Anforderungsspezifikation

Version	Datum	Autoren	Beschreibung
0.1	11.04.2018	Alle	Initialversion, Verteilung der Aufträge
0.2	11.04.2018	Martin	Preface
0.3	11.04.2018	Lorenz, Matthias	Erarbeitung Systemarchitektur
0.3	11.04.2018	Andreas	Beginn Erfassung Anforderungen
0.4	13.04.2018	Andreas	Weiterführung Anforderungen
0.5	14.04.2018	Cyrill	System Model eingefügt
0.6	14.04.2018	Matthias	Weiterführung System Requirements
0.7	15.04.2017	Raphael	Testing finalisiert
0.8	16.04.2017	Andreas	Evolution erfasst
1.0	16.04.2018	Cyrill	Abgabefertig

Vorwort

Dieses Dokument beschreibt die funktionalen Anforderungen an das zu entwickelnde PMS und dient der Auftragnehmerin als Grundlage für die Weiterführung des Projekts und unterstützt den Auftraggeber bei der Verifikation des Ergebnisses.

Einleitung

Der Auftraggeber, eine regionale Gesundheitsbehörde, hat uns den Auftrag erteilt, ein Patientenmanagementsystem mit dem Fokus auf Depressionskrankheiten zu entwickeln. Dabei müssen wir ausschliesslich die Funktionen umsetzen, welche der Arzt braucht.
Im Rahmen dieses Auftrags wurden mittels der Design-Thinking-Methode verschiedene wiederverwendbare Ergebnisse erstellt. In einem ersten Schritt haben wir ein grobes Modell abgebildet, um den Scope einzugrenzen:

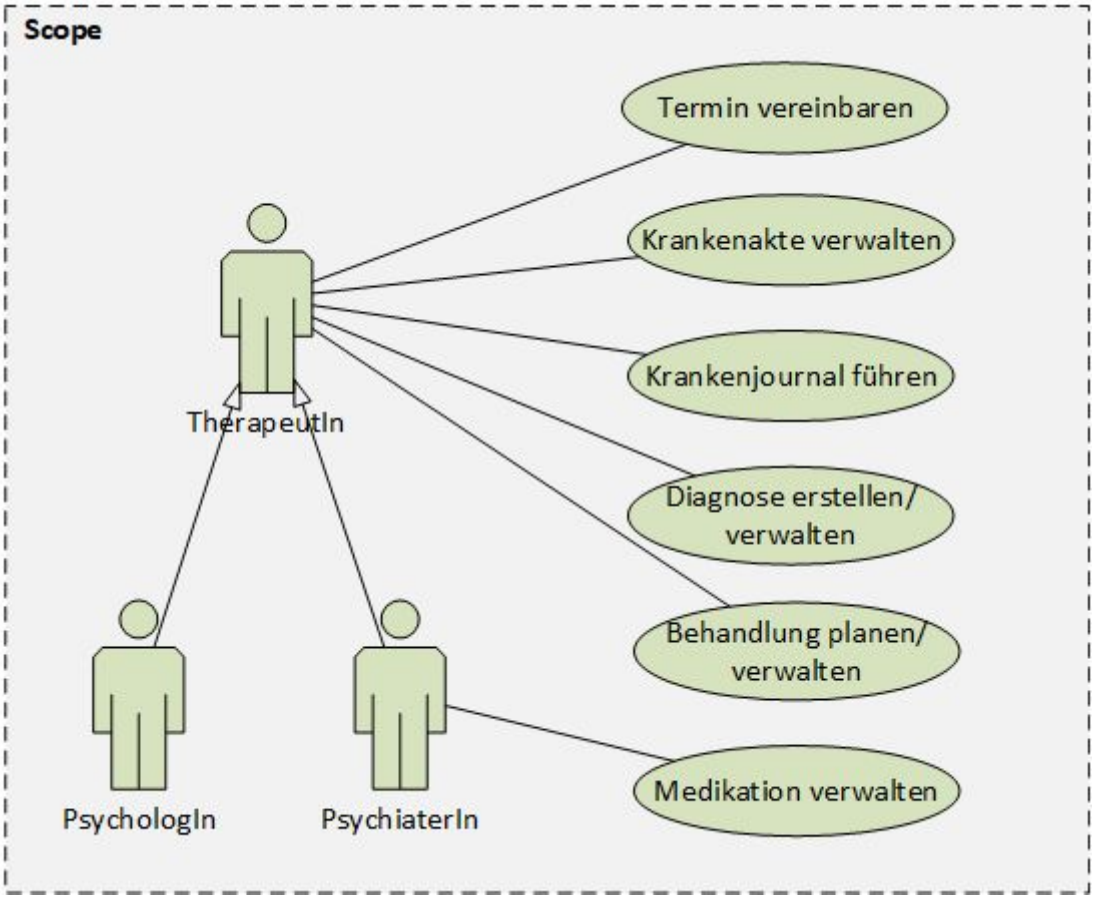


Abbildung 1: Rollen und Use Cases

Auf dieser Übersicht sind die Funktionen des zukünftigen PMS ersichtlich. Wir haben bewusst keine Schnittstellen zu anderen Systemen vorgesehen, da in diesem ersten Projekt das Kernsystem entwickelt werden soll. Zusätzliche Module, welche Schnittstellen zu externen Systemen benötigen bzw. anbieten, folgen in späteren Projekten. Das Ziel der Software ist es, die Ärzte bei der Behandlung von Depressionskranken zu unterstützen und durch die verschiedenen Workflows innerhalb der Use Cases zu führen. Das System muss

möglichst einfach zu bedienen sein und eine Vereinfachung der bisherigen Abläufe ermöglichen. Um dem Leser einer Einblick in die Abläufe zu geben, werden die Use Cases “Krankenjournal führen” und “Medikation verwalten” genauer betrachtet.. Zudem werden wird die Anforderungen sowie den Aufbau des Systems beschreiben und es wird aufgezeigt, welche Tests wir wie durchführen wollen.

Glossar

Backend	Der auf dem Server laufende Programmteil.
Black-Box Test	Black-Box-Test bezeichnet eine Methode des Softwaretests. Hierbei werden Tests anhand der Spezifikation / Anforderung entwickelt ohne Kenntnisse über den Code.
Business Logic	Bezeichnet in einem IT-System jene Ebene, welche die Kommunikation zwischen dem Frontend (siehe oben) und der Datenbank übernimmt.
CSS	CSS bedeutet ausgeschrieben Cascading Style Sheets und ist eine Gestaltungs- und Formatierungssprache vor allem für Internetseiten.
Framework	Ein Softwaregerüst das den konzeptionellen Rahmen sowie generische Funktionalität für die Entwicklung einer Applikation liefert.
Frontend	Das Frontend bezeichnet den Teil der Applikation, welcher für den User sichtbar ist.
Html	Steht für Hypertext Markup Language. Der html-Code gibt dem Html-Dokument (z.B. einer Webseite) Anweisungen zur Strukturierung und Formattierung der darzustellenden Inhalte.
JavaScript	JavaScript ist eine Programmiersprache, die es ermöglicht, dass Webseiten sich dynamisch aufbauen und dem Benutzer anpassen können.
PMS	Steht für Patientenmanagement System und meint eine Software für die Verwaltung und Verarbeitung von Patienten- und Falldaten.
Relationale Datenbank	Eine relationale Datenbank bezeichnet eine Sammlung von Tabellen (den Relationen), in denen Datensätze abgespeichert sind.
Sprint	Ein Sprint ist ein Arbeitsabschnitt, in dem ein Inkrement einer Produktfunktionalität implementiert wird.

Systemarchitektur	Die Darstellung der Struktur und Komponenten eines IT-Systems.
Unit-Test	Wird in der Softwareentwicklung angewendet, um die funktionalen Einzelteile (Units) von Computerprogrammen zu testen.
User centered design	Methode des Softwaredesigns in der der Anwender ins Zentrum gestellt wird und sich die Software nach ihm ausrichten soll.
Vier-Augen-Prinzip	Ist eine Sonderform des Mehr-Augen-Prinzips und besagt, dass wichtige Entscheidungen nicht von einer einzelnen Person getroffen werden oder kritische Tätigkeiten nicht von einer einzelnen Person durchgeführt werden sollen oder dürfen. Ziel ist es, das Risiko von Fehlern und Missbrauch zu reduzieren.
VPN	Steht für virtual private network. Darüber kann bspw. ein Mitarbeiter von zuhause aus auf das firmeninterne Netzwerk zugreifen.
Webapp	Steht für Webapplikation und meint ein Programm, das direkt im Webbrowser aufgerufen/ bedient werden kann.

Anforderungen

Definition der Benutzeranforderungen

Untenstehenden noch einmal eine nummerierte Übersicht der vorhandenen Use Cases. Auf Use Case 3 und 6 wird explizit eingegangen. Auf Basis der aller Use Cases werden anschliessend die Benutzeranforderungen erstellt.

Nr.	Use Case
1	Termin vereinbaren
2	Krankenakte verwalten
3	**Krankenjournal führen**
4	Diagnose erstellen
5	Behandlung planen/verwalten
6	**Medikation verwalten**

Use Case 3: Krankenjournal führen

Szenario:
Arzt beendet Sitzung mit Patient und trägt Neuerungen in des Krankenjournal ein.

Kurzbescheibung:
Der Arzt öffnet den Fall des Patienten und wählt im Dashboard den Punkt “Journal” aus. Dies kann auch über den Termin-Workflow passieren. In der danach angezeigten Journalübersicht fügt er einen neuen Eintrag hinzu und speichert diesen, sodass die Behandlungssitzung für andere Ärzte oder das Pflegepersonal sicht- und nachvollziehbar ist.

Beteiligte Akteure:
Therapeut

Auslöser/Vorbedingung:
Beenden einer Therapiesitzung

Ergebnisse/Nachbedingung:
Neuer Eintrag im Krankenjournal eines Patienten

Ablauf:

Nr.	Wer	Was
1	Therapeut	Öffnet Fall (oder den aktuellen Termin) des betreffenden Patienten
2	Therapeut	Öffnet das Journal im betreffenden Fall (oder im Termin-Workflow).
3	Therapeut	Klickt auf “Neuer Eintrag” um das Journal zu ergänzen
4	Therapeut	Schreibt Informationen zur Sitzung mit dem Patienten in neuem Eintrag nieder
5	Therapeut	Speichert neuen Eintrag

Use Case 6: Medikation verwalten

Szenario:
Arzt entscheidet vor, während oder nach der Sitzung die Medikamente, die der Patient einnimmt, anzupassen.

Kurzbeschreibung:
Der Arzt öffnet den Fall des Patienten und wählt im Dashboard den Punkt “Medikation” aus. Dies kann auch im Termin-Workflow passieren. In der danach angezeigten Medikationsübersicht kann der Arzt Medikamente hinzufügen, bestehende auf aktiv/inaktiv setzen oder Medikamente entfernen.

Beteiligte Akteure:
Therapeut

Auslöser/Vorbedingung:
- Der Patient benötigt ein neues Medikament
- Wiederaufnahme/Absetzung eines Medikaments zur Analyse der folgenden Reaktion auf den Patienten

Ergebnisse/Nachbedingung:
Neuer Medikationsplan des Patienten

Ablauf:

Nr.	Wer	Was
1	Therapeut	Öffnet Fall (oder den aktuellen Termin) des betreffenden Patienten
2	Therapeut	Öffnet die Medikation im betreffenden Fall (oder im Termin-Workflow).
3a	Therapeut	Klickt auf “neue Medikation”
3a1	Therapeut	Wählt gewünschtes Medikament, dessen Dosierung und die geplante Verabreichungszeit aus
3a2	Therapeut	Speichert neue Medikation
3b	Therapeut	Wählt bestehende Medikation aus
3b1	Therapeut	Setzt Status der Medikation auf aktiv/inaktiv
		Wenn aktiv: Passt den Zeitraum der Medikation an
4	*Therapeut*	*Wenn in Termin-Workflow: Bestätigt Medikationsübersicht mit “erledigt”.*

Folgende Benutzeranforderungen ergeben sich aus den erwähnten Use Cases (in Tabelle mit UC abgekürzt):

Nr.	Anforderung	Basierende Use Cases
1	Der Arzt muss einen Termin erfassen können	UC1
2	*Das Pflegepersonal muss einen Termin erfassen können (out of Scope)*	*UC1*
3	Der Arzt muss Änderungen am Termindatum machen können	UC1
4	*Das Pflegepersonal muss Änderungen an einem Termindatum machen können (out of Scope)*	*UC1*
5	*Die Administration muss Änderungen an den Personendaten eines Patienten durchführen können (out of Scope)*	*UC2*
6	Der Arzt muss das Journal eines Patientenfalls erweitern können	UC2,3,4,5
7	Der Arzt muss Einträge anderer Ärzte im Journal eines Patientenfalls sehen können	UC3,4,5
8	*Das Pflegepersonal muss Einsicht in das Journal eines Patienten haben (out of Scope)*	*UC5*
9	Der Arzt muss eine Diagnose für einen Patienten erfassen können	UC4
10	Der Arzt muss im Termin-Workflow die Diagnose eines Falls eines Patienten ändern können	UC4
11	*Das Pflegepersonal muss Einsicht in die Diagnose haben (out of Scope)*	*UC4*
12	Der Arzt muss im “News Feed” alle wichtigen Informationen zum Termin finden	UC5
13	Der Arzt muss die Medikation anpassen können direkt aus Dashboard und/oder aus dem Terminworkflow her anpassen können	UC5,6
14	Der Arzt muss den Medikationszeitraum anpassen können	UC5,6
15	Der Arzt muss die Möglichkeit haben, den Terminworkflow unvollständig zu speichern	UC1,5
16	Im Fall-Dashboard sollen Zugriffe auf den News-Feed/Termin-Feed, das Falljournal, die	UC2,3,4,5,6

	Medikation und die Diagnose direkt möglich sein	
16	*Das Pflegepersonal soll die Möglichkeit haben, Notizen zum Patienten im aktuellen Fall für den Arzt zu hinterlegen (out of Scope)*	*UC4,5*

Es sind zudem folgende NFR Anforderungen vorhanden/erwünscht:

Nr.	Anforderung	Basierende Use Cases
1	Das Fall-Dashboard soll übersichtlich gestaltet sein	-
2	Die Übersicht zu “Medikation” und “Journal” soll bei Zugriff aus Terminworkflow analog zum Direktzugriff aussehen.	-
6	Das Navigieren in den unterschiedlichen Applikationsteilbereichen soll in annehmbarer Zeit geschehen, auch bei grösseren Fällen	-

Spezifikation der Systemanforderungen

Die folgende Tabelle listet die funktionalen Systemanforderungen zu den jeweiligen Benutzeranforderungen auf.

Nr.	Funktionale Systemanforderung	Zugrunde liegende Benutzeranforderung
1	Beim Erfassen oder Ändern von Terminen werden allfällige Termin- und Ressourcenkonflikte überprüft (z.B. Raumbelegung).	1,3
2	Sobald ein Fall zum Patienten existiert hat der Arzt die Möglichkeit einen neuen Journal-Eintrag zu eröffnen. Jedem Journaleintrag wird beim Speichern automatisch der Zeitstempel und der Autor angefügt.	6
3	In der Journalansicht sind die Einträge chronologisch geordnet (neueste zuoberst) und sie enthält Filter nach Rolle der Autoren.	7
4	In der Diagnose kann der Arzt den Namen der Erkrankung, die Hauptsymptome, die Zusatzsymptome, deren Ausprägung sowie weitere Kommentare zum Krankheitsbild des Patienten erfassen. Überdies wird die Suizidneigung hier festgehalten. Änderungen der Diagnose sind jederzeit	9,10

	möglich.	
5	Änderungen im elektronischen Patientendossier generell werden vollständig geloggt, mit Autor, Fallnummer und Zeitstempel, damit sämtliche Änderungen nachvollziehbar sind.	7
6	Der Newsfeed listet neben den neusten Journaleinträgen auch die wichtigsten Informationen aus dem elektronischen Patientendossier sowie die letzten Therapie-Termine des Patienten auf.	12
7	Das System umfasst eine interne Medikamentendatenbank, wo zu jedem Medikament zusätzlich zu den üblichen Informationen allfällige Wechselwirkungen mit anderen Medikamenten oder Substanzen enthalten sind. Die Daten werden regelmässig von einer externen Medikamentendatenbank importiert.	
8	Beim Erfassen und Ändern von Medikationen werden jeweils Wechselwirkungen mit anderen Medikamenten geprüft und bei allfälligen Komplikationen wird eine Warnung angezeigt.	13
9	Der Arzt kann ein neues Medikament zum Dossier hinzufügen und dabei ein Startdatum sowie bei Bedarf ein voraussichtliches Enddatum eingetragen. Diese Zeitdaten können jederzeit von einem Arzt geändert werden.	14
10	Die Arbeit am PMS kann jederzeit gespeichert und unterbrochen werden, auch wenn ein Workflow noch nicht beendet wurde.	15
11	Das PMS bietet die Möglichkeit interne Links zu anderen Programmbereichen hinzuzufügen.	16

Die nicht funktionalen Systemanforderungen umfassen die folgenden Punkte:

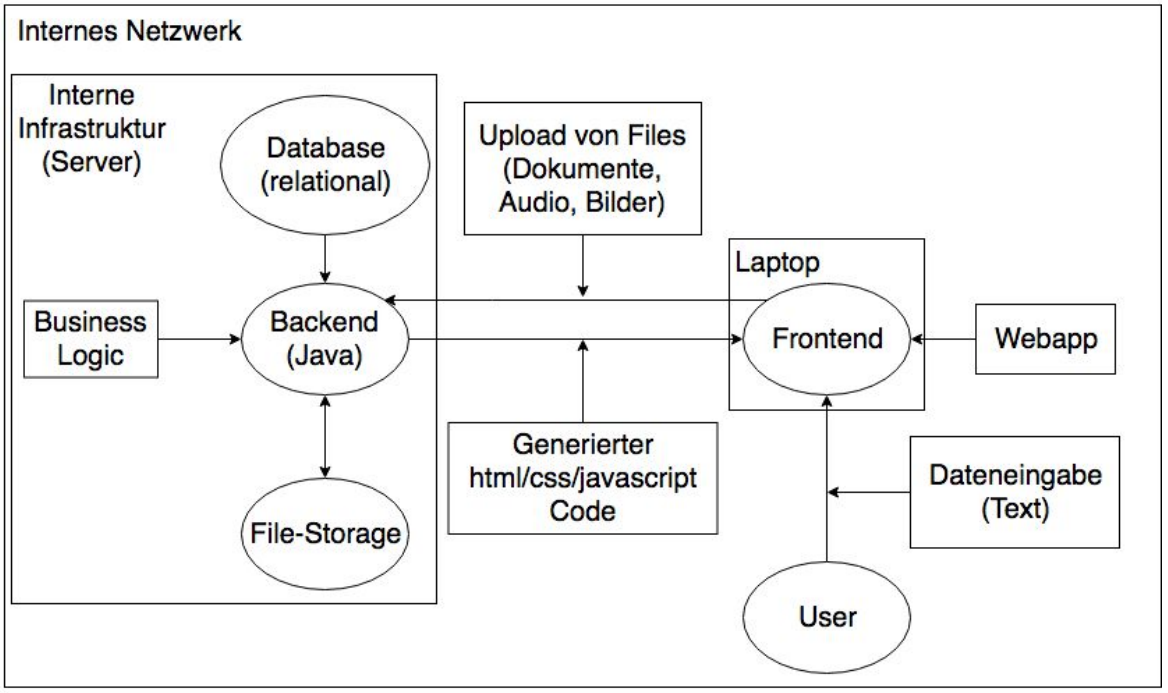
Nr.	**Nicht funktionale Systemanforderung**
1	Die Gestaltung des Systems ist user centered mit starkem Fokus auf leichte Bedienbarkeit.
2	Die Applikation soll in allen Webbrowsern brauchbar sein
3	*Die Applikation soll auf Tablets/Mobilgeräten brauchbar sein (out of Scope, nice to have)*

4	Die Applikation soll betriebssystemunabhängig verwendbar sein
5	Die Applikation soll mit dem Framework “Vaadin” umgesetzt werden. Das Backend basiert auf Java 8.
6	Die Umsetzung des elektronischen Patientendossier folgt den gesetzlichen Richtlinien (Gesetzessammlung EPDG).
7	Rechte müssen rollenspezifisch vergeben werden können. (Out of Scope, da nur die Rolle der Ärzte implementiert wird).

Systemarchitektur

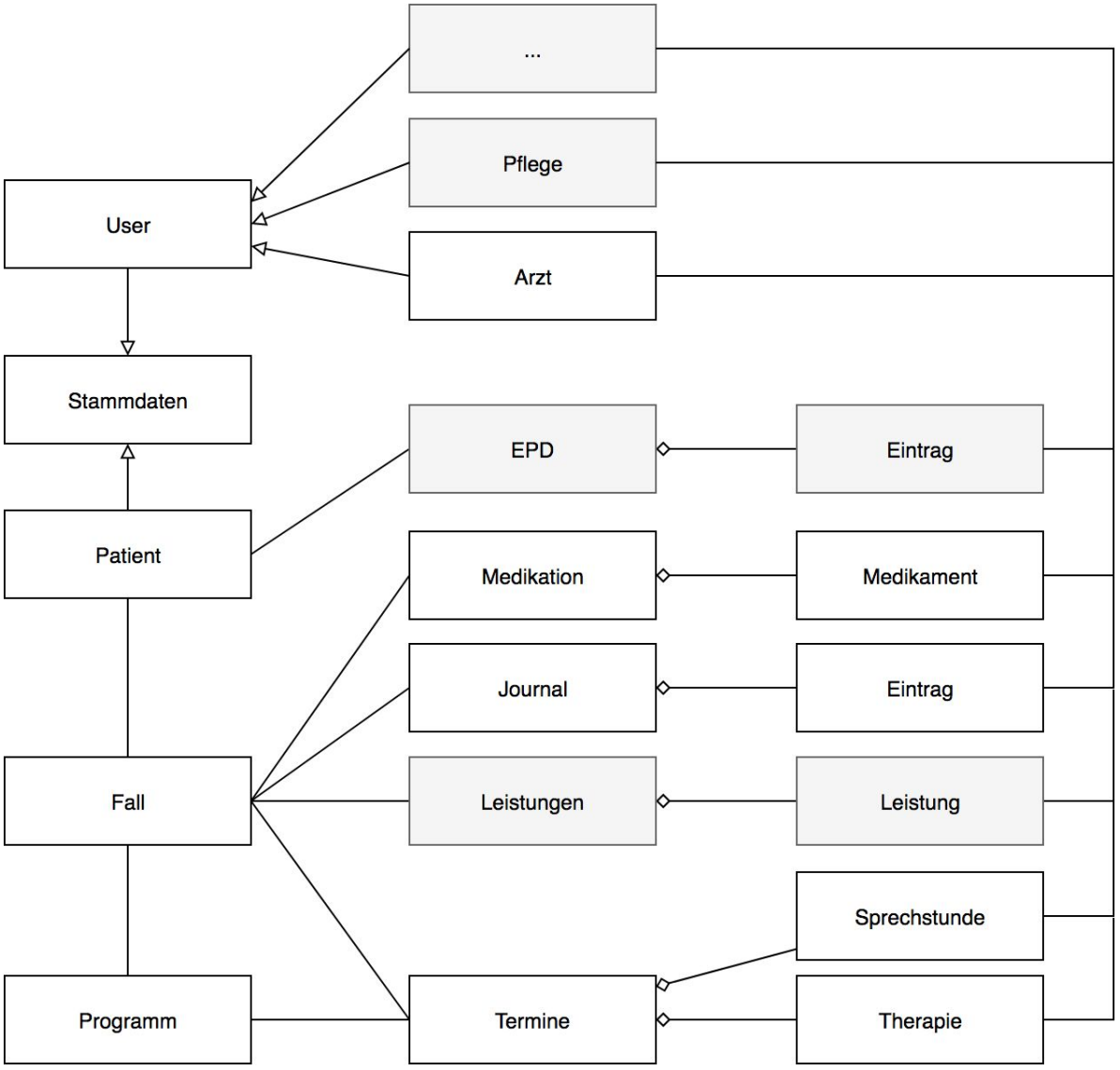
Die Systemarchitektur basiert auf einem Client-Server Modell. Im System werden sensible Patientendaten verarbeitet und gespeichert, wodurch die Anforderungen an den Datenschutz erhöht sind. Um letzteren gerecht zu werden befindet sich
das ganze System in einem internen Netzwerk und ist von aussen nicht (resp. nur via VPN) erreichbar.

Der Benutzer kann nur auf das Frontend zugreifen. Dieses kommuniziert mit dem Backend, welches direkt mit der Datenbank und dem Filesystem interagiert. Das Backend, welches in Java geschrieben ist, generiert aus diesen Daten für den Webbrowser verständlichen Code und schickt diese an das Frontend weiter.



System Modelle

Das folgende Modell beschreibt beschreibt die Zusammenhänge zwischen den Systemkomponenten und zeigt auf, wie die realen Objekte im Patientenmanagementsystem abgebildet werden.



Die Zusammenhänge lesen sich analog des folgenden exemplarischen Beschriebes für eine Sprechstunde: “Ein Patient mit der Adresse X (Stammdaten) hat betreffend seiner aktuellen Depression (Fall) zu einem bestimmten Termin eine Sprechstunde beim Arzt Z”.

Die grau hinterlegten Bereiche seien hier nur der Vollständigkeit wegen angedeutet. Sie liegt ausserhalb des Scopes.

Systemweiterentwicklung

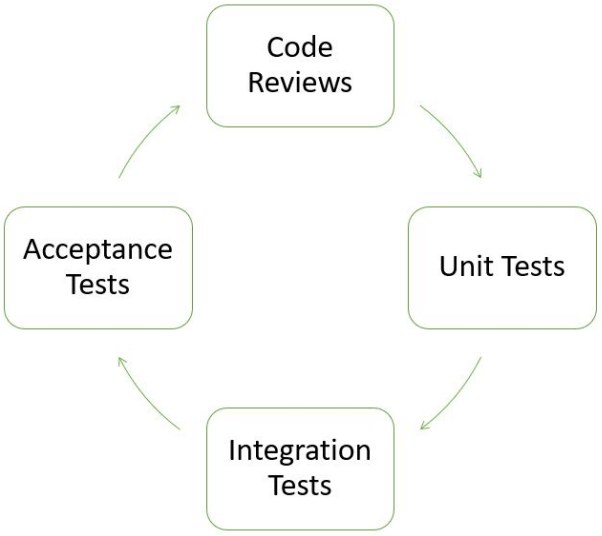
Das System bietet nach der Entwicklung die grundlegende Management-Möglichkeit für Ärzte, Patientenfälle zu koordinieren und zu verwalten. Durch den Terminablauf ist eine nachvollziehbare Struktur gegeben, bei der Informationen zum betreffenden Fall hinterlegt werden können.

Sollten neue Workflows nötig sein, können diese ins bestehende System integriert werden um auch hier eine einheitliche Informationsstruktur zu haben. Durch Berechtigungsänderungen können Informationen auch anderen Leuten zur Verfügung gestellt werden (z.B Pflegerinnen und sonstigen Direkt-Beteiligten). Das System ist zudem auch erweiterbar für andere Krankheitsbilder, da es die grundlegenden Komponenten eines alltäglichen PMS besitzt. Dadurch dass es eine Webapplikation ist, wäre auch der Einsatz auf mobilen Geräten wie Tablets denkbar, sofern das eingesetzte Framework dies unterstützt.

Hardwarespezifische Änderungen durch Wachstum der Datensammlung könnten durchaus anfallen, jedoch kann unabhängig von Applikationsanpassungen geschehen.

Qualitätssicherung und Testing

Um die Qualität unserer Software zu gewährleisten, haben wir vier wichtige Eckpfeiler definiert. Sie werden in jedem Sprint durchgeführt und sind somit iterativ abarbeitbar.



Code Reviews

Um unsere Code Qualität hoch zu halten, werden Code Reviews durchgeführt. Dies wird bei Pull Requests auf Github gemacht und ist somit dort auch dokumentiert. Weil niemand etwas alleine in den Develop Branch mergen kann, ohne dass jemand anderes den Pull

Request akzeptiert, ist dies automatisch gegeben. Dank dem Vier-Augen-Prinzip welches hier zur Anwendung kommt, können Fehler früh entdeckt und eliminiert werden. Beim Code Stil werden wir uns an das Buch Clean Code von Robert Cecil Martin orientieren.

Unit Tests

Wir versuchen eine möglichst hohe automatisierte Testabdeckung unseres Codes zu erreichen. Dadurch wollen wir eine Verbesserung der Wartbarkeit und eine Minimierung der Fehleranfälligkeit erreichen. Bei jedem Pull Request müssen alle Tests auf grün sein, ansonsten wird er automatisch abgelehnt.

Integration Tests

Die abgeschlossenen Tasks werden nach dem Pull Request auf einem Testsystem getestet. Dabei werden verschiedene Arten von Integration Tests durchgeführt. Neben teils automatisierten Tests werden auch Black-Box Testings durchgeführt. Bei Black-Box Tests kennt die Testperson den Code nicht und achtet somit auf andere Aspekte, als jemand der den Code kennen würde.

Acceptance Tests

Alle Stories in einem Sprint haben Akzeptanzkriterien. Anhand dieser Kriterien werden vom Kunden die Acceptance Tests durchgeführt und die Stories abgenommen. Diese Tests werden meist erst im Sprint nach der Umsetzung der Story durchgeführt, sobald das Deployment auf der Integration Umgebung des Kunden durchgeführt wurde.