
# Green.

## Software Requirement Specification

| | |
|---|---|
| Schule: | Berner Fachhochschule (BFH) |
| Departement: | Technik und Informatik (TI) |
| Modul: | Software Engineering and Design (BTI7081) |
| Typ: | Semesterarbeit |
| Autoren: | Kevin Amalathas, Yannis Biasutti, Markus Joder, Alexandre Moeri, José Rindlisbacher |
| Betreuer: | Prof. Urs Künzler und Prof. Dr. Jürgen Vogel |
| Ort und Datum: | Bern, den 14. April 2020 |

# Inhaltsverzeichnis

# 1 Einleitung

## 1.1 Zweck

Dieses Software Requirement Specification Dokument beschreibt die, im Rahmen des Software Engineering and Design (BTI7081) Moduls, durch das Team Green, zu entwickelnde Software Applikation.
Darin werden insbesondere die, in der Design Thinking Phase eruierten, Anforderungen der Nutzer an das System erläutert und in funktionale und nicht funktionale Anforderungen ausformuliert.
Desweiteren bietet das Dokument eine Grundlage zur Sicherstellung der Qualität und die dafür anzuwendenden Methoden.

### 1.1.1 Leserschaft

Dieses Dokument bildet die Grundlage für ein gemeinsames Verständnis der Ziellösung zwischen den Auftraggebern, Prof. Urs Künzler und Prof. Dr. Jürgen Vogel, und Team Green.

### 1.1.2 Versionierung

| Version | Datum | Autor | Beschreibung | Status |
|---|---|---|---|---|
| 0.1 | 01.04.2020 | moera1 | Dokument erstellt | Draft |
| 0.2 | 01.04.2020 | moera1 | Definition der Dokumentstruktur | Draft |
| 0.3 | 09.04.2020 | amalk1 | Use-Case Diagramme erstellt | Draft |
| 0.4 | 09.04.2020 | amalk1 | Nicht-Funktionale Anforderungen | Draft |
| 0.5 | 09.04.2020 | jodem1 | Use-Case Diagramme dokumentiert | Draft |
| 0.6 | 10.04.2020 | amalk1 | Funktionale Anforderungen | Draft |
| 0.7 | 12.04.2020 | jodem1 | Benutzer- und Systemanforderungen | Draft |
| 0.8 | 13.04.2020 | biasy1 | Qualitätssicherung dokumentiert | Draft |
| 0.9 | 13.04.2020 | biasy1 | Schnittstellen dokumentiert | Draft |
| 1.0 | 14.04.2020 | moera1 | Kap. 1, 3.3, 5 | Version 1 |

**Tabelle 1:** Versionierung

## 1.2 Umfang

Das Ziel des Projekts ist es einen Teil einer Applikation zur Patientenverwaltung im Gesundheitswesen zu entwickeln. Der Fokus dabei liegt auf die Betreuung von depressiven Patienten durch Pfleger am Wohnort des Patienten (Spitex).
Die Applikation soll den Pflegern, durch die digitalisierung von Prozessen und neuen Funktionalitäten, bei Ihrer täglichen Arbeit mit den Patienten dienen.

### 1.2.1 Anwendungsbereich (Scope)

Folgende Anwendungsfälle gehören zum Scope des Projekts:

- Einsehen von zugewiesenen Patienten
- Einsehen von geplanten Einsätzen
- Einsehen, erstellen, modifizieren und verwalten von Besuchsprotokollen
- Nutzung einer Wissensbasis zum Wissenstransfer zwischen den Pflegern

### 1.2.2 Ausschlüsse (Out of scope)

Folgende Themengebiete werden aus dem Scope des Projekts ausgeschlossen:

- Patientenverwaltung
- Terminverwaltung
- Ressourcenverwaltung/planung
- Verwaltung des Leistungskatalogs
- Aktivitäten der Pfleger ohne direkten Zusammenhang zu Patientenbesuchen
- Beschaffung der Patienten-, Termin-, und Leistungs-Daten (aus Umsystemen)
- Authentifizierung und Autorisierung innerhalb der Applikation

## 1.4 Glossar

| Begriff | Erklärung |
|---|---|
| Spitex | Die Bezeichnung Spitex, Abkürzung für «spitalexterne Hilfe und Pflege», ist eine im deutschschweizerischen Sprachraum verwendete allgemeine Bezeichnung für die Hilfe und Pflege zu Hause. Es entspricht dem Begriff ambulante Pflege in Deutschland.[1] |
| Smoke Tests | Smoke testing wird von den Entwicklern vor einem ersten Release durchgeführt, oder auch von Testern, bevor sie einen Build für weitere Tests akzeptieren. [2] |
| Regressionstests | Unter einem Regressionstest versteht man in der Softwaretechnik die Wiederholung von Testfällen, um sicherzustellen, dass Modifikationen in bereits getesteten Teilen der Software keine neuen Fehler („Regressionen“) verursachen.[3] |
| Exploratives testen | Exploratives Testen ist ein Ansatz zum Testen von Software, der als gleichzeitiges Lernen, Testdesign und Testausführung beschrieben wird.[4] |

**Tabelle 2:** Glossar

[1] Quelle: siehe Q1 in 5.3 Quellenverzeichnis
[2] Quelle: siehe Q2 in 5.3 Quellenverzeichnis
[3] Quelle: siehe Q3 in 5.3 Quellenverzeichnis
[4] Quelle: siehe Q4 in 5.3 Quellenverzeichnis

# 2 Benutzer- und Systemanforderungen

## 2.1 Annahmen

Die Personendaten der Klienten lassen sich von einem Umsystem beziehen und werden nicht in der hier beschriebenen Anwendung verwaltet.

Die vorliegende Applikation unterstützt die Pfleger der Spitex bei ihren Klienten vor Ort. Da bei der Spitex das Backoffice für die Terminvereinbarung und -verwaltung zuständig ist, werden auch diese Daten aus einem bestehenden Umsystem bezogen.

Da es sich beim fiktiven Auftraggeber um eine regionale Gesundheitsorganisation handelt, wird Deutsch als Standardsprache behandelt und auf eine internationalisierung der Applikation verzichtet.

Da eine webbasierte Applikation vom Auftraggeber vorgeschrieben wird, gehen wir in vorliegendem Dokument davon aus, dass sämtliche Benutzer bei Verwendung dieser Applikation über Zugang zu einem internetfähigen Gerät mit stabiler Funkverbindung verfügen. Dadurch wird die Unterstützung von Offline-Funktionalitäten hinfällig.

## 2.2 Funktionale Anforderungen

| ID | Name | Beschreibung | Use-Case Nr. |
|---|---|---|---|
| FA1 | Anwendungsmeldungen | Das System soll für die Entwickler eine Protokollierung besitzen, um die Fehlersuche eines Problems ausfindig zu machen. Die Protokollierungsebenen sollen gemäss Standard eingeführt werden. | |
| FA3 | Benachrichtigungen | Die Mitarbeiter sollen wichtige Information wie anstehende Besuche per Benachrichtigungen erhalten, um jederzeit auf dem aktuellen Stand zu sein. | |

| | | | |
|---|---|---|---|
| FA4 | Berechtigungssystem | Anhand des Berechtigungskonzeptes soll ein Berechtigungssystem eingeführt werden, damit die Benutzer nur die nötigen Informationen einsehen können. Es gibt mindestens die Rollen “Benutzer” und “Administrator” (Out of Scope) | |
| FA5 | FAQ | Das System soll eine Funktion besitzen, um bei Bedarf möglichst einfach Hilfe zu einer bestimmten Problemstellung zu erhalten. Es handelt sich dabei um eine FAQ-Seite, auf welcher erfahrene Pfleger ihr Wissen zusammentragen und damit an unerfahrene Pfleger weitergeben können. | |
| FA6 | Hilfefunktion | Das System soll über eine Hilfefunktion verfügen, damit ein Benutzer möglichst schnell Unterstützung (bspw. telefonische Unterstützung durch einen erfahrenen Pfleger) beim Backoffice anfordern kann. | 3 |
| FA7 | Profil | Jeder Pfleger besitzt ein Profil, welches je nach Berechtigung verwaltbar ist. | |
| FA8 | Transaktionalität | Die Erfassung soll transaktional durchgeführt werden können. Die Erfassung ist erst abgeschlossen, sobald diese vollständig beendet wurde. Dadurch können Protokolle bereits vor einem anstehenden Termin vorerfasst, und falls anschliessend nicht mehr benötigt, wieder verworfen werden. | |
| FA9 | Authentifizierung | Bevor ein Pfleger mit der Applikation interagieren kann, muss er sich mit einem Login-Verfahren authentifizieren. (Out of Scope) | |

| | | | |
|---|---|---|---|
| FA10 | Protokollierung | Ein Pfleger kann auf einem Klienten Protokolle erfassen. Es gibt verschiedene Protokolltypen. Diese entstehen entweder aus einem Termin (Besuchsprotokoll) oder einem Telefongespräch (Telefonprotokoll). Zusätzlich gibt es die Möglichkeit Notiz-Einträge (“Gedächtnisprotokolle”) zu erfassen, welche Unabhängig von einem Ereignis erstellbar sind.<br>Mit einem Protokoll können vordefinierte Leistungen verknüpft werden (Pflege, Medikation, Vitalwerte). Zusätzlich kann eine Notiz (Freitext) erfasst werden. | 1 |
| FA11 | Protokolle verwalten | Protokolle können jederzeit beliebig editiert werden. Das Löschen von erfassten Protokollen ist jedoch nur mit der Rolle “Administrator” möglich. | 2 |
| FA12 | Protokoll-Suchfunktion | Der Benutzer kann die Protokoll-Historie seiner Klienten nach Stichwörtern oder vordefinierten “Tags” durchsuchen und filtern. | 4 |
| FA13 | Klienten-Profil | Für jeden Klienten gibt es ein Profil mit folgenden Informationen:<br>● Personendaten des Klienten<br>● Alle Pfleger welche für diesen Klienten im Einsatz sind/waren<br>● Neuigkeiten mit den neuesten Protokoll-Einträgen auf dem Klienten<br>● Protokoll-Historie mit allen jemals erfassten Einträgen<br>● Terminübersicht | |
| FA14 | Klienten-Übersicht | Ein Pfleger kann die Liste aller Klienten durchsuchen.<br>Standardmässig wird diese Liste nach Klienten gefiltert, welche durch den Benutzer betreut werden. | |

| | | | |
|---|---|---|---|
| FA15 | Kalender | Im Kalender soll der Benutzer eine Übersicht über alle vergangenen und zukünftig geplanten Aktivitäten erhalten. | |

**Tabelle 3:** Funktionale Anforderungen

### 2.2.1 Use Case 1: Protokoll erstellen

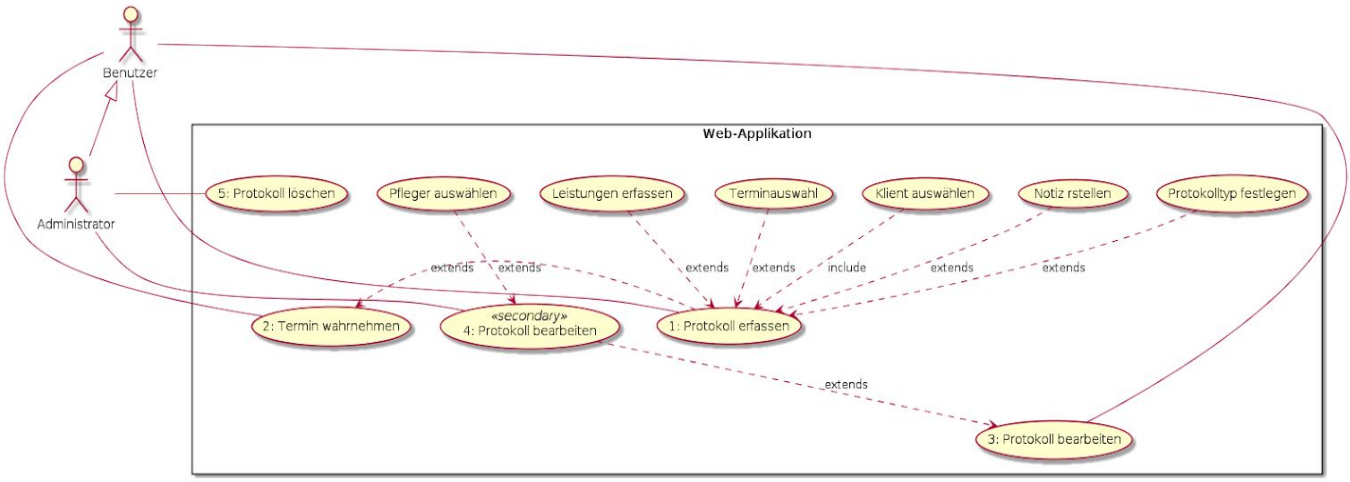


**Abbildung 1:** Use Case - Protokoll erstellen

| | |
|---|---|
| **Nr. und Name:** | 1: Protokoll erstellen |
| **Szenario:** | Kontakt mit Klienten soll protokolliert werden |
| **Kurzbeschreibung:** | Der Pfleger erstellt ein Protokoll |
| **Beteiligt Akteure:** | Normalbenutzer (Pfleger) |
| **Auslöser / Vorbedingung:** | Kontakt mit Klienten fordern die Erfassung von Leistungen, Auswahl Klient getroffen |
| **Ergebnisse / Nachbedingung:** | Das erstellte Protokoll ist für alle Benutzer einsehbar |

**Ablauf**

| Nr. | Wer | Was |
|---|---|---|
| 1.0 | Benutzer | Eröffnet ein neues Protokoll und wählt den Typ |
| 1.1 | System | Erstellt das Protokoll auf dem System |

| | | |
|---|---|---|
| 1.2 | System | Legt den Klienten fest |
| 1.3 | System | Hinterlegt den Pfleger |
| 1.4 | Benutzer | Option: Termin erstellen |
| 1.5 | Benutzer | Wählt einen Termin |
| 1.6 | System | Liefert die Auswahl der Verfügbaren Leistungen |
| 1.7 | Benutzer | Wählt Leistung aus |
| 1.8 | Benutzer | Erfasst Leistung |
| 1.9 | System | Speichert Leistung |
| 1.10 | Benutzer | Option: Weitere Leistung |
| 1.11 | Benutzer | Bestätigt Leistungen |
| 1.12 | System | Fragt ob Benutzer Notiz erstellen will |
| 1.13 | Benutzer | Option: Erstellt Notiz |
| 1.14 | System | Speichert und gibt Feedback |
| 1.15 | System | Zeigt Klienten mit neuem Protokoll in den Neuigkeiten |

**Ausnahmen, Varianten**

| Nr. | Wer | Was |
|---|---|---|
| | | |

### 2.2.2 Use Case 2: Protokolle bearbeiten

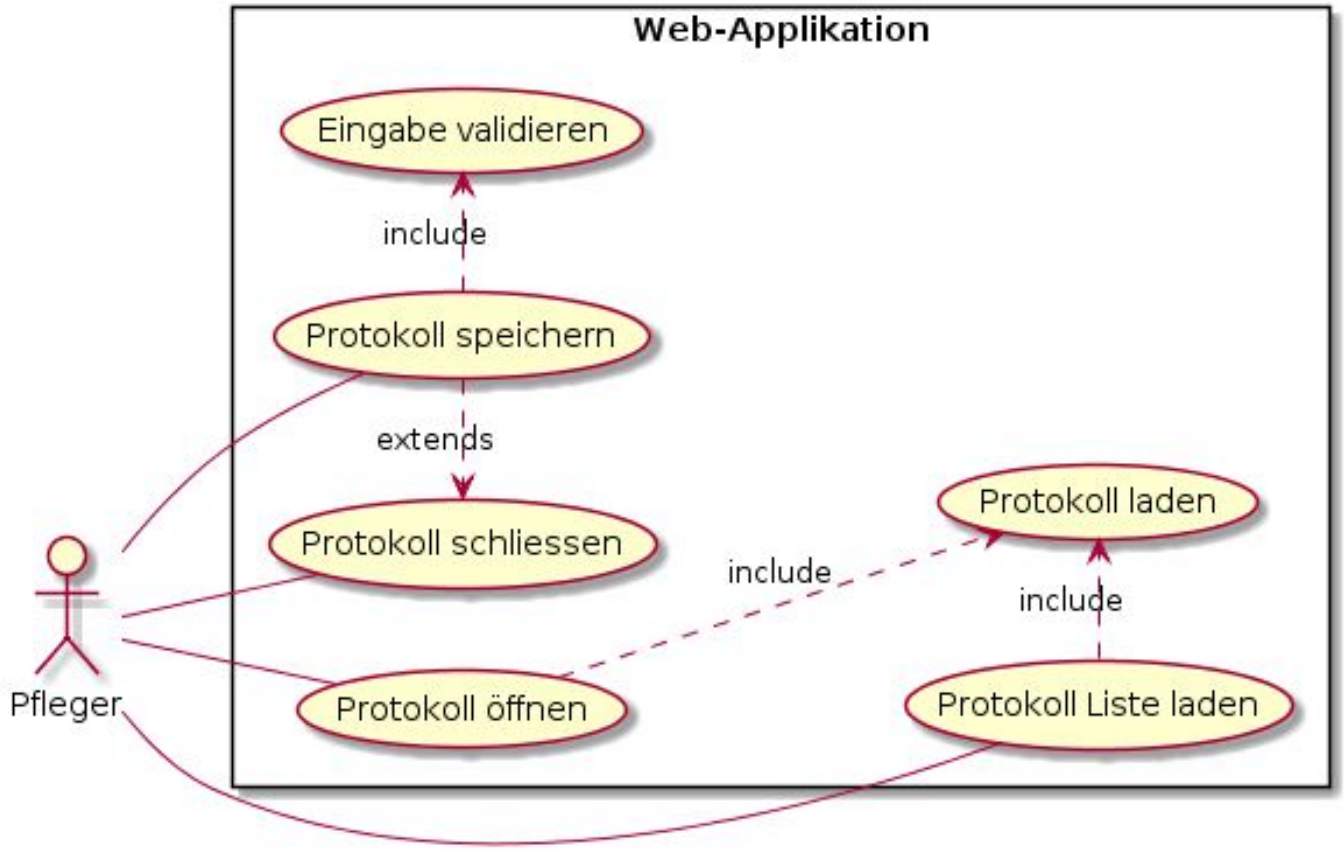


**Abbildung 2:** Use Case - Protokolle bearbeiten

### 2.2.3 Use Case 3: Hilfe anfordern

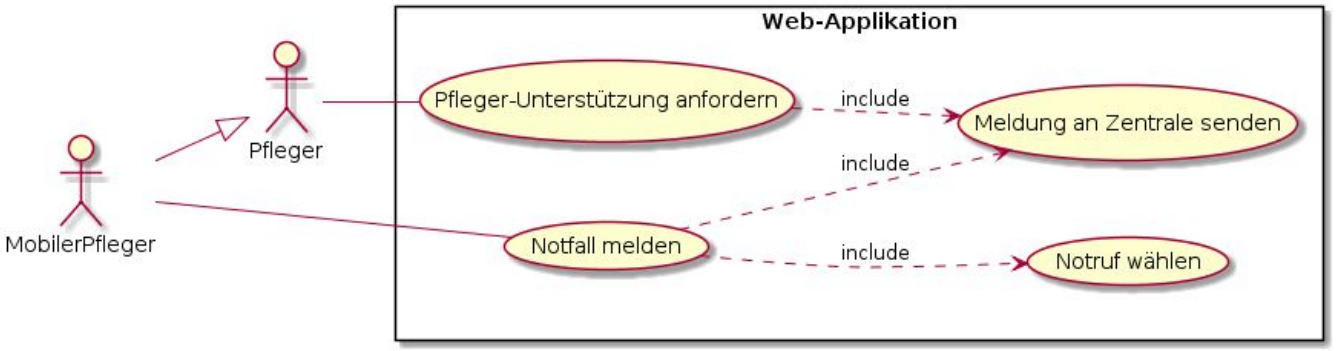


**Abbildung 3:** Use Case - Hilfe anfordern

#### 2.2.3.1 Abgrenzung

Im Rahmen des Moduls BTI7081 werden wir auf den Akteur MobilerPfleger verzichten und nur die Weblösung implementieren.

### 2.2.4 Use Case 4: Protokoll-Historie durchsuchen

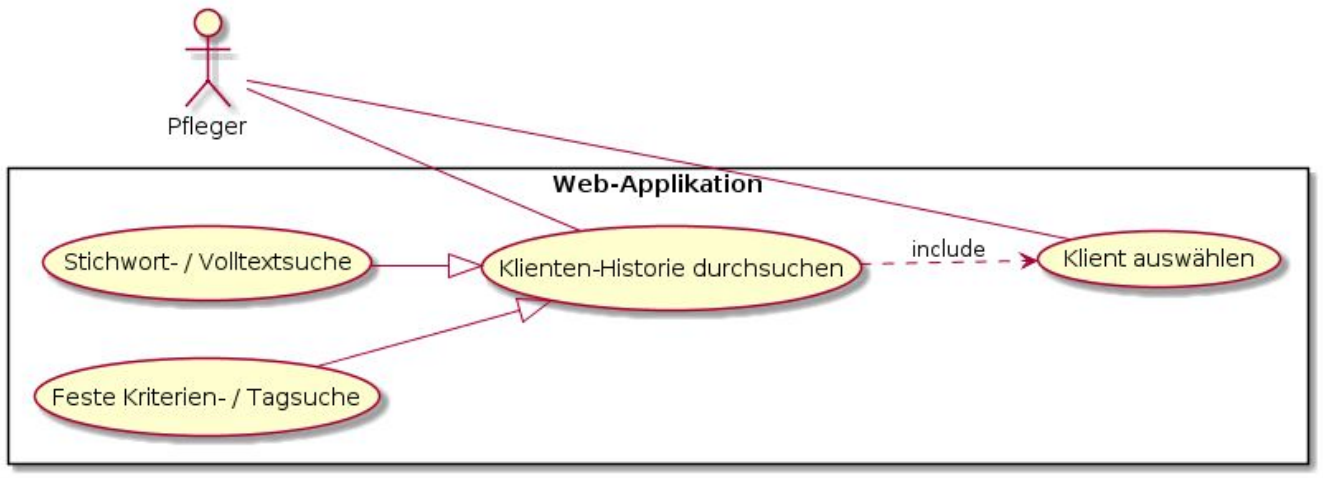


**Abbildung 4:** Use Case - Protokoll Historie

| | |
|---|---|
| **Nr. und Name:** | 4: Protokoll-Historie durchsuchen |
| **Szenario:** | Ein Pfleger möchte vor seinem Besuch die früheren Protokolle des nächsten Klienten durchsuchen, um sich auf den anstehenden Besuch bestmöglich vorbereiten zu können. |
| **Kurzbeschreibung:** | Pfleger durchsucht die Protokoll-Historie eines Klienten |
| **Beteiligt Akteure:** | Pfleger (Benutzer) |
| **Auslöser / Vorbedingung:** | Protokoll-Historie ist geladen |
| **Ergebnisse / Nachbedingung:** | Gewünschtes Suchergebnis wird angezeigt |

**Ablauf**

| Nr. | Wer | Was |
|---|---|---|
| 1.1 | Benutzer | Löst Suche für den ausgewählten Klienten aus |
| 1.2 | System | **Varianten:** Durchsucht Protokolle des Klienten |

| | | |
|---|---|---|
| 1.3 | System | **Ausnahme:** Keine Ergebnisse gefunden |
| 1.4 | System | Resultat wird für den Benutzer aufbereitet |
| 1.5 | Benutzer | Benutzer kann das Resultat in einer übersichtlichen Ansicht kontrollieren/verifizieren |

**Ausnahmen, Varianten**

| Nr. | Wer | Was |
|---|---|---|
| 6.2 | System | **Variante a)** Suche nach Stichworten |
| 6.2.1 | System | Durchsucht Protokolle (Volltext) in der Datenbank nach Übereinstimmungen |
| 6.2 | System | **Variante b)** Suche nach vordefiniertem Kriterium |
| 6.2.1 | System | Durchsucht vordefinierte Protokoll-Attribute in der Datenbank nach Übereinstimmungen |
| 6.3 | System | **Ausnahme** Keine Ergebnisse gefunden |
| 6.3.1 | System | Generiert entsprechende Fehlermeldung für den Nutzer |
| 6.3.2 | Benutzer | Kann den Suchvorgang mit neuem Suchbegriff erneut starten |

## 2.3 Nicht-Funktionale Anforderungen

| ID | Name | Beschreibung |
|---|---|---|
| NFA1 | Backup | Ein tägliches Backup sollte gewährleistet werden. Für eine optimale Sicherung, sollte ein Backup Konzept erstellt werden. |
| NFA2 | Benutzerfreundlichkeit | Die Bedienung sollte für Mitarbeiter in allen Altersgruppen und ohne Informatik Hintergrund bedienbar sein. Die einzelnen Bedienelemente sollen deshalb mit einem hohen Wiedererkennungswert gestaltet werden. |
| NFA3 | Berechtigungskonzept | Es soll ein Berechtigungskonzept existieren, um das System für verschiedene Nutzer |

| | | |
|---|---|---|
| | | einzuschränken. (Out of Scope) |
| NFA4 | Datenaufbewahrung | Die Daten sollten gemäss der Aufbewahrungspflicht hinterlegt werden. |
| NFA5 | Datensicherheitsrichtlinien | Da es sich um medizinische Daten handelt, sollte das System alle Datenschutzrichtlinien einhalten. Alle Standards, die durch die GDPR (General Data Protection Regulation) definiert wurden, sollen eingehalten werden. |
| NFA6 | Dokumentation | Nebst dem Code-Repository soll eine Benutzerdokumentation existieren. |
| NFA7 | Sicherheit | Das System soll soweit wie möglich sicherstellen, dass unberechtigte Zugriffe und Manipulationen von Aussen untersagt sind. Es existiert ein Authentifizierung- und Berechtigungssystem um dies zu gewährleisten. (Out of Scope) |
| NFA8 | Skalierbarkeit | Das System soll fähig sein bei wachsenden Nutzerzahlen zu skalieren, da das initiale System nur einen kleinen Kreis von Benutzern beinhaltet, jedoch mit der Zeit auf grössere Benutzergruppen ausgeweitet werden könnte. |
| NFA9 | Stabilität | Das System muss bei Verwendung stabil sein und zuverlässig funktionieren. |
| NFA10 | Verfügbarkeit | Da Spitex Mitarbeiter das System ohne Unterbrechung nutzen müssen muss eine hohe Verfügbarkeit gewährleistet werden. |
| NFA11 | Wartbarkeit | Um das System aktuell zu halten, ohne die Arbeit der Mitarbeiter zu unterbrechen, müssen Wartungsfenster frühzeitig angekündigt werden. |
| NFA12 | Responsive Design | Die Applikation wird als mobile Browser-Lösung realisiert, damit die Pfleger bei den Klienten vor Ort auf das System zugreifen können. Das Layout soll so flexibel gestaltet werden, dass dieses auf dem Desktop, Tablet und Smartphone eine gleichbleibende Benutzererfahrung gewährleistet. |

**Tabelle 4:** Nicht-funktionale Anforderungen

## 2.4 Schnittstellen

Unsere Applikation wird in ein Umfeld gepflanzt welches bereits diverse andere elektronischen Systeme besitzt. Um eine nahtlose Integration zu gewährleisten werden in diesem Kapitel die internen und externen Schnittstellen definiert.

### 2.4.1 Externe Schnittstellen

#### 2.4.1.1 Benutzer

Die Klienten und Pfleger werden bereits in einem anderen System verwaltet und gewartet um eine Duplizierung dieser Daten zu verhindern wird das BenutzerSystem an unsere Applikation angebunden. Das BenutzerSystem stellt hierfür eine Rest Schnittstelle zur Verfügung. Benutzer können geladen, verändert aber nicht gelöscht werden.

#### 2.4.1.2 Leistungen

Die Leistungen sind von den Versicherungen vorgegeben und werden von einem zentralen System verwaltet. Leistungen können von diesem System über eine Rest Schnittstelle geladen werden. Veränderungen können nur im Quellsystem vorgenommen werden.

### 2.4.2 Interne Schnittstellen

In der Applikation werden hauptsächlich Daten miteinander verknüpft. Diese Verknüpfung muss persistiert werden können. Daher wird eine es eine Schnittstelle zu einer Datenbank geben.

# 3 Modellierung

## 3.1 Systemarchitektur

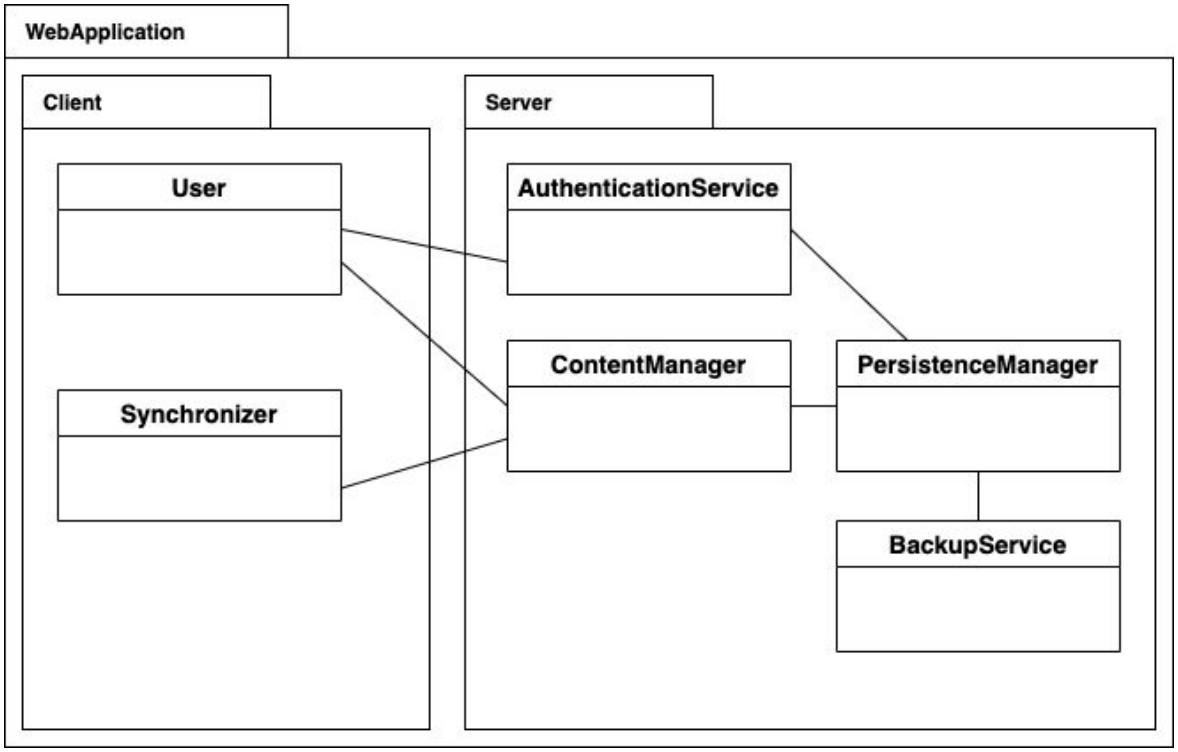


**Abbildung 5**: Systemarchitektur

Interne Systemkomponenten unter Berücksichtigung einer möglichen Client-Server bzw. Front-, Backend Implementation.

Der **User** ist hier stellvertretend für alle Benutzerinteraktionen über das UI am Frontend. Während der **Synchronizer** jegliche Funktionalitäten bietet, welche dafür sorgen, dass Inhalte stets aktualisiert werden und gegebenenfalls bei Netzwerkunterbrechungen keine Informationen verloren gehen. Dazu arbeitet er eng mit dem serverseitigem **ContentManager** zusammen, welcher wiederum durch den **PersistenceManager** Informationen persistieren oder diese für den Benutzer bzw. Client zur Verfügung stellen kann.
Der **AuthenticationService** dient der Authentifizierung der Benutzer inklusive ihren Rollen und dient somit automatisch auch der Autorisierung. Weiter soll der **BackupService** sicherstellen, dass die für die Applikation und ihren Betrieb relevante Daten einem regelmässigem Backup-Plan folgen.

## 3.2 Datenmodell

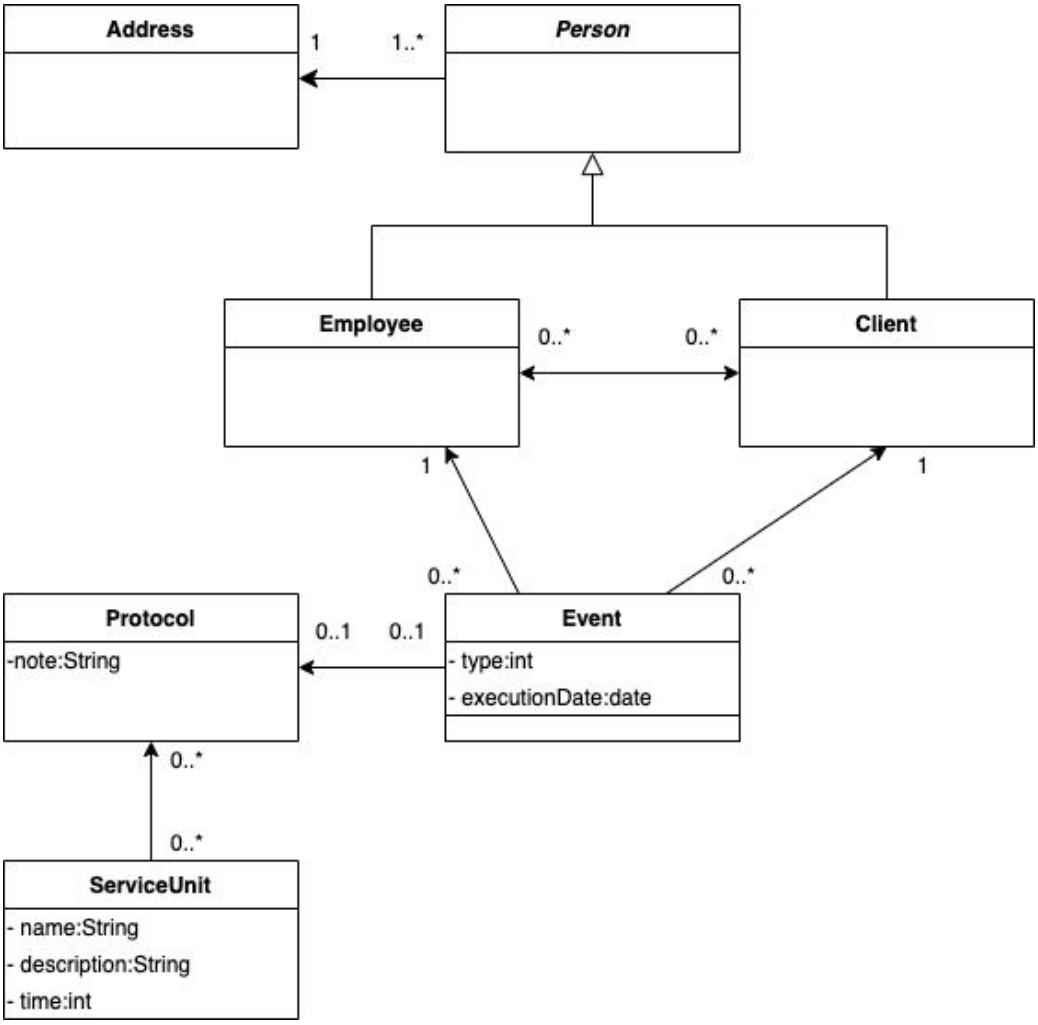


**Abbildung 6**: Datenmodell

Diese schematische Darstellung stellt vereinfacht die internen Objekte der Applikation dar und zeigt im übertragenen Sinn die logischen Zusammenhänge zwischen bekannten Entitäten wie zum Beispiel den Pfleger, den Klienten oder den Protokollen.
Das zentrale Element stellt das **Event** dar, es ist stellvertretend als Kundenkontakt zu betrachten. Es enthält Informationen über die Art des Kontakts und hält den Zeitpunkt fest. Dazu existieren relationen zum Pfleger (**Employee**), dem Klienten (**Client**) und selbsterklärend, wie selbstverständlich dem Protokoll. Dieses **Protokoll** hat die möglichkeit annotiert zu werden und stellt einen recht simplen Container für die durch den Pflegenden erbrachten Leistungen (**ServiceUnits**) dar.

## 3.3 Ansichten

### 3.3.1 Dashboard


**Abbildung 7:** Ansicht - Dashboard

Das Dashboard dient als Einstiegsansicht in die Applikation und soll dem Benutzer einen Überblick über Aktuelles geben, sowie effiziente Navigationsoptionen zur Verfügung stellen.

Zur Übersicht gehören Neuigkeiten, wie noch nicht visierte Terminänderungen oder neue Protokolle von Patienten zu denen eine Beziehung besteht, sowie die nächsten Terminen.

Die Navigation bietet einen schnellen Wechsel zur Übersicht der Haupt-Ressourcen: Termine, Patienten, Protokolle, sowie Zugriff zur Wissensbasis.

Das Dashboard ist für den Benutzer personalisiert.

### 3.3.2 Patienten-Übersicht

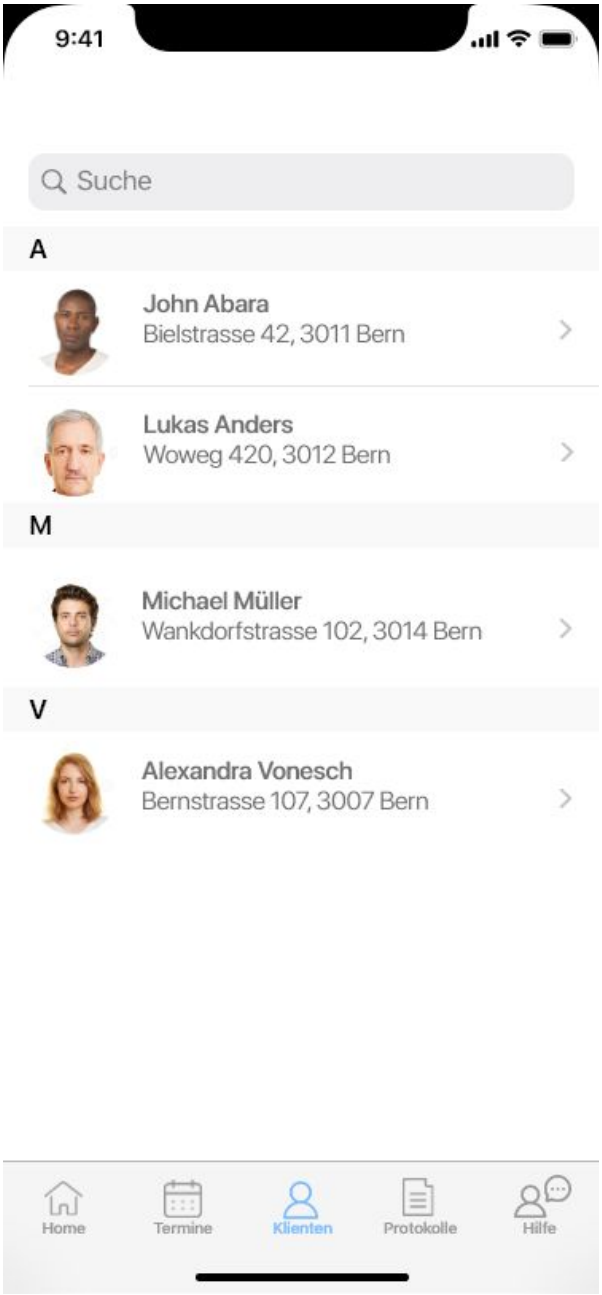


**Abbildung 8:** Ansicht - Patieten-Übersicht

Die Patientenübersicht bietet dem Benutzer (Pfleger) eine Übersicht der Patienten mit der eine Beziehung (Vergangene oder Zukünftige Termine) besteht.

Eine scrollbare, alphabetisch sortierbare Liste zeigt alle Patienten, sowie die wichtigsten Eckdaten, an.

Eine Text-Suchfunktion soll ebenfalls einfach die Möglichkeit bieten die Liste zu filtern.

### 3.3.3 Patient-Profil


**Abbildung 9:** Ansicht - Patient-Profil

Das Patienten-Profil zeigt auf einem Blick die wichtigsten Informationen und Eckdaten eines Patienten an.

Dazu gehört nebst eines Profil-Fotos um den Patienten rasch zu erkennen, den Namen und Kontaktdaten.

Ein Knopf (Telefon-Icon) soll ermöglichen den Patienten ausserhalb der Applikation telefonisch zu erreichen.

Patienten-Spezifische Neuigkeiten, sowie das letzte Protokoll und den nächsten Termin werden angezeigt.

Einen entsprechenden Link (Alle anzeigen) führt weiter zu einer Übersicht aller geplanter Termine, respektive erfassten Protokolle des Patienten.

Über einen weiteren Knopf (+) wird ermöglicht direkt ein Protokoll zum angezeigten Patienten zu erstellen.

### 3.3.4 Protokoll-Ansicht


**Abbildung 10:** Ansicht - Protokoll-Ansicht

Die Protokoll-Ansicht stellt ein Protokoll übersichtlich dar.

Dabei werden zuerst die involvierten Parteien (Pfleger und Patient) angezeigt.

Weiter werden die mit dem Protokoll in Zusammenhang gebrachten Leistungen gelistet.

Zuletzt wird Freitext, welches dem Protokoll beigefügt wurde angezeigt.

Über einen Link an gewohnter Stelle lässt sich zum Patienten-Profil (zurück-)navigieren.

### 3.3.5 Protokoll-Erstellung

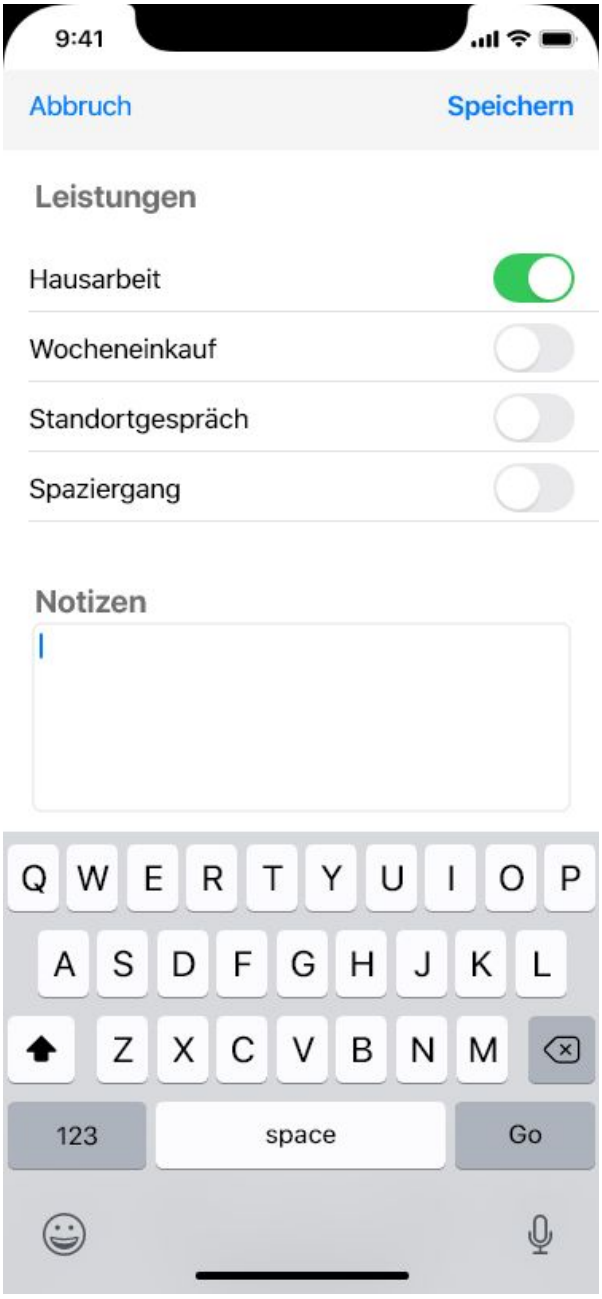


**Abbildung 11:** Ansicht - Protokoll-Erstellung

Bei der Protokollerstellung soll mittels gewohnten Interaktionsmöglichkeiten die Erfassung erleichtert werden.

Dabei werden aus dem Leistungskatalog die dem Patienten zustehenden Leistungen angezeigt, welche sich mittels Slider dem Protokoll hinzufügen lassen.

Weiter steht ein Freitext-Feld für Notizen zur Verfügung.

Das Protokoll lässt sich Speichern, welches die angegebenen Daten persistiert und anschliessend zum Patienten-Profil zurückführt oder Abbrechen, wobei die neuen Daten/Änderungen nicht persistiert werden.

Siehe Use Case 1 (Seite 10).

# 4 Qualitätsmanagement[5]

## 4.1 Qualitätsanforderungen

Die Qualität unserer Applikation ist unser höchstes Gut und wird mit einem strikt definierten Verfahren eingehalten. Die Qualitätssicherung wird auf verschiedenen Ebenen durchgeführt und mittels definierten Erwartungswerten messbar gemacht. Die Teststufen werden auf für diesen Zweck ausgelegten Umgebungen durchgeführt. (Kapitel 4.2 Umgebungen)

## 4.2 Umgebungen

Die Qualitätssicherung benötigt auf allen Teststufen eine Umgebung welche sich in einem definierten und reproduzierbaren Zustand befindet. Aus diesem Grund wird in diesem Kapitel definiert wann welche Umgebung gepatcht wird und welche Tests auf diesen ausgeführt werden.

### 4.2.1 Entwicklung

Fortlaufend wird auf dieser Umgebung die neuste Version vorhanden sein. Diese Umgebung richtet sich hauptsächlich an Entwickler. Hier werden automatisierte Tests, konkret Unit Tests, ausgeführt. (Kapitel 4.3.1 Komponententests)

### 4.2.2 Test

Die Testumgebung wird nach jeder Iteration mit dem neu erarbeiteten Arbeitspaket bespielt, vorausgesetzt alle Tests auf der Entwicklungsumgebung sind erfolgreich verlaufen. Ziel dieser Umgebung ist, dass alle funktionalen Anforderungen geprüft werden können ohne Abhängigkeit von Umsystemen. (Kapitel 4.3.3 Systemtests)

### 4.2.3 Integration

Die Integrationsumgebung widerspiegelt, so weit als möglich, die produktive Umgebung wie sie auch vom Kunden verwendet wird. Ziel ist es auch nicht funktionale Anforderungen zu testen und ggf. Lasttests durchzuführen. Diese Umgebung wird in die Systemlandschaft integriert und von Umsystemen angebunden, daher muss diese Umgebung hochverfügbar sein. Deployments finden punktuell vor produktiven Releases

[5] Quellen: Q5 in 5.3 Quellenverzeichnis

statt auch in Abhängigkeit der Testresultate aus der Testumgebung. (Kapitel 4.3.4 Abnahmetests)

### 4.2.4 Produktion

Verfügbarkeit und Fehlerfreiheit sind das oberste Gebot dieser Umgebung. Ausfallrisiko muss auf ein minimum reduziert werden können. Sollten alle vorhergegangenen Umgebungen eine neue Softwareversion als ausgereift befunden haben, so wird ein produktives Deployment gemacht. Dieses darf den Betrieb nicht beeinträchtigen. Nach erfolgtem Deployment werden Smoke Tests durchgeführt.

## 4.3 Vorgehen

### 4.3.1 Komponententests

Die Komponententests werden automatisiert mittels Unit Tests bei jedem Build der Applikation durchgeführt. Zu jeder Klasse besteht eine Test Klasse welche eine Code Coverage von mindestens 80% aufweist. Die Unit Tests werden auch als Teil der Regressionstests angesehen.

### 4.3.2 Integrationtests

Integrationstests sollen wo es Sinn macht automatisiert werden und in den Buildprozess eingebaut werden. Manuelle Integrationstests werden ergänzend auf der Testumgebung durchgeführt. Dies insbesondere wenn es spezielle Testdaten Konstellationen benötigt welche die Testdaten “zerstört”, bzw. in einen nicht reproduzierbaren Zustand bringt. Im Fokus steht das Testen in die Tiefe.

### 4.3.3 Systemtests

Das Testen in die Breite wird bei den Systemtests erreicht. Da es sich in diesem Fall um eine Webapplikation handelt, setzen wir das Sessionbased Testing Verfahren ein. (4.3.3.1 Sessionbased Testing) Dieses Verfahren erlaubt es einem eine weite Variabilität in die Testabdeckung zu bringen. Verschiedene Geräte und Browser können so relativ schnell abgedeckt werden und die Fehlerrate auch für weniger populäre Konfigurationen drastisch reduzieren. Hiermit werden Funktionale sowie nicht Funktionale Anforderungen geprüft.

#### 4.3.3.1 Sessionbased Testing

Bei einer Test Session testen alle Teammitglieder explorativ während einer timeboxed Session, normalerweise 20 Minuten, einen vorher definierten Teil der Applikation aus

Usersicht in einem ebenfalls vor der Session definierten Gerät/Browser. Nach der Session präsentiert jedes Teammitglied seine Findings dem Team und es wird zusammen beschlossen ob dies ein Fehler ist oder nicht. Das Dokumentieren des Fehlers liegt in der Verantwortung des jeweiligen Teammitgliedes.

### 4.3.4 Abnahmetests

Als letztes werden auf dem Integrationssystem die Abnahmetests durchgeführt. Diese sind Anwendungsfall basiert und müssen vom Kunden abgenommen werden. Der Kunde selbst führt auch Akzeptanztests aus und entscheidet anhand dieser Ergebnisse ob ein produktives Deployment gemacht wird.

## 4.4 Abgrenzung

Wir behalten uns vor im Rahmen des Moduls, Software Engineering und Design (BTI7081), Umgebungen zusammenzufassen und die Verfügbarkeit des System zu vernachlässigen. Des weiteren werden wir die Qualitätsanforderungen wie folgt umsetzen:

- Die Rolle Kunde wird durch unser privates Umfeld repräsentiert.
- Wir beschränken uns auf folgende Plattformen:
    - Desktop: Die neusten Versionen von Firefox und Google Chrome.
    - Mobile: Die vorhandenen Geräte der Teammitglieder
- Auf Lasttests wird verzichtet

# 5 Verzeichnis

## 5.1 Abbildungsverzeichnis



## 5.2 Tabellenverzeichnis

## 5.3 Quellenverzeichnis


| | | |
|---|---|---|
| **Q1** | Spitex | https://de.wikipedia.org/wiki/Spitex<br>aufgerufen am 14.04.2020 |
| **Q2** | Smoke Tests | https://de.wikipedia.org/wiki/Smoke_testing<br>aufgerufen am 14.04.2020 |
| **Q3** | Regressionstests | https://de.wikipedia.org/wiki/Regressionstest<br>aufgerufen am 14.04.2020 |
| **Q4** | Explorative testen | https://de.wikipedia.org/wiki/Exploratives_Testen<br>aufgerufen am 14.04.2020 |
| **Q5** | Kapitel Qualitätmanagement | Basiswissen Softwaretest (de)<br>Buch vom dpunkt.verlag, ISBN: 978-3-86490-024-2 |
| **Q6** | Allgemein | Foliensatz Modul BTI7081 |