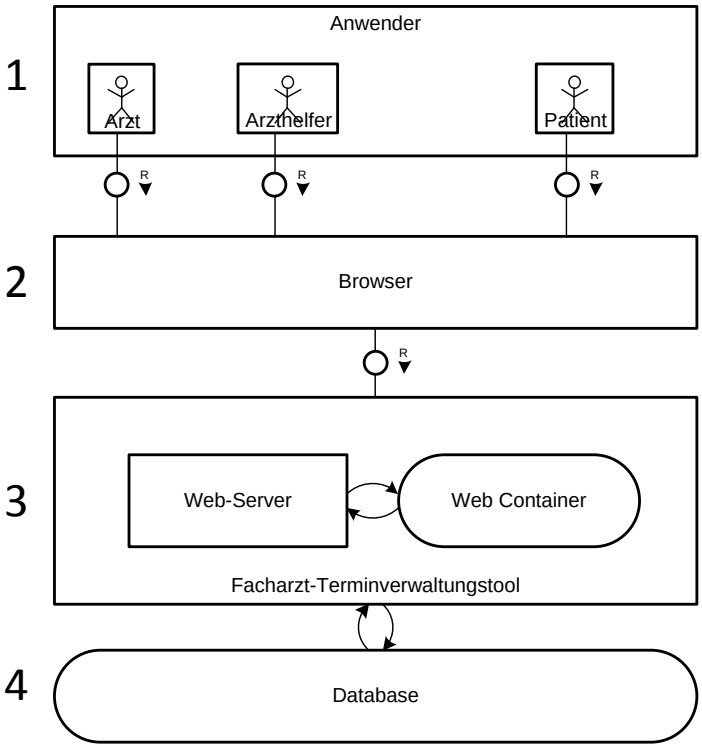

Abbildung 7: Architekturskizze

Wie in der Abbildung 7 zu entnehmen, werden vier verschiedene Ebenen benötigt. In der ersten Ebene befinden sich die Anwender der Applikation. Diese sind die Ärzte selbst, deren Arzthelfer und letztendlich die Patienten. Für die Darstellung der Oberflächen und um mit der Applikation arbeiten zu können, wird ein entsprechendes Tool benötigt. Daher befindet sich in der zweiten Ebene der Browser, da die Applikation eine webbasierte Anwendung ist. So wird eine mögliche weite Verbreitung und Verwendbarkeit sichergestellt. In der dritten Ebene befindet sich die Applikation selbst, mit seinem Web-Server und Web Container. Der Web-Server verwaltet dabei sämtliche HTTP / HTTPS Anfragen, während der Web Container unter anderem die Servlets (Java Klassen für Anfragen der Web Server) beinhaltet. In der letzten Ebene werden alle Daten in einer Datenbank persistiert.

Um einen hohen Qualitätsstandart zu erreichen, muss jede Ebene verschiedene nicht-funktionale Anforderungen erfüllen. Eine generelle Übersicht befindet sich in Kapitel 3.2.2[4]. Im Detail müssen für jede Ebene also beispielshaft folgende Anforderungen erfüllt werden:

- **Ebene 1: Anwender**
  - Zugriff auf die Applikation erfolgt über eine User/Passwort-Authentifizierung
  - Verschiedene Rollen ermöglichen unterschiedliche Berechtigungen
- **Ebene 2: Browser**
  - Webbasierte Anwendung und damit kompatibel zu gängigen Browsern und Betriebssystem unabhängig
  - Per HTTPS verschlüsselte Verbindung
  - Intuitive Bedienoberfläche
- **Ebene 3: Applikation**
  - Kurze Antwortzeiten durch schlanke Datenbanken
  - Gute Wartbarkeit durch strukturierten Quellcode
- **Ebene 4: Datenbank**
  - Regelmäßige Durchführung von Datensicherungen
  - Überwachung der Systemaktivitäten durch Erstellung von Logs

Die neun genannten funktionalen Anforderungen sind nur ein kleiner Ausschnitt aller, die die Applikation erfüllen muss.

[4] Siehe Kapitel 3.2.2: Nicht-funktionale Anforderungen, S. 8

5.4 Verwendete Frameworks und Technologien

Im folgenden Unterkapitel werden abschließend die angemeldeten Technologien und Frameworks (Rahmenwerke) näher aufgezeigt und erläutert.

5.4.1 Java Web App

Der Prototyp wird mit der weit verbreiteten objektorientierten Programmiersprache Java mit der Version 8 realisiert. In Java gibt es eine Vielzahl verschiedenster Frameworks, Technologien und Tools, die es dem Entwickler ermöglichen soll, seine Software zu schreiben.

5.4.2 MySQL

Die dauerhafte Datenhaltung wird über die relationale Datenbank MySQL realisiert. Dazu wird die OpenSource-Standardvariante genutzt, sodass keinerlei Lizenzkosten oder ähnliches anfallen. Der Prototyp wird dabei nur lokal auf dem Rechner laufen und dieser muss somit über eine MySQL - Instanz verfügen. Das wird in diesem Fall durch das unter der GNU General Public License stehende XAMPP mit der integrierten MySQL-Datenbank erfolgen.

5.4.3 SpringBoot

SpringBoot ist ein auf Spring basierendes Java-Framework zur Erzeugung von Standalone-Applikationen. Dabei werden neben Spring-Bibliotheken auch grundlegende Frameworks wie Hibernate (Persistenzframework auf der JPA-Spezifikation) oder JUnit und Mockito (zum Testen) eingebunden. Des Weiteren gibt es einen Embedded Webserver (für den Prototyp wird der Tomcat benutzt, ein Open Source Webserver für Java Webtechnologien.), sodass das entstehende WAR-File nirgendwo explizit deployed werden muss. Sämtliche Spring-Konfigurationen werden nach Möglichkeit von Spring Boot weggekapselt und standardmäßig mit sinnvollen Werten vorbelegt, sodass der Konfigurationsaufwand minimiert wird. Benutzt wird die aktuellste Version 1.5.2.

5.4.4 Vaadin

Die GUI der Applikation wird über Vaadin realisiert. Vaadin bietet eine Vielzahl verschiedenster GUI-Webkomponenten, die vollständig in Java realisiert werden können, sodass in der gesamten Applikation nur eine Programmiersprache verwendet werden muss. Auch der Zugriff vom Controller und auf das Modell ist leicht möglich, da Vaadin ebenfalls im Webserver ausgeführt wird. Es wird das aktuellste Release 8.0.5 verwendet.

5.4.5 Maven

Für das Erstellen und Verwalten der Fremdbibliotheken wird das Build-Management-Tool Maven verwendet. Dies ist ein Teil der Apache Software Foundation und wird ausschließlich für Java Programme verwendet. Maven wird dabei über eine zentrale Datei im Hauptverzeichnis konfiguriert: pom.xml. Dort werden die benötigten Maven-Plugins definiert, abhängige Fremdbibliotheken (wie die oben genannten Frameworks) eingebunden und die Applikation grundsätzlich definiert. Des Weiteren gibt Maven eine gängige Standard-Verzeichnisstruktur für Java-Projekte vor, die ebenfalls benutzt wird.

5.4.6 JPA

Die Java Persistence API (JPA) ist eine Schnittstelle für Java Anwendungen, die die Zuordnung für Java Objekte zu den dazugehörigen Datenbankeinträgen vereinfacht. So können innerhalb der Applikation Zugriffe auf die MySQL Datenbank erfolgen. Dabei findet die Konfiguration hauptsächlich über Annotationen statt und die Abfragesprache ist dem Standard-SQL angelehnt.

6 Grobentwurf

Im Grobentwurf wird der Anwendungsfall „Terminreservierung“ mit Hilfe verschiedener Diagramme detailliert betrachtet und aufgeschlüsselt, um für die Implementierung des Prototypens notwendigen Informationen zusammenzustellen. Dabei kommen folgende Modelle zum Einsatz:

- Komponentendiagramm
- Sequenzdiagramm
- Aktivitätsdiagramm
- Zustandsdiagramm

Zusätzlich wird das Klassendiagramm der Applikation gezeigt, welches für den Anwendungsfall die benötigten Klassen mit den dazugehörigen Attributen und Methoden aufzeigt.

6.1 Komponentendiagramm

In der nachfolgenden Abbildung ist das Komponentendiagramm abgebildet:

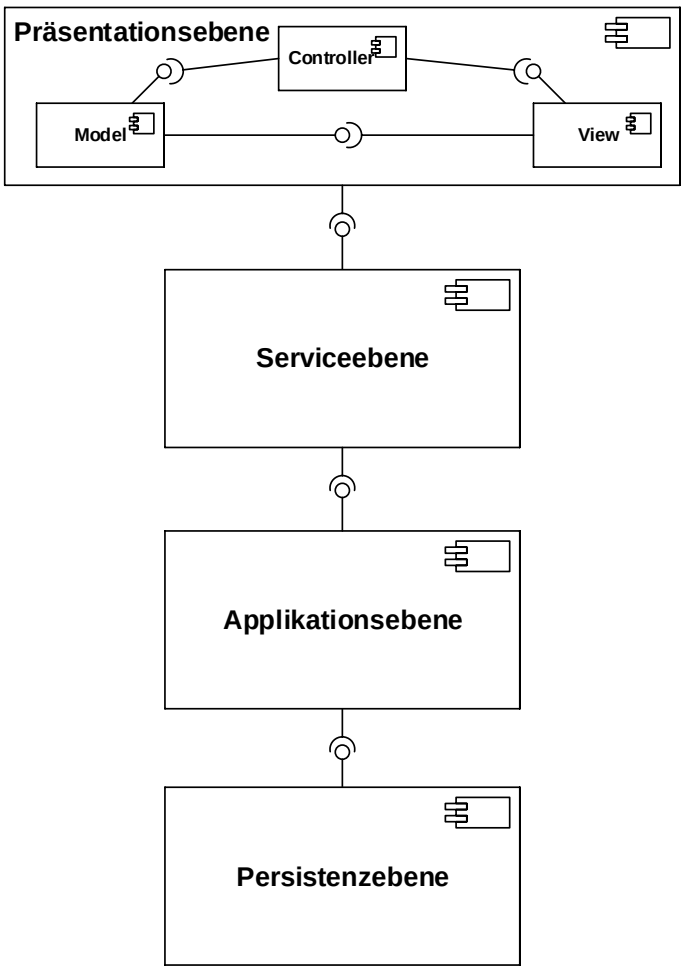

Abbildung 8: Komponentendiagramm

Die Applikationen wird in der Abbildung 8 in vier Ebenen unterteilt, anfangend mit der Präsentationsebene, der Service-, Applikations- und der Persistenzebene. Die Präsentationsschicht enthält drei weitere Komponenten, angelehnt an das MVC-Model[5]. Die Controller-Komponente greift auf die Schnittstellen der beiden anderen Komponenten zu, und zwar Model und View. Zwischen diesen beiden findet ebenfalls ein Datenaustausch statt, wobei die Model-Komponente eine Schnittstelle für View bereitstellt. In diesem hierarchischen Diagramm stellen die niederen Ebenen den höher gestellten Ebenen eine Schnittstelle zu Verfügung, sodass der Datenfluss von den höheren Schichten ausgelöst wird. Dementsprechend greift die Präsentationsebene auf die Serviceebene, welche wiederum auf die Applikationsschicht aufsetzt. Diese greift im letzten Schritt auf die Daten, die in der Persistenzschicht gespeichert wird, zu. Die angeforderten Daten werden zurück zu den oberen Ebenen durchgereicht, sodass der Nutzer der Applikation letztlich das Ergebnis seiner Anfrage sieht. Die Präsentationsschicht ist die Schnittstelle zur Außenwelt und dem Anwender selbst. In der Serviceebene werden die Anfrage des Anwenders an die Applikationslogik übergeben, welche die verschiedenen Klassen und Operationen enthält. Die Daten werden in der Persistenzebene gespeichert und von dort aus abgerufen.

[5] Siehe Kapitel 5.1: MVC, S. 17

6.2 Aktivitätsdiagramm

Das Aktivitätsdiagramm des Anwendungsfalles einer Terminreservierung befindet sich in nachfolgender Abbildung:

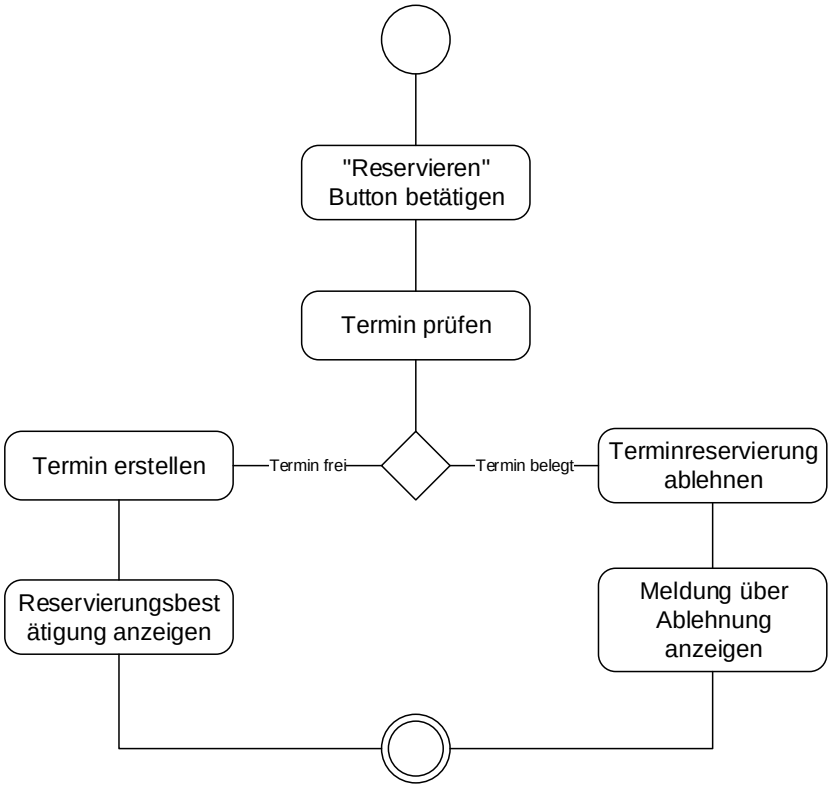

Abbildung 9: Aktivitätsdiagramm

Die Aktivitäten, die der Benutzer im Anwendungsfall „TerminReservieren“ durchführt, wurden anhand des UML-Standard Diagramms dargestellt. Der Benutzer beginnt mit der Aktivität „TerminReservieren“, indem dieser den dafür vorgesehenen Button betätigt. Daraufhin wird mit der Funktion „Terminprüfen“ geprüft, ob der gewünschte Termin noch zur Verfügung steht oder bereits belegt ist. Dies kann bspw. passieren, wenn der Benutzer den Termin nicht schnell genug reserviert und jemand anderes sich diesen Termin reserviert hat. Ist der gewünschte Termin noch frei, wird der Termin im System für den angemeldeten Benutzer erstellt. Die erfolgreiche Erstellung des Termins wird anschließend mittels Reservierungsbestätigung angezeigt. Sollte der gewünschte Termin nicht mehr zur Verfügung stehen, wird die Reservierung abgelehnt. Dieses Ereignis wird dem Benutzer ebenfalls über eine Meldung angezeigt. Die Aktivität „TerminReservieren“ ist nach dem jeweiligen Anzeigen der erfolgreichen oder fehlgeschlagenen Terminreservierung abgeschlossen und es kann eine neue Aktivität begonnen werden.

6.3 Zustandsdiagramm

Das Zustandsdiagramm für den bekannten Anwendungsfall ist in der nachfolgenden Abbildung zu sehen. Dabei spielt das Objekt *Termin* die entscheidende Rolle:

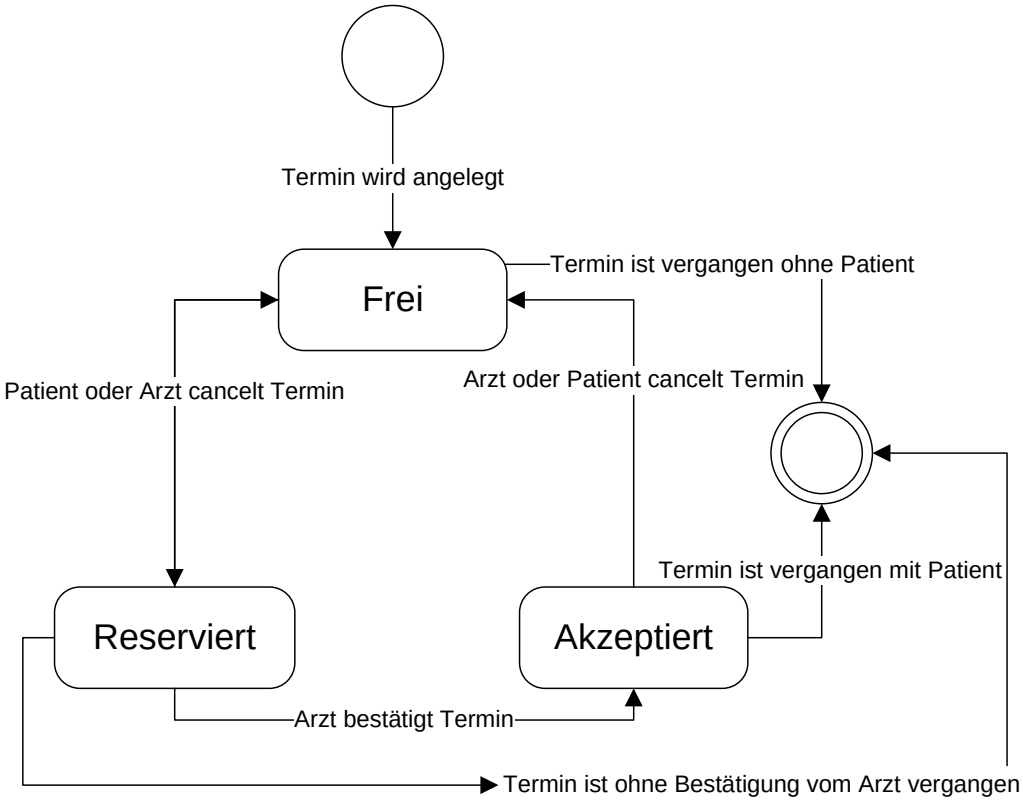

Abbildung 10: Zustandsdiagramm

Dabei gibt es im System für das Objekt *Termin* drei Zustände:

- Frei
- Reserviert
- Akzeptiert.

Zu Beginn ist ein Termin noch nicht für einen Benutzer reserviert und weist den Zustand „frei“ auf. Erst bei einer Buchung wird ein Termin angelegt und der Prozess der Reservierung wird gestartet. Der Zustand des Termins wechselt zu reserviert, wenn der Anwender einen freien Termin buchen möchte und diesen auswählt. Der Arzt bestätigt oder storniert diesen Termin, sodass der Zustand zu akzeptiert bzw. frei wechselt. Ein Termin erlischt, sobald der Termin in der Vergangenheit liegt oder der Arzt diesen nie bestätigt.

6.4 Sequenzdiagramm

In folgender Abbildung ist das Sequenzdiagram des bekannten Anwendungsfalles:

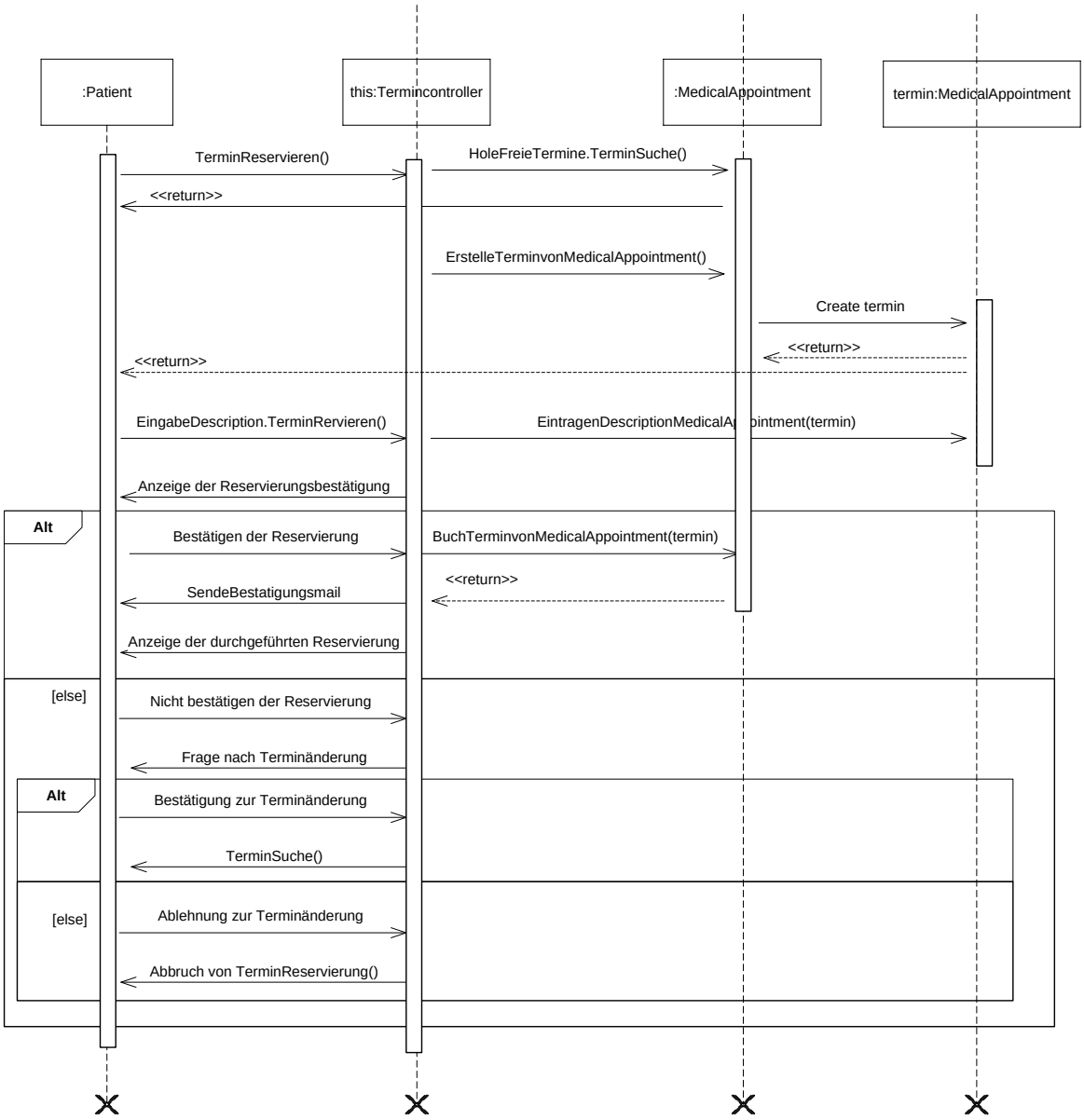

Abbildung 11: Sequenzdiagramm

Die Abbildung 11 beschreibt mit Hilfe eines Sequenzdiagramms die Funktion der Terminreservierung. Dazu wählt der Patient einen freien Termin, der über die Funktion „Termin finden“ angezeigt wird. Diese ist Bestandteil der Klasse Termincontroller, welches die wichtigen Operationen wie *TerminSuche* und *TerminReservieren* enthält. Die Auswahl eines freien Termins erzeugt eine Instanz der Domänenklasse *MedicalAppointment*. Der Patient fügt eine Beschreibung, welches zum Beispiel die akuten Krankheitserscheinungen oder die gewünschte Behandlung sein können, dem Termin hinzu. Nach der Übergabe der Daten und dem Betätigen der Schaltfläche „reservieren“ kann der Patient auf die Nachfrage vom System, ob der Termin wirklich reserviert werden soll, auf zwei unterschiedlichen Arten reagieren. Die eine Möglichkeit ist die positive Beantwortung der Frage und damit einhergehend auch das Anzeigen einer erfolgreich durchgeführten Reservierung. Aber auch das Versenden einer Bestätigungsmail an die im System hinterlegte E-Mail Adresse des Patienten. Die alternative Handlungsmöglichkeit liegt in der negativen Beantwortung der Reservierungsanfrage. Diese Aktion hat zur Folge, dass der Termincontroller nach einer Terminänderung nachfragt. Diese kann der Patient entweder Zustimmen oder Ablehnen und gelangt dementsprechend entweder in dem Fenster der Funktion *TerminReservieren* oder der Übersicht von freien Terminen.

6.5 Klassendiagramm

In folgender Abbildung befindet sich das Klassendiagram nach UML Standard für den Anwendungsfall einer Terminreservierung:

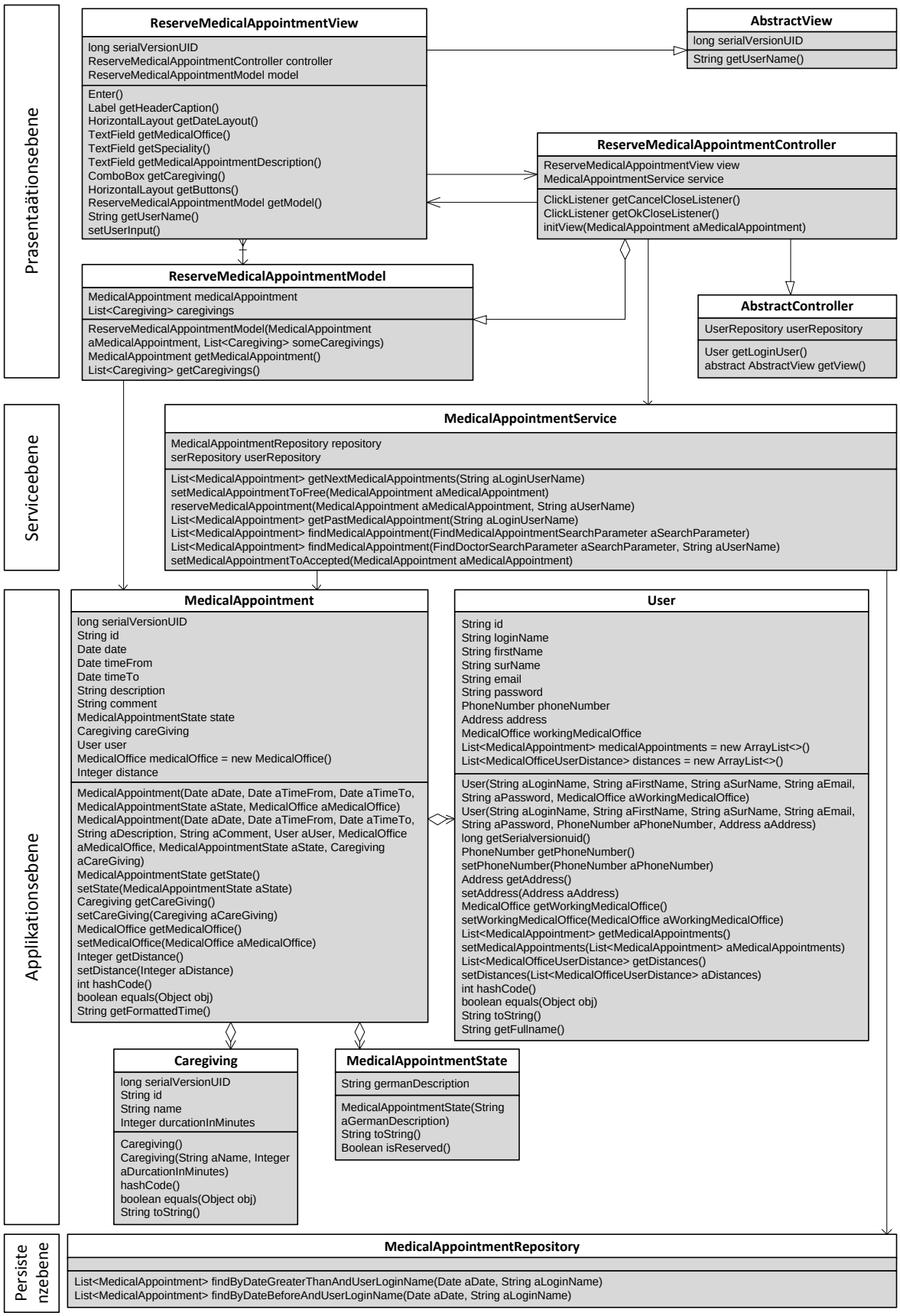

Abbildung 12: Klassendiagramm

In Abbildung 12 werden alle Klassen in einem Klassendiagramm dargestellt, die für den Anwendungsfall einer Terminreservierung benötigt werden. Alle Klassen werden gemäß dem Architekturkonzept in vier Ebenen unterteilt, die bereits in Kapitel 5 innerhalb der Konzeption der Architektur dargelegt worden sind[6]. Bedingt der Übersichtlichkeit wurden alle Standard Getter- und Setter-Methoden innerhalb der Klassen weggelassen.

In der Präsentationsebene befinden sich insgesamt fünf verschiedene Klassen für die Abbildung des MVC Pattern. Dabei stellt die Klasse *ReserveMedicalAppointmentModel* die Daten für die GUI und deren Geschäftslogik bereit, die Klassen *ReserveMedical-AppointmentView* und *AbstractView* stellen diese grafisch bereit und die Klasse *Reser-veMedicalAppointmentController* mit dem *AbstractController* sind für deren Steuerung verantwortlich.

Über die genannte Klasse *ReserveMedicalAppointmentController* des Controllers werden alle Daten des Anwenders an die nächste Ebene, der Serviceebene, überreicht und steuert somit die gesamte Applikation. Diese Schicht enthält die Klasse *MedicalAppoint-mentService*, die daher die angefragten Informationen der Präsentationsschicht bereitstellt und Daten aus der unterliegenden Schicht, der Applikationsschicht, anfordert. Dies geschieht anhand verschiedener Methoden (z.B. *List<MedicalAppointment> getPastMedi-calAppointment(String aLoginUserName*) für die Auflistung bereits vergangener Termine).

Alle dieser Methoden greifen auf die Klasse *MedicalAppointment* der Applikationsebene zu, welche die Informationen der Termine beinhaltet und verwaltet. Zusätzlich kommen hier die Klassen *User* für die Benutzer, *Caregiving* für die Arten von Behandlungen und *MedicalAppointmentState* für den Status der Termine.

Die Daten selbst werden in der Persistenzebene gehalten. Für den Zugriff wird die Klasse MedicalAppointmentRepository verwendet. Diese beinhaltet bspw. Methoden für den Zugriff auf zukünftig möglicher Termine oder Termine die bereits vergangen sind. Ein Zugriff auf diese Methoden erfolgt hierbei über die Serviceebene.

[6] Siehe Kapitel 5: Architekturkonzeption, S. 17

7 Implementierung des Prototyps

Der Prototyp wurde in einer mehrschichtigen, objektorientierten Java-Applikation implementiert, wobei die verschiedenen Schichten die einzelnen Teilbereiche der Software übernehmen und separat kapseln. Es wurde während der gesamten Implementationsphase versucht, nach Möglichkeit die gestellten Anforderungen zu bedienen. Damit die Abhängigkeiten möglichst gering bleiben, ist nur der Zugriff von den höheren Schichten in die niedrigeren Schichten erlaubt.

Innerhalb der Applikation existieren vier Schichten.

- In der Präsentationsebene werden die fachlich verschiedenen GUIs und ihre Interaktionen untereinander gekapselt. Dazu findet der Aufbau in dieser Schicht nach dem MVC Design Pattern statt und wurde während der gesamten Implementierung stringent eingehalten. Zum MVC Pattern gehören die Elemente Model, View und Controller. Das Model enthält die Daten für die GUI und die dazugehörige GUI Geschäftslogik und ist unabhängig von der View und dem Controller. Die View stellt das Modell graphisch auf einer Oberfläche dar und kann direkt auf das Model zugreifen. Die eingehenden Anfragen vom Benutzer werden direkt an den Controller weitergeleitet. Der Controller steuert hingegen die Applikation. Er verwaltet die verschiedenen GUIs, leitet die Benutzeranfragen an die anderen Schichten weiter und nimmt deren Ergebnisse entgegen. Das MVC Pattern wir für jede einzelne GUI angewendet, so dass es in der Applikation mehrfach auftritt. Durch den einheitlichen Aufbau sowie die Verwendung des gängigen Patterns, wird die Einarbeitungszeit deutlich verkürzt. Zudem werden die Wartbarkeit und die Lesbarkeit des Codes erhöht.
- Die zweite Schicht ist die Serviceebene. In der Serviceebene werden die Anfragen aus der GUI zentral entgegengenommen und weiterverarbeitet. Zusätzlich wird in dieser Schicht eine entitätsübergreifende Geschäftslogik gehalten.
- Als weitere Schicht werden in der Applikationsebene die Daten gehalten und die Geschäftslogik innerhalb einzelner Entitäten realisiert.
- Die Persistenzebene ist die vierte Schicht. Sie regelt den Zugriff auf die rationale MySQL-Datenbank und sorgt für das Mapping zwischen den Tabellen und den eingesetzten JAVA-Objekten.

Es wurde darauf geachtet eine klare und strikte Schichtentrennung, anhand des Komponentendiagramms zu realisieren. Die Anwendungsfälle, die durch die Applikation erfüllt werden sollen, wurden mittels Klassendiagramm nach dem UML Standard skizziert und abgebildet. Dort wurde ebenfalls auf eine strikte Schichtentrennung geachtet und auch umgesetzt. Der jeweilige Ablauf von Anwendungsfällen wurde anhand von Sequenzdiagrammen realisiert. Dieses Diagramm war sehr hilfreich, um die einzelnen Schritte und Operationen innerhalb des Anwendungsfalles abzubilden. Innerhalb eines Anwendungsfalles gibt es mehrere Aktivitäten, die der Benutzer durchführen kann. Die wichtigsten Aktivitäten konnten anhand eines Aktivitätsdiagrammes, ebenfalls im UML-Standard, skizziert und umgesetzt werden. Nach einer Aktivität ändern sich innerhalb des Softwareprototypen die jeweiligen Zustände. Diese Zustandsänderungen konnten mittels Zustandsdiagramm aufgenommen und in die Implementierung des Prototypens einfließen. Insgesamt wurde versucht, möglichst nah an der Dokumentation und den Diagrammen zu implementieren und auch deren Begrifflichkeiten zu verwenden, um eine Wiederauffindbarkeit zwischen Dokumentation und Implementierung zu ermöglichen.

Im nachfolgenden wird nun der Prototyp beschrieben: Dabei ist zu beachten, dass nicht alle Funktionalitäten implementiert wurden, ein entsprechender Hinweis nach dem Drücken des Buttons weist den Anwender darauf hin.

Die Applikation verfügt über eine Benutzerverwaltung. Nach dem Aufruf des Programms muss der Benutzer sich mit Benutzername und einem selbst festgelegtem Kennwort an der Applikation anmelden und die Anmeldung durchführen:

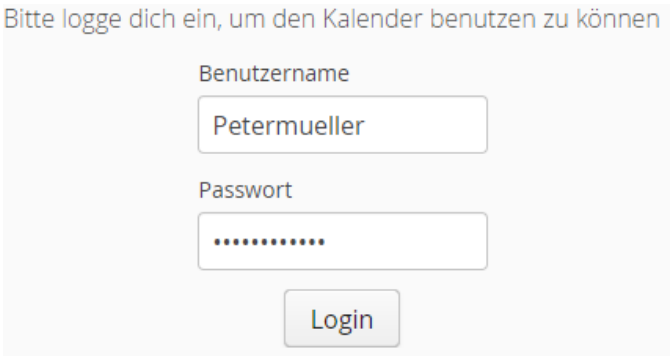

Abbildung 13: Anmeldebildschirm

Nach erfolgreicher Anmeldung gelangt der Benutzer in seine Terminübersicht, wo er auf einen Blick seine bevorstehenden Termine angezeigt bekommt:

Abbildung 14: Übersicht bevorstehender Termine

Von hier aus stehen weitere Funktionen zur Verfügung. Der Benutzer kann unter anderem sich seine vergangenen Termine anzeigen lassen:

Datum	Uhrzeit	Arztpraxis	Fachrichtung	Grund
2017-06-18	09:00 - 09:20	Zahnarztpraxis Benrath	Zahnarzt	Kontrolle
2017-06-28	09:00 - 09:15	Arztpraxis Dr. Günther	Allgemeinmedizin	Blutabnahme
2017-06-23	09:00 - 10:00	Zahnarztpraxis Benrath	Zahnarzt	Zahn ziehen
2017-06-03	15:00 - 15:30	Arztpraxis Dr. Günther	Allgemeinmedizin	EKG

Abbildung 15: Übersicht vergangener Termine

Oder er kann sich auf die Suche nach einem neuen Termin in einer Facharztpraxis begeben. Eine weitere nützliche Funktion in dieser Übersicht ist über den Button Termin absagen erreichbar. Hier kann der Benutzer bereits gebuchte Termine absagen:

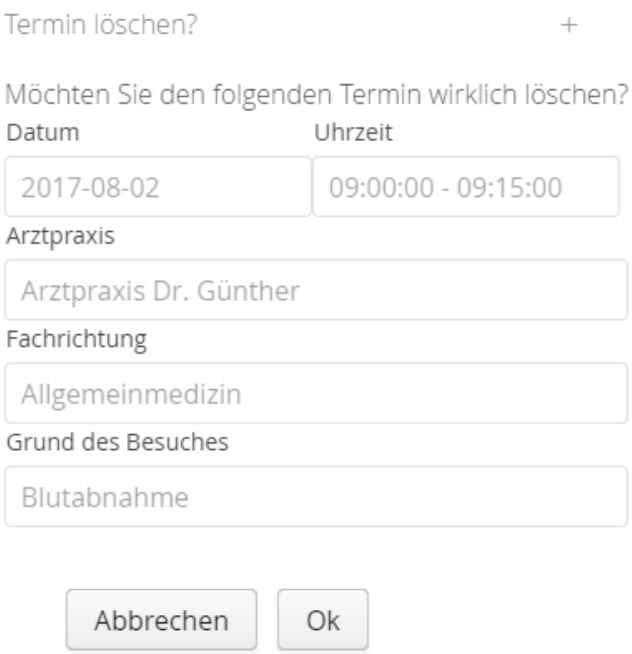

Abbildung 16: Absage eines reservierten Termins

Falls der Benutzer genauere Informationen der Facharztpraxis benötigt, kann er sich diese ebenfalls in der Übersicht anzeigen lassen. Das kann notwendig werden, wenn der Benutzer die Telefonnummer, Email-Adresse oder die Anschrift der Praxis benötigt.

Der Benutzer kann in der Applikation neue Termine suchen und für sich reservieren. In der Suchmaske kann der Benutzer verschiedene Suchkriterien ausfüllen, um ein exaktes Ergebnis für seine Suche zu erhalten. Zu den Kriterien gehören der Praxisname, der Zeitraum in dem der Termin liegen soll, die Uhrzeiten, der Fachbereich und die Entfernung der Praxis.

In der Ergebnisliste tauchen dann die Treffer zu der Suche auf:

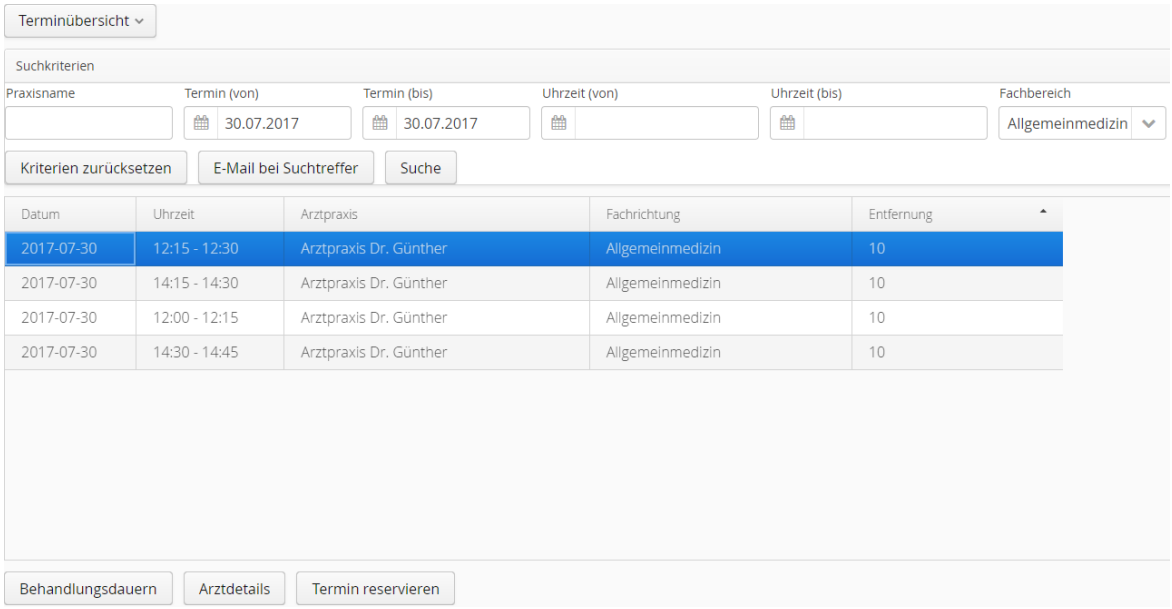

Abbildung 17: Suchmaske für freie Termins

Dort kann der Benutzer dann einen passenden Termin auswählen und vorab reservieren. Diese Reservierungsanfrage wird dann an die jeweilige Praxis übermittelt. Die Praxis kann dann den gewünschten Termin bestätigen oder ablehnen. In der Reservierungsmaske kann der Benutzer über ein optionales Eingabefeld seinen Behandlungswunsch äußern, so dass die Praxis den Grund für den Termin im Vorfeld weiß und sich ggf. darauf einstellen kann:

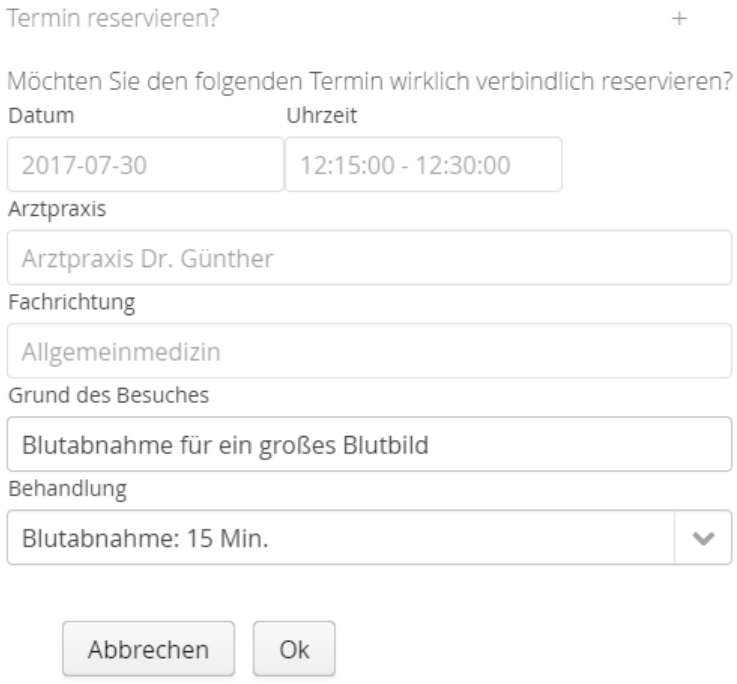

Abbildung 18: Maske für die Reservierung eines Termins

Die Praxis und der Arzt haben über die Applikationsoberfläche verschiedene Funktion zur Verfügung, worüber Termine angeboten, abgesagt, bearbeitet und zugesagt werden können:

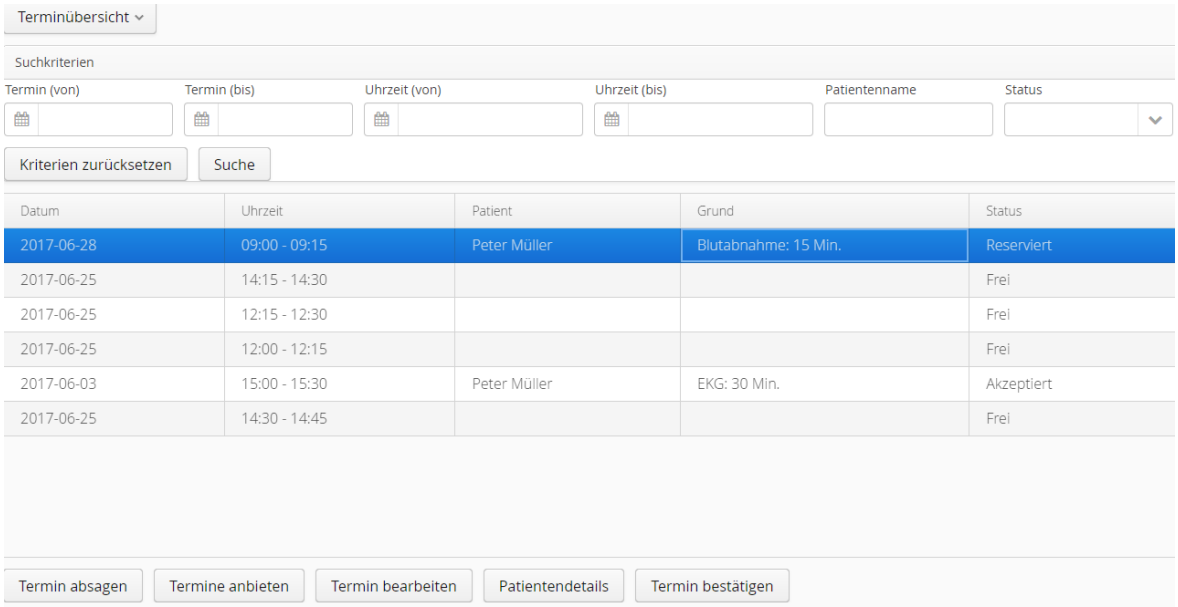

Abbildung 19: Übersicht aller möglichen Termine mit dem aktuellen Status

Das Facharzt-Termin-Verwaltungstool bietet Patienten und Facharztpraxen eine effiziente und effektive Möglichkeit Termine online zu verwalten und einen Überblick über alle anstehenden oder bereits vergangenen Termine zu erhalten.

8 Glossar

In der nachfolgenden Tabelle befinden sich eine Beschreibung aller benötigten Attribute inklusive ihrer Datentypen und deren Beschreibung:

Deutsch	Englisch	Datentyp	Beschreibung
Adresse	Address	Object	Wohnsitz des Patienten oder Sitz der Arztpraxis.
Arztpraxis	Medical Office	Object	Eine Arztpraxis stellt über seine Benutzer (=Mitarbeiter) Termine für andere Benutzer (=Patienten) ein und verwaltet diese.
Behandlung	Caregiving	Object	Behandlung bezeichnet eine Dienstleistung eines Arztes an einem Patienten, die eine bestimmte Zeit in Anspruch nimmt.
Benutzer	User	Object	Ein Benutzer ist jeder, der sich bei der Applikation angemeldet hat. Dies kann auf der einen Seite ein Patient sein oder auf der anderen Seite der Mitarbeiter einer Arztpraxis.
Benutzername	LoginName	String	Name mit dem sich der Benutzer in Verbindung mit dem Passwort authentifiziert.
Beschreibung	Description	String	Kurze Beschreibung der Behandlung.
Datum	Date	Date	Datum, an dem der Termin stattfindet.
Dauer	Duration	Integer	Behandlungsdauer in Minuten, die eine Behandlungsart normalerweise in Anspruch nimmt.
E-Mail	E-Mail	String	E-Mail Adresse zur Kontaktaufnahme.
Fachrichtung	Speciality	Enum	Fachrichtung der Arztpraxis.
Hausnummer	HouseNumber	String	Hausnummer als Teil der Adresse.
Kommentar	Comment	String	Möglichkeit des Arztes, die Behandlung zu kommentieren.
Nachname	Surname	String	Nachname des Benutzers.
Nummer	Number	Long	Rufnummer der Telefonnummer.
Offen bis	OpenTo	Date	Öffnungszeit der Arztpraxis.

Offen von	OpenFrom	Date	Öffnungszeit der Arztpraxis.
Passwort	Password	String	Passwort mit dem sich der Benutzer in Verbindung mit dem Namen authentifiziert.
Patient	Patient	Object	User, der die Dienstleistung der Arztpraxis in Anspruch nimmt.
Postleitzahl	Zipcode	Integer	Postleitzahl als identifizierendes Merkmal der Stadt.
Stadt	City	String	Stadt als Teil der Adresse.
Status	State	Enum	Zustand eines Termins mit den Ausprägungen frei, reserviert und bestätigt.
Straße	Street	String	Straße als Teil der Adresse.
Telefonnummer	Phone Number	Object	Rufnummer unter der ein Patient oder die Arztpraxis erreichbar ist.
Termin	Medical Appointment	Object	Ein Termin ist die Vereinbarung über eine Behandlung zwischen einer Arztpraxis und eines Patienten zu einem definierten Zeitpunkt über eine bestimmte Dauer.
Uhrzeit bis	Time to	Date	Uhrzeit, an der der Termin endet.
Uhrzeit von	Time from	Date	Uhrzeit, an der der Termin beginnt.
Vorname	FirstName	String	Vorname des Benutzers.
Vorwahl	DialingCode	String	Vorwahl der Telefonnummer, auf Grund führender Nullen als String gespeichert.
Webseite	Website	String	Webseite der Arztpraxis um als Patient an weitere Informationen zu gelangen.

Tabelle 6: Glossar mit Datenbeschreibungen

9 Fazit

Mit der Entwicklung des Prototyps des Facharzt-Termin-Verwaltungstool ist in relativ kurzer Zeit eine Applikation entwickelt worden, die es dem Benutzer ermöglicht, einfach und komfortabel, online Facharzttermine zu reservieren und zu verwalten. Dabei wurden die wichtigsten Funktionalitäten dokumentiert und implementiert, um einen technischen und fachlichen Durchstich zu schaffen.

Fachärzte und deren Mitarbeiter haben über diese Plattform die Möglichkeit, dem Patienten freie Termine anzubieten und die Auslastung in den Praxen zu steigern. Kurzfristig freigewordene Termine, die sonst nicht erneut vergeben werden konnten, lassen sich nun online anbieten und können von den Patienten reserviert und gebucht werden.

Der Prototyp beinhaltet außerdem eine Facharzt-Suchfunktion, die dem Patienten die Möglichkeit bietet, Fachärzte in seiner Umgebung zu suchen. Neben der Suche und der Terminverwaltung lassen sich noch weitere nützliche Funktionen implementieren, die aber wegen des zeitlichen Aspektes nicht weiterverfolgt wurden. Denkbar wäre zum einen eine Schnittstelle zum Arztinformationssystem, worin die Befunde und Berichte der Patienten abgelegt sind. Diese Systeme werden in der Regel zur Terminverwaltung eingesetzt. Damit Termine nicht in zwei verschiedenen Systemen gepflegt und verwaltet werden müssen, wäre eine Schnittstelle zu unserem online basierten Facharzt-Termin-Verwaltungstool sehr nützlich. Dabei könnte ein Austausch von Informationen zwischen den Systemen in dem Maße erfolgen, dass z.B. Befunde, Arztbriefe oder Rezepte zur Verfügung gestellt werden könnten. Natürlich alles im Rahmen der geltenden Datenschutzbestimmungen und den notwendigen Sicherheitsmechanismen.

Ein weiteres hilfreiches Feature wäre eine Benachrichtigungsfunktion für den Patienten über kürzlich freigewordene Termine in einer bestimmten Facharztpraxis. Wenn der Patient sich auf unserer Plattform für einen Termin in einer bestimmten Praxis interessiert, könnte der Patient per E-Mail über einen freien Termin informiert werden. Der Patient kann sich dann am System anmelden und den freien Termin für sich reservieren. Das Verteilen von allgemeinen Informationen einer Facharztpraxis als eine Art Newsletter mit Informationen über die Öffnungszeiten, Veranstaltungen, Urlaub der Praxis, etc. könnten über ein Email-System an die Patienten erfolgen.

Es lassen sich sicherlich nach einem Feedback der Anwender noch weitere Funktionen finden, die man in ein Facharzttermin-Verwaltungs-Tool implementieren kann.

Abschließend kann man sagen, dass die Entwicklung und die Implementierung einer Software-Lösung, für nicht Programmierer, ein komplexes und aufwendiges Projekt darstellen. Die ganze Dokumentation und Reflektion war mit sehr viel Aufwand und Fleißarbeit verbunden. Dieser Aufwand hat aber zum erfolgreichen Gelingen des gesamten Softwareentstehungsprozesses beigetragen und war somit sehr hilfreich.

Dieses Dokument zeigt dabei in einem verkürzten Rahmen die Dokumentation von der ersten Vision bis zur Implementierung. Weitere Diagramme (z.B. weitere UseCases) befinden sich in den übrigen Meilensteinen, die im Github zu finden sind.

Nächste Schritte wären nun eine Erweiterung des Prototyps um fehlende Funktionalitäten, wie zum Beispiel einer Registrierung und verschiedene Informationsdialoge, sowie einer Inbetriebnahme auf einem ständig verfügbaren Server mit anschließender Vorstellung bei Ärzten, die als Kunden gewonnen werden müssen.

10Anhang

10.1 AHP Analyse

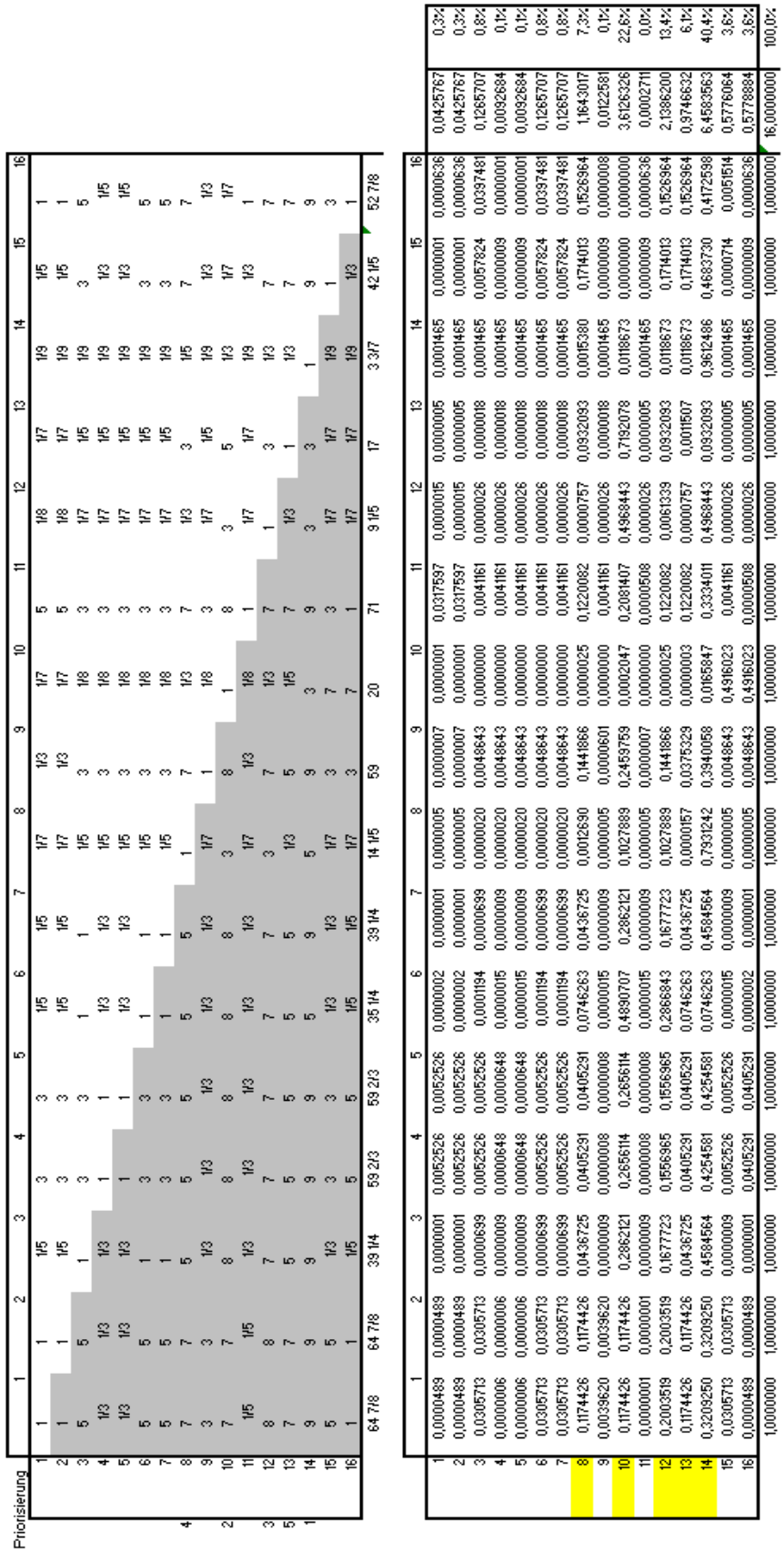

Abbildung 20: AHP Analyse