Technology
Arts Sciences
TH Köln

Konzept
zum Projekt
"DocTimer"

Im Rahmen des Faches
Entwicklungsprojekt interaktive Systeme
WS 17/18

Team

Jiman Saeed 11110055
Elena Correll 11115121

Betreuer

Prof. Dr. Gerhard Hartmann
Prof. Dr. Kristian Fischer

Ngoc-Anh Dang
Franz-L. Jaspers

Inhaltsverzeichnis

1. Einleitung

Im Rahmen des Faches „Entwicklungsprojekt interaktive Systeme“ haben wir das Konzept zu DocTimer entwickelt. Im Folgenden werden wir die Idee zu diesem Projekt und Recherchearbeit vorstellen, sowie die Überlegungen ausführen, die auf Basis der Module Mensch-Computer-Interaktion und Web basierte Anwendungen getroffen wurden.

1.1. Problemszenario

Zeit ist ein kostbares Gut im Leben eines jeden Menschen, daher gewinnt Zeitmanagement, vor allem in der heutigen Gesellschaft, immer mehr an Relevanz. Doch nicht in allen Bereichen ist Zeitmanagement einfach. Denken sie doch nur mal an ihren Hausarzt - finden Sie dass er ein gutes Zeitmanagement hat? Wahrscheinlich nicht, denn im Schnitt verbringt man bei jedem Arztbesuch 24 Minuten im Wartezimmer. Das Problem der Praxen ist, dass sie die Termine oft nicht im Voraus planen können, da Patienten oft kurzfristig einen Termin buchen, was die Zeiten im Terminkalenders nach hinten schiebt und zu dem die Sprechzeiten verkürzt.

1.2. Idee

DocTimer soll das Zeitmanagement von Ärzten und Patienten erleichtern. Wir wollen den Ärzten die Terminvereinbarung abnehmen und eine Lösung dafür finden wie man auch kurzfristige Termine sinnvoll in einen vollen Terminkalender eingliedern kann. Zudem soll den Ärzten eine strukturierte Übersicht zu allen anstehenden Terminen und den jeweiligen Patienten zu Verfügung gestellt werden.

Den Patienten soll die Möglichkeit gegeben werden rund um die Uhr Termine zu vereinbaren und an diese erinnert zu werden. Außerdem soll eine Lösung gefunden werden die Wartezeiten im Wartezimmer so gut wie möglich zu umgehen.

2. Zielhierarchie

Anhand der folgenden Zielhierarchie wollen wir Ihnen unsere Produktvision von DocTimer näherbringen.

2.1 operative Ziele

Das operative Ziele unseres Projektes ist es den Praxen eine Übersicht über Termine zu schaffen, die logisch und chronologisch umgesetzt ist.
Diese Übersicht muss von Ärzten bearbeitet und angepasst werden können und kann ihnen zusätzlich die Möglichkeit bieten Informationen zu den jeweiligen Patienten, wie etwa die Patientenakte, aufzurufen.
Das System muss einem Patienten eine Oberfläche bieten, in der er sich registrieren und Termine vereinbaren kann.

2.2 taktische Ziele

Das taktische Ziel ist das Terminmanagement des Arztes zu automatisieren.
Das System muss einem Patienten einen freien Termin, abhängig vom Anliegen, ausgeben können und diesen nach erfolgreicher Bestätigung speichern.
Es sollen Erinnerungen an einen Termin sowie aktuelle Wartezeiten, die das System berechnen wird, gesendet werden.
Das Erreichen dieser Ziele ist notwendig um die strategischen Ziele umsetzen zu können.

2.3 strategische Ziele

Langfristiges Ziel von DocTimer ist ein System, dass das Zeitmanagement hinsichtlich der Terminvereinbarung von alleine perfektioniert.
Das System soll fähig sein, auf Basis von gespeicherten Daten, Sprechzeiten immer besser anpassen zu können und somit immer genauer die Wartezeit voraussagen zu können.

Unser Ziel den Menschen Zeit zu schenken und Stress zu vermeiden wäre somit erfüllt.

3. Recherche

Um das Produkt und seine spätere Nutzungsumgebung einschätzen zu können, sollte man im Vorfeld Recherchen hierzu durchführen. Diese haben wir in diesem Kapitel zusammengefasst.

3.1. Domänenrecherche

In welchem Bereich unsere Anwendung eingebunden wird und ob sich gesetzliche Einschränkungen ergeben, wird in der Domänenrecherche untersucht. Für DocTimer werden die Domänen der Terminvereinbarungen und Patientenakte betrachtet.

Domäne Terminvereinbarung

Eine Terminvereinbarung findet zwischen zwei oder mehreren Personen statt und ist durch eine Zeitangabe und ggf. einen Ort definiert. Es gibt verschiedene Arten von Terminen, doch wir wollen die allgemeinen Interessen, die Personen bei einer Vereinbarung haben, betrachten.
- an dem Termin sollen alle entscheidenden Personen Zeit haben
- der Termin soll in den persönlichen Zeitplan passen
- der Termin soll an einem geeigneten Ort sein
- der Termin soll eine angemessene Dauer haben
- der Termin soll eingehalten werden
- der Termin soll bei Ausfall angemessen abgesagt oder verschoben werden

bei einem Arzttermin kommt es sehr oft vor, dass sich der vereinbarte Termin nach hinten verschiebt. Wir haben untersucht, dass sich die Patienten nicht unbedingt über diese Tatsache beklagen, sonder über die Ungewissheit wie lange die Wartezeit andauert.

Domäne Patientenakte

Die Patientenakte ist für den behandelnden Arzt ein wichtiger Einblick, denn sie dokumentiert die Untersuchungen, Behandlungen, Therapien und weitere ärztliche Maßnahmen eines Patienten. Das heißt es handelt sich um persönliche Informationen über das Krankheitsbild einer Person, weshalb hier besonders auf die Gewährleistung des Datenschutzes zu achten ist. Die Patienten müssen Nutzungsbedingungen zustimmen um im System die entsprechenden Daten verwalten zu können.

3.2. Marktrecherche

Auf dem Markt gibt es viele Anwendungen, die sich mit dem Thema Terminvereinbarung in der Arztpraxis beschäftigen. Im Folgenden werden wir die zwei Produkte, die in unseren Augen am interessantesten sind, beschreiben, sowie die Vor- und Nachteile gegenüber unserem Produkt aufzeigen.

• *Arzttermine.de* ist eine Website auf der man online einen Termin mit einem Arzt buchen kann. Nach Eingabe der gesuchten Fachrichtung und einer PLZ erscheint eine Liste von Ärzten aus der jeweiligen Region, Informationen zu diesen und den verfügbaren Terminen. Man kann nun einen Termin auswählen und diesen buchen und auch nach Bedarf stornieren.

+ Man bekommt alle Ärzte aus der Umgebung ausgegeben
  man hat Einsicht in alle freien Terminslots
- die Website stellt keine Informationen zu Ärzte in Notdienst
  bietet den Praxen kein Produkt, dass mit der Terminvereinbarung verknüpft ist

• *Doctolib* bietet wie Arzttermine.de den Patienten die Möglichkeit rund um die UhrTermine zu vereinbaren, sowie eine automatische Erinnerung an einen Termin per E-Mail. Das Produkt stellt einen Kalender für Ärzte zu Verfügung, der ihnen die Terminverwaltung vereinfacht.

+ übersichtlicher und gut strukturierter Terminkalender
- informiert Patienten nicht über Wartezeiten
  keine Verknüpfung von Sprechzeiten und Patientenakte

Aus der Marktrecherche hebt sich hervor, was unser Produkt von anderen unterscheidet. Dies ist in dem Kapitel Alleinstellungsmerkmale zusammengefasst.

3.3. Alleinstellungsmerkmale

Die Alleinstellungsmerkmale unseres Systems spiegeln sich in den Teilfunktionen wieder, die zusätzlich zu der automatischen Terminvereinbarung zur Verfügung gestellt werden. Zudem ist das Ziel unseres Projektes ein System, das mit jeder Benutzung präziser wird und somit der Konkurrenz, die einfache automatisierte Terminvereinbarung anbietet, überlegen ist.

DocTimer ist die einzige Applikation, welche auch den Patienten eine Übersicht über ihre anstehenden Termine gibt, sie erinnert und sogar über Wartezeiten informiert. Ein weiteres Alleinstellungsmerkmal ist die Verknüpfung von Terminkalender und Patientenakten - dem Arzt wird ein Produkt zur Verfügung stehen mit dem er eine Übersicht über seine Sprechzeiten und den entsprechenden Patienten in einem hat.

Durch diese Alleinstellungsmerkmale kann DocTimer sich von der Konkurrenz abheben und zu einem einzigartigen und gefragten Produkt werden.

4. Methodischer Rahmen

4.1 Design-Prinzipien

Vor der Entscheidung für ein Vorgehensmodell setzt man die Design-Prinzipien fest. Hierbei wird grundsätzlich zwischen zwei Prinzipien unterschieden, dem "User Centered Design" und dem "Usage Centered Design". Für unser Projekt wählen wir das “User Centered Design“ aus.
Das Ziel, die Attraktivität des Produkts zu steigern, wird durch zentrale Förderung des Benutzungserlebnisses beim Gebrauch des Produkts erreicht. So gut wie möglich soll es die Arbeitsziele der Praxen unterstützen und das Zeitmanagement der Patienten verbessern, daher wollen wir uns besonders auf die Interessen und Probleme der Nutzer konzentrieren und stellen diese in den Mittelpunkt.

4.2 Vorgehensmodelle

Im Folgenden werden wir die uns bekannten Vorgehensmodelle genauer betrachten und erläutern warum wir uns für das Modell „Usability Engineering Lifecycle“ von Deborah Mayhew entschieden haben.

Discount Usability Engineering

Das Modell „Discount Usability-Engineering“ von Nielsen steht für eine kostengünstige, sichtbare Verbesserung der Gebrauchstauglichkeit. Die Evaluation in diesem Modell besteht aus den zehn Heuristiken nach Nielsen auf die besonders Wert gelegt wird, wie auch aus lautem Nachdenken und aus Prototypen die als Szenarien in Papier-basierter Form dargestellt werden.
Für unser Projekt ist dieses Modell eher ungeeignet, da es schwächen bezüglich der Anforderungsanalyse aufweist. Unser Produkt muss sehr exakt und zuverlässig laufen, weshalb wir großen Wert darauf legen eine ausführliche Anforderungsanalyse zu entwickeln.

Scenario Based Usability Engineering

Das Modell von Rosson und Carrol „Scenario Based Usability Engineering“ spezialisiert sich auf das Verstehen, Beschreiben und Modellieren menschlichen Handels anhand der Nutzung von Szenarien. Aus Zeitgründen ist dieses Modell nicht geeignet, da es sich auf das Verstehen mehrerer Nutzungskontexte bezieht, wobei unser Projekt sich nur auf den mobilen Nutzungskontext beschränkt.

Usability Engineering Lifecycle

Das Vorgehensmodell „Usability Engineering Lifecycle“ von Deborah Mayhew konzentriert sich auf die Benutzer und ihre Anforderungen an das System, welche in diesem Projekt eine hohe Priorität besitzen. Um ein fehlerfreies Ergebnis zu erzielen, was für unser Produkt sehr wichtig ist, eignen sich die iterativen Prozesse, die nach dem Modell immer wieder durchlaufen werden, sehr gut.

In User-Profiles finden Sie folgende Informationen: die relevanten Anforderungen die analysiert, Stakeholder die ermittelt und festgelegt werden. Für unsere Projektidee ist es wichtig ein User-Profiles zu erstellen um uns in die Lage eines Benutzers, sei es Arzt oder Patient, hineinversetzen zu können. Dadurch fallen einem ggf. neue Ideen und Anregungen ein. Dies hat für unser Projekt höchste Priorität, da wir unser Produkt den Bedürfnissen und Wünschen der Nutzer anpassen möchten.

Auf dieser Grundlage werden Style-Guides entwickeln, die die Gestaltungsprinzipien, die Gebrauchstauglichkeit und die präzisen Systemanforderungen bestimmen. Da unser System viele Informationen besitzen wird, die gut strukturiert werden müssen, finden wir diesen Punkt besonders hilfreich für die Umsetzung.

Ein konzeptionelles Modell für das zukünftige System wird aus den Ergebnissen der Anforderungsanalyse entwickelt. Diesbezüglich werden Prototypen entworfen. Die Ergebnisse werden erneut in adäquate Style Guides verfasst. Mit Hilfe eines iterativen evaluierten Prototyps werden die Standards des Screen Designs erarbeitet und festgelegt der am Ende alle festgelegten Gestaltungsziele enthalten muss.

Nachfolgend werden die gesamten Elemente des Interfaces erfasst, für alle Tests die Testmodelle generiert und die Benutzerschnittstelle wird in einem iterativen Prozess angepasst, bis alle Gestaltungsziele erfüllt sind, dementsprechend werden die Style Guides überarbeitet.

Nach der Installation des bis dahin entwickelten Systems wird ein Benutzerfeedback eingeholt, aus dem Ideen und Anregungen für weitere Optimierung am System gewonnen werden können. Sollte es erforderlich sein am System weiterzuentwickeln, wird wieder iterativ am System gearbeitet, ansonsten endet der Gesamtprozess. Bei Einhaltung dieses Verhaltensmodells schätzen wir unsere Wahrscheinlichkeit für ein hervorragendes Ergebnis größer ein.

5. Kommunikationsmodell

Es wird zwischen zwei Kommunikationsmodellen unterschieden, dem deskriptiven Kommunikationsmodell, welches den "Ist-Zustand" veranschaulicht und dem präskriptiven Kommunikationsmodell, welches den "Soll-Zustand" verdeutlichen soll.

Im Folgenden werden beide Kommunikationsmodelle an unserem Projekt angewendet und erläutert.

5.1 Deskriptives Kommunikationsmodell

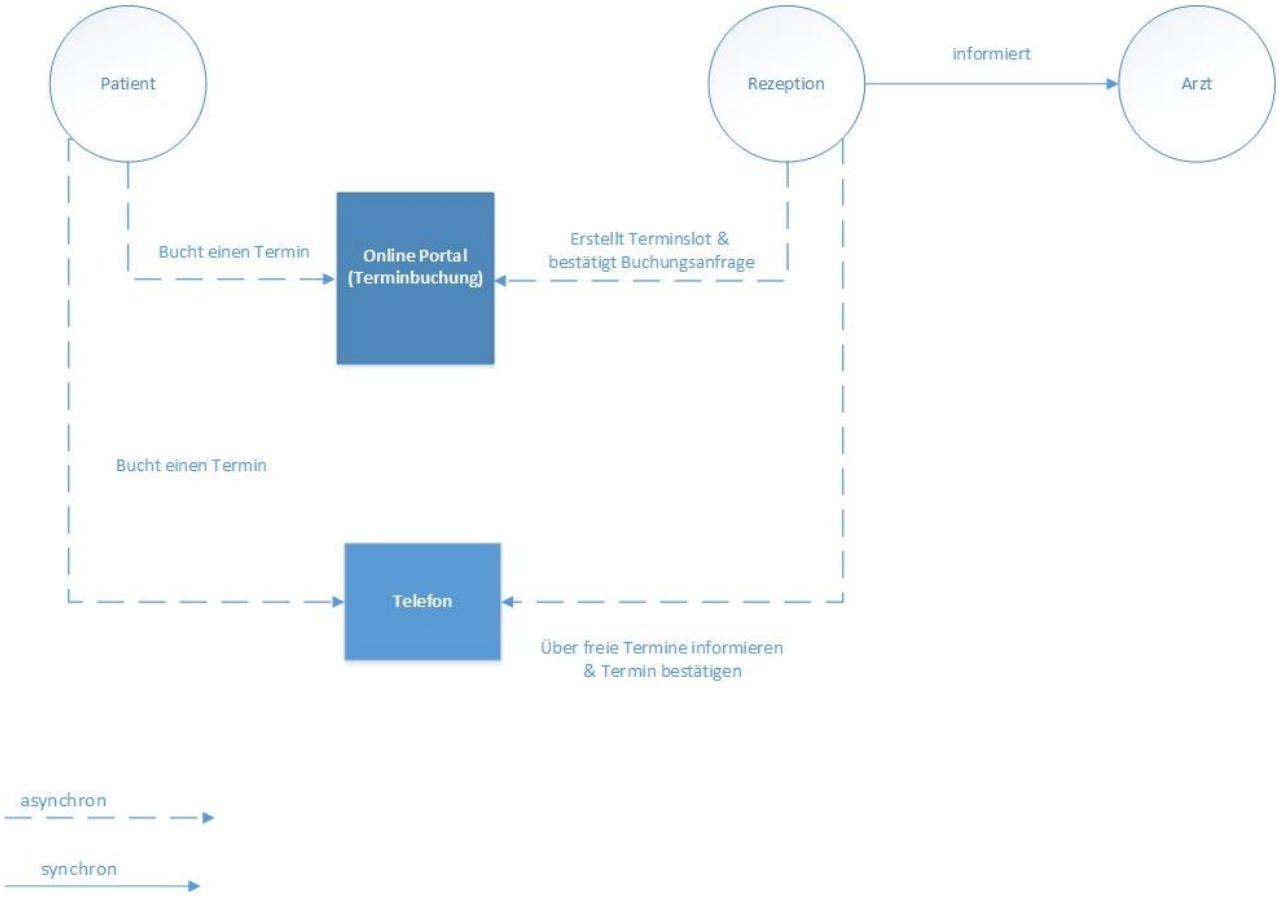

Abbildung 1: deskriptives Kommunikationsmodell

Abbildung 1 zeigt das deskriptive Kommunikationsmodell, d.h. den aktuellen Vorgang einer Terminbuchung beim Arzt.

Der Patient hat zwei Möglichkeiten einen Termin zu buchen, entweder er geht auf das Online Portal der Terminbuchung des jeweiligen Arztes, wo ihm ein freier Terminslot vorgeschlagen wird und er eine Bestätigung der Buchungsanfrage erhält, oder er bucht einen Termin per Telefon. Zum Schluss informiert die Rezeption den Arzt über den Termin.

Was uns an diesem Zustand nicht gefällt ist das Zeitmanagement der Termine. Kurzfristige Termine werden nicht beachtet und Patienten nicht über Terminverschiebungen informiert, dies hat die Folge, dass Patienten lange im Wartezimmer warten müssen, trotz pünktlicher Einhaltung ihres Termins.

5.2 Präskriptives Kommunikationsmodell

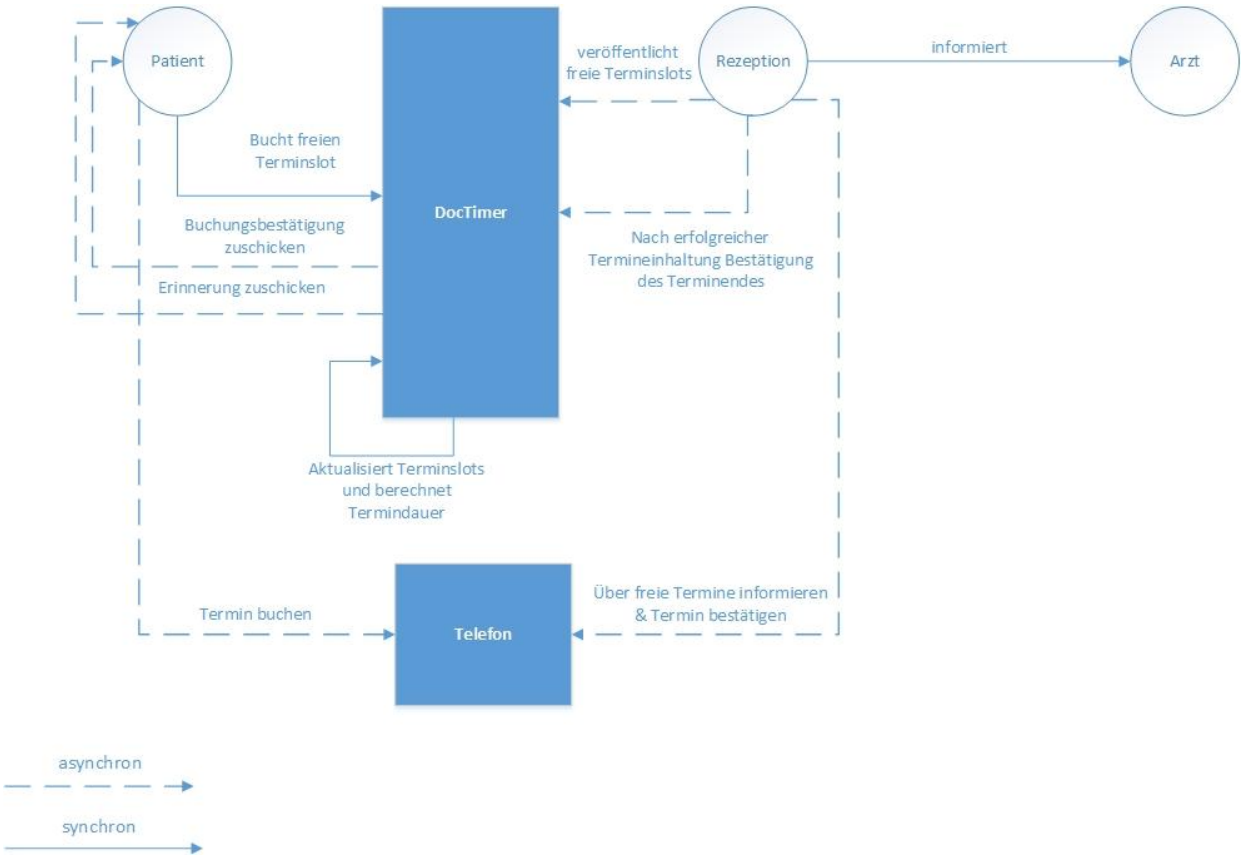

Abbildung 2: präskriptives Kommunikationsmodell

In Abbildung 2 ist das präskriptive Kommunikationsmodell abgebildet, das den Zustand, den wir mit unserem Projekt erreichen wollen, darstellt.

Auf die Möglichkeit telefonisch einen Termin zu vereinbaren wollen wir nicht nochmal näher eingehen, sondern den Fokus auf die Kommunikation mittels den Produkt DocTimer setzen.

Die Rezeption erstellt mithilfe von DocTimer Terminslots. Ein Patient sucht einen Arzt, danach wird er nach dem Grund für den Termin gefragt und ein freier Termin wird ausgegeben, den er buchen kann. Nach Bestätigung wird der Termin in den Kalender des Arztes automatisch eingetragen. Das System schickt am Tag des vereinbarten Termins eine Erinnerung, sodass der Patient seinen Termin nicht vergisst.

Damit DocTimer die Termin- und Wartezeitenberechnung verbessern kann muss die Rezeption dem System jeweils Anfangszeit und Endzeit eines Termins mitteilen.

6. Systemarchitektur

6.1 Architekturdiagramm

In Abbildung 3 ist das Architekturdiagramm zu sehen, dass wir für unser Projekt erstellt haben.

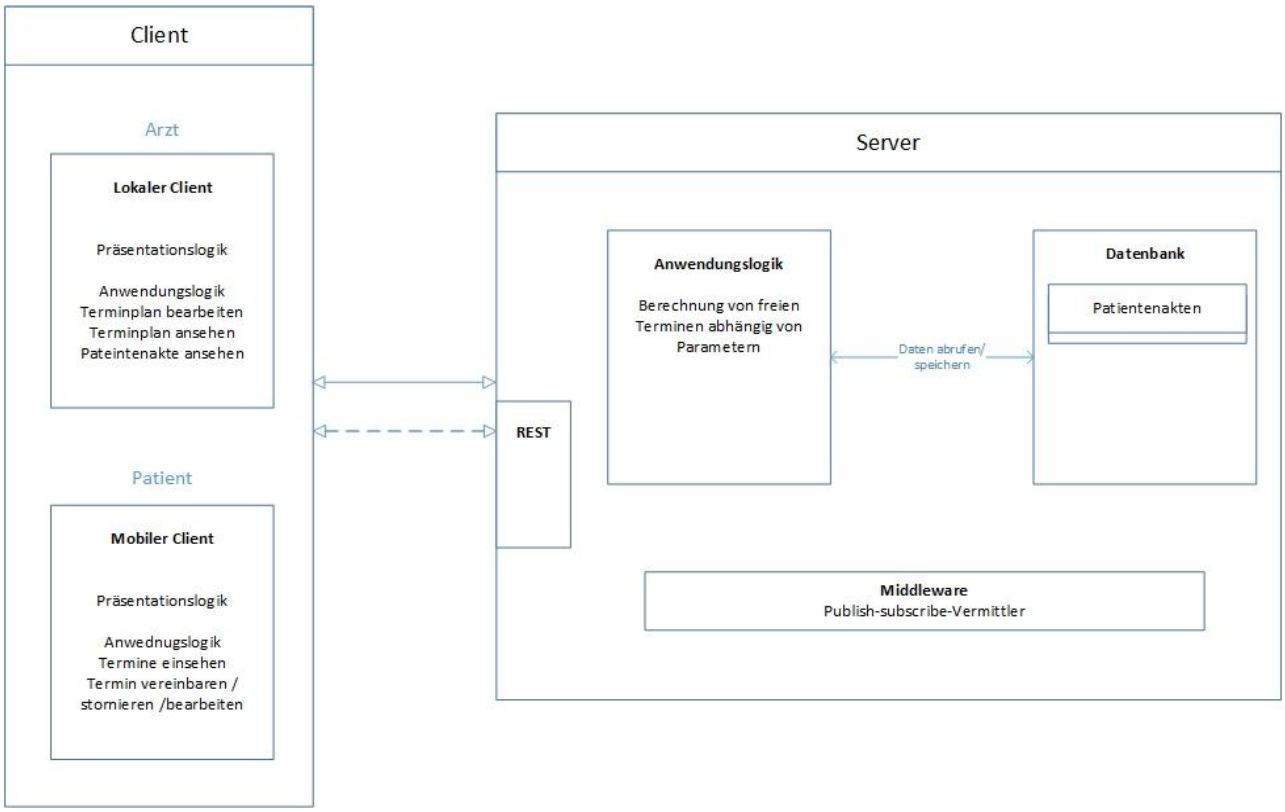

Abbildung 3: Architekturdiagramm

6.2 Architekturbegründungen

Das Projekt wird über ein verteiltes interaktives System realisiert. In diesem Kapitel werden die einzelnen Komponenten genauer erläutert.

Client

Unser System wird einen lokalen und einen mobilen Client haben, die mit Java und JavaScript umgesetzt werden. Der lokale Client der Praxis soll die Terminübersicht präsentieren und die Möglichkeit bieten Patientenakten aufzurufen sowie Termine zu bearbeiten.

Der mobile Client des Patienten ruft Daten, wie Terminart und –dauer, Praxen und freie Termine aus der Datenbank ab, gibt sie aus und bietet die Möglichkeit diese zu bearbeiten. Zudem kann der Benutzer Nachrichten erhalten.

Sever

Der Server von DocTimer beinhaltet die Datenbank, Middleware und einen Teil der Anwendungslogik. Das Grundgerüst wird über NodeJS realisiert werden.
Die Datenbank soll Praxis-, Patienten- und Termindaten persistent speichern, was wir mit einer relationalen Datenbank in MySQL umsetzen werden. Die Datenbank speichert veränderte oder neu erstellte Daten und gibt gefragte Daten aus.
Entscheidungen über die Middleware werden wir nach der Anforderungsanalyse treffen.

Anwendungslogik

Die Anwendungslogik ist auf drei Komponenten verteilt: den mobilen Client, den lokalen Client und den Server.
Der lokale Client verwaltet in einer Datenbank allgemeine Termindaten, die synchron mit der Datenbank des Servers verknüpft sind. Der Server kann aufgrund dieser Daten den nächsten freien Termin berechnen und an den mobilen Client sowie an den lokalen Client asynchron weitergeben. Der mobile Client kann daraufhin eine asynchrone Bestätigungs-Nachricht an den Server senden, worauf ein Termin in die Datenbank eingetragen wird, was wiederum dem lokalen Client asynchron mitgeteilt wird.
Der Server sendet an den Patienten asynchrone Benachrichtigungen zu Wartezeiten und Erinnerung an einen Termin.

Die Anwendungslogik des Servers soll außerdem so programmiert sein, dass sie die tatsächlichen Zeiten eines Termins speichert und auf Grund dieser Daten den Algorithmus für eine Termin und Wartezeitberechnung verbessern kann.

7. Risiken

Bei der Umsetzung des Projekts können Probleme auftrete, die unseren Zeitplan nach hinten verschieben oder die Realisierung erschweren werden. Unter diesem Kapitel, werden die Risiken, die auftreten können, aufgelistet.

Projektspezifische Risiken

Probleme bei der Implementierung
Das Projekt könnte mit den geplanten Technologien nicht umsetzbar sein, daher sollte man sich vorher Alternativen überlegen.

Anforderungen nicht erfüllt
Das Projekt könnte auch ohne Anwendungslogik realisierbar sein und würde die Anforderungen somit nicht erfüllen. In diesem Fall muss das Projekt erweitert werden.

Probleme bei der Berechnung von Terminen
Die Berechnung von Terminen könnte fehlerhaft sein und so zu Lücken und noch mehr Verschiebung im Terminplan führen, deshalb muss großen Wert auf den Berechnungsalgorithmus gelegt werden, was zu Verschiebungen im Projektplan führen würde.

„Intelligente" Berechnung nicht möglich
Die Anpassung der Berechnung und somit die automatisch Perfektionierung des Systems könnte nicht implementierbar sein, was zum Nichterfüllen unseres strategischen Zieles führen wird.

Projektinterne Risiken

Fehlendes Know-How
Die Umsetzung des Projektes kann durch fehlendes Wissen gefährdet sein. Man muss sich frühzeitig um Beschaffung von Informationen und das Auffüllen von Wissenslücke kümmern um ein gutes Endergebnis erzielen zu können.

Ausfall eines Teammitgliedes
Durch Ausfallen eines Teammitgliedes könnte der Projektplan nicht eingehalten werden und zu einem Ressourcenmangel führen, der das Nicht-Erreichen des Zieles zur Folge haben könnte.

8. Proof of Concepts

Um Risiken besser einschätzen zu können, werden Proof of Concepts erstellt.

1. Terminbuchung

Zur Buchung eines Termins gibt das System dem User den nächsten freien und passenden Termin aus.

Exit: DocTimer gibt dem Patienten einen passenden Termin aus, den dieser bestätigt.

Fail: Der vorgeschlagene Terminslot ist für den Patienten nicht passend

Fallback: Der Patient hat die Möglichkeit einen neuen Terminslot anzufordern.

2. Berechnung der Termindauer

Das System legt je nach Anliegen eine Dauer für einen Termin fest.

Exit: Die Berechnung der Termindauer entspricht der tatsächlichen Dauer

Fail: Die Termindauer ist kürzer/länger als DocTimer berechnet hat

Fallback: Die Rezeption bestätigt das Ende eines Termins und der Terminkalender wird dementsprechend angepasst. Das System speichert diese Information und kann diese verwenden

3. Internetverbindung auf lokalem Client stürzt ab

Um als Praxis immer einen aktualisierten Terminplan zu haben, wir eine Internetverbindung vorausgesetzt.

Exit: Das System hat eine stabile Internetverbindung und bekommt alle Termine eigetragen

Fail: Der Client ist nicht mit dem Internet verbunden und es werden Termine in den Kalender eingetragen, die die Praxis nicht abrufen kann

Fallback: Der Server prüft ob der Client verbunden ist und bestätigt dem Patienten den Termin erst nach erfolgreicher Eintragung.

9. Projektplan

Im Folgenden sieht man den Projektplan den wir auf Grundlage des Vorgehensmodells erstellt haben. Zunächst ist eine Übersicht des Workloads, der für die einzelnen Phasen und Teammitglieder geplant ist, in Tabelle 1 dargestellt, dann die einzelnen Aktivitäten und Meilensteine, die in den Wochen bis zur Projektpräsentation anstehen, in Tabelle 2.

Tabelle 1: Projektplan Übersicht

Name	Workload geplant	Workload tatsächlich
Konzept	140 h	
Modellierung	230 h	
Implementierung	200 h	
Jiman Saeed	280 h	
Elena Correll	280 h	
Zusammen	560 h	

Tabelle 2: Projektplan zu einzelnen Aktivitäten und Meilensteinen (MS)

KW	Datum	Aktivität	Unteraktivität	Unteraktivität	WL gep	WL tat
42	16.10.17	Am **Workshop** teilnehmen			2 h	2 h
		Projektplan erstellen			17 h	16 h
			Grobe Übersicht aufstellen		7 h	5 h
			Verfeinern der einzelnen Phasen		9 h	11 h
				Phase 1 verfeinern	2 h	2 h
				Phase 2 verfeinern	2 h	2 h
				Phase 3 verfeinern	2 h	2 h
				Überarbeiten	5 h	4 h
		Projekt definieren			16 h	15 h
			Recherchieren		3 h	4 h
			Idee finden		3 h	2 h
				Brain Storming	2 h	1 h
				Idee endgültig festlegen	1 h	1 h
			Exposé schreiben		8 h	9 h

				Exposé 1 verfassen	5 h	5 h
				Exposé 2 verfassen	5 h	4 h
		Git Repository erstellen und erste Artefakte hochladen			1 h	1 h
		GitHub-Dokumentation schreiben			2 h	2 h
	20.10.17	**Persönlicher MS**	Exposé fertig			
43	23.10.17	**Workshop** und Beratungstermin			2 h	2 h
		Ggf. **Projektidee** überarbeiten			4 h	4 h
		Überarbeitetes **Exposé** schreiben			4 h	4 h
		Konzept erstellen			23 h	21 h
			Einleitung schreiben		2 h	1 h
			Vorgehensmodell auswählen		6 h	5 h
				Einzelne Aktivitäten der Modelle betrachten und abwägen	3 h	2 h
				Ausgewähltes Modell argumentieren	3 h	3 h
			Zielhierarchie schreiben		3 h	3 h
				Strategische Ziele formulieren	1 h	1 h
				Taktische Ziele formulieren	1 h	1 h
				Operative Ziele formulieren	1 h	1 h
			Marktrecherche durchführen		5 h	5 h
				Recherchieren, Vor- und Nachteile formulieren	3 h	3 h
				Alleinstellungsmerkmal argumentieren	2 h	1 h
			Domänenrecherche durchführen		4 h	5 h
				Internetrecherche und Personenbefragung	2 h	3 h
				Recherche ausformulieren	2 h	2 h
			Projektspezifische Risiken formulieren		3 h	3 h
		Projektplan überarbeiten			5 h	6 h
		GitHub-Dokumentation			2 h	1 h

		schreiben				
	27.10.17	**Persönlicher MS**	Projektplan fertig			
44	30.10.17	**Workshop** & Beratungstermin			2 h	2 h
		Konzept schreiben			14 h	12 h
			Kommunikationsmodell entwerfen		6 h	6 h
				Deskriptives KM erstellen	3 h	3 h
				Präskriptives KM erstellen	3 h	3 h
			Systemarchitektur		6 h	5 h
				Software-Schichten & Komponenten ermitteln	1 h	1 h
				Architekturdiagramm entwerfen	2 h	2 h
				Architektur-Begründungen schreiben	3 h	2 h
			Allgemeine Struktur und Formulierung verbessern		2 h	1 h
		Rapid Prototyping			20 h	19 h
			Server implementieren		10 h	10 h
			Client implementieren		10 h	9 h
		GitHub-Dokumentation schreiben			2 h	1 h
	3.11.17	**Persönlicher MS**	Konzept fertig			
45	6.11.17	Beratungstermin				
		Rapid Prototyping			36 h	34 h
			Server implementieren		11 h	11 h
			Client implementieren		10 h	10 h
			Oberfläche implementieren		8 h	7 h
			Präsentation vorbereiten		5 h	5 h
			Präsentation üben		2 h	2 h
		Artefakte ins **GitHub** hochladen				
		GitHub-Dokumentation schreiben			2 h	1 h
	10.11.17	**Persönlicher MS**	Rapid Prototyping fertig			
	12.11.17	**MS 1**	Abgabe Projektplan, Konzept, Rapid Prototyping			
46	13.11.17	**Präsentation Rapid Prototyping**				

		Dokumentation schreiben			38 h	
			Grobe Übersicht erstellen		4 h	
			Deckblatt und erstes Inhaltsverzeichnis entwerfen		2 h	
			Requirement Analysis durchführen		16 h	
				Anforderungen analysieren	4 h	
				Stakeholder analysieren	2 h	
				User Profile erstellen	2 h	
				Projektspezifische Aufgabenanalyse durchführen	2 h	
				Plattform Capabilities Constraints schreiben	2 h	
				Konzeptionelles Modell entwickeln	4 h	
			Anforderungen überarbeiten		6 h	
				Funktionale Anforderungen definieren	2 h	
				Qualitative Anforderungen definieren	2 h	
				Organisatorische Anforderungen definieren	2 h	
			Level 1 des Vorgehensmodells		10 h	
				Konzeptionelles Modell entwickeln	4 h	
				MockUp erstellen	3 h	
				Iterative Evaluation durchführen	3 h	
		GitHub-Dokumentation schreiben			2 h	
46	20.11.17	**Open Space**	Fragen & freies Arbeiten Dokumentation		6 h	
		Dokumentation schreiben			32 h	
			Level 2 des Vorgehensmodells		10 h	
				Screen Design Standards festlegen	4 h	
				Prototyp entwerfen	4 h	
				Iterative Evaluation	2 h	

				durchführen		
			Level 3 des Vorgehensmodells		8 h	
				Detailliertes UI Design entwerfen	5 h	
				Iterative Evaluation durchführen	3 h	
			Interaktion definieren		10 h	
				Paradigmen formulieren	4 h	
				Modi formulieren	3 h	
				Stile formulieren	3 h	
			REST Spezifikation durchführen		4 h	
		GitHub-Dokumentation schreiben			2 h	
	24.11.17	**Persönlicher MS**	Dokumentation hat 600 Wörter			
47	27.11.17	**Open Space**	Frei an der Doku arbeiten, Fragen klären		6 h	
		Dokumentation schreiben			30 h	
			Komponenten beschreiben		8 h	
			Interaktionen beschreiben		9 h	
				Synchrone Interaktion	2 h	
				Asynchrone Interaktion	3 h	
				Daten	4 h	
			Datenschutz zu Web-Services schreiben		5 h	
			Endgeräte beschreiben		4 h	
			Sprache, Typsystem und Speicherung der Daten beschreiben		4 h	
		GitHub-Dokumentation schreiben			2 h	
48	04.12.17	**Open Space**	Fragen zu fertiger Dokumentation stellen		6 h	
		Dokumentation überarbeiten			12 h	
		Implementieren			20 h	
			Anwendungslogik skizzieren		5 h	
			Server implementieren		12 h	
				Datenbanklogik des Kalenders implementieren	6 h	
				Anwendungslogik implementieren	3 h	

				Anwendungslogik implementieren	3 h	
			Datenstruktur spezifizieren		3 h	
		GitHub-Dokumentation schreiben			2 h	
	8.12.17	**Persönlicher MS**	Dokumentation fertig			
49	11.12.17	**Open Space**	Frei an der Implementierung arbeiten, Fragen klären		6 h	
		Implementieren			27 h	
			Server implementieren		20 h	
				Datenbanklogik implementieren	14 h	
				Anwendungslogik implementieren	6 h	
			Client implementieren		7 h	
				Anwendungslogik implementieren	7 h	
		GitHub-Dokumentation schreiben			2 h	
		Installations-dokumentation schreiben			3 h	
	15.12.17	**Persönlicher MS**	Server ist implementiert			
50	18.12.17	**Open Space**	Frei an der Implementierung arbeiten, Fragen klären		6 h	
		Implementieren			27 h	
			Client implementieren		27 h	
				Präsentationslogik implementieren	14 h	
				Anwendungslogik implementieren	13 h	
		GitHub-Dokumentation schreiben			3 h	
		Installations-dokumentation schreiben			2 h	
	22.12.17	**MS 2**	Abgabe Projekt-dokumentation			
51	25.12.17	*Frohe Weihnachten & Guten Rutsch!*				
52	01.01.18	*Let´s do it*				
		Implementieren			24 h	
			UI Design installieren		6 h	
			Webservice einbinden		12 h	

			Client fertig implementieren		6 h	
		Auf **Code Audit** vorbereiten			9 h	
			Inhalte zusammenfassen		5 h	
			Präsentation erstellen		4 h	
		GitHub-Dokumentation schreiben			2 h	
		Installations-dokumentation schreiben			3 h	
	05.01.17	**Persönlicher MS**	Client ist implementiert			
53	08.01.18	**Code Audit**				
		Implementieren			33 h	
			UI Design installieren		16 h	
			Webservice einbinden		17 h	
		GitHub-Dokumentation schreiben			2 h	
		Installations-dokumentation schreiben			3 h	
	12.01.18	**Persönlicher MS**	Web-Service ist eingebunden			
54	15.01.18	**Open Space**	Implementieren und Fragen klären		6 h	
		Implementieren			18 h	
			Code aufräumen		9 h	
			Oberfläche implementieren		9 h	
		Poster erstellen			6 h	
			Brainstorming		4 h	
			Skizzieren des Posters		2 h	
		Prozess-assessment zusammenstellen			3 h	
		GitHub-Dokumentation schreiben			3 h	
		Installations-dokumentation schreiben			2 h	
	19.01.18	**Persönlicher MS**	Oberfläche ist implementiert			
55	22.01.18	**Implementierung** verbessern			10 h	
		Präsentation erstellen			6 h	
			Wichtigste Informationen zusammenstellen		4 h	

			Fachsprache beachten		2 h	
		Poster erstellen			12 h	
		GitHub-Dokumentation schreiben			2 h	
		Installations-dokumentation schreiben			10 h	
			Überarbeiten		7 h	
			Formatieren		3 h	
	28.01.18	**MS 3**	Abgabe von Implementation, Implementations-dokumentation, Fazit, Prozess-assessments			
56	29.01.18	**Code Inspektion**				
		Präsentation üben			20 h	
	02.02.18	**Abgabe Poster**				
57	05.02.18	**Projektpräsentation**				