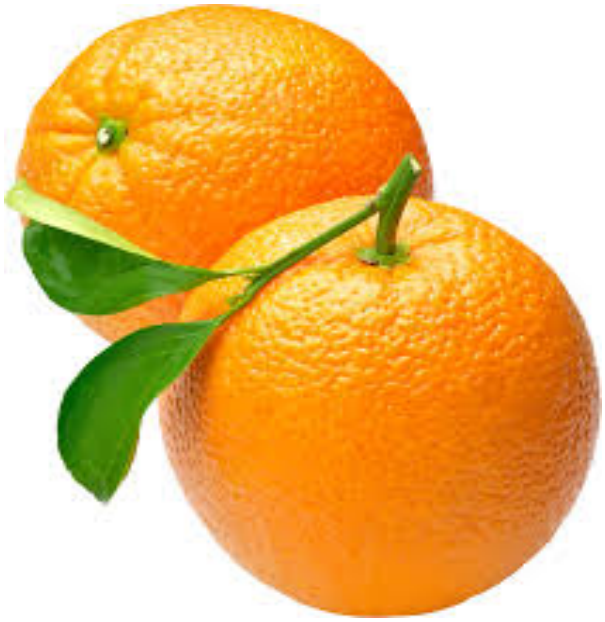
MHC-PMS Team Orange

Teamarbeit im Fach Software Engineering & Design
BTI7081 - BFH Bern – Vogel/Künzler



	Namen	**Datum**	**Visum**
Erstellt/ bearbeitet	Team Orange	05.04.2019	
Überprüft:			
Freigegeben:			

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

1. Allgemeines

Im Rahmen eines fiktiven Kundenauftrags erarbeiten Studenten der Berner Fachhochschule im Modul Software Engineering and Design (SoED) ein Projekt, welches ein Patienten Management Systems (PMS) als Resultat ergeben soll.

Dieses Dokument beschreibt die Anforderungen an das System, welche von dem Team Orange im Rahmen des Requirements Engineering ausgearbeitet wurden. Das Team Orange setzt sich aus den Studenten Adrian Berger, Gian, Demarmels, Matthias Ossola Lars Peyer, Kevin Riesen und Yannis Schmutz zusammen.

Das Dokument richtet sich, nebst dem nichtexistenten Kunden, an System- und Software-Engineers, welche das Projekt erarbeiten sowie deren, für die Bewertung verantwortlichen, Dozenten.

Das Dokument dient als Grundlage zur weiteren Umsetzung des Projekts

1.1. Release Übersicht

Version	Datum	Autor	Beschreibung	Status
1.0.0	01.04.19	Y. Schmutz	Dokument erstellt	Entwurf
1.0.1	01.04.19	Y. Schmutz	Definition der Dokumentstruktur	Entwurf
1.2.0	01.04.19	Y. Schmutz	Einleitender Text (Allgemeines)	Entwurf
1.3.0	01.04.19	Y. Schmutz	Versionisierungskonzept	Entwurf
1.3.1	01.04.19	Y. Schmutz	Korrektur Rechtschreibefehler	Entwurf
1.4.0	04.04.19	Y. Schmutz	Einführung	Entwurf
1.5.0	04.04.19	Y. Schmutz	Systemübersicht	Entwurf
1.6.0	04.04.19	A. Berger	Anforderungen an die Entwicklungsumgebung	Entwurf
1.7.0	04.04.19	A. Berger	Testkonzept	Entwurf
1.7.1	05.04.19	L. Peyer	Anpassungen Systemarchitektur	Entwurf
1.7.2	05.04.19	L. Peyer	Abgrenzung	Entwurf
1.8.0	05.04.19	K. Riesen M. Ossola	Benutzeranforderungen	Entwurf
1.8.1	05.04.19	G. Demarmels Y. Schmutz	UseCase Diagramme Bilder	Entwurf
1.9.0	05.04.19	L. Peyer	Nichtfunktionale Anforderungen	Entwurf
1.10.0	05.04.19	L. Peyer	Domain Anforderungen	Entwurf
2.0.0	08.04.19	G. Demarmels	Review	Review

Tabelle 1: Änderungshistorie

Der Filename dieses Dokumentes soll stets die neuste Version beinhalten.

1.1.1. Versionisierungskonzept

Dieses Kapitel definiert das Format der Versionsnummer sowie deren Inkrementierung.

Das Format ist durch die Struktur X.Y.Z definiert, wobei X, Y, Z einer natürlichen Zahl entsprechen. Die Gewichtung der Ziffern zwischen den Punkten ist von links nach rechts absteigend. So wird bei jeder Änderung des Dokumentes, welche in der *Tabelle 1: Änderungshistorie* zu erfassen ist, mindestens die Zahl an der Stelle Z inkrementiert.

Die einzelnen Stellen werden in folgenden Fällen angepasst:

X: Sofern das Dokument durch das Review abgesegnet und falls nötig korrigiert wurde.

Y: Wesentliche Änderungen oder Abschlüsse von Kapiteln.

Z: Korrektur von Rechtschreibe- und Flüchtigkeitsfehlern, kleinere Inhaltliche Anpassungen an bestehenden Inhalten.

Sofern die Positionen X oder Y inkrementiert werden, starten die rechts davon liegenden wieder bei 0.

1.2. Einführung

Auf Wunsch der schweizerischen Gesundheitsbehörde ist eine Lösung zu erarbeiten, welche den Umgang zwischen Ärzten und Patienten sowie anderen involvierten Berufsgruppen vereinfacht.

Zurzeit kämpfen etliche psychiatrischen Kliniken und Spitälern mit uneinheitlichen Kommunikationskanälen sowie unstrukturierten Prozessen. Probleme, die bereits intern schwerwiegende organisatorische Mehraufwände verursachen, sind im institutionsübergreifenden Umfeld kaum zu bewältigen.

Darum wurde das Team Orange damit beauftragt, ein Patienten Management System zu entwerfen, welches nicht nur die interne Organisation der Anstalt, sondern auch betriebsübergreifende Prozesse strukturiert und vereinheitlicht.

Die entstehende Lösung ermöglicht einen systematischen Austausch zwischen Ärzten und Berufskollegen, die Automatisierung und Effizienzsteigerung administrativer Tätigkeiten sowie eine vereinfachte Betreuung von Patienten.

1.3. Referenzen

ID	Quelle
[1]	Die UML-Kurzreferenz 2.5 (6. Auflage) ISBN 978-3-486-74909-0
[2]	Interview mit Beat Stucki (Leiter Informatik bei LUPS)
[3]	Inhalte aus der Vorlesung Software Engineering and Design (BTI7081) https://www.cpvrlab.ti.bfh.ch/bachelor/wm/BTI7081/soed/
[4]	https://en.wikipedia.org/wiki/Atomic_commit
[5]	https://github.com/angular/angular/blob/master/CONTRIBUTING.md#type

Tabelle 2: Referenzen

Auf Referenzen wird wie folgt verwiesen: (**REF**: [x])

1.4. Abkürzungen und Begriffe

LUPS	*Luzerner Psychiatrie*
PMS	*Patienten Management System*
SoED	*Software Engineering and Design*
WAN	*Wide Area Network*

2. Systemübersicht

Die zu entwickelnde Lösung soll die Interoperabilität diverser unterschiedlicher Institutionen im Gesundheitswesen gewährleisten. Dies erfordert möglichst universelle und permanente Zugriffsmöglichkeiten auf das Patienten Management System auch ausserhalb der Organisation. Die Lösung wird mit sensitiven Patientendate in Kontakt kommen und soll entsprechend geschützt und gesichert werden. Für das Hosting der Applikation steht ein internes Rechenzentrum zur Verfügung.

Dementsprechend gilt es die Systemarchitektur zu designen.

Abbildung 1: Systemübersicht

2.1. Systemarchitektur

Das PMS wird durch eine webbasierte Client-Server-Architektur realisiert. Somit entfällt für den Kunden die Notwendigkeit der Anschaffung innerbetrieblicher Infrastruktur. Für die Benutzung des Systems erfordert es lediglich einen Computer mit Zugang zum LAN. Damit die Lösung auch von externen Standorten erreichbar ist, werden Zugänge zum WAN geöffnet. Dabei gilt es die nötigen Sicherheitsstandards zu beachten.

Die, für den Betrieb des PMS notwendige Infrastruktur wird von Team Orange durch ein eigenes Datencenter aufgebaut. Das Datencenter befindet sich zwingend in der Schweiz und untersteht eHealth Swiss und den Schweizer Datenschutzrichtlinien. Dies stellt mehrere Webserver zur Verfügung, welchen den universellen Zugriff auf das System ermöglichen. Die Webserver werden alle 15min repliziert und bieten so, auch bei Serverausfällen, einen zuverlässigen Zugriff auf die Daten. Damit bei erhöhter Last die Applikation nicht schwächelt, sind den Webservern ein Loadbalancer vorgesetzt. Dieser verteilt die einkommenden Anfragen auf die Webserver.

Daten werden, wo nötig, persistent auf einem Datenbankserver abgelegt und durch ein kontinuierliches Backup gesichert. Der Server wird analog zu den Webservern auch jede 15min repliziert und bietet so auch eine maximale Ausfallsicherheit. Jeglicher Datenaustausch zwischen den Clients und dem PMS wird verschlüsselt übertragen. Zugriffe auf die Applikation von externen Standorten wird zudem über ein VPN Zugang gesichert.

Alle Server befinden sich hinter einen Firewall und es werden nur die absolut nötigsten Ports geöffnet um eine maximale Sicherheit zu bieten. Hier soll dennoch ein Kompromiss zwischen Usability und Sicherheit eingegangen werden. Die Applikation soll den Benutzern das Leben nicht unnötig erschweren.

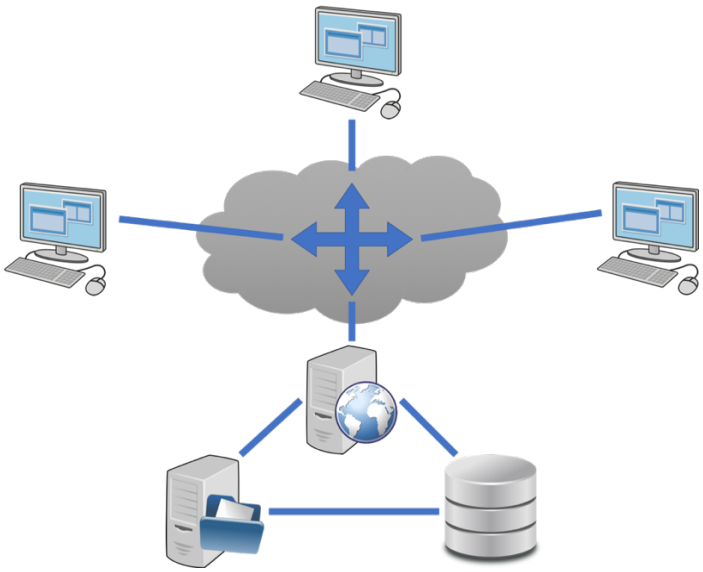


Abbildung 2: Systemarchitektur

2.2. Systemevolution

Da davon auszugehen ist, dass das PMS nach einer erfolgreichen Einführung innert Kürze landesweit (oder später gar europa-/weltweit) eingesetzt werden wird, muss die Architektur dementsprechend modular aufgebaut werden.

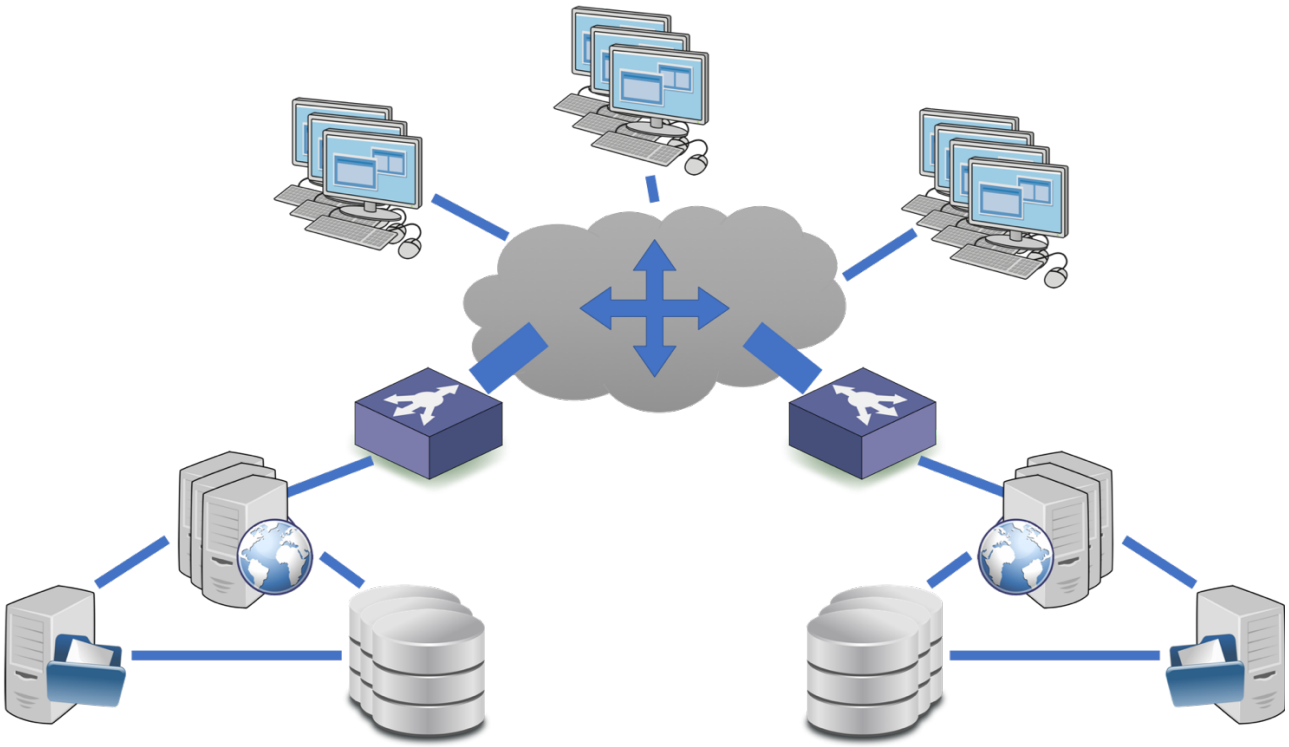


Abbildung 3: Erweiterte Systemarchitektur

Um eine permanente Verfügbarkeit zu gewährleisten, wird das PMS dezentral in mehreren Datencentern aufgebaut. Jedes dieser Center verfügt über einen Load-Balancer, welche den Datenverkehr auf verschiedene Server verteilt. Die Webserver wie auch die Datenbanken sind redundant aufgebaut. Persistente Daten werden kontinuierlich gesichert und über das ganze PM-System konsistent gehalten.

3. Zielbestimmung (Benutzeranforderungen)

Die wichtigsten 6 Anforderungen sind als Use cases definiert. Diese sind in die Kategorien Muss, Soll und Kann unterteilt:

Muss: Unverzichtbar

Soll: Wichtig, aber bei zu hohen Kosten / fehlender Zeit verzichtbar

Kann: Schön zu haben, aber nicht essenziell

3.1. Muss-Ziele

3.1.1. Ziele vereinbaren/planen

Es soll möglich sein, mit einem Patienten Ziele zu erfassen.

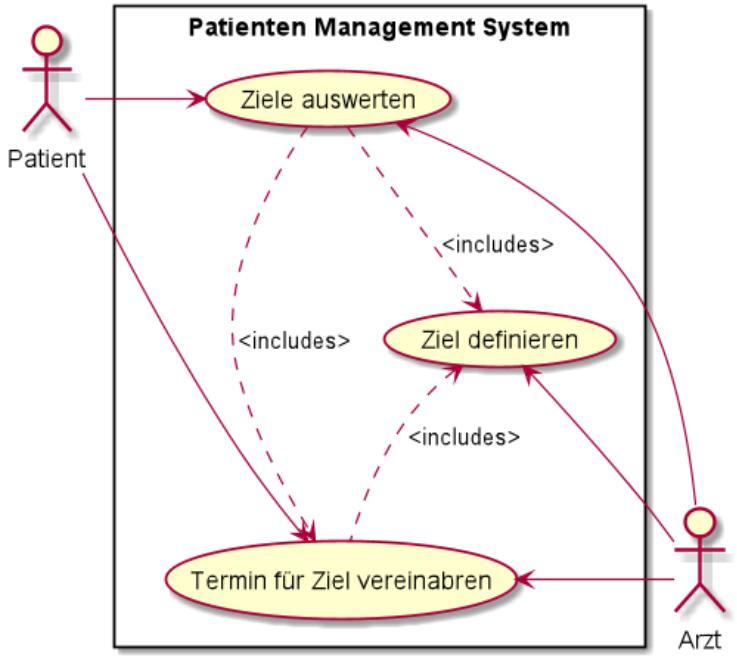


Abbildung 4: Usecase Ziele vereinbaren/planen

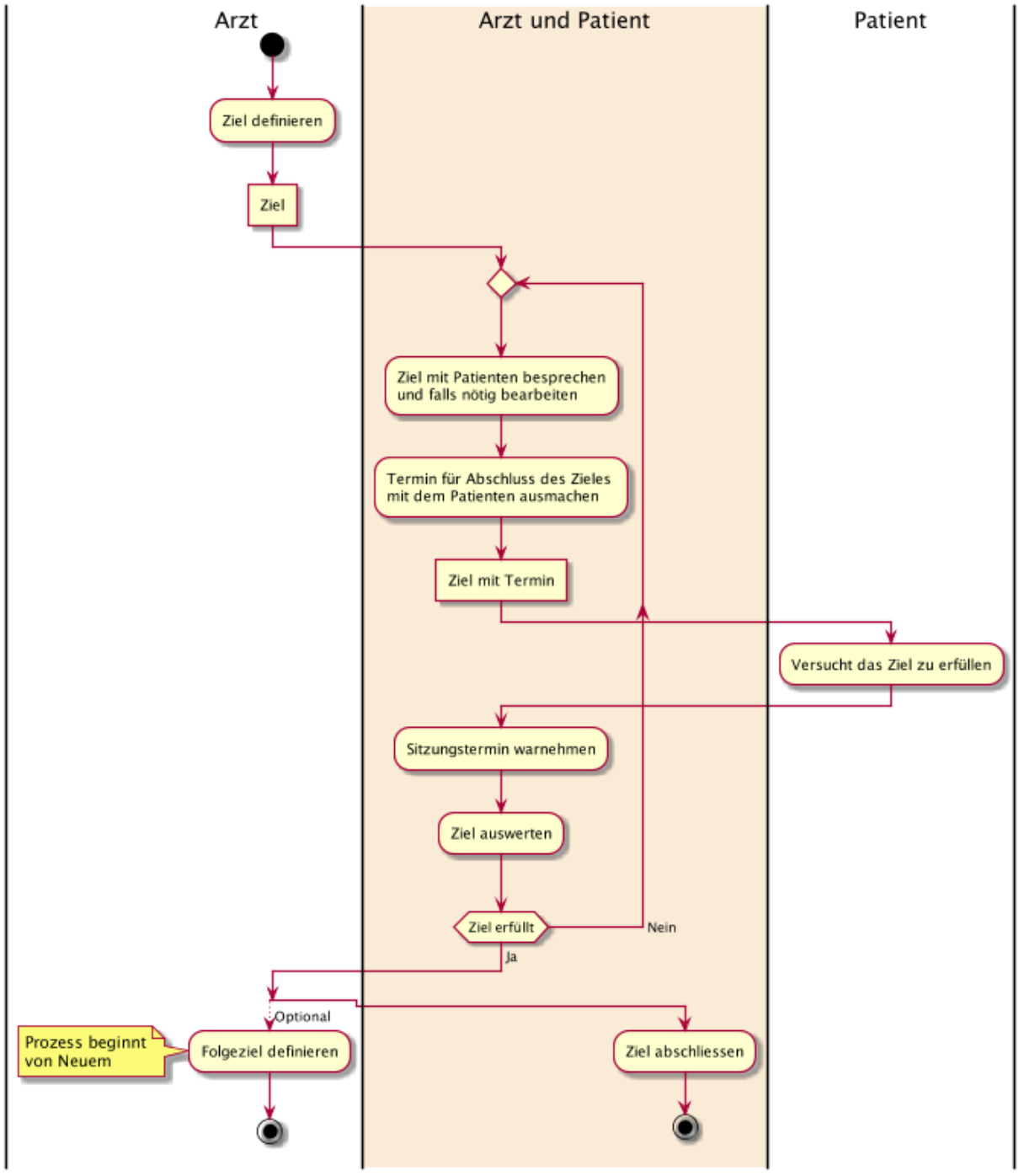


Abbildung 5: Activity Ziele vereinbaren/planen

3.1.2. Wegen Unklarheiten rückfragen

Es soll möglich sein, bei Unklarheiten zu einem Patienten eine Rückfrage zu erstellen.

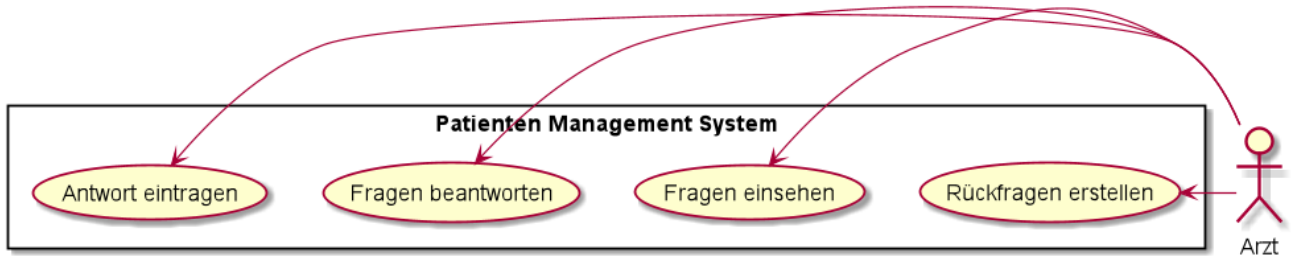


Abbildung 6: Usecase Unklarheiten

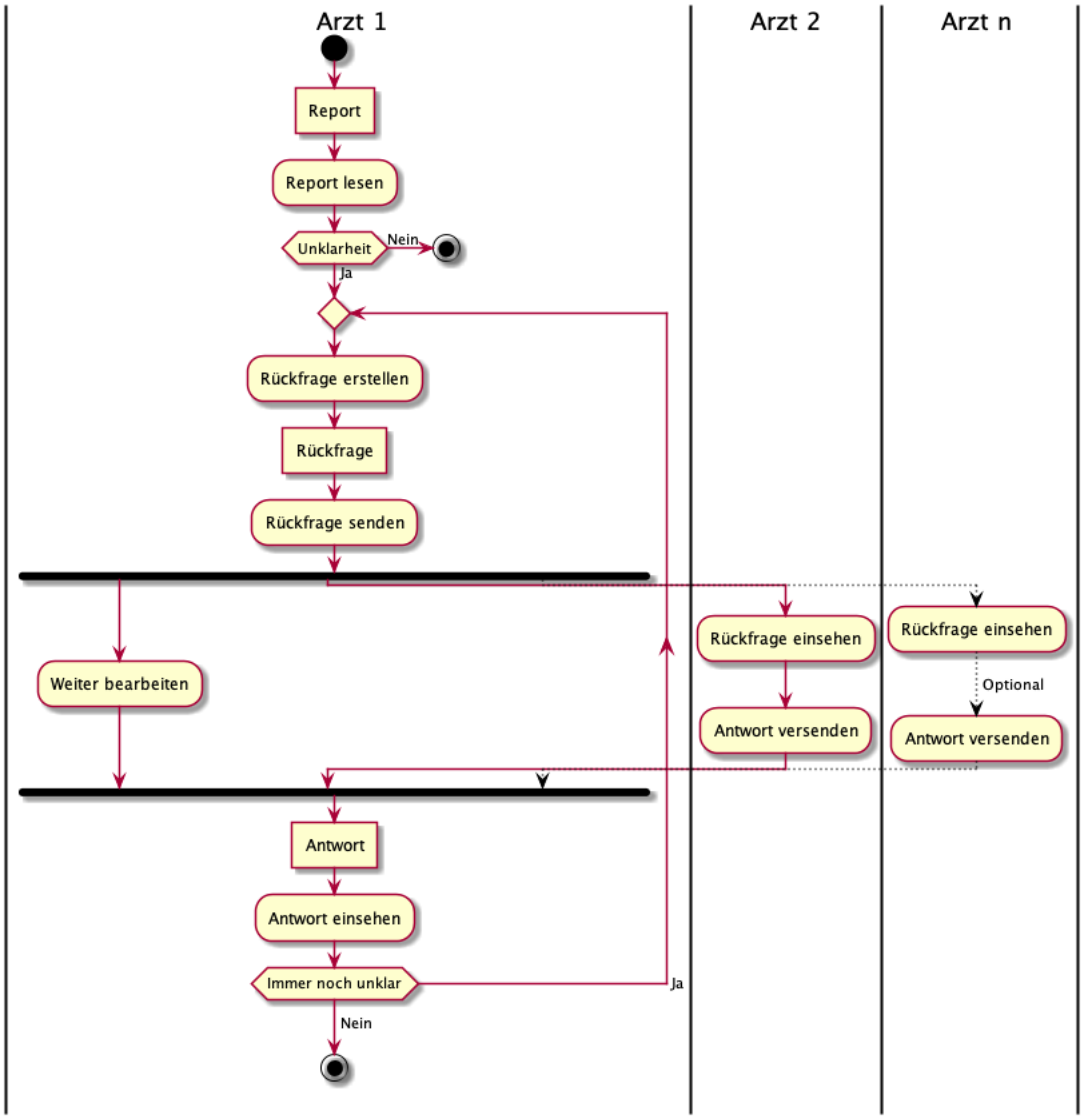


Abbildung 7: Activity Unklarheiten

3.2. Soll-Ziele

3.2.1. Fortschritt dem Patienten zeigen

Es soll möglich sein, einen Report zu generieren und diesen dem Patienten zu zeigen.

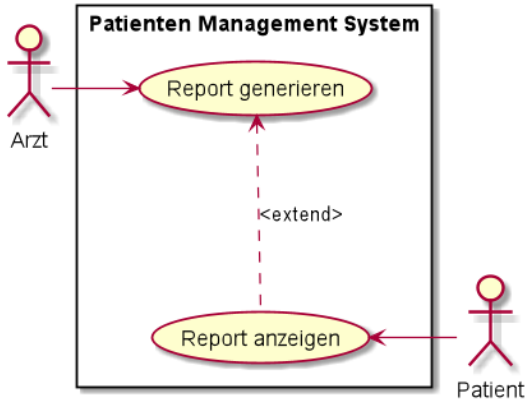


Abbildung 8: Usecase Patientenfortschritt

3.2.2. Sitzungstermin vereinbaren

Es soll möglich sein, einen Termin zwischen Arzt und Patient zu definieren.

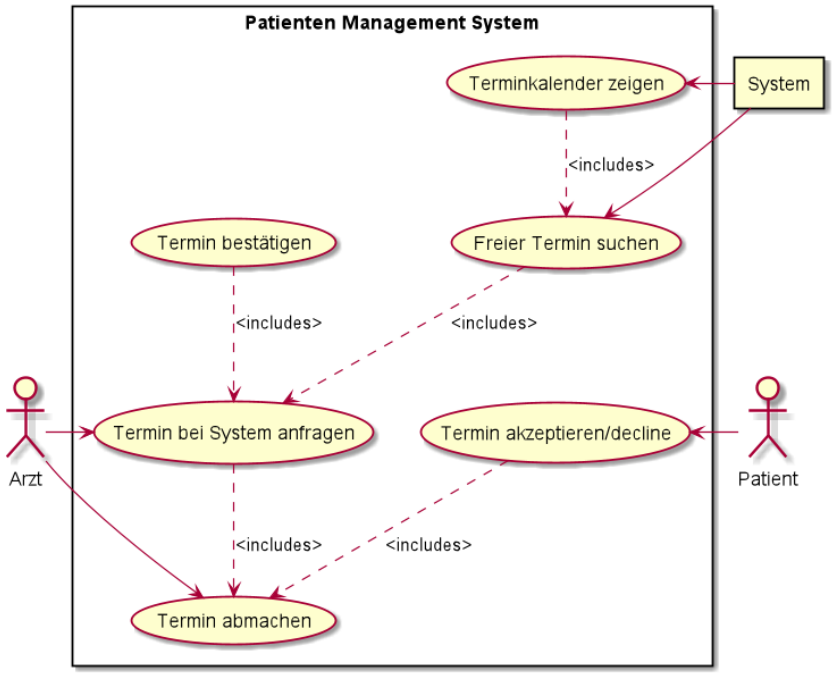


Abbildung 9: Usecase Sitzungstermin

3.3. Kann-Ziele

3.3.1. Sitzungsgespräch aufnehmen und in Report umwandeln

Es soll möglich sein, ein aufgenommenes Gespräch zwischen Arzt und Patient im System zu erfassen.

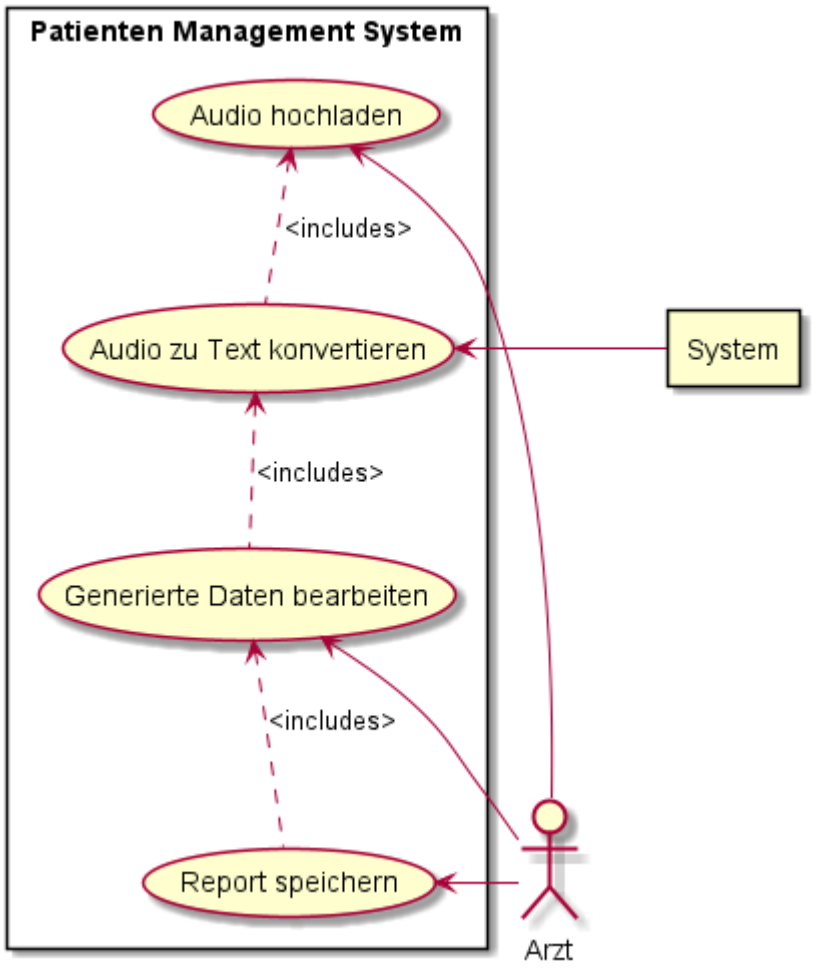


Abbildung 10: Usecase Gespräch aufnehmen

3.3.2. Handgeschriebenen Patientenreport importieren

Es soll möglich sein, einen von Hand geschriebenen Report im System zu erfassen.

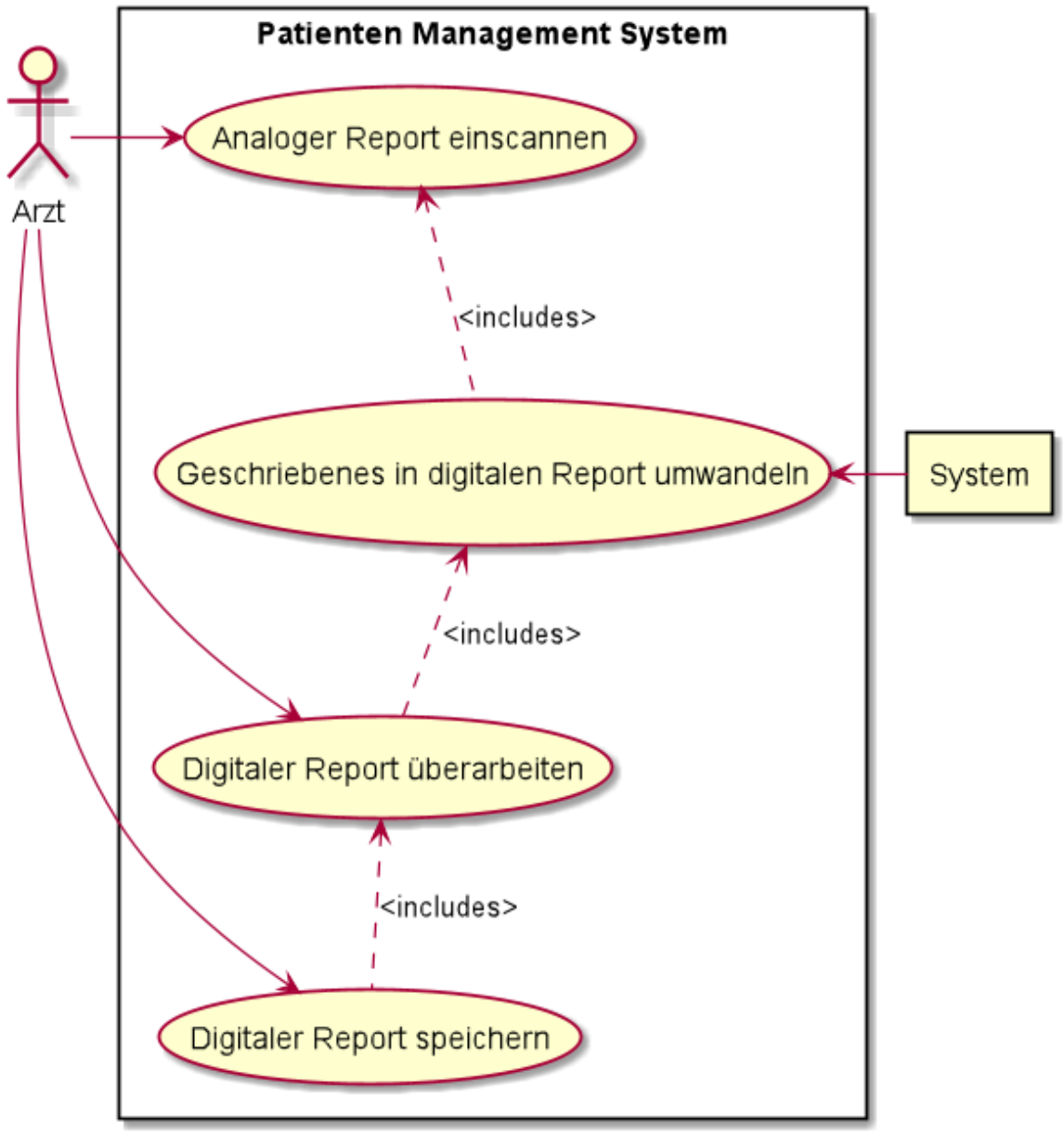


Abbildung 11: Usecase Geschriebenes digitalisieren

3.4. Abgrenzungen

Da es den Rahmen der Projekt Aufgabe sprengen würde, die ganze PMS Applikation umzusetzen, wird eine klare Abgrenzung definiert. Team Orange fokussiert sich bei der Umsetzung auf die Interaktion zwischen süchtigen Patienten und Fachärzten. Dabei geht es hauptsächlich um den Arzte Besuch von Patienten und die Definition von Zielen und Festhalten von Abmachungen. Standard-Funktionen wie Login, Profil, Berechtigungen, usw. werden in der Projektieret ausgeschlossen und nicht umgesetzt.

4. Spezifikation der Systemanforderungen

Die Spezifikation der Systemanforderungen ist in funktionale und nichtfunktionale Anforderungen unterteilt.

4.1. Funktionale Anforderungen

Die Muss Ziele der Benutzeranforderungen werden hier noch genauer spezifiziert.

4.1.1. Ziele vereinbaren/planen

Name	Ziele vereinbaren/planen
Beschreibung	Der Arzt definiert Ziele, deren Erfüllung er anschliessend mit dem Patienten plant. Zum definierten Zeitpunkt wird die Erfüllung ausgewertet und das Ziel wird entweder erneut in Angriff genommen oder abgeschlossen.
Akteure	Arzt, Patient
Ergebnis(se)	Erfüllung eines definierten Behandlungszieles
Eingehende Daten	Behandlungsstatus des Patienten, Sitzungstermine
Quellen der Daten	Applikation, Kalender, Patient
Erzeugte Date	Behandlungsziel
Vorbedingungen	Der Patient muss beim Arzt in Behandlung sein, der Arzt muss Kenntnis über den Behandlungsstatus haben
Nachbedingung	Der Patient hat das definierte Ziel erfüllt.
Essenzieller Ablauf	Siehe Activity-Diagramm unter Abbildung 5: Activity Ziele vereinbaren/planen
Offene Punkte	-

Tabelle 3: Detailbeschrieb Ziele vereinbaren

4.1.2. Wegen Unklarheiten rückfragen

Name	Wegen Unklarheiten rückfragen
Beschreibung	Bei Unklarheiten zu einem Patienten kann ein Arzt eine Rückfrage an die anderen Ärzte, die diesen Patienten behandelt haben, erstellen. Diese Rückfrage kann von diesen Ärzten beantwortet werden.
Akteure	Ärzte
Ergebnis(se)	Beantwortung einer Unklarheit/Frage
Eingehende Daten	Patientenreport
Quellen der Daten	Applikation
Erzeugte Date	Antwort durch Arzt
Vorbedingungen	Ein Arzt hat eine Frage zu einem erfassten Report
Nachbedingung	Dem Arzt wurde seine Frage beantwortet
Essenzieller Ablauf	Siehe Activity-Diagramm unter Abbildung 7: Activity Unklarheiten
Offene Punkte	Es bleibt noch auszuwerten, ob wir einen Chat, Messageboard oder beides implementieren.

Tabelle 4: Detailbeschrieb Unklarheiten

4.2. Nichtfunktionale Anforderungen

4.2.1. Produktanforderungen

Damit die Software auch für behinderte Personen bedienbar ist, sollten die gängigsten Barrierefreiheiten beachtet werden. Ein Muss sind Screenreader Unterstützung und die Navigation mittels Tastatur.

Die Ladezeiten der einzelnen Seiten sollten so optimiert werden, damit ein flüssiges Arbeiten ermöglicht wird. Gib es Seiten bei der die Ladezeit länger dauern solle, muss ein Ladeindikator eingeblendet werden, damit der User sieht, dass Daten geladen werden und die Software sich nicht aufgehängt hat.

Unsere Organisation ist an den Authentifizierungs-Dienst von Imprivata angebunden. Damit sich alle User mit ihren bereits bestehenden Login-Daten an der Software anmelden können, muss die Software eine Schnittstelle zu Imprivata bieten.

4.2.2. Organisationsanforderungen

Wenn möglich sollte die Software auf unseren eigenen Servern in Betrieb genommen werden können. Sollte dies aus technischen Gründen nicht möglich sein, wird ein Hoster in der Schweiz bevorzugt. Werden Hoster aus dem Ausland beigezogen, müssen der Gerichtstand sowie der Authentifizierungs-Schlüssel zwingen in der Schweiz liegen.

Folgendes Backupszenario muss vorhanden sein, damit die Ausfallsicherheit unserer Organisation gewährleistet werden kann:

- Alle 15 Minuten ein Snapshot des Applikationsservers
- Nächtliches Backup
- Wöchentliches Backup: Jeweils am Montag
- Monatliches Backup: Jeweils am ersten Tag des Monats
- Jährliches Backup: Jeweils am Ende des Jahres

Die Server müssen mittels USV (Unterbrechnungsfreie Stromversorgung) gespeist werden und Minimum 3 Minuten per Batterie betrieben werden können. Danach kann das Dieselagregate übernehmen.

Die Software sollte auch ausserhalb der Organisation erreichbar sein, damit Mitarbeiter auch von externen Standorten auf die nötigen Daten zugreifen können.

Alle Software, welche nach dem 01.01.2019 in Betrieb genommen wird, muss die ScreenShot Funktionalität unterbinden. Dies ist Teil unserer Datenschutzrichtlinie.

4.2.3. Externe Anforderungen

Die Software muss eHealth Swiss konform sein und ganz klar das Datenschutzgesetz der Schweiz befolgen. Ethische Aspekte sind in der Behandlung von Patienten sehr wichtig und dürfen keinen Falls verletzt werden. Organisation intern gibt es hier keine speziellen Richtlinien.

4.3. Domain Anforderungen

Die meisten der Domain Anforderungen sind zugleich auch nicht-funktionale Anforderungen und durch den vorangegangenen Abschnitt abgedeckt. Da es sich bei diesem RFQ um eine öffentliche Ausschreibung handelt, müssen zwingend alle Entwicklungspartner sowie Sub-Contracters nach ISO zertifiziert sein.

5. Anforderungen an die Entwicklungsumgebung

5.1. Betriebssystem und Entwicklungsumgebung

Beim Betriebssystem haben wir uns entschieden, dass jeder die freie Wahl hat. Einzige Voraussetzung ist, dass die Entwicklungsumgebung auf dem verwendeten System selbständig installiert werden kann.

Für die Wahl der IDE haben wir entschieden, eine Vorgabe zu definieren. Damit verfolgen wir das Ziel, dass wir uns im Problemfall gegenseitig helfen können. Zudem verlieren wir keine unnötige Zeit aufgrund unterschiedlichen Funktionsweisen unterschiedlicher IDEs.

Wir verwenden hierbei die folgenden IDEs und Tools:

- Intellij für die Programmierung mit Java
- MySQL Workbench für die Arbeit mit der Datenbank
- GIT mittels GitHub als Versionierungssystem

5.2. Versionierung

Versioniert werden die Dateien mittels GIT. Das Online-Repo wird auf GitHub geführt.

5.2.1. Commit Regeln

Commits sollten grundsätzlich so atomar (**REF**: [4]) wie möglich sein. Zudem sollten alle Änderungen, welche publiziert werden immer doppelt geprüft werden. Sobald die Änderung also fertig ist, sollte erneut geprüft werden, ob alle Anpassungen korrekt sind.

Für die commit Medungen orientieren wir uns an den Angular Contibuting Guidelines (**REF:** [5])

Folgendes Muster muss dabei eingehalten werden: <value>: <message> (<reference>)

Als <value> sind die folgenden Optionen wählbar:

- build: Änderungen die das build System oder externe Abhängigkeiten betreffen
- ci: Änderungen, die die continuous Integration betreffen
- docs: Änderungen an der Dokumentation
- feat: Generell für neue Features
- fix: Generell für Bugfixes
- perf: Änderungen die die Performance des Systems betreffen
- refactor: Änderungen die weder einen Bugfix, noch ein neues Feature hinzufügen
- style: Kleine Änderungen, welche den Sinn des Codes nicht ändern (Formatierungen, Leerschläge)
- test: Änderungen die das Testing betreffen

Die <message> sollte mit einem Imperativ beginnen und die grundlegende Änderung beschrieben.

Als <reference> dient das Präfix des Branches (siehe Abschnitt 5.2.2Branching Regeln)

5.2.2. Branching Regeln

- Der Hauptbranch ist wie üblich der master
- Als Entwicklungsbranch wird der Branch develop verwendet
- Für jedes Feature wird ein neuer Branch erstellt
    - Branchname: [Prefix]-[Kurzbeschrieb]
        - Prefix: task[Nummer der Tasks]
            - Bsp: task2-patient-use-case
        - Für generelle Tasks, wird der Prefix general verwendet
            - Bsp: general-reorganize-folder-structure
- Auf den master, sowie den develop werden können keine Commits erstellt werden
- Pull Request auf den master erfolgen immer vom Branch develop aus
- Alle Features werden per Pull Request in den develop Branch gemerged
- Damit ein Pull-Request germeged werden kann, benötigt dieser mindestens das Review einer Person. Dabei dürfen sämtliche Mitglieder des Teams sämtliche Pull-Requests reviewen und akzeptieren.

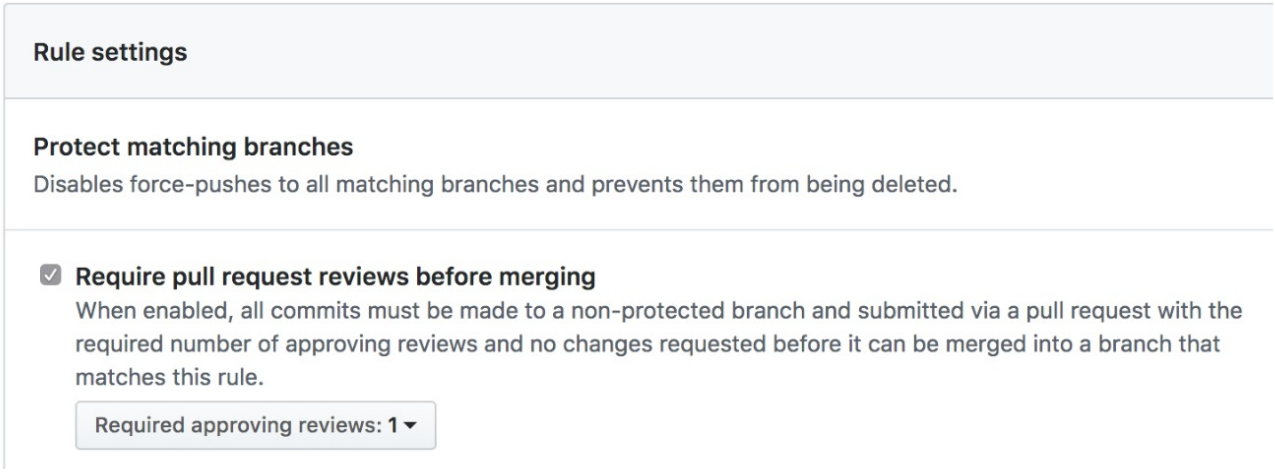


Abbildung 12: GIT Regeleinstellungen

Diese Regeln gelten ab dem nächsten Task.

6. Testkonzept

6.1. Testziele

Ziel des Testens ist es, die Funktionalität des Codes sicherzustellen, indem Fehler gefunden und behoben werden können.

6.2. Testobjekte

Nr.	Objekt	Beschreibung
1	GUI	Die Benutzeroberfläche soll den Anforderungen entsprechen
2	Code	Die Funktionalität des Codes soll geprüft werden

6.3. Testarten

Nr.	Testart	Beschreibung
1	Manuelle Tests	Der Tester führt auf Basis von Testfällen verschiedenen Szenarien aus und vergleicht das Resultat mit dem erwarteten Ergebnis.
2	jUnit Test	Automatisierte Test, welcher der Tester in der Konsole seines Computers ausführen kann. Diese prüfen ebenfalls bereits bestehende Funktionalitäten.

6.4. Testrahmen

6.4.1. Testvoraussetzungen

- Das Testkonzept muss in fertiger Form vorliegen, damit alle Testfälle definiert sind
- Die Software muss vom Entwickler für die entsprechenden Tests freigegeben werden

6.4.2. Fehlerklassen

Festgestellte Fehler werden in Klassen 0-4 eingestuft

Nr.	Fehlerklassen	
0	*Fehlerfrei*	Das erhaltene Resultat stimmt eindeutig mit dem erwarteten Resultat überein.
1	*Unwesentlicher Mangel*	Es existiert ein kleiner Fehler, welcher jedoch keinen Einfluss auf die Verwendung der Software hat. Meist sind dies optische Ungenauigkeiten. Der Fehler sollte notiert werden, muss jedoch nicht zwingend behoben werden.
2	*Leichter Mangel*	Die Software hat einen kleineren Fehler, der jedoch für die Produktivität des Produktes kein Problem darstellt. Der Fehler sollte jedoch notiert werden, damit er später behoben werden kann.
3	*Schwerer Mangel*	Die Software kann nicht produktiv verwendet werden. Sicherheitsmängel fallen mindestens in diese Fehlerklasse. Eine sofortige Behebung ist erforderlich.
4	*Kritischer Mangel*	Die Software ist nicht bedienbar und kann so nicht genutzt werden. Eine sofortige Behebung ist erforderlich.

6.5. Start- und Abbruchbedingungen

Nr.	Testart	Bedingungen
1	Manuelle Tests	Der Entwickler hat dem Tester den Testauftrag erteilt und die Unit Tests wurden erfolgreich absolviert. Wird ein Fehler der Fehlerklasse 3 oder 4 festgestellt, muss dies unverzüglich dem Entwickler mitgeteilt werden und die Tests werden für den Moment abgebrochen.
2	jUnit Test	Diese Tests werden durchgeführt, sobald der Entwickler das Gefühl hat, die Anforderungen seien erfüllt.

6.6. Testinfrastruktur

Die Tests werden vom Tester lokal auf seiner Installation geprüft. Sollte im späteren Projektverlauf festgestellt werden, dass eine Online Installation der Applikation notwendig ist, könnten die Tests auf die Online Version verlagert werden.

6.7. Testfallbeschreibungen

ID	T-001
Beschreibung	Der Arzt kann ein Ziel erfassen.
Testvoraussetzung	-
Testschritte	Der Arzt meldet sich am System an und navigiert zu der Zielübersicht. Der Arzt klickt auf Ziel erfassen und füllt das Formular aus Der Arzt sendet das Formular ab
Erwartetes Ergebnis	Ziel wurde erfolgreich erfasst und persistiert.

ID	T-002
Beschreibung	Der Arzt kann ein Ziel bearbeiten.
Testvoraussetzung	Das zu bearbeitende Ziel muss existieren.
Testschritte	Der Arzt meldet sich am System an und navigiert zu der Zielübersicht. Der Arzt navigiert zum zu bearbeitenden Ziel und klickt auf Ziel bearbeiten Der Arzt bearbeitet das Formular und sendetes.
Erwartetes Ergebnis	Das Ziel wurde erfolgreich bearbeitet und die Änderungen persistiert.

ID	T-003
Beschreibung	Der Arzt kann ein Ziel auswerten (ansehen und auswerten).
Testvoraussetzung	Das Ziel muss bereits existieren
Testschritte	Der Arzt meldet sich am System an und navigiert zu der Zielübersicht. Der Arzt navigiert zum zu auszuwertenden Ziel. Der Arzt wertet Übersicht aus.
Erwartetes Ergebnis	Arzt sieht eine Übersicht des Ziels mit allen relevanten Infos.

ID	T-004
Beschreibung	Der Arzt kann eine Rückfrage stellen.
Testvoraussetzung	Rapport muss existieren
Testschritte	Der Arzt navigiert zum Rapport. Der Arzt klickt auf Rückfrage stellen Der Arzt erfasst seine Rückfrage und sendet diese ab.
Erwartetes Ergebnis	Rückfrage wird am Rapport angehängt und angezeigt.

ID	T-005
Beschreibung	Der Arzt kann eine Rückfrage bantworten.
Testvoraussetzung	Rapport mit Rückfrage muss existieren
Testschritte	Der Arzt navigiert zum Rapport. Der Arzt klickt auf Rückfrage beantworten Der Arzt erfasst seine Antwort und sendet diese ab.
Erwartetes Ergebnis	Rückfrage wird am Rapport angehängt und angezeigt.

ID	T-006
Beschreibung	Der Arzt kann die Antwort einer Rückfrage einsehen.
Testvoraussetzung	Rapport mit Rückfrage und Antwort muss existieren.
Testschritte	Der Arzt navigiert zum Rapport. Der Arzt sieht im Bereich die Rückfragen & Antworten.
Erwartetes Ergebnis	Die Rückfragen und Antworten sind ersichtlich.

6.8. Testplan

Sobald eine Funktionalität vom Entwickler freigegeben wird, werden sämtliche Tests durchgeführt.

6.9. Abnahmetest

Sobald sämtliche Tests des Testkonzeptes erfolgreich durchgeführt wurden, gilt der Abnahmetest als erfüllt.

7. Anhang

7.1. Interview Beat Stucki (LUPS)

Um die nicht-funktionalen und Domain Anforderungen besser definieren zu können, haben wir Beat Stucki (Leiter Informatik der Psychiatrie Luzern) befragt. Es handelt sich dabei um Gespräch-Notizen:

• Generell soll die Software WAN tauglich sein (Terminalserver)

• Barrierefreiheit: nicht festgelegt - wir schauen drauf - Datenschutz -

Kein Screenshot (Im Smartphone Bereich)

• Seitenladezeiten Vorgabe: Gibt es nicht - normales arbeiten soll ermöglicht werden

• SSO Anbindung: Haben sie nicht. Möchten sie gerne. Imprevita ist Med. Standard

• Hosting: Inhouse wenn Patientendaten beinhaltet (Mame, geb.Dat., Konfession). Sonst nicht zwingend. Patiententests können in der Cloud laufen. Wenn nicht Inhouse dann meistens in der Schweiz. Schlüssel in CH und Gerichtsstand in CH wenn Hoster im Ausland.

4

• Backup: Alle 15min Snapshot / 1x Nachtbackup / Wöchentlich / Monatlich / Jährlich (Kein definierter Standard bei LUPS)

• Ausfallsicherheit: 2 RZ / 180 Server / 6 Hosts. 3 davon können ohne Problem ausfallen (Leistungseinbusse). Kernapplikationen müssen ohne Internet funktionieren. Hat ein Notfallzenario - Kerapp exportiert alle 3h ein PDF der Patientendok. Verteilt diese PDF’s auf dedizierte Notebooks (gezipped). Küche hat Notvorrat für 3 Tage :). 2 Wochen USV Power (Dieselmotoren). Erste 3 Min. über Batterie.

• Regelwerke: eHealth Swiss muss beachtet werden. Datenschutzgesetz.

• Ethische Aspekte: Normale Ethische Aspekte - Darf nicht gegen Gesetze

verstossen

• ISO/IEEE: Ist nicht relevant da Datenhoheit bei LUPS und keine Fremddaten gehalten werden. Nur wenn Datacenter ausgelagert würde dann ISO270001 / BSI7799

• Hersteller zertifiziert: Wenn eine Ausschreibung denn ja.