Hochschule für Angewandte Wissenschaften Hamburg
*Hamburg University of Applied Sciences*

# Bachelorarbeit

**Leon Fausten**

## Konzeption und Entwicklung eines webbasierten Systems zur Krankmeldung

*Fakultät Technik und Informatik*
*Studiendepartment Informatik*

*Faculty of Engineering and Computer Science*
*Department of Computer Science*

Leon Fausten

# Konzeption und Entwicklung eines webbasierten Systems zur Krankmeldung

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

im Studiengang Bachelor of Science Angewandte Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Sarstedt
Zweitgutachter: Prof. Dr. Steffens

Eingereicht am: 28. August 2014

**Leon Fausten**

**Thema der Arbeit**
Konzeption und Entwicklung eines webbasierten Systems zur Krankmeldung

**Stichworte**
Webentwicklung, Softwareengineering, Client-Server Architektur, Rest, Symfony2, AngularJS

**Kurzzusammenfassung**
Diese Arbeit befasst sich mit der Entwicklung eines webbasierten Systems, um Krankmeldungen innerhalb eines Unternehmens durchzuführen. Ziel dieser Arbeit ist es, einen möglichen Weg der Entwicklung, von den anfänglich gegebenen Anforderungen, bis zu einem realisierbaren System, zu zeigen. Vorgestellt werden dabei sowohl Konzepte und Praktiken der Analyse, als auch die Entwicklung eines Realisierungskonzeptes, sowie dessen Umsetzung.

Teil dieser Arbeit ist die Umsetzung einer Auswahl der Anforderungen.

**Leon Fausten**

**Title of the paper**
Conception and development of a web based system for notification of sickness

**Keywords**
web development, software engineering, client-server architecture, rest, Symfony2, AngularJS

**Abstract**
This work is about the development of a web based system to handle notification of sickness. Goal of this thesis is to show the development of a feasible system, starting with the given requirements. Concepts and practices for the analysis and the development of a feasible concept and its implementation will be presented.

Part of this work is the implementation of selected requirements.

# Inhaltsverzeichnis

# Abbildungsverzeichnis

# Auflistungen

# 1 Einleitung

Die Einleitung gibt einen groben Überblick über diese Arbeit. Sie beschreibt die Beweggründe zu Anfang und das gewünschte Ziel zum Ende des Projektes.

Innerhalb der Motivation soll deutlich werden, weshalb das aktuelle Verfahren durch ein neues, softwaregestütztes Verfahren ersetzt werden soll.

Die Zielsetzung soll das gewünschte Resultat zum Ende dieser Arbeit zeigen, dies betrifft den Weg der Entwicklung, sowie die Realisierung der Anwendung.

Bei jeder Erwähnung einzelner Personengruppen innerhalb dieser Arbeit, sind sowohl alle männlichen, wie auch weiblichen Personen gemeint.

## 1.1 Motivation

Viele Firmen haben in den letzten Jahren neue *Web 2.0* Plattformen eingeführt. Dies umfasst in den meisten Fällen Plattformen zum Informationsaustausch, z.B. Wikis. Software, die es ermöglicht gemeinsam Dokumente auszutauschen und zu bearbeiten, sodass alle Mitarbeitenden immer auf dem aktuellen Stand sind, wird außerdem häufig eingeführt.[1]

Für andere alltägliche Aufgaben existieren oft keine passenden Lösungen, weil das Unternehmen eigene individuelle Abläufe besitzt, die eine Software abbilden muss. Somit müssten die Unternehmen entweder ihre Arbeitsabläufe anpassen oder sich eine für sie angepasste Lösung entwickeln lassen. Im Falle eines händischen, manuellen Vorgangs, müssen die betroffenen Mitarbeiter den Ablauf genau kennen, damit dieser vollständig abgearbeitet werden kann. Bei Krankmeldungen, einem häufig manuell ausgeführtem Vorgang mit mehreren Beteiligten, müssen bestimmte Personen umgehend unterrichtet werden. Wird dies versäumt, kann unter Umständen, nicht mehr rechtzeitig auf den Ausfall reagiert werden, z.B. mit einer Terminverschiebung.

[1] vgl. http://www.sueddeutsche.de/karriere/social-media-in-unternehmen-wie-das-web-machtstrukturen-aendert-1.1424446, Abgerufen am 23.07.2014

Eine Software, die durch den Prozess führt und auf die Korrektheit des Ablaufes achtet, reduziert diese Problematik. Der Benutzer wird innerhalb der Software automatisch durch den Vorgang geführt, deshalb muss ihm dieser nicht mehr genau bekannt sein.

Zur Verdeutlichung der Problematik, wird die Vorgehensweise bei der Firma Silpion untersucht. Die Firma Silpion verwendet einen manuellen Prozess, mit E-Mails und Excel Tabellen um neue Krankmeldungen einzustellen und zu verwalten. Für eine Firma, die genau solch eine Software ihren Kunden verkaufen möchte, ist es besonders wichtig als gutes Vorbild voran zu gehen und diesen Ablauf zu modernisieren.

Eine neue, individuell angefertigte Software, soll es ermöglichen die Abläufe zu vereinfachen und zu beschleunigen. Schwachpunkte sollen in der aktuellen Vorgehensweise gefunden und ausgebessert werden.

Ein Beispiel, aus der ursprünglichen Vorgehensweise, ist bei der Einreichung von Krankmeldungen zu sehen. Hierbei ist wichtig, dass der Vorgang für eine initiale Krankmeldung oder einer Verlängerung möglichst einfach gehalten wird, damit dies für die betroffene Person möglichst schnell und ohne große Anstrengung möglich ist.

Viele, vorher manuell ausgeführten Schritte, können durch einen automatisierten Prozess ersetzt werden. So wird fehleranfälligen Teilschritten vorgebeugt, wie z.B. dem Versäumen einer Frist oder dem Versenden einer E-Mail an den jeweiligen Zuständigen einer Abteilung.

Da die meisten Personen inzwischen ein Smartphone besitzen, ist es sehr wichtig, dass das System ebenfalls komfortabel auf mobilen Geräten verwendet werden kann. Dies kann entweder mithilfe einer App geschehen oder durch eine Webseite, dessen GUI an die kleineren Displaygrößen angepasst ist.

## 1.2 Zielsetzung

Ziel dieser Bachelorarbeit ist es einen Weg zu zeigen, ein System zu entwickeln, mit dessen Hilfe der Krankmeldungsprozess innerhalb der betreuenden Firma unterstützt und vereinfacht wird. Der vorhandene Prozess soll analysiert werden und als Grundlage für einen neuen, optimierten Prozess dienen. Mithilfe von festen Vorgaben aus dem Anforderungsdokument und Gesprächen mit Firmenangehörigen soll der neue Prozess entstehen.

Um diesen Prozess praktisch umzusetzen, wird ein Softwarekonzept erarbeitet, welches den Benutzer einfach durch die einzelnen Prozessschritte leitet. Es wird iterativ und inkrementell

entwickelt und dadurch ständig im Verlauf dieser Arbeit erweitert und ausgebessert. Für spätere Anpassungen und Erweiterungen ist es wichtig, dass dies beim Konzept beachtet wird.

Am Ende dieser Arbeit soll ein Server mit einer allgemeinen Schnittstelle für beliebige Clients existieren. Als Client soll eine Webapplikation entstehen, dessen GUI sich automatisch an die Displaygrößen anpasst. Dies soll gewährleisten, dass die Anwendung sowohl auf einem Desktop PC, wie auch auf einem Smartphone, bedienbar ist. Im Rahmen dieser Arbeit wird nur eine Auswahl der Anforderungen umgesetzte. Für einen Betrieb zusätzlich notwendige Anforderungen, sind nicht Teil dieser Arbeit und werden erst im Anschluss realisiert.

Die Firma Silpion, die die Arbeit unterstützt, hat sich auf die Softwareentwicklung mithilfe von agilen Vorgehen spezialisiert. Deshalb ist es wichtig, dass ein geeignetes Vorgehen für ein Ein-Mann-Team mit unregelmäßigen Entwicklungszeiten gefunden wird.

## 1.3 Struktur der Arbeit

Diese Arbeit ist in fünf Kapitel aufgeteilt, die Gliederung wird in folgendem Abschnitt beschrieben.

Anfänglich wird im Kapitel Analyse (2) ein Überblick über den aktuellen Vorgang gezeigt. Der neue Vorgang wird mithilfe des Anforderungsdokuments erweitert und optimiert. Dabei werden die gewünschten Funktionalitäten vorgestellt und mit derzeit erhältlichen Anwendungen verglichen. Aus diesem Vergleich resultiert, ob es lohnend ist eine eigene Applikation zu entwickeln oder einzukaufen.

Für die erarbeiteten Anforderungen wird im Design Kapitel (3) ein Realisierungskonzept entwickelt. Die Umsetzung wird vollständig zwischen dem Client und dem Server unterschieden. Die für beide Komponenten gleichermaßen wichtigen Entscheidungen, wie die Kommunikation über die Rest-Schnittstelle und dessen Absicherung, wird beschrieben. Der interne Ablauf einzelner Szenarien wird vorgestellt. Die verwendeten Frameworks werden festgelegt.

Anschießend wird in der Realisierung (4) beschrieben, wie die Anwendung umgesetzt wird. Dabei wird sowohl auf das Vorgehen bei der Entwicklung eingegangen, als auch auf Tools, die diese direkt unterstützten.

Am Ende dieser Arbeit werden im Kapitel Schluss (5) die Ergebnisse besprochen und ein Ausblick über die zukünftige Verwendung und Weiterentwicklung der Anwendung gegeben.

# 2 Analyse

In diesem Kapitel werden die Funktionalitäten des Systems, mithilfe der Zielsetzung aus dem ersten Kapitel erarbeitet.

Der Ist-Zustand und der Soll-Zustand werden vorgestellt und miteinander verglichen. Schwachstellen im aktuellen Zustand werden hervorgehoben.

Vorhandene Anwendungen, die die Möglichkeit geben Krankmeldungen zu erfassen und zu bearbeiten, werden vorgestellt und mit Blick auf die Verwendbarkeit innerhalb der Firma bewertet.

Innerhalb der Problemstellung wird dargestellt, wieso die ursprüngliche Lösung den aktuellen Anforderungen nicht mehr gerecht wird.

## 2.1 Problemstellung

Eine Krankmeldung ist in einer Firma ein alltäglicher Vorgang. Für die Durchführung müssen die betroffenen Mitarbeiter und Mitarbeiterinnen den genauen Ablauf kennen. Der Prozess kann sich von Firma zu Firma unterscheiden. Es ist z.B. der Firma überlassen, ab welchem Tag eine Arbeitsunfähigkeitsbescheinigung[1] erforderlich ist. Welchen Personen der oder die Betroffene die Krankmeldung melden muss kann auch unterschiedlich sein, ist aber sehr wichtig, damit auf den Ausfall reagiert werden kann. Die Bearbeitung einer Krankmeldung und das Erstellen von Auswertungen ist für die Abteilungsleitung ein eher störender Vorgang, der die eigentliche Arbeit unterbricht und deswegen schnell, ohne großen Aufwand erfolgen sollte.

Bei der Erfassung und Auswertung der Daten müssen rechtliche Beschränkungen des Betriebsrates beachtet werden. Personenbezogene Daten, wie die gesamten Krankheitstage einer Person müssen vertraulich behandelt werden und dürfen nicht veröffentlicht werden. Wenn dies nicht beachtet wird, darf die Software nicht eingeführt werden.

[1] AU - wird vom Arzt ausgestellt

Die Studie TK Gesundheitsreport 2014[2] hat ergeben, dass die durchschnittlichen Arbeitsunfähigkeitstage pro erwerbstätigen Person seit 2006 stetig steigen.


Abbildung 2.1: Statistik: Arbeitsunfähigkeitstage je Erwerbsperson 2014 (Zeit Online 2014)

Mithilfe von Umfragen, wie der folgenden von der IKK classic und genaueren Analysen, wird versucht herauszufinden, woran die steigende Arbeitsunfähigkeitsquote liegen kann.

Die Umfrage der IKK classic hat ergeben, dass sich circa 22% der Erwerbstätigen zwischen 18 und 29 Jahren schon einmal ohne Krankheit krank gemeldet haben. Bei Betrachung aller Erwerbstätigen sind es etwa 11%. Als Begründung gaben die Meisten (43%) an, dass sie sich überlastet fühlten und eine Auszeit benötigten, an zweiter Stelle liegen private Gründe oder private Termine (36%). Ursachen für die Überlastung können Zeitdruck, zu viel Arbeit oder mangelnde Wertschätzung im Beruf sein.[3]

[2] http://www.tk.de/centaurus/servlet/contentblob/644772/Datei/124007/Gesundheitsreport-2014.pdf, Abgerufen am 29.07.2014

[3] vgl. https://www.ikk-classic.de/presse/pressemitteilungen/bundesweit/aktuell/19032014-blaumachen-oder-krank-zur-arbeit.html, Abgerufen am 29.07.2014

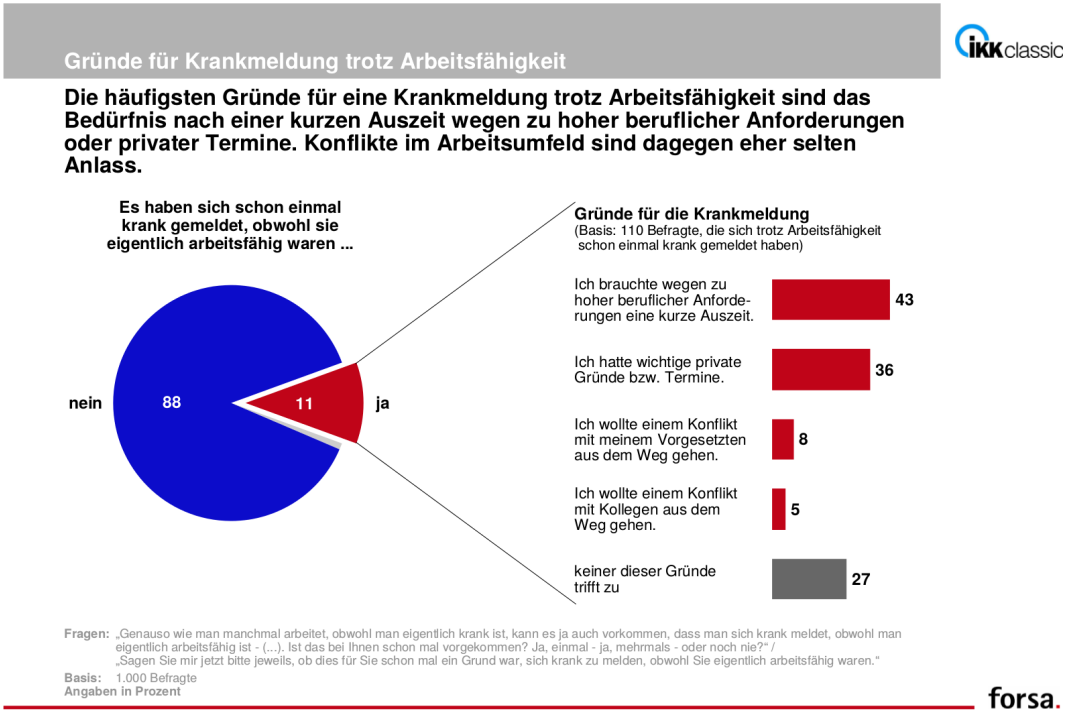


Abbildung 2.2: Statistik: Gründe für Krankmeldung trotz Arbeitsfähigkeit (IKK classic 2014)

Durch eine Analyse der Krankmeldungen kann auf die Zufriedenheit einer Abteilung geschlossen werden. Wenn sich in einer speziellen Abteilung besonders viele Leute krankmelden ist dies oft ein Indiz dafür, dass etwas nicht stimmt. Dies kann z.B. viel Stress und im extrem Fall Burn-Outs durch zu viel Arbeit bedeuten. Ein Grund dafür kann die ständige Erreichbarkeit sein.[4]

Silpion IT Solutions hat sich seit der Gründung im Jahr 2000 sehr verändert. Silpion ist seit dem stark gewachsen, aus dem Unternehmen mit circa 10 Angehörigen sind inzwischen um die 120 feste Mitarbeiter und Mitarbeiterinnen und circa 180 Freelancer und Freelancerinnen geworden. Die Prozesse und Strukturen von Silpion, wie der Prozess der Krankmeldung, stammen teilweise allerdings noch aus den Anfangszeiten der Firma.[5]

Die steigende Arbeitsunfähigkeitsquote und dass Silpion immernoch am wachsen ist, verdeutlicht, dass eine modernisierte Lösung mit der Zeit immer mehr an Bedeutung gewinnt.

[4] vgl. http://www.zeit.de/karriere/beruf/2014-04/infografik-zunahme-fehltage-arbeitslosenquote, Abgerufen am 29.07.2014

[5] vgl. Anforderungsdokument

Innerhalb der Firma zeigt die Praxis, dass bei der aktuellen Lösung alles manuell, ohne automatische Vorgänge, durchgeführt wird. Durch diese Lösung werden schnell Krankmeldungen übersehen oder nicht richtig bearbeitet. Da die Abteilungsleitung und die Buchhaltung jeweils eine eigene Liste führen, kann dies schnell zu Inkonsistenzen führen. Die Abarbeitung durch die Abteilungsleitung ist nicht fest vorgegeben, deshalb kann es vorkommen, dass Abteilungen das Vorgehen unterschiedlich handhaben.

Ein fester Prozess für Krankmeldungen ist für das Einreichen von Krankmeldungen und Arbeitsunfähigkeitsscheinen vorhanden. Allerdings existiert kein fester Ablauf wie die Bearbeitung innerhalb unterschiedlicher Abteilungen geregelt ist. Es ist klar geworden, dass der Vorgang mithilfe einer softwareunterstützten Lösung erheblich verbessert werden kann.

Der aktuelle Ablauf macht deutlich, dass es wichtig ist, dass ein neuer klar definierter Prozess entsteht, der mithilfe einer Software geleitet wird. Dadurch sollen manuelle, leicht übersehbare Erinnerungen, wie z.B. zum Abgeben einer AU, unnötig werden.

### 2.1.1 Ist-Zustand

In diesem Abschnitt wird der aktuelle Ablauf einer neuen Krankmeldung vorgestellt. Zur Visualisierung wurde ein Business Prozess Diagramm (BPMN) gewählt. Diese bieten eine übersichtliche Möglichkeit Geschäftsvorfälle zu beschreiben.

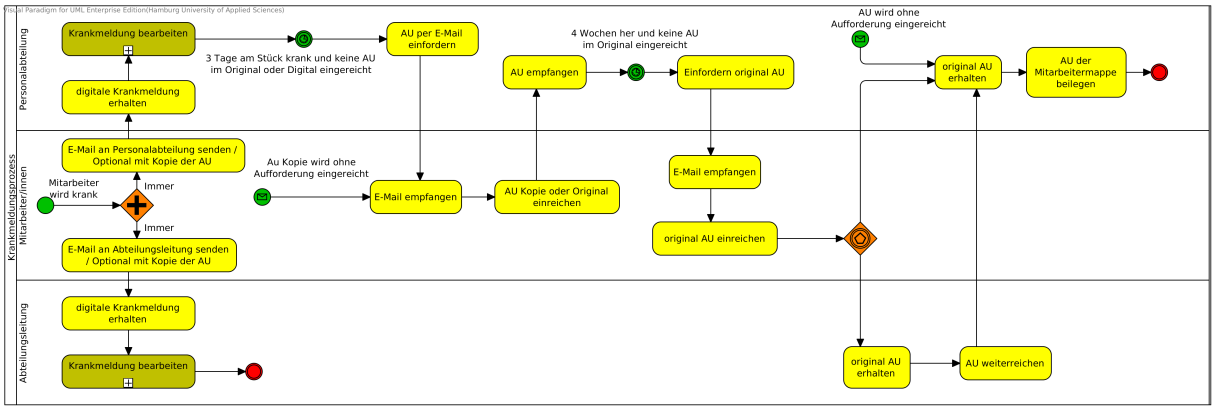


Abbildung 2.3: BPMN Diagramm: Ist Zustand: Erstellen einer Krankmeldung

Die Abbildung 2.3 zeigt den Ablauf beim Erzeugen einer neuen Krankmeldung. Im Krankheitsfall müssen aktuell mehrere Schritte durchgeführt werden. Die Schritte werden nicht direkt ineinander übergeleitet, sondern starten zeitabhängig.

Am ersten Tag der Krankheit muss am Morgen der Abteilungsleitung und der Personalabteilung eine E-Mail geschickt werden. Optional kann dabei eine Kopie der Arbeitsunfähigkeitsbescheinigung (AU) im Anhang mitgesendet werden. Die Abteilungsleitung und Personalabteilung bearbeiten die Krankmeldung intern auf unterschiedliche Arten. Relevante Kollegen und Kolleginnen, aus gemeinsammen Projekten, werden von der Abteilungsleitung informiert. Eventuell führt die Abteilungsleitung noch eigene Listen, wann welche Personen ihrer Abteilung krank war.

Wenn der oder die Mitarbeitende am dritten Tag infolge immernoch krank ist, muss eine AU eingereicht werden. Es ist ausreichend, wenn diese eingescannt oder abfotografiert wird und später im Original nachgereicht wird. Die AU kann auch direkt vorbeigebracht oder per Post geschickt werden. Wenn am dritten Tag in Folge noch keine AU per E-Mail geschickt oder im Original eingereicht wurde, fordert die Personalabteilung diese per E-Mail ein. Die Arbeitsunfähigkeitsbescheinigung sollte eigentlich immer zur Personalabteilung geschickt werden. Es hat sich allerdings herrausgestellt, dass diese oft an die Abteilungsleitung geschickt werden, welche diese dann wiederum weiterleiten müssen[6].

Spätestens 4 Wochen nach dem ersten Tag der Krankheit muss die AU im Original vorliegen. Wenn dies noch nicht der Fall, ist fordert die Personalabteilung die AU ein.

Die Personalabteilung archiviert die AU in der Mitarbeitermappe, wenn Fragen von der Krankenkasse oder des Steuerberaters beantwortet werden müssen, wird diese gegebenenfalls kopiert. Die AU wird per Kopie oder im Original an den Steuerberater geschickt und die Fragen parallel per E-Mail beantwortet. Die Fragen der Krankenkassen werden traditionell per Briefpost beantwortet.

Verlängerungen einer Krankmeldung werden fast wie eine neue Krankmeldung gehandhabt. Der Unterschied besteht darin, dass eine AU weiterhin nach einer Gesamtdauer von drei Tage, ab der initialen Krankmeldung, eingereicht werden muss. Die 40 Tage für das Einreichen der Original AU gilt weiterhin ab der initialen Krankmeldung.

Die Verlängerung ist in der Grafik nicht dargestellt.

Die aktuelle Vorgehensweise beinhaltet ein paar Schwachstellen. Durch die vielen E-Mails, die versendet werden müssen, werden manche einfach vergessen oder beim Empfänger über-

[6] vgl. Anforderungsdokument

sehen. Da die Personalabteilung für jede Krankmeldung eine E-Mail erhält, müssen sie alle nacheinander abarbeiten. Dies kostet viel Zeit.

Bei den zeitbedingten Aktivitäten kommt es schnell vor, dass diese nicht durchgeführt werden, wenn die Personalabteilung intern keine automatischen Erinnerungen wie z.B. Kalendereinträge verwendet.

Damit der gesamte Prozess vollständig durchgeführt werden kann, ist es notwendig, dass alle Mitarbeiter alle für sie erforderlichen Schritte kennen. Davon kann nicht ausgegangen werden, da es Personen gibt, die häufiger krank sind und andere nur sehr selten.

Dieser Vorgang macht Auswertungen über die gesamten Krankheitstage eines Mitarbeitenden, einer Abteilung oder der gesamten Firma, sehr aufwendig.

### 2.1.2 Soll-Zustand

Der gewünschte Endzustand des Vorganges zum Anlegen einer Krankmeldung wird in diesem Abschnitt vorgesellt. Dabei werden die notwendigen Aktionen der betroffenen Person beschrieben und die unterschiedlichen Benachrichtigungen dargestellt, die an die verschiedenen Personengruppen gehen. Der Kontrollfluss innerhalb des Systems wird visualisiert. Der Zustand wurde mithilfe des Anforderungsdokuments und durch Mitarbeitergespräche entwickelt.

Abbildung 2.4 zeigt den Soll-Zustand des neuen Vorganges. Der Mitarbeiter oder die Mitarbeiterin meldet sich am morgen der Arbeitsunfähigkeit über das System krank. Optional kann ein Bild der Vorderseite und ein Bild der Rückseite der AU hochgeladen werden.

Sobald die Krankmeldung abgesendet wurde, erstellt das System automatisch eine Benachrichtigung für die Abteilungsleitung und die Personalabteilung.

Wurde keine AU hochgeladen oder als Original abgegeben, wird dem Mitarbeitenden, der Personalabteilung und der jeweiligen Abteilungsleitung nach drei Tagen eine Benachrichtigung gesendet. Sie erinnert alle Beteiligten daran, dass noch eine AU nachzureichen ist, bzw. eine zu bearbeitende Krankmeldung aussteht.

Weitere Benachrichtigungen werden nach einer Gesamtdauer von vier und sechs Wochen der Arbeitsunfähigkeit gesendet. Wenn dieser Fall eintritt, müssen die Abteilungsleitung und Personalabteilung entsprechende nicht weiter definierte Schritte durchführen. Bei einer

Arbeitsunfähigkeit, ab einer Dauer von vier Wochen, hat ein Arbeitnehmer das Recht auf eine Lohnfortzahlung.[7] Der Anspruch auf eine Fortzahlung besteht für bis zu sechs Wochen.[8]

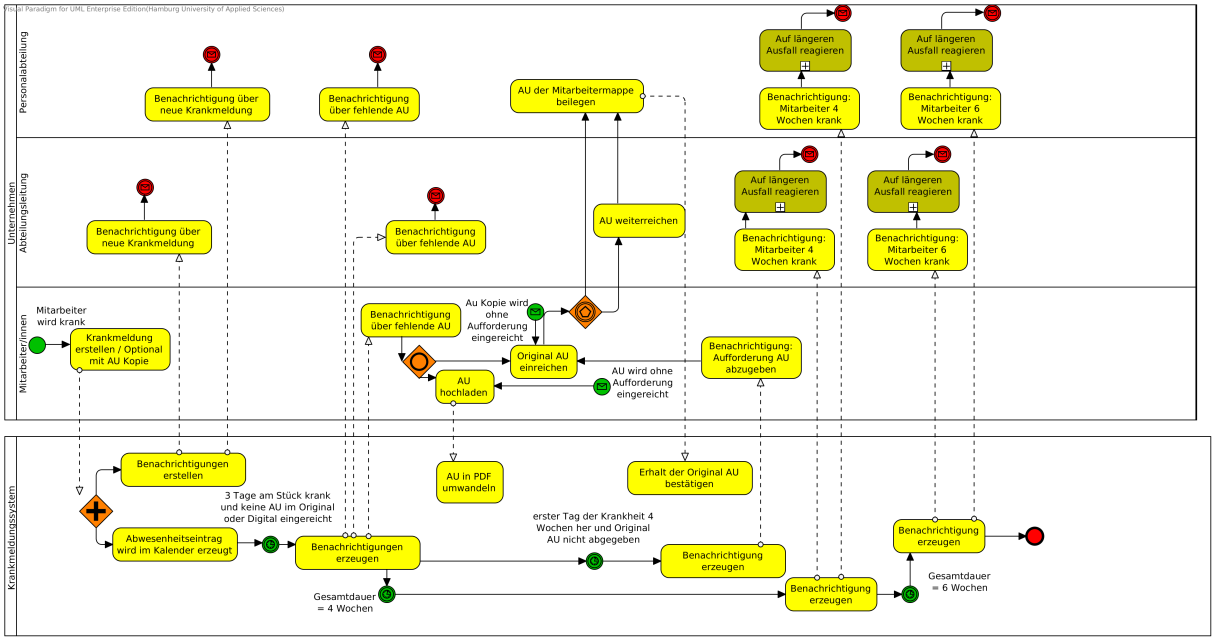


Abbildung 2.4: BPMN: Soll Zustand: Erstellen einer Krankmeldung

Nach vier Wochen muss die AU im Original bei der Personalabteilung vorliegen, falls dies nicht der Fall ist, wird der betroffenen Person, der Personalabteilung und der Abteilungsleitung ebenfalls eine Benachrichtigung gesendet.

In der Grafik ist nur die initiale Krankmeldung dargestellt. Die Verlängerung wird dort nicht behandelt. Diese wird aber ebenfalls fast wie eine neue Krankmeldung behandelt. Der Unterschied besteht darin, dass die Gesamtdauer aufaddiert wird.

### 2.1.3 Vergleich der Zustände

In der alten Lösung kann nur sehr schwer über individuell geführte Tabellen herausgefunden werden, wer wann krank war und ob eine AU eingereicht wurde. Dabei kann sehr schnell ein Fehler gemacht werden.

[7] Entgeltfortzahlungsgesetz §3 Abs. 3
[8] Entgeltfortzahlungsgesetz §3 Abs. 1

Die Abteilungsleitung und Personalabteilung hat sehr viel zu tun, deswegen sind Auswertungen über die Krankheitstage neben ihrer alltäglichen Arbeit sehr mühselig. Die Praxis hat gezeigt, dass sie immer weiter nach hinten geschoben oder vollständig ausgelassen werden.

Die gewünschte Lösung führt die Benutzer durch die einzelnen Schritte des Prozesses, dadurch muss nicht mehr jeder Nutzer die genauen Schritte kennen. Zeitgesteuerte Events werden automatisch ausgeführt, wenn sie notwendig sind und können so nicht mehr in Vergessenheit geraten. Die Events generieren automatisch Benachrichtigungen, wenn das Handeln eines Mitarbeitenden nötig ist. Außerdem sind keine manuellen geführten Listen mehr erforderlich, dadurch ist automatisch ein konsistenter Zustand garantiert.

Durch die neue Lösung kann jeder Benutzer selber nachsehen, wann er oder sie sich krank gemeldet hat. Dies war vorher nur möglich, indem sie selber eine Liste geführt haben oder die Personalabteilung gefragt haben.

Mithilfe der zentralen Server Lösung können alle Daten, wie die Informationen einer Krankmeldung, dessen Verlängerungen und digitale Kopien der Arbeitsunfähigkeitsbescheinigungen, an einer zentralen Stelle aufbewahrt und zugänglich gemacht werden. Die zentrale Haltung der Daten und des aktuellen Bearbeitungsstandes erleichtert die Bearbeitung durch mehrere Mitglieder der Personalabteilung. Die zentrale Lösung ermöglicht es außerdem schneller die Fragen des Steuerberaters und der Krankenkassen zu bearbeiten. Durch vorgefertigte Vorlagen kann schnell eine Übersicht mit den gewünschten Kriterien erzeugt werden.

## 2.2 Use-Case

Anwendungsfälle (engl. Use-Case) stellen die fachlichen Anforderungen eines Systems dar. Sie drücken das Ziel einer Aktion aus. Die Interaktionen verschiedener Benutzer (Aktoren) mit dem System wird dargestellt. Sie werden in den meisten Fällen in Verbindung mit agilen Entwicklungsvorgehen verwendet.

In dieser Arbeit soll ein System entwickelt werden, mit dessen Hilfe sich die Firmenangehörigen, über einen einfachen Weg, schnell krankmelden können. Alle benötigten Informationen sollen an zentraler Stelle gesammelt werden. Erinnerungen, wie z.B. falls der Arbeitsunfähigkeitsschein fehlt, sollen automatisch per E-Mail versendet werden.

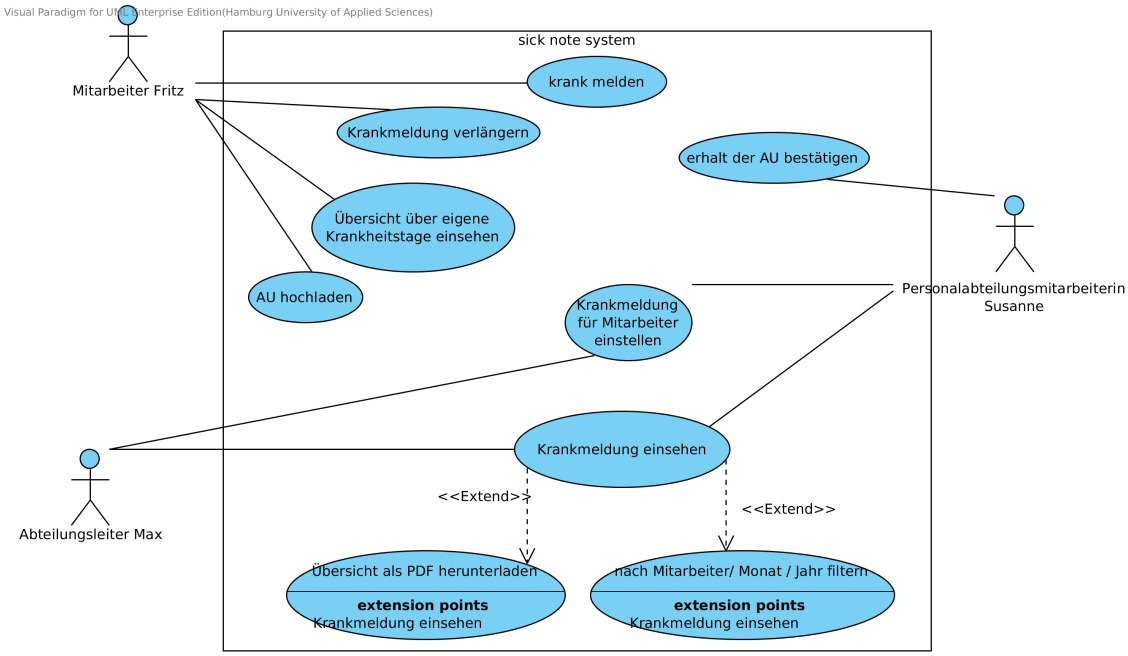


Abbildung 2.5: Use Case Diagramm

Eine genauere Beschreibung der einzelnen Anforderungen ist im Abschnitt über die Funktionalen Anforderungen (Kapitel 2.2.2) zu finden.

### 2.2.1 Personas

Eine Persona ist eine fiktive Person, die das Produkt verwenden soll. Diese Persona besitzt Eigenschaften, wie zum Beispiel Ziele, Wünsche und Fähigkeiten, die relevant für das System sind.

#### Mitarbeiter Fritz

Fritz ist normaler Mitarbeiter der Firma. Er ist 32 Jahre alt und arbeitet als Entwickler bei Silpion. Privat verfügt er über einen Internetanschluss und besitzt außerdem ein Smartphone. Er hat eine Tochter, die vier Jahre alt ist. Fritz ist verheiratet, seine Frau ist ebenfalls berufstätig. Wenn ihr gemeinsames Kind krank wird, muss sich ein Elternteil zuhause um die Tochter kümmern.

#### Abteilungsleiter Max

Max ist 35 Jahre alt und Abteilungsleiter der Web Engineering Abteilung. Er hat viele interne und externe Termine am Tag. Er pflegt viele externe Firmenkontakte und hat auch intern einige Aufgaben zu erledigen. Durch seine zahlreichen Tätigkeiten hat er viele Meetings und bekommt einige E-Mails pro Tag.

#### Personalabteilungsmitarbeiterin Susanne

Susanne ist 27 Jahre alt und arbeitet in der Personalabteilung der Firma. Sie kann gut mit einem Computer umgehen, hat aber nicht viel Wissen über den technischen Hintergrund.

### 2.2.2 Funktionale Anforderungen

Funktionale Anforderungen legen fest, was das System leisten soll. Durch das bereits erarbeitete Anforderungsdokument ergeben sich die nachfolgend beschriebenen funktionalen Anforderungen.

#### F1 - 3 teiliges Rollensystem

Durch die Unternehmensstruktur gibt es drei vorgegebene Rollen, mit jeweils unterschiedlichen Berechtigungen. Die Berechtigungen werden jeweils durch die einzelnen Anforderungen beschrieben.

**Mitarbeiter/innen** Zu dieser Rolle gehören alle Entwickler/innen aber auch Personen der Infrastruktur, Buchhaltung und sonstigen Abteilung.

**Abteilungsleitung** Sie sind für ihre Mitarbeiter und Mitarbeiterinnen verantwortlich und müssen bei Ausfall entsprechend reagieren.

**Personalabteilung** Dies ist eine eigene Abteilung. Sie beantworten die Fragen vom Steuerberater und den Krankenkassen und kümmern sich um alles was den Personalstamm betrifft.

#### F2 - Mitarbeiter/in meldet sich krank

Ein Mitarbeiter oder eine Mitarbeiterin ist krank. Er, bzw. sie meldet sich online an dem System an und kann optional einen Arbeitsunfähigkeitschein hochladen. Außerdem muss ein

Anfangsdatum (meist das aktuelle) und ein voraussichtliches Enddatum eingetragen werden. Optional kann auch eine Bemerkung hinzugefügt werden.

### F3 - Mitarbeiter/in meldet pflegebedürftige Person krank

Eine pflegebedürftige Person, um die sich der Mitarbeiter oder die Mitarbeiterin kümmern muss, ist krank.

Es muss die Möglichkeit geben, zwischen einer Krankmeldung des Arbeitnehmers, der Arbeitnehmerin selbst oder einer pflegebedürftigen Person unterscheiden zu können.

Eine weitere Unterscheidung, auf die geachtet werden muss, ist ob es sich bei der pflegebedürftigen Person um ein Kind handelt. Im Falle eines Kindes bekommt die Firma für jeden Tag Krankengeld von der Versicherung. Die Kranktage des Angestellten selbst werden vom Gehalt abgezogen und der Angestellte bekommt das Geld von der Krankenkasse wieder.

Dies kann allerdings je nach gesetzlicher oder privaten Krankenkasse unterschiedlich sein.[9]

### F4 - Übersicht über Krankmeldungen erzeugen

Benutzer können sich eine Übersicht über ihre Krankmeldungen anzeigen lassen. Sie können filtern nach:

- Jahr
- Monat

Die Abteilungsleitung kann eine Übersicht über die Krankmeldungen ihrer Abteilung generieren. Sie können zusätzlich nach Personen filtern. Die Personalabteilung kann sich eine Übersicht über alle Personen erzeugen lassen.

### F5 - PDF aus Übersicht erzeugen

Aus jeder aus F4 erzeugbaren Übersichten soll ein PDF generiert werden, das ggf. der Krankenkasse oder dem Steuerberater zugeschickt werden kann.

[9] vgl. http://www.arbeitsrecht.org/arbeitnehmer/krankheit/voraussetzungen-fuer-die-entgeltliche-freistellung-zur-pflege-von-kranken-kindern/, Abgerufen 24.07.2014

### F6 - Stellvertretend Krankmeldungen durchführen

Die Abteilungsleitung kann für ihre Mitarbeitenden stellvertretend eine Krankmeldung erzeugen. Die Personalabteilung für alle Firmenmitglieder.

### F7 - Erhalt der original AU bestätigen

Die Personalabteilung kann eine Checkbox setzen, wenn eine AU im Originalen eingereicht wurde. Für User anderer Rollen, ist der Status nur sichtbar.

### F8 - Upload einer AU

Es gibt die Möglichkeit optional eine AU hochzuladen. Der Upload muss durch je ein Bild der Vorderseite und ein Bild der Rückseite geschehen. Die Bilder sollen zu einem PDF zusammengefasst werden. Das PDF kann von der Abteilungsleitung und der Personalabteilung eingesehen werden.

### F9 - Automatische Benachrichtigung

Die Abteilungsleitung und die Personalabteilung haben die Möglichkeiten die Art der Benachrichtigung wie folgt zu konfigurieren:

- per E-Mail
- per Benachrichtigung im System

Sie bekommen Benachrichtigungen wenn:

- sich ein/e Mitarbeiter/in initial krank meldet
- sich ihre/seine Krankmeldung verlängert
- ein/e Mitarbeiter/in inzwischen 3 Tage krank ist und keine AU hochgeladen hat
- ein/e Mitarbeiter/in inzwischen mehr als 4 Wochen krank ist
- ein/e Mitarbeiter/in inzwischen mehr als 6 Wochen krank ist

Mitarbeitende bekommen eine Benachrichtigung wenn:

- sie am dritten Tag der Krankheit noch keine AU hochgeladen haben und keine AU im Original eingereicht haben

- sie nach vier Wochen Krankheit noch nicht alle AUs im Original eingereicht haben

#### F10 - Eintragung in Kalender

Nach der Krankmeldung soll automatisch ein Abwesenheitseintrag für die Person im Urlaubskalender generiert werden.

### 2.2.3 Nichtfunktionale Anforderungen

Nichtfunktionale Anforderungen geben Eigenschaften des zu entwickelnden Systems vor. Sie beinhalten technische Anforderungen an das System.

#### NF1 - zentrales Login über das LDAP-System

In der Firma ist bereits durch ein LDAP-System[10] ein zentraler Authentifizierungsdienst vorhanden. Dieser soll verwendet werden, um sich bei dem Krankmeldungssystem einzuloggen. Das LDAP-System gibt außerdem Informationen über den User, welcher Rolle dieser angehört und wer dessen Vorgesetzter ist.

Der Login wird vorerst nicht Teil der Bachelorarbeit sein. Stattdessen wird ein sehr vereinfachtes Usermanagement umgesetzte, das später ohne große Anpassungen durch die zentrale Lösung ersetzt oder erweitert werden kann.

#### NF2 - Single Sign-On für mehrere Systeme

Es soll ein zentrales Loginsystem für alle teilnehmenden Systeme existieren. Sobald sich ein User bei einem dieser Systeme anmeldet, soll er automatisch alle weiteren, ohne erneute Authentifizierung, verwenden können. Für dieses System ist die Verwendung der allgemein gültigen Userdaten notwendig (siehe NF1), aus diesem Grund wird wird diese Anforderung ebenfalls nicht im Rahmen der Bachelorarbeit umgesetzt.

Im folgenden wird eine Möglichkeit beschrieben, wie diese Anforderung im Anschluss an diese Arbeit umgesetzt werden kann.

[10] Lightweight Directory Access Protocol - Verzeichnisdienst zum Auslesen von z.B. personenbezogenen Daten, Möglichkeit einen allgemeinen Authentifizierungsdienst bereitzustellen

Für ein Single Sign-On kann z.B. das Kerberos Protokoll verwendet werden. Kerberos wurde in den 80er Jahren am MIT entwickelt und ist inzwischen in der fünften Version veröffentlicht worden.

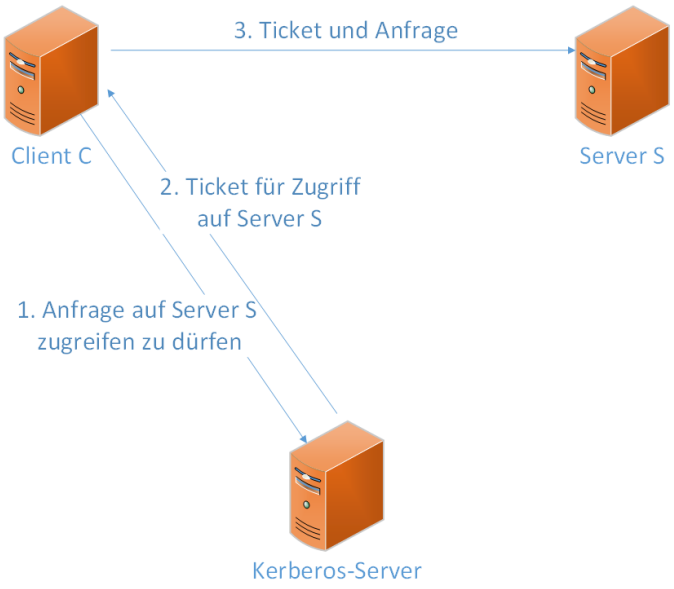


Abbildung 2.6: Stark vereinfachte Funktionsweise des Kerberos Protokoll

Abbildung 2.6 zeigt den Ablauf des Kerberos Protokolls, stark vereinfacht. Im ersten Schritt fragt der User mit seinen Benutzerdaten beim Kerberos-Server eine Zugriffsberechtigung für den Server S an. Der Kerberos-Server gibt dem Client ein gültiges Ticket für den Zugriff auf den Server zurück. Mithilfe des Tickets kann der User einen erfolgreichen Zugriff zum Server S erlangen und dort seine gewünschten Anfragen stellen. Der Kerberos-Server kennt alle Passwörter und Schlüssel (zur asynchronen Verschlüsselung) der Benutzer und Server. Die Tickets sind dementsprechend verschlüsselt, sodass diese nur vom Kerberos-Server erzeugt werden können und nur vom Ziel Server gelesen werden können. Innerhalb der verschlüsselten Informationen sind zusätzliche Daten hinzugefügt worden, die die Gültigkeit des Tickets einschränken.

#### NF3 - lose Kopplung des Servers und Clients, Kommunikation mittels eines Rest-Services

Das User-Interface und die serverseitige Programmlogik sollen voneinander entkoppelt sein. Die Clients und Server sollen mithilfe einer Rest Schnittstelle (siehe Kapitel 3.1.3) miteinander kommunizieren. Dies macht es bei Bedarf möglich weitere Clients hinzuzufügen, ohne dass der Server bearbeitet werden muss. Weitere Clients können z.B. eine native iOS oder Android-App sein. Diese sind nicht Teil dieser Arbeit.

#### NF4 - responsive Webdesign

Es gibt optimierte User-Interfaces für PCs, Tablets und Smartphones. Das System soll über die mobilen Geräte gut bedienbar sein. Dies soll mithilfe des Twitter Bootstrap Frameworks unterstützt werden. Silpion besitzt bereits ein auf das Unternehmen angepasstes Design.

## 2.3 Evaluierung des Marktes

In diesem Abschnitt werden bereits auf dem Markt verfügbare Lösungen vorgestellt und analysiert.

Da die Anforderungen individuell für die Firma angepasst sind, ist es schwierig eine vorhandene, passende Lösung zu finden, die nicht aufwendig angepasst werden müsste.

Die Anwendung wird kurz beschrieben und erläutert, welche Anforderungen durch das Produkt nicht erfüllt werden und weshalb es nicht für den Einsatz geeignet ist.

### 2.3.1 ESIS- Elektronisches Schüler Informationssystem

Für Schulen existiert ein System names ESIS (Elektronisches Schüler Informations System). ESIS ist ein nicht kommerzielles Produkt. Die Anwendung wurde erstmals 2007 innerhalb einer Schule eingesetzt, inzwischen verwenden es circa 260 Schulen.

Neben der webbasierten Krankmeldung existieren Module für die Neuanmeldung an einer Schule, zum Buchen der Sprechzeiten der Lehrer und zur allgemeinen Informationsverbreitung an die Eltern. Die Schüler und Eltern können durch das System kostengünstiger und schneller informiert werden.[11]

[11] vgl. http://www.esis.de/esis.html, Abgerufen am 26.03.2014

Für die Firma Silpion ist nur das Modul für die Krankmeldung interessant. Die übrigen Module sind überflüssig und werden nicht benötigt.

An dem Prozess der Krankmeldung müsste einiges geändert werden. ESIS benötigt keine Anmeldung, um eine Person krank zu melden. Die Formulare enthalten schulspezifische Felder, wie z.B. Klasse, die ebenfalls nicht benötigt werden. Die Möglichkeiten eine Arbeitsunfähigkeitsbescheinigung hochzuladen, den Erhalt der original AU zu bestätigen oder Auswertung automatisch zu erzeugen fehlen. Das System deckt die gewünschten Benachrichtigungen ebenfalls nicht ab.

Die technische Umsetzung eignet sich nicht für eine professionelle Software Entwicklungsfirma. Die Webanwendungen werden auf einem externen Server von ESIS gehostet. Innerhalb der Schule befindet sich nur ein Administrationprogramm.

Durch die vielen nötigen Änderungen und die nicht vorhandene Rest Schnittstelle eignet sich dieses System leider nicht für den Einsatz bei Silpion und wird deswegen nicht weiter betrachtet.

### 2.3.2 HCM Fehlzeitenverwaltung

Von der Firma HCM CustomerManagement GmbH gibt es eine webbasierte Software für das Abwesenheitsmanagement. Diese Lösung bietet neben der Krankmeldung außerdem die Möglichkeiten Urlaubsanträge und Dienstreiseanträge durchzuführen. Für Urlaubsanträge existiert bereits eine Software, die ebenfalls als Bachelorarbeit entwickelt wurde und sich aktuell in einer Testphase befindet. Ein Tool für Dienstreisen wird zurzeit nicht benötigt.

Die Dienste werden wie gewünscht als Webapplikation zur Verfügung gestellt.

Die Software bietet es an die benötigten Formulare frei zu definieren. Ebenfalls ist die gewünschte Authentifikation über den LDAP Server möglich.

Die Firma bietet an die Software als SaaS[12] aber auch die Möglichkeit eine Lizenz zu kaufen, um das System auf einem eigenen Server auszuführen und anzupassen oder sich eine angepasste Version liefern zu lassen.

Die Hersteller empfehlen ebenfalls ihre Personalverwaltungs-Lösung zu verwenden, da dadurch erst alle Prozesse mithilfe der Personalstammdaten durchgeführt werden können.

[12] Software as a Service - Anmieten einer bereits laufen und vorkonfigurierten Version auf einem fremden Server

Die Software stellt keine Rest-Schnittstelle für eine App oder der Interaktion mit anderen Systemen zur Verfügung.[13]

Diese Lösung entspricht zum großen Teil den Anforderungen, für den vollen Funktionsumfang muss aber noch eine zweite Software für die Personalstammdaten erworben werden. Durch diese Software enstehen größere Kosten für die Lizenzen und Anpassungen als durch eine Neuentwicklung im Rahmen einer Bachelorarbeit. Die Module für die Urlaubsplanung und Dienstreiseanträge werden nicht benötigt und verursachen bei der Anschaffung und Wartung nur unnötige Kosten.

### 2.3.3 SAP ERP System

Das SAP ERP System[14] beinhaltet die Komponente Personalzeitwirtschaft. Der Funktionsumfang des Systems ist wählbar, es können sowohl Krankheitszeiten und Urlaubszeiten erfasst werden, aber auch der allgemeine Personalbedarf kann ermittelt werden. Ein Einsatz als Zeiterfassungssystem ist möglich.

Das System kann mithilfe von beliebigen Modulen erweitert werden, dadurch können z.B. Funktionen wie das Organisieren von Veranstaltungen, mit Berücksichtigung der Anwesenheit, verwendet werden.

Mithilfe von Self-Service Anwendungen, wie einer Web-Anwendung, kann die gewünschte Anforderung, dass sich Mitarbeiter und Mitarbeiterinnen selbst krank melden können, erfüllt werden. [15]

Eine Rest-Schnittstelle, für eine eventuell später entwickelte Mobile App, ist nicht vorhanden.

Bisher verwendet die Firma Silpion kein SAP Modul, aus diesem Grund ist auch kein spezielles Wissen darüber vorhanden.

Das SAP System ist sehr komplex und aufwendig einzuführen. Durch die Komplexität wird die Installation und Wartung erheblich Zeit in Anspruch nehmen.

Durch die erforderlich Zeit und die benötigten Lizenzen entstehen erhebliche Kosten für die Firma.

[13] vgl. http://hcm-infosys.com/vdoc/easysite/hcm/losungen/urlaub-krank-dienstreise, Abgerufen am 26.03.2014
[14] ERP - Enterprise-Resource-Planning: Produkt der Firma SAP zum organisieren und leiten aller wichtigen Prozesse innerhalb eines Unternehmens
[15] vgl. http://help.sap.com/saphelp_470/helpdata/de/8a/98459c46c411d189470000e829fbbd/content.htm, Abgerufen am 27.07.2014

### 2.3.4 Fazit

Aktuell gibt es ein paar Softwarelösungen, die eine ähnliche Richtung einschlagen, wie es gewünscht ist.

Leider bringen die Lösungen viele weitere Komponenten mit, die nicht benötigt werden. Da die Anforderungen individuell auf Silpion zugeschnitten sind, stellt eine fertige, existierende Lösung keine akzeptable Option bereit. Es ist vorteilhaft, wenn die Software im eigenen Haus entwickelt wird. Dadurch kann die Lösung an die individuellen Anforderungen angepasst werden und bei Bedarf zu einem späteren Zeitpunkt weiterentwickelt oder Änderungen vorgenommen werden.

Keines der erwerbbaren Produkte bietet eine offene Rest-Api an, um weitere Systeme anzubinden. Dies ist ein Nachteil, da so keine externen Systeme und Apps auf die Daten zugreifen können.

Wegen der genannten Nachteile, empfiehlt sich die schnellere Lösung, ein Produkt zu kaufen, nicht, stattdessen wird eine neue Software entwickelt.

# 3 Design

In diesem Kapitel wird ein Konzept für die Realisierung des Krankmeldungssystems erarbeitet. Zur Realisierung dieses Systems tragen sowohl funktionale, wie auch nichtfunktionale Anforderungen bei.

Da die Unabhängigkeit zwischen dem Client, mit dem User-Interface (Frontend), und dem Server (Backend), mit der eigentlichen Logik, durch die Anforderungen sehr wichtig ist, wird beim Design vollständig zwischen ihnen unterschieden.

Für die Kommunikation ist eine gut ausgearbeitete Rest-Api nötig, damit bei späteren Änderungen die vorhandenen Clients nicht unbrauchbar werden. Die Api wird inkrementell während der Entwicklung der Software entwickelt.

Am Anfang dieses Kapitels werden Designentscheidungen beschrieben, die relevant für den Client und den Server sind. In den darauf folgenden Sektionen werden für den Server und anschließend für den Client relevante Kriterien vorgestellt.

## 3.1 Allgemein

Designentscheidungen, die für den Client und den Server gemeinsam wichtig sind, werden hier beschrieben. Die Verteilungssicht wird in diesem Abschnitt vorgestellt, sowie der Aufbau als Client-Server Architektur. Die Kommunikation über die Rest-Schnittstelle und die Absicherung der gesamten Anwendung wird vorgestellt.

### 3.1.1 Verteilungssicht

Verteilungsdiagramme (engl. deployment diagram) zeigen eine Gesamtübersicht über das System, inklusive Clients, Server und relevanter Dritt-Systeme. Die Anwendungen und dessen Komponenten werden konkreten Hardwaremaschinen zugeordnet. Die Kanäle der Kommunikation und Abhängigkeiten untereinander werden visualisiert.

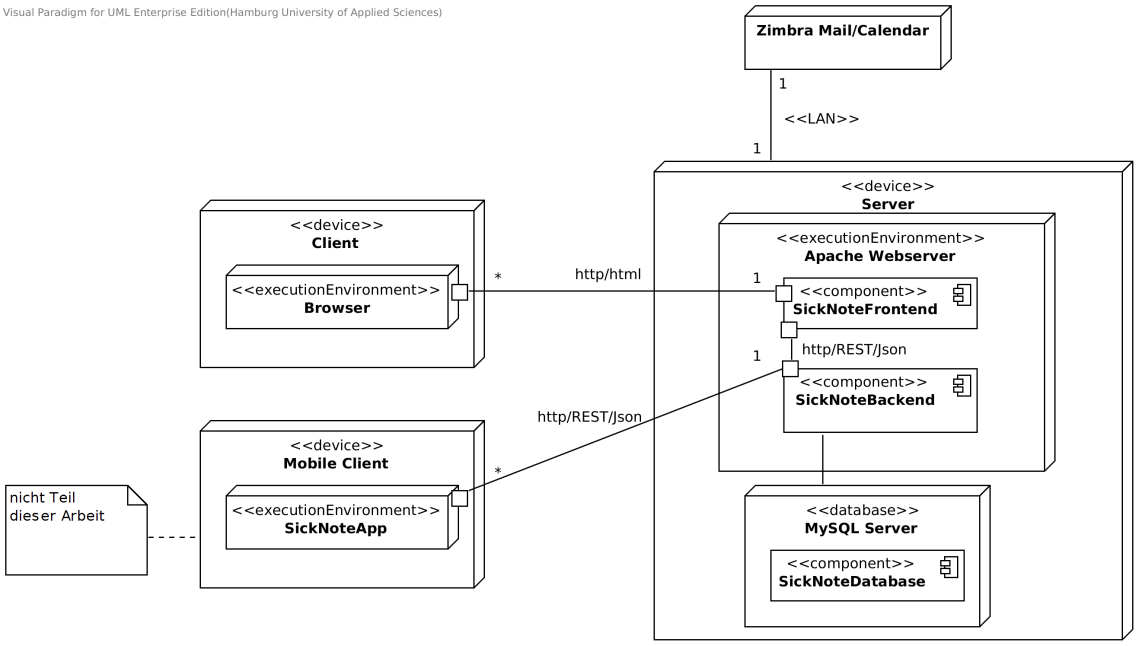


Abbildung 3.1: Verteilungsdiagramm Server und Client

Auf dem Server läuft ein Apache Webserver[1] und ein MySQL Datenbankserver von Oracle[2]. Da diese Server bei einem Großteil der Webprojekte innerhalb der Firma verwendet werden, bietet es sich an diese ebenfalls zu nutzen. Dies hat den Vorteil, dass kein neues Wissen für die Installation, Konfiguration und Wartung erforderlich ist. Da die Server auch im kommerziellen Bereich kostenfrei genutzt werden können ,entstehen dadurch keine zusätzlichen Lizenzgebühren. Kostenpflichtige Lizenzen würden zusätzliche Kosten verursachen, welche in einem Kosten- und Nutzungs-Vergleich keinen Vorteil bringen würden.

Der Server und der Webclient werden von einem Webserver ausgeführt. Eine andere Konfiguration, in der beide Komponenten auf unterschiedlichen Servern laufen, ist technisch möglich. Dies ist aber nicht empfehlenswert, da dadurch weiterer Konfigurations- und Wartungsaufwand für den zweiten Server entsteht. Die zu erwartende Last, ist nicht so enorm, dass diese auf einen separaten Datenbankserver verteilt werden müsste.

Die Aufteilung, auf zwei physikalisch verschiedene Maschinen, wird durch die Kommunikation über HTTP und Rest ermöglicht. Die Schnittstellen sind hier mithilfe von Ports dargestellt.

[1] http://httpd.apache.org/, Abgerufen am 06.08.2014
[2] http://www.mysql.com/, Abgerufen am 06.08.2014

Die Kommunikation über die gemeinsame Schnittstelle ermöglicht es ebenfalls, dass gleichzeitig mehrere Clients existieren können, ohne das Anpassungen am Server gemacht werden müssen. Eine zweite Version ist eine mögliche Mobile App, die nicht Teil dieser Bachelorarbeit ist.

Alle Clients müssen sich an die Richtlinien der Serverschnittstelle halten. Der Server kann theoretisch auch mehrfach existieren, dies ist aktuell aber nicht empfehlenswert, weil dadurch die gleichen Funktionalitäten, bei unterschiedlichen Implementationen, andere Resultate liefern könnten. Dies passiert, wenn innerhalb der Anwendung genutzte Methoden unterschiedlich umgesetzt werden. In Zukunft kann es allerdings Sinn machen, mehrere Server mit der gleichen Implementierung auszuführen. Dadurch kann die Ausfallsicherheit erhöht und eine Lastverteilung erreicht werden. Zurzeit ist dies nicht notwendig, da die Firma noch in einer Größe ist, bei der noch auf die alte Methode, per E-Mail oder Telefon zurückgegriffen werden kann. Jede Instanz des Servers muss mit der gleichen Datenbank arbeiten, damit ein konsistenter Zustand garantiert ist.

Der Server hat eine Verbindung zu dem externen Mail und Kalender Server. Dieser dient dazu, die Benachrichtigungen zu versenden und die Abwesenheit in den Kalender einzutragen.

Der Browser beim Client greift, wie bei anderen Webseiten üblich, mithilfe von HTTP und HTML auf das Webfrontend zu.

### 3.1.2 Client-Server Architektur

Für das Krankmeldungssystem wird eine Client-Server Architektur entwickelt. Abbildung 3.2 zeigt die möglichen Aufteilungen einer verteilten Anwendung. Abbildung a und b werden als Thin-Clients bezeichnet. Thin-Clients beinhalten keinerlei programmspezifische Logik. D und e sind Fat-Clients, sie führen die gesamte programmspezifische Logik aus. Der Server ist nur für die Datenhaltung verantwortlich.

Diese Art der Aufteilung wird Two-Tier Architektur genannt, weil die Anwendung in Client und Server unterteilt ist. Wenn die Datenbank ebenfalls auf einem eigenen Server läuft, handelt es sich um eine Three-Tier Architektur. Bei der gewählten Two-Tier Lösung ist der Datenbankserver exklusiv nur für das Krankmeldungssystem verfügbar.[3] Bei einer Three-Tier ist es einfacher möglich den gleichen Datenbankserver auch anderen Anwendungen mit eigenen Datenbanken zur Verfügung zu stellen.[4]

[3] Dedizierter Server
[4] Shared Server

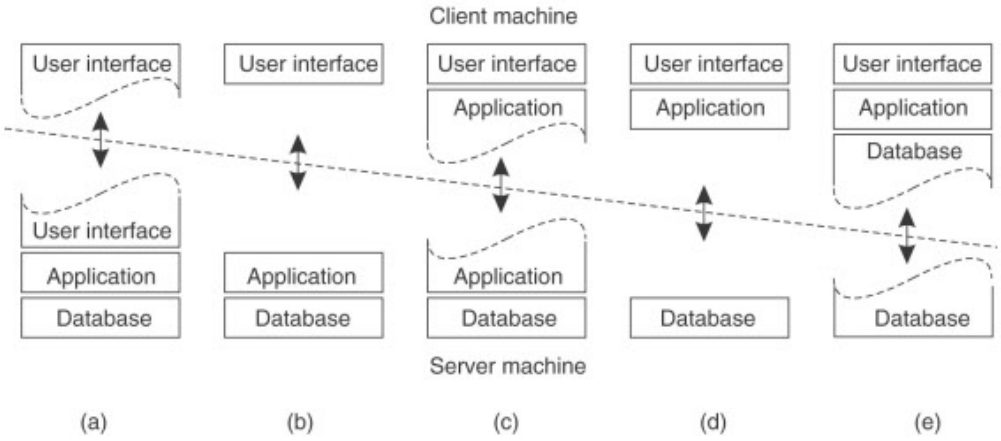


Abbildung 3.2: Client-Server Architektur (Tanenbaum, 2007)

Abbildung a stellt die Aufteilung der Webapplikation dar. Das User-Interface wird zum Teil auf dem Client mit dem Browser erzeugt und zum Teil auf dem Webserver. Der Server beinhaltet in diesem Fall den Webserver mit der Anwendung und den Datenbankserver mit der Datenbank.

Die Aufteilung der zukünftige Mobile App ist in Abbildung b zu sehen. Das User-Interface wird vollständig von der App realisiert und auf dem Client ausgeführt.[5] Der Server bietet nur die Dienste mit den zugehörigen Berechnungen und den Zugriff auf die Daten an. Im Vergleich zur Lösung a wird das benötigte Datenvolumen verringert. Die zur Darstellung erforderlichen Elemente müssen nicht zum Client übertragen werden. Verglichen mit der Webserverlösung (a), werden keine Designelemente mehr übertragen, sondern nur noch die eigentlich relevanten Informationen.

Je nach Implementierung kann die Mobile App wie in Abbildung c umgesetzt werden. Dies ist der Fall, wenn z.B. innerhalb der App bereits geprüft wird ob die Daten gültig sind, bevor sie abgeschickt werden. Die Prüfung wird aber in jedem Fall erneut auf dem Server durchgeführt. Durch die zusätzliche Prüfung, werden mehrfache Übertragungen verhindern, wenn die Daten inkorrekt sind.

Die Abbildungen d und e ergeben für diese Art von Anwendung keinen Sinn. Sie setzen die Anwendung vollständig auf dem Client um. Der Server hält nur die Daten. Bei Abbildung e wird ein Teil der Daten auf dem Client gespeichert. Dies kann Sinnvoll sein, wenn die Anwendung ebenfalls offline arbeiten soll. Sobald der Client wieder online arbeitet, muss er die

[5] Native App - App wurde für eine spezielle Plattform entwickelt

Daten synchronisieren. Mit dieser Lösung ist die Anwendung automatisch hoch skalierbar, da jeder Client alle Berechnungen selber ausführt. Die einzige Schwachstelle, die die Anwendung ausbremsen könnte ist der Zugriff auf die Datenbank.

### 3.1.3 Kommunikationsschnittstelle

Die Kommunikation des Servers mit den Clients geschieht über eine Rest-Schnittstelle. Representational State Transfer (Rest) in ein Architektur-Stil für verteilte Netzwerk Applikationen. Rest verfolgt das Prinzip, dass jede Ressource eindeutig durch eine URL abstrahiert wird. Ressourcen können ein unterschiedliches Format haben. Es kann sich bei ihnen um Bilder aber auch Text handeln. Bei dieser Applikation werden die Ressourcen als Text im Json Format dargestellt. Die Repräsentationen sind unabhängig von den Ressourcen selbst, es kann mehrere verschiedene Repräsentationen für eine Ressource geben (z.B. XML[6] und Json[7]).

Der Zugriff auf die Ressourcen wird bei Rest mithilfe der Standardmethoden von HTTP realisiert. Zu diesen zählen GET, PUT, POST und DELETE (Tabelle 3.1).

| **Methode** | **Verwendungszweck** |
|---|---|
| GET | Lesen einer Ressource, es dürfen keine Veränderungen vorgenommen werden. |
| PUT | Erstellen oder bearbeiten einer Ressource, dessen URL bereits bekannt ist. |
| POST | Erstellen einer neuen Ressource. |
| DELETE | Löschen von Ressourcen. |

Tabelle 3.1: Verwendung der HTTP Methoden (http://www.oio.de/public/xml/rest-webservices.htm)

Die Kommunikation mit einer Rest-Schnittstelle sollte immer zustandslos sein. Die Skalierbarkeit wird durch die Zustandslosigkeit sehr vereinfacht. Der Server muss den Kontext der Anfrage nicht kennen, dies hat zur Folge, dass mehrere Anfragen an unterschiedliche Server gehen können. Gegen dieses Prinzip verstößt diese Anwendung, weil eine PHP Session durch die Authentifizierung aufgebaut wird. Dies könnte behoben werden, indem eine andere Authentifizierungsmethode verwendet wird. HTTP Basic garantiert einen zustandslosen Status.

[6] Extensible Markup Language - flexible Möglichkeit zur Darstellung von hierarchischen Informationen
[7] JavaScript Object Notation - menschlich lesbare Notation im Attribut-Wert Format zur Darstellung von Informationen

Bei dieser Methode wird bei jeder Anfrage der Benutzername und das Passwort mitgesendet. Das HTTP Basic Verfahren ist Sicherheitstechnisch kritisch, da das Passwort nur Base64[8] codiert an den Server übertragen wird. Dies ist genauso unsicher, wie als wenn das Passwort direkt im Klartext übertragen wird. Es ist ausreichend, wenn das kodierte Passwort bekannt ist. Der Klartext ist für eine Anmeldung nicht erforderlich. Bei der HTTP Basic Authentifizierung kommt vom Browser immer ein Login Fenster. Dieses ist nur sehr schwer, oder unmöglich zu unterdrücken. Die Möglichkeit eines Logout existiert bei diesem Verfahren nur über die Möglichkeit die, bei der Anmeldung im Client, gespeicherten Anmeldedaten zu löschen.

Bei dem verwendeten Verfahren handelt es sich um ein Login über ein Formular mit Aufbau einer Session. Der Benutzer gibt seine Userdaten beim Client ein, diese Daten werden direkt per POST an das Backend gesendet und anschließend eine Session aufgebaut. Da abzusehen ist, dass Silpion nicht in dem Umfang und der Geschwindigkeit wächst, welche eine Skalierung für dieses System erfordert reicht die Anmeldung über das Formular mit Aufbau einer Session aus.

Die Kommunikation über HTTP mit dem Standard Port 80 hat den Vorteil, dass dieser bei fast jeder Firewall offen ist. Bei einem anderen Kommunikationsweg kann es zu Schwierigkeiten bei mobilen Clients führen, wenn diese sich in einem Netzwerk befinden, dessen Firewall die Möglichkeiten einschränkt. Für eine bessere Sicherheit der Anwendung und bei der Übertragung der Login Daten, ist es besser eine HTTPS Verbindung zu verwenden. HTTPS läuft standardmäßig über den Port 443, dieser ist ebenfalls über fast jede Firewall erreichbar.

Durch die vollständige Kommunikation über die Rest-Schnittstelle ist die Kopplung der Clients (Frontends) von dem Server (Backend) sehr lose. Der Server benötigt keinerlei Kenntnisse über die Clients.

Im folgenden Abschnitt wird ein kurzer Überblick über die einzelnen Funktionalitäten der Api gegeben. Eine genauere Dokumentation befindet sich im Anhang A. Nach Beendigung dieser Arbeit werden weitere Anfragen hinzukommen, durch PUT Methoden werden zum Beispiel Methoden zum bearbeiten einer Krankmeldung hinzugefügt.

### POST Anfragen

Für folgende Fälle wird mittels POST eine Anfrage gesendet:

- Einstellen einer neuen Krankmeldung.
- Erzeugen einer Verlängerung für eine Krankmeldung.

[8] Base64 kodiert Binärdateien in allgemein lesbare ASCII Zeichen ohne Sonderzeichen

- Hochladen einer AU.
- Login im System.

#### POST Antworten

Bei den POST Anfragen können folgende Antworten kommen:

**HTTP 200 OK** Die Anfrage wurde korrekt und erfolgreich durchgeführt. Beim Upload einer AU wird zusätzlich dessen ID zurückgegeben.

**HTTP 203 Created** Eine neue Krankmeldung oder Verlängerung wurde erzeugt.

**HTTP 400 Bad Request** Die Anfrage war inkorrekt oder unvollständig.

**HTTP 401 Unauthorized** Es ist kein User angemeldet oder die Anmeldung war nicht erfolgreich.

**HTTP 403 Forbidden** Die Anfrage ist unter dem angemeldeten User nicht erlaubt.

#### GET Anfragen

Für folgende Fälle wird eine GET Anfrage gesendet:

- Überprüfung welcher User aktuell angemeldet ist.
- Abfragen der Userdaten, wie Abteilung und Position eines Mitarbeiters.
- Alle User der gesamten Firma oder einer Abteilung ermitteln.
- Krankmeldungen für eine/n Mitarbeiter/in, einer Abteilung oder aller Mitarbeiter/innen abfragen.
- Alle möglichen Typen von Krankmeldungen ermitteln.
- Download einer hochgeladenen AU.
- Abmelden des aktuellen Users.

#### GET Antworten

**HTTP 200 OK** Die Anfrage wurde korrekt und erfolgreich durchgeführt. Mit Ausnahme der AU Download Anfrage werden Antworten als Text im Json Format zurückgegeben. Die AUs werden als PDF zurückgegeben.

**HTTP 404 Not Found** Unter der angefragten Url wurde keine Ressource gefunden.

Wenn die Berechtigungen nicht stimmen, kann wie auch bei den POST Antworten, der Status HTTP 401 oder HTTP 403 zurückgegeben werden.

### 3.1.4 Sicherheit

Eine Absicherung der Rest-Schnittstelle ist zwingend erforderlich, damit zwischen den einzelnen Rollen unterschieden werden kann und Unbefugten der Zugriff auf sensible Daten vorenthalten bleibt.

Die Benutzer melden sich mit ihren Benutzerdaten direkt am Server an. Da der Client die Daten im Klartext an den Server schickt, ist es zwingend erforderlich, dass die Verbindung mithilfe von HTTPS verschlüsselt ist. Um eine verschlüsselte Verbindung zu erstellen, muss zuvor ein Zertifikat für die Webseite erzeugt werden. Innerhalb des Zertifikats befindet sich der öffentliche Schlüssel des Servers. Das Zertifikat versichert, dass der Schlüssel wirklich zu dem angegebenen Inhaber gehört. Bei Aufruf der Seite wird es zum Client geschickt. Durch den Schlüssel kann der Client Daten verschlüsseln, die nur vom Webserver gelesen werden können[9]. Der Client generiert einen Sitzungsschlüssel und überträgt ihn mit dem öffentlichen Schlüssel des Servers verschlüsselt zum Server. Ab hier ist die gesamte Kommunikation symmetrisch[10] verschlüsselt.[11]

Am Anfang wird nur ein selbst generiertes Zertifikat verwendet. Das selbst generierte Zertifikat ist nicht von einer vertrauenswürdigen Stelle generiert worden. Deshalb wird dem Benutzer im Browser eine Warnung, wie in Abbildung 3.3 angezeigt. Sobald die offizielle Testphase der Anwendung beginnt, wird auf ein offiziell für Silpion beglaubigtes Zertifikat umgestellt. Die Warnung im Browser ist dann nicht mehr vorhanden. Das Zertifikat muss von einer Stelle ausgestellt werden, der der Browser vertraut. Diese Stellen werden bei der Installation des Browsers und des Betriebssystems mitgeliefert.[12]

Die umgesetzte Webseite fragt bei jedem Aufruf das Backend, welcher User mit welchem Usernamen angemeldet ist. Wenn kein User angemeldet oder die Session des Users abgelaufen ist, wird ein HTTP Unauthorized Status zurückgegeben. Mit dieser Abfrage wird festgestellt, ob ein User bereits angemeldet ist oder sich noch über den Login authentifizieren muss. Weil die Clientdaten nicht auf dem Client gespeichert werden, ist diese häufige Abfrage notwendig.

[9] Asymmetrische Verschlüsselung - unterschiedliche Schlüssel für Ver- und Entschlüsselung
[10] Symmetrische Verschlüsselung - ein Schlüssel für Ver- und Entschlüsselung
[11] vgl. http://www.softed.de/fachthema/https.aspx, Abgerufen am 09.08.2014
[12] z.B. von der Deutschen Telekom

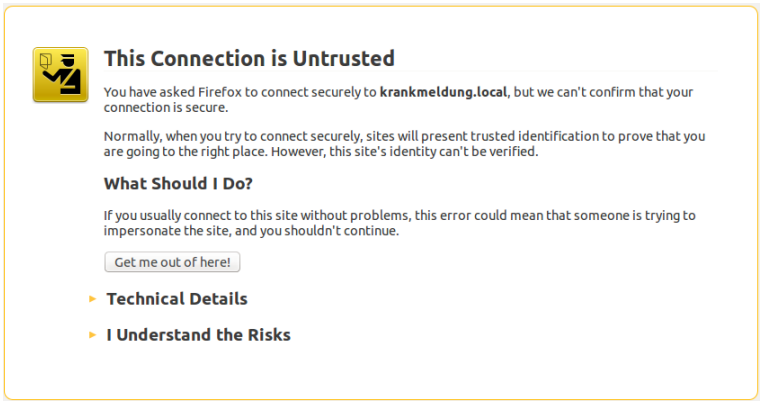


Abbildung 3.3: Firefox Meldung bei einem nicht durch eine vertrauenswürdige Stelle beglaubigtem Zertifikat

Die Daten können nur persistent in einem Cookie gespeichert werden, dabei entsteht aber das Problem, dass der Client erst bei konkreten Anfragen bemerkt, dass die Session abgelaufen ist.

Im nachfolgenden Schritt muss der Client sich die User Daten inklusive dessen Rollen per Anfrage vom Server holen.

## 3.2 Server

Der Server stellt alle benötigten Schnittstellen per Rest-Api für das System zur Verfügung. Er ist für die gesamte Logik und Datenhaltung zuständig. Von ihm werden keine Informationen über die Clients benötigen, dadurch ist er vollständig unabhängig von ihnen.

### 3.2.1 Drei-Schichten Architektur

Bei einer drei Schichten Architektur wird der Server in drei Bereiche aufgeteilt. Den Presentation Layer, Business Layer und in den Data Access Layer. Die einzelnen Schichten dürfen nur von Schichten abhängig sein, die über ihnen liegen. Die Schichten sind in Abbildung 3.4 sichtbar.

Die Unabhängigkeit der Schichten, wird durch dessen Trennung mit den jeweiligen Zugriffsrechten erhöht. Dies hat den Vorteil, dass z.B. die Präsentations Schicht ausgetauscht werden kann, ohne dass die anderen Schichten angepasst werden müssen. Die Entwicklung der unte-

ren Schichten kann unabhängig von oben liegenden Schichten geschehen. Kommunikation geschieht mithilfe von Methodenaufrufen und dessen Rückgabewerten.

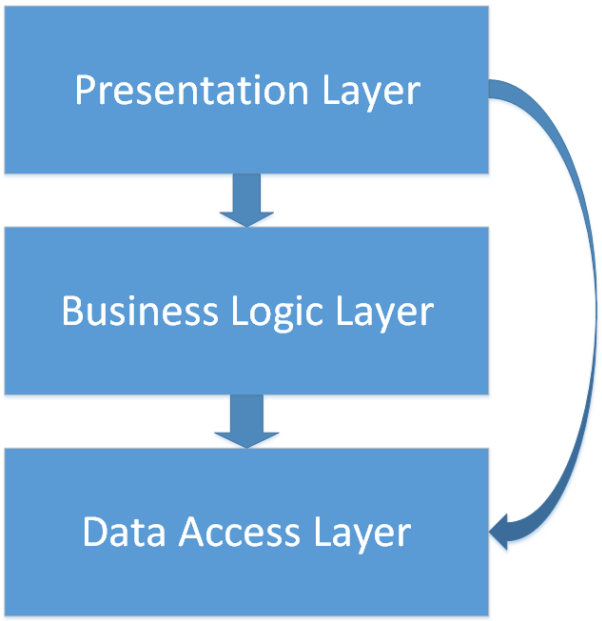


Abbildung 3.4: Drei-Schichten Architektur

Die oberste Schicht ist der Presentation Layer, er stellt in diesem Fall die Rest Schnittstelle dar. In dieser Schicht werden alle Schnittstellen für die Anwendungen zur Verfügung gestellt. In Anwendungen, die die GUI selber implementieren ist diese auf dieser Schicht angesiedelt. Hier werden alle grafischen Komponenten und Steuerelemente zur Interaktion mit dem Benutzer erzeugt. Der Presentation Layer führt keinerlei Berechnungen aus, er ruft lediglich die entsprechenden Methoden aus dem Layer mit der Businesslogik auf.

Die Businesslogik bildet die zweiten Schicht. Innerhalb dieser Schicht werden alle notwendigen Berechnungen durchgeführt. Hier werden alle Geschäftsvorfälle bearbeitet, bei dieser Anwendung sind es z.B. die Erinnerungsmails, die an Personalabteilung gesendet werden müssen. Diese Schicht greift nicht direkt auf die Daten einer Datenbank oder anderer Quellen[13] zu. Für den Zugriff verwendet sie die unterhalb liegende Data-Access Schicht.

Die dritte Schicht ist für den Zugriff und die Haltung der Daten verantwortlich, sie wird Data-Access Schicht genannt. Sie implementiert die Schnittstelle zur Datenbank. Hier werden die Entitäten, ihre Repositories und die Data Transfer Objects zur Verfügung gestellt. Statt

[13] z.B. Webservice

einer Datenbank kann es sich bei der Datenquelle unter anderem auch um einen Zugriff auf einen Webservice handeln, der diese zur Verfügung stellt. Die benötigten Daten werden in ein für die anderen Schichten verwendbares Format, wie z.B. zu Objekten umgewandelt.

### 3.2.2 Komponentendiagramm

In einem Komponentendiagramm wird die Struktur des Systems dargestellt. Die Darstellung beinhaltet die Aufteilung des Systems in die einzelnen Komponenten und deren Schnittstellen. Komponenten können ausgetauscht werden, wenn die neue Komponente die gleichen Schnittstellen anbietet.

In diesem Abschnitt werden die Komponenten des Servers aus Abbildung 3.5 vorgestellt.

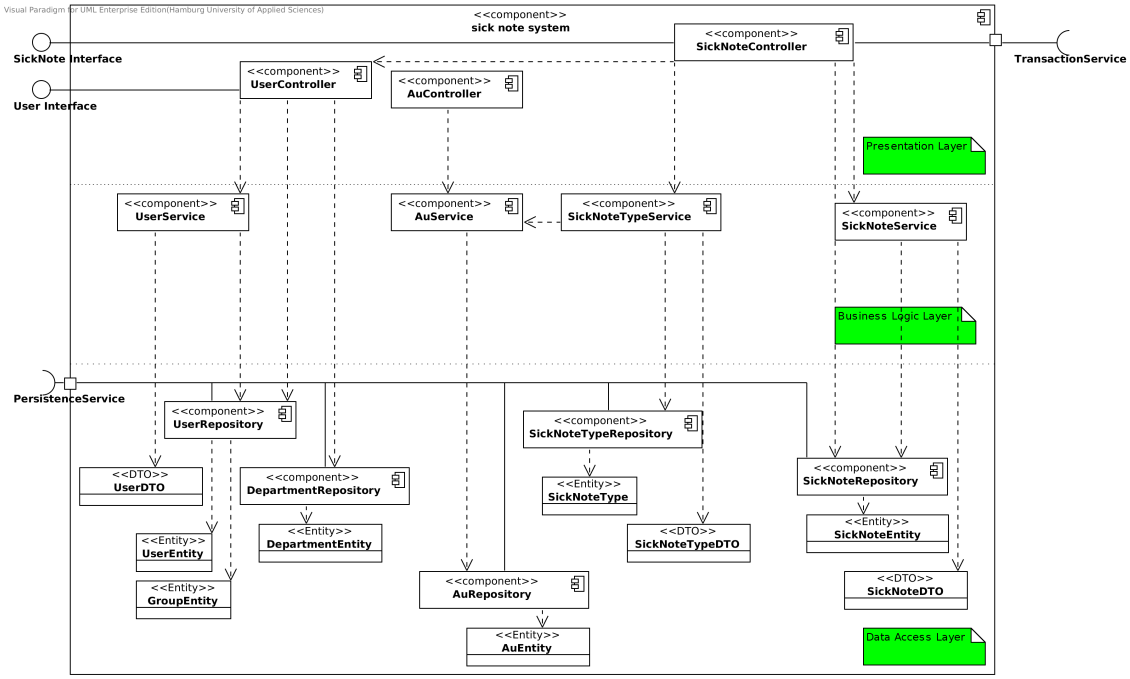


Abbildung 3.5: Komponentendiagramm des Servers

#### Controller

Die Controller beinhalten die Schnittstellen für die Api. Sie dienen als Fassade nach außen. Bei den Controllern werden alle ein und ausgehenden Nachrichten bearbeitet. In ihnen wird

geprüft, ob eingehende Daten syntaktisch korrekt sind und ob der anfragende User berechtigt ist die Anfrage durchzuführen.

**UserController:** Der UserController beinhaltet alle Schnittstellen, um Informationen über Benutzer herauszufinden. Dazu gehört das Abfragen der Informationen, wie Abteilung oder Position in der Firma. Außerdem kann der aktuell authentifizierte Benutzer abgefragten werden. Innerhalb dieses Controllers befinden sich die Schnittstellen um alle Benutzer oder die Benutzer einer Abteilung herauszufinden.

**AuController:** Der AUController beinhaltet die Schnittstellen, um einen Arbeitsunfähigkeitsschein hochzuladen und herunterzuladen.

**SickNoteController:** Dieser Controller beinhaltet die Schnittstellen, um Krankmeldungen und deren Verlängerungen zu erzeugen und zu finden.

#### Services

In den Services befindet sich die eigentliche Geschäftslogik. Hier geschieht auch die Umwandlung der Entitäten in die Data Transfer Objects (siehe Kapitel 3.2.2, Abschnitt DTO) und umgekehrt. Die Services erzeugen mithilfe der Repositories neue Objekte und suchen vorhandene Objekte in der Datenbank. Alle notwendigen Berechnungen passieren in den Services. Services sind von außen nur über die Controller erreichbar.

**UserService:** Der UserService implementiert die Methoden zur Suche von Usern und Abteilungen.

**AuService:** Dieser Service setzt das Erzeugen, Suchen und Löschen von ungenutzten Arbeitsunfähigkeitsscheinen um.

**SickNoteTypeService:** Dieser Service bietet eine Methode zum Finden von allen möglichen Typen einer Krankmeldung mit Bezeichnung und Beschreibung an.

**SickNoteService:** Der SickNoteService realisiert die Methoden des dazugehörigen Interfaces, um Krankmeldungen zu erzeugen, zu verlängern und zu suchen.

#### Repositories

Die Repositories sind für alle Datenbankabfragen zuständig. In ihnen werden die SQL Statements generiert. Andere Objekte sollten aus Gründen der Aufgabentrennung und Wiederverwendbarkeit keine direkten SQL Statements enthalten. Diese sollten auf die Repository-

Methoden zurückgreifen. Durch die zentrale Haltung der Datenbankabfragen wird es einfacher die zugrundeliegende Datenstruktur oder die gesamte Datenbank auszutauschen. Noch weiter vereinfacht dies ein Database Abstraction Layer (DBAL). Dies wird im Abschnitt über das Framework Symfony (Kapitel 3.2.5) genauer erläutert.

Jeder Zugriff auf die Daten erfolgt über ein Repository. Die Daten werden in ihnen gefiltert und zusammengefasst. Sie können auf mehrere Entitäten zugreifen, wenn es notwendig ist.

Sie sind für die sogenannten CRUD Operationen zuständig: Da die grundlegenden Funktionali-

| **Operation** | **Aufgabe** |
|---|---|
| Create | Das Speichern von Daten in der Datenbank. |
| Read | Das Lesen von Daten aus der Datenbank. |
| Update | Das Verändern von Daten in der Datenbank. |
| Delete | Das Löschen von Daten aus der Datenbank. |

Tabelle 3.2: CRUD Operationen

täten der Repositories schon beschrieben sind, werden die einzelnen speziellen Repositories nicht genauer erläutert.

#### Entitäten

Instanzen der Entitäten stellen Objekte der Datenbank dar. In ihnen werden die eigentlichen Informationen gespeichert. Sie enthalten Möglichkeiten zum Bearbeiten und Auslesen der Informationen.

#### Data Transfer Objects

Die Data Transfer Objects (DTOs) sind Objekte die nach außen gegeben werden. In ihnen werden nur Informationen gespeichert, die für das Ziel sinnvoll sind. Außerdem halten sie die Daten nur, sie implementieren keinerlei Verhalten. Dadurch kann die Größe der zu übertragenden Daten minimiert werden.

Sie besitzen nur Methoden zum Lesen, können also nicht verändert werden. Dadurch wird verhindert, dass über gespeicherte Referenzen Daten von Objekten verändern werden können.

Da Objekte oft Referenzen auf anderen Objekte halten, muss entschieden werden, wie diese in einem DTO repräsentiert werden. In dieser Arbeit werden z.B. die Verlängerungen einer Krankmeldung rekursiv innerhalb der Krankmeldungen gespeichert und zurückgegeben.

```
{
   "sickNotes":
      [
         {
            "id":186,
            "user": {"username":"leon", ...},
               "first_date":"2014-07-11",
               "last_date":"2014-07-11",
               ...
               "child":
                  {
                     "id":189,
                     "user": {...},
                     "first_date":"2014-07-11",
                     "last_date":"2014-07-15",
                     ...
                  }
         }
      ]
}
```

Auflistung 3.1: Beispiel Response einer Abfrage nach Krankmeldungen

In der Auflistung 3.1 wurden aus Gründen der Übersichtlichkeit Informationen ausgelassen. Hier ist zu sehen, dass die Verlängerung unterhalb von Child eingehängt wird. Das Child Objekt wird wie eine normale Krankmeldung behandelt, wenn es eine weitere Krankmeldung gibt, wird diese innerhalb des Child Objekt wieder mit der Bezeichnung Child eingehängt.

Bei dieser Lösung ist sichtbar, dass die Abhängigkeiten vollständig mit übertragen werden. Dies kann sehr schnell zu Problemen führen, wenn auf zu viele unterschiedliche Objekte verwiesen wird. Bei zu vielen, ineinander verschachtelten Referenzobjekten wird diese Darstellung schnell unübersichtlich. Auch die Datenmenge, die möglichst niedrig gehalten werden sollte, steigt. Diese Lösung kann schnell zu Problemen führen, wenn versucht wird zirkuläre Abhängigkeiten zu referenzieren. Theoretisch bekommen die Objekte dabei eine unendliche Tiefe, dies führt zu einem Fehler auf dem Server.

In dieser Anwendung besteht das Risiko nicht, da eine Krankmeldung nur drei Referenzen (auf den betreffenden Benutzer, den erzeugenden Benutzer und optional eine Verlängerung) hält. Theoretisch könnte eine Krankmeldung sehr oft verlängert werden, doch bis dies problematisch wird, müssten es schon sehr viele werden.

Folgender Ansatz löst das Referenzierungsproblem: Statt die Daten selbst direkt mitzusenden, wird eine ID oder ein anderer Identifier, der die Ressource eindeutig identifiziert mitgegeben. Die zusätzlich verfügbaren Informationen können so bei Bedarf über weitere Anfragen abgefragt werden.

```
{
   "sickNotes":
      [
         {
            "id":186,
            "user": "leon",
               "first_date":"2014-07-11",
               "last_date":"2014-07-11",
               ...
               "child": 189,
         }
      ]
}
```

Auflistung 3.2: Beispiel Response einer Abfrage nach Krankmeldungen mit IDs

Bei dieser Art der Lösung besteht kein Problem mit zirkulären Abhängigkeiten. Häufig ist es sinnvoll eine Mischung aus beiden Varianten zu verwenden.

In dieser Arbeit wird eine Mischung aus beidem verwendet. Für die gespeicherten AUs wird nur eine ID mit übergeben. Dies bewirkt, dass die AUs nur heruntergeladen werden, wenn dies gewünscht ist.

### 3.2.3 Sequenzdiagramme

Ein Sequenzdiagramm beschreibt einen konkreten Ablauf eines Szenarios. Es werden real vorhandene Operationen verwendet und ausgetauschte Nachrichten mit Beachtung der zeitli-

cher Reihenfolge gezeigt. Mithilfe dieser Art von Diagrammen wird das Zusammenspiel der einzelnen Objekte visualisiert.

Die Erstellung einer neuen Krankmeldung, das Hochladen einer Arbeitsunfähigkeitsbescheinigung und das Abfragen der Krankmeldungen einer Abteilung wird in diesem Abschnitt exemplarisch vorgestellt.

Die interne Vorgehensweise des Clients wird hier nicht dargestellt, da dies von Client zu Client unterschiedlich gehandhabt werden kann.

### Mitarbeiter/in übermittelt neue Krankmeldung an das System

Der Mitarbeiter Fritz wird krank und meldet sich online über das System krank. Er lädt keine AU hoch.

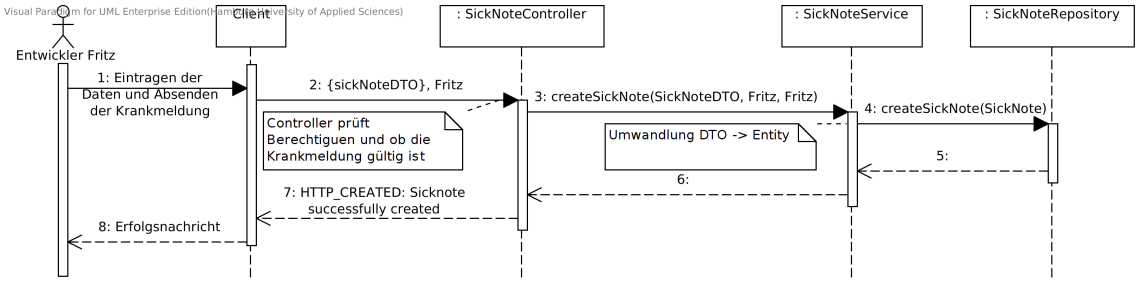


Abbildung 3.6: Sequenzdiagramm: Mitarbeiter/in übermittelt neue Krankmeldung

Abbildung 3.6 zeigt den Ablauf bei der Erstellung einer neuen Krankmeldung.

1. Die Daten werden in dem Client durch den User eingegeben.
2. Der Client übermittelt die Daten per Rest im Json Format an den Server.

   Der betroffene User wird in der Url mitgegeben.
3. Der Controller prüft, ob die eingegebenen Daten korrekt sind und ob der angemeldete User berechtigt ist, für den in der Url angegeben User eine Krankmeldung zu erzeugen. Nach erfolgreicher Prüfung, ob es sich um einen korrekten Zeitraum handelt, wird die Krankmeldung, der betroffene User und der authentifizierte User an den SickNoteService weitergereicht.

4. Der Service wandelt die Krankmeldung in das Entitäts Objekt um und gibt das Objekt an das Repository weiter. Das Repository speichert die Krankmeldung dauerhaft in der Datenbank.
7. Der Kontrollfluss wird zurück an den Controller gereicht.

   Der Controller sendet einen erfolgreichen HTTP Created Status an den Client zurück.
8. Der Client zeigt dem User eine Erfolgsnachricht an.

```
{
   "firstDate": "2014-04-04",
   "lastDate": "2014-04-04",
   "comment": "Termin mit XYZ werde ich absagen",
   "type": "user"
}
```

Auflistung 3.3: Beispiel Request Krankmeldung erstellen

In dem Beispiel Request in Abbildung 3.3 ist sichtbar, wie der Json String für eine neue Krankmeldung aussehen kann.

#### Mitarbeiter/in lädt eine Kopie der AU bei Erstellung der Krankmeldung hoch

Hier wird der Ablauf des Uploads erläutert, der gesamte zusätzlich notwendige Ablauf zur Erstellung einer Krankmeldung ist in Abbildung 3.6 zu sehen. Zwischen den Schritten zwölf und dreizehn werden die Schritte zwei bis sieben aus Abbildung 3.6 ausgeführt. Sie sind aus Gründen der Übersichtlichkeit hier ausgelassen.

1. Die Daten werden in dem Client durch den User eingegeben.
2. Der User öffnet das Fenster zum hochladen einer AU, wählt die Bilder der Vorder- und Rückseite der AU aus und drückt den Upload Button.
3. Der Client schickt die Daten den Server.
4. Der Controller prüft ob die Vorderseite vom richtigen Typ ist.
5. Der Controller prüft ob die Rückseite vom richtigen Typ ist.
6. Der Controller wandelt die beiden Bilder in ein PDF um.
7. Das erzeugte PDF und der aktuell autorisierte User wird an den AuService gegeben.

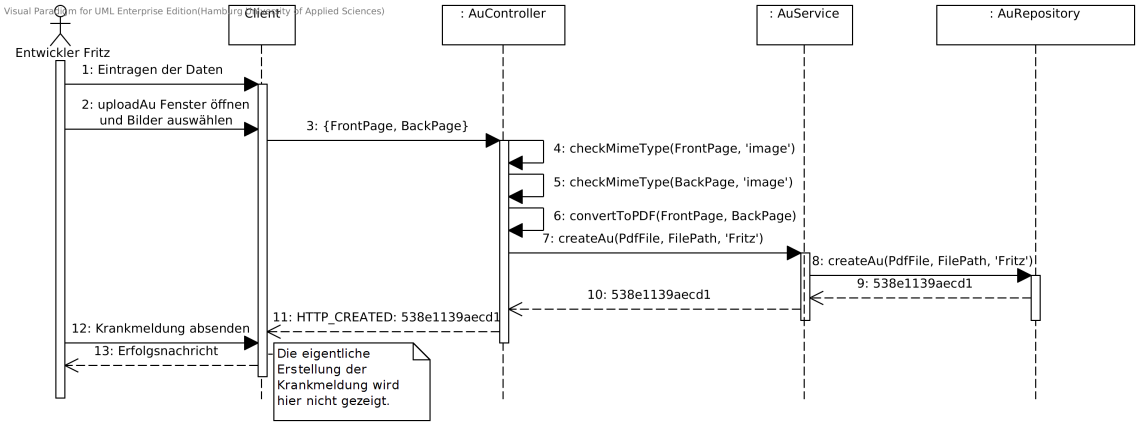


Abbildung 3.7: Sequenzdiagramm: Mitarbeiter/in lädt eine Kopie der AU bei Erstellung der Krankmeldung hoch

8. Der Service reicht die Daten weiter an das Repository.
9. Das Repository generiert eine eindeutige ID, speichert das PDF mit dieser in der Datenbank und gibt die ID zurück.
10. Die ID wird weitergereicht.
11. Der Controller erzeugt eine HTTP Created Nachricht mit der ID als Text.
12. Das gesamte Formular für die neue Krankmeldung wird abgesendet.
13. Der Client zeigt dem User eine Erfolgsnachricht an. Dieser Schritt entspricht dem 8. aus der Abbildung 3.6.

Die dadurch erhaltene ID des PDFs wird mit an den Request zur Erstellung einer neuen Krankmeldung gehängt, wie im folgenden Beispiel zu sehen ist.

```
{
   "firstDate": "2014-04-04",
   "lastDate": "2014-04-04",
   "comment": "Termin mit XYZ werde ich absagen",
   "type": "user",
   "au": "538e1139aecd1"
}
```

Auflistung 3.4: Beispiel Request Krankmeldung mit AU Kopie erstellen

Dadurch, dass die neuen PDFs vor der Erstellung der Krankmeldung in der Datenbank gespeichert werden, kann es passieren, dass nicht verwendete PDFs gespeichert werden. Dieser Fall tritt ein, wenn nach dem Hochladen der Prozess der Erstellung nicht abgeschlossen wird. Um dies zu beheben, gibt es einen Job, der alle nicht zugewiesenen PDFs löscht. Dieser kann bei Bedarf oder als regelmäßiger Cron Job[14] ausgeführt werden.

Das unnötige Generieren von PDFs kann dadurch verhindert werden, dass die Reihenfolge geändert wird. Der Upload einer Krankmeldung ist bei der Änderung erst im zweiten Schritt, nachdem die neue Krankmeldung erzeugt wurde möglich. Bei der Erstellung einer Verlängerung ist dies bereits bedacht. In einer neuen Version sollte diese Änderung in Betracht gezogen werden, der Cron Job wird dadurch überflüssig.

Die Dateien werden aus Sicherheitsgründen direkt in der Datenbank gespeichert und nicht als Datei im Dateisystem. Da die Arbeitsunfähigkeitsbescheinigungen sensible Daten sind, die nicht von jedem eingesehen werden dürfen, ist dies besonders wichtig. Bei der Zuweisung wird geprüft, ob der User, der die AU hochgeladen hat, auch derjenige ist, der versucht die aktuelle Zuweisung durchzuführen. Herunterladen dürfen nur die User selbst, deren Abteilungsleitung und die Buchhaltung.

Eine Möglichkeit, um bereits hochgeladene und zugewiesene AUs zu löschen, existiert derzeit nicht. Eine mögliche Lösung ist, dass dies nur der Personalabteilung erlaubt ist. Die Löschmöglichkeit nur eine gewisse Zeit zuzulassen, z.B. bis 15 Minuten nach dem Upload, ist außerdem denkbar. Eine genaue Regelung muss bei einem Wunsch dieser Funktion vorher innerhalb der Firma abgeklärt werden.

### Abfragen aller Krankmeldungen einer Abteilung für einen speziellen Zeitraum

Die Abteilungsleitung möchte sich alle Krankmeldungen ihrer Abteilung ansehen. Für die Abfrage muss sie den Zeitraum eingrenzen. In diesem Diagramm wird exemplarisch dargestellt, wie die Abfrage einer Liste von Krankmeldungen für einen speziellen Zeitraum abläuft.

1. Die Abteilungsleitung ruft die Seite für die erstellten Krankmeldungen auf.

   Die vorhandenen Filter, wie Zeitraum, stellt er wie gewünscht ein.

2. Der Controller empfängt die Anfrage.

   Er prüft, ob der User die Rolle Buchhaltung hat oder ob er zur Abteilungsleitung der gesuchten Abteilung gehört.

[14] Automatische zeitgesteuerte Aufgabenausführung

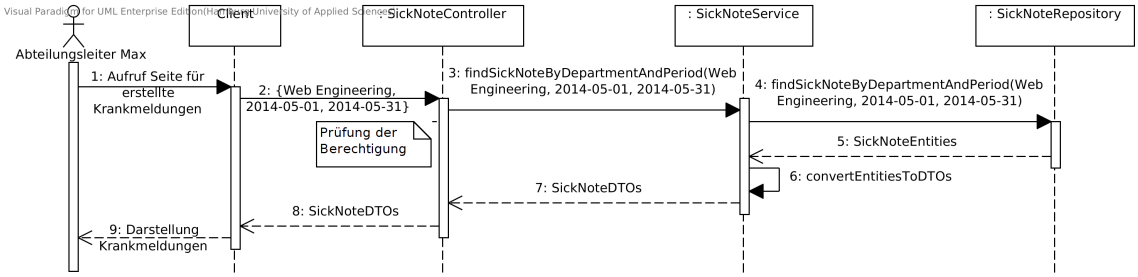


Abbildung 3.8: Sequenzdiagramm: Abteilungsleitung fragt alle Krankmeldungen einer Abteilung für einen Zeitraum ab

3. Der Service leitet die Anfrage an das Repository weiter.
4. Das Repository sucht die Krankmeldungen aus der Datenbank.
5. Das Repository gibt die gefundenen Entitäten zurück.
6. Der Service wandelt die Entitäten in DTOs um.
7. Die DTOs werden an den Controller zurück gegeben.
8. Der Controller wandelt das Ergebnis in einen Json String um und sendet ihn an den Client zurück.

   Ein Beispiel für die Darstellung ist in Abbildung 3.1 zu sehen.
9. Der Client stellt dem User das Ergebnis dar.

### 3.2.4 Datenmodell

In diesem Abschnitt wird das zugrundeliegende Datenmodell für die Datenbank beschrieben. Abbildung 3.9 zeigt das Datenmodell als Entity-Relationship-Modell Diagramm in Chen Notation.

Jeder User kann beliebig viele Krankmeldungen (SickNote) haben. Wenn der Benutzer (User) seit Einführung der Software noch nicht krank war, besitzt er noch keine Krankmeldung. Jeder Mitarbeiter (User) gehört zu einer Abteilung (Department) und hat außerdem eine Position (Group) in der Firma. Zu Anfang gibt es die Positionen Mitarbeitende, Abteilungsleitung und Personalabteilung. User besitzen Attribute für Vorname, Nachname, E-Mail Adresse und einige weitere, diese sind in der Abbildung nicht mit dargestellt.

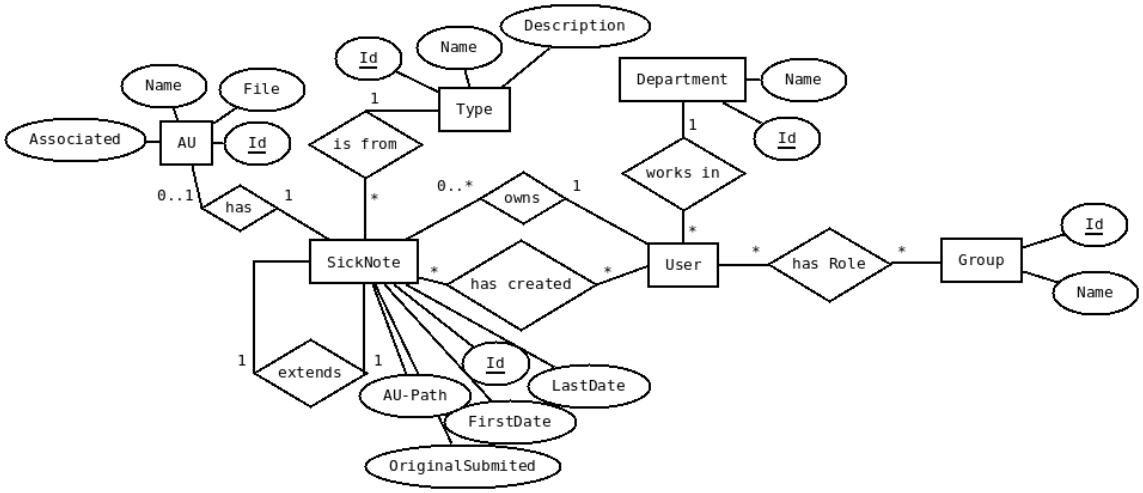


Abbildung 3.9: ER Diagramm nach Chen Notation

Eine Krankmeldung besitzt einen Start- (FirstDate) und End-Zeitpunkt (LastDate). Die AUs werden per FileUpload hochgeladen und in der Datenbank gespeichert. Die Bescheinigung wird nicht als Datei im Dateisystem gespeichert, da dort die Zugangsbeschränkungen sehr aufwendig wären. Jede AU Entität hat ein Attribut Associated, dies wird gesetzt, wenn die AU mit einer Krankmeldung verwendet wird. Dies ist notwendig, da das Hochladen und das Erstellen der AU in zwei unabhängigen Schritten geschieht. So kann es vorkommen, dass nach dem Hochladen der Erstellungsvorgang abgebrochen wird und überflüssig generierte PDFs gespeichert werden.

Krankmeldungen besitzen außerdem ein Attribut, um kennzuzeichnen, ob eine original AU abgegeben wurde (OriginalSubmitted).

Die Verlängerung einer Krankmeldung ist mithilfe einer rekursiven Beziehung der SickNote Entität gelöst (extends). Eine Verlängerung muss immer an die letzte Verlängerung, bzw. an die initiale Krankmeldung gehängt werden, falls bisher keine weitere Verlängerung existiert.

Die Entität Type dient dem Zweck festzulegen, ob eine Krankmeldung den Mitarbeiter selbst oder eine pflegebedürftige Person, für die er verantwortlich ist, betrifft.

Da Krankmeldungen auch stellvertretend erstellt werden können, besitzt ein User zwei Beziehungen zu einer Krankmeldung. Die erste (owns) stellt dar, dass die Meldung den Mitarbeiter betrifft. Die zweite (has created) stellt dar, von wem sie erzeugt wurde.

### 3.2.5 Symfony2

Der Server des Systems wird in PHP mithilfe des Frameworks Symfony2 entwickelt. Symfony2 ist ein PHP Framework für die Webentwicklung.

Symfony1 wurde 2005 unter Verwendung der MIT Lizenz[15] von der Firma SensioLabs[16] veröffentlicht. Seit 2011 ist Version 2 verfügbar. Die Versionen sind nicht kompatibel zueinander, die neue Version ist eine vollständige Neuentwicklung. Schulungen und Support sind gegen Gebühren von SensioLabs möglich. Diese sind aber nicht notwendig, da es eine große Community gibt.

Durch das Framework werden Strukturen vorgegeben, die das System einfacher wartbar und erweiterbar machen. Es stellt allgemein wiederverwendbare Funktionalitäten, wie ein Login Formular zur Verfügung, die von vielen Webdiensten benötigt werden. Symfony setzt sich aus vielen verschiedenen Komponenten zusammen, dazu gehören unter anderem die DependencyInjection[17], das Templating [18] oder der Translator[19]. Jede dieser Komponenten kann ebenfalls ohne das Framework als Bibliothek verwendet werden.

Es arbeitet nach dem Model-View-Controller (MVC) Pattern. Dieses Pattern[20] wird von unterschiedlichen Frameworks meistens unterschiedlich umgesetzt. In Symfony kümmert sich das Modell um die von der Präsentation unabhängige Verwaltung der Daten. Mithilfe der Views werden die Daten dargestellt. Die benötigten Informationen erhalten sie von dem Modell. Die Controller verarbeiten Benutzereingaben, ändern Modelldaten und aktualisieren die entsprechenden Views. Die Views beschränken sich in diesem Fall auf die Erstellung eines Json Strings und den PDF Templates für die Umwandlung der AU Bilder und als Vorlage für die Benachrichtigungs E-Mails. In anderen Fällen kann die vollständige Oberfläche damit designt werden.

Wichtig ist, dass diese Art der Aufteilung das Separation of Concerns Design Prinzip erfüllt.[21]

[15] Erlaubt Verwendung in Projekte mit öffentlichem und nicht öffentlichem Quellcode
[16] http://sensiolabs.com/, Abgerufen am 08.08.2014
[17] Übergeben der Abhängigkeiten, anstatt sie innerhalb der Objekte zu erzeugen
[18] Twig ist die Template Sprache die Symfony verwendet
[19] Übersetzungsmodul für mehrsprachige Webseiten
[20] Allgemein wiederverwendbare Programmierstrukturen, um häufig ähnlich auftretende Probleme zu lösen
[21] klare Aufteilung der Aufgabenbereiche in einzelne Module und Komponenten

### Externe Komponenten

Symfony verwendet viele Open Source Projekte als Teil des Frameworks. Diese sind zum Teil standardmäßig integriert oder können anschließend hinzugefügt werden. Die für dieses Projekt relevantesten Pakete werden hier kurz erläutert.

Der Großteil dieser Komponenten ist auch ohne das Framework einsetzbar.

**Doctrine:**

Doctrine ist ein objektrelationales Abbildungssystem (Object-Relational-Mapping - ORM) um Objekte aus dem Backend persistent in einer relationalen Datenbank zu speichern. Das ORM System verbindet die Einträge der Datenbank mit Objekten der Programmiersprache und wandelt diese automatisch in beide Richtungen um. Mithilfe von Doctrine werden Entitäten definiert. Die Instanz einer Entität repräsentiert einen Datensatz der Datenbank. Die Umwandlung geschieht automatisch.

Außerdem bietet Doctrine eine Datenbankabstraktionsschicht (Database Abstraction Layer - DBAL) , dadurch wird die Kommunikation mit der Datenbank vereinheitlicht. Hiermit wird es möglich die zugrundeliegende Datenbank auszutauschen, ohne dass das System angepasst werden muss. Bei der Implementierung werden keine realen SQL Befehle geschrieben. Mithilfe eines Query Builders wird eine abstrakte Syntax definiert, diese wird von dem System in die Syntax für die konkrete Datenbank umgewandelt.

Das DBAL System kann genutzt werden, um das Schema der Datenbank zu erzeugen. Dazu müssen an den Attributen die entsprechenden Annotationen für die Beziehungen ergänzt werden.[22]

**Swift Mailer:**

Swift Mailer ist eine Bibliothek, die zum Senden von E-Mails verwendet wird.[23]

**PHPUnit:**

PHPUnit wird für Unit Tests verwendet.[24]

**FOSUserBundle:**

[22] http://www.doctrine-project.org/, Abgerufen am 09.08.2014
[23] http://swiftmailer.org/, Abgerufen am 09.08.2014
[24] http://phpunit.de/, Abgerufen am 09.08.2014

Das FOSUserBundle ist nicht standardmäßig in Symfony integriert. Die gesamte Userverwaltung kann hiermit umgesetzt werden. Es bietet individuelle anpassbare User und Gruppen Entities. Von Haus aus existieren Funktionen, um sich mit optionaler Bestätigung per E-Mail zu registrieren und eine Passwort vergessen Anfrage zu stellen. Die User werden automatisch in der Datenbank gespeichert.

Durch automatisch geworfene Events, kann das Verhalten beliebig angepasst werden.[25]

**FOSRestBundle:**

Dieses Paket stellt Tools um eine Rest-Api zu entwickeln zur Verfügung. Zu den Tools zählt unter anderem das automatische Erzeugen von URLs oder die Möglichkeit spezielle Mime Types[26] unterschiedlich zu behandeln. Ein Exception Controller zum versenden von HTTP Statuscodes wird ebenfalls unterstützt. Funktionalitäten um das Format der Antworten zu bestimmen können genutzt werden. Es ist möglich an eine Request Url „.json“ oder „.xml“ anzuhängen um das Format zu bestimmen.[27]

**JMSSecurityExtraBundle:**

Mithilfe von Annotationen können, durch dieses Bundle, Berechtigungen überprüft und Methoden abgesichert werden. Es sendet automatisch die passenden Statusmeldungen an den Anfragenden zurück.[28]

**NelmioApiDocBundle:**

Um die Api später einfach wiederverwenden zu können, ist eine übersichtliche und aktuelle Dokumentation notwendig. Diese kann mithilfe dieses Bundles sehr einfach erstellt werden. Ebenfalls mithilfe von Annotationen, werden die Texte direkt an der Methode erzeugt. Dadurch ist es einfach sie aktuell zu halten, ohne dass in einem externen Dokument Änderungen vorgenommen werden müssen. Die Dokumentation kann mithilfe einer Subroute des Webservers abgerufen und angesehen werden. Sie ist in Form einer Webseite dargestellt. [29]

Im Laufe der Entwicklung haben sich bei dem Bundle ein paar Nachteile herausgestellt. Die Rückgabewerte der einzelnen Methoden lassen sich nicht gut beschreiben. Es ist nur möglich eine Klasse anzugeben, die zurückgegeben wird. Dabei fehlt aber

[25] https://github.com/FriendsOfSymfony/FOSUserBundle, Abgerufen am 16.07.2014
[26] Zur Identifikation des Dateitypes im Internet
[27] https://github.com/FriendsOfSymfony/FOSRestBundle, Abgerufen am 16.07.2014
[28] http://jmsyst.com/bundles/JMSSecurityExtraBundle/master, Abgerufen 17.07.2014
[29] https://github.com/nelmio/NelmioApiDocBundle, Abgerufen am 02.08.2014

die Möglichkeit zu beschreiben, dass es sich z.B. um ein Array mit Objekten handelt. Eigene frei definierte Rückgabewerte lassen sich bisher ebenfalls nicht darstellen. Diese Einwände sind bereits von mehreren Benutzern in der Community angemerkt worden, deswegen kann es gut sein, dass dies in zukünftigen Versionen möglich wird.

## 3.3 Client

Der Client ist vollständig verantwortlich für die Darstellung der Dienste. Bei der Umsetzung muss entschieden werden, welche Dienste der konkrete Client umsetzen soll. Die Vorgaben der Schnittstelle des Servers müssen bei der Umsetzung beachtet werden.

Der hier vorgestellte Client ist ein beispielhafter Client, der als Webservice aufgebaut wird.

Ein anderer Client kann die gleichen Funktionen vollständig anders umsetzen.

### 3.3.1 3 Schichten Architektur

Der Client ist ebenfalls in einer 3 Schichten Architektur aufgebaut. Es handelt sich um ein Model-View-Controller Pattern.

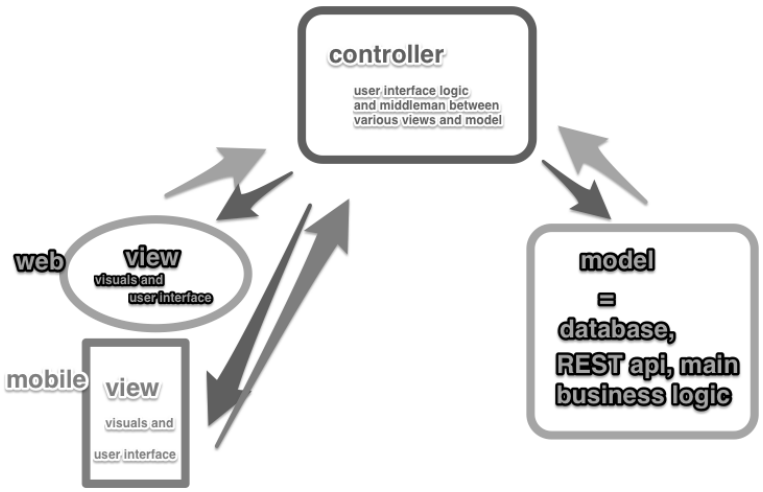


Abbildung 3.10: Model-View-Controller Entwurfsmuster (http://www.angularjstutorial.com/, 2014)

Die Anwendung besitzt eine Schicht für Views, die durch Templates generiert werden. Mit diesen interagiert der User.

Die Controller versorgen die Views mit den Daten, mithilfe der Services.

Services realisieren die Rest Schnittstelle, das bedeutet, dass sie die Model Aufgaben übernehmen.

Dieses Entwurfsmuster teilt die Verantwortlichkeiten von Benutzerschnittstellen in drei Teile auf. Diese Umsetzung erleichtert es dieselben Informationen unterschiedlich zu repräsentieren[30].

### 3.3.2 Komponentendiagramm

Hier wird der Aufbau des Clients dargestellt. Die Länge der Komponenten hat keine Bedeutung, dies dient nur übersichtlicheren Verbindungen.

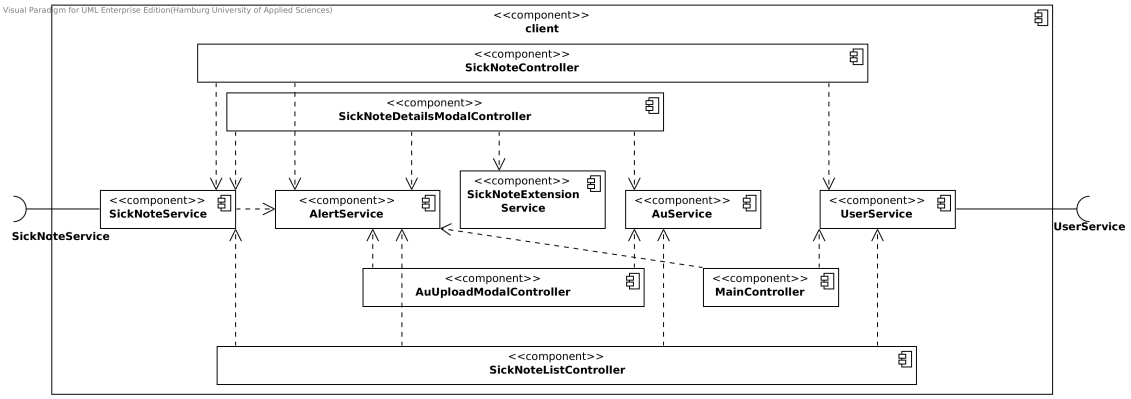


Abbildung 3.11: Komponentendiagramm

#### MainController

Der MainController umgibt die gesamte App. Er stellt die Funktionalitäten für Login und Logout zur Verfügung. Außerdem lädt er die User Daten des angemeldeten User.

[30] vgl. Patterns kompakt, Kapitel 5.6

#### SickNoteController

Der SickNoteController ist für die Erstellung neuer Krankmeldungen zuständig. Er stellt die Funktionalitäten für den Datepicker zur Verfügung. Der Controller sorgt dafür, dass bei Usern mit der Rolle Buchhaltung ein Feld angezeigt wird um direkt den Erhalt der Original AU zu bestätigen. Außerdem können User der Rolle Abteilungsleitung oder Buchhaltung Krankmeldungen stellvertretend für andere User erstellen.

#### SickNoteListController

Der SickNoteListController sorgt für die Darstellung der bereits abgegebenen Krankmeldungen. Die Abteilungsleitung kann sich alle Krankmeldungen der eigenen Abteilung ansehen. Die Buchhaltung kann alle Krankmeldungen einsehen. Hier werden auch die nötigen Filter für die Darstellung erzeugt.

#### AuUploadModalController

Der AuUploadModalController ist dafür zuständig ein Modal Dialog zu öffnen, in welchem es möglich ist ein Bild der Vorder- und Rückseite der AU hochzuladen. Bei einem Modal handelt es sich um ein Fenster, welches sich auf der Webseite öffnet.

#### SickNoteDetailsModalController

Dieser Controller öffnet einen Dialog, in dem die Einzelheiten einer Krankmeldung angezeigt werden. Der SickNoteDetailsModalController ist für die Erstellung von Verlängerungen zuständig. Bei einer Verlängerung kann nur ein Enddatum eintragen werden.

#### SickNoteService

Der SickNoteService stellt alle Rest-Schnittstellen zur Verfügung, um neue Krankmeldungen zu erstellen und abzufragen.

#### SickNoteExtensionService

Dieser Service stellt eine Funktionalität zur Erzeugung einer Verlängerung zur Verfügung.

#### AlertService

Dieser Service ist für die Anzeige von Alert Meldungen zuständig. Er beendet sie, sobald deren Timeout abgelaufen ist.

#### UserService

Der UserService beinhaltet die Rest-Schnittstelle um einen User anzumelden, abzumelden und den aktuell angemeldeten User abzufragen.

#### AuService

Der AuService stellt Funktionalitäten zum Hochladen und Herunterladen einer AU zur Verfügung.

### 3.3.3 Sequenzdiagramme

Die interne Vorgehensweise des Servers wird hier nicht dargestellt (siehe Kapitel 3.2.3). Hier wird nur der Ablauf innerhalb des Clients dargestellt. Die dargestellten Szenarien setzen voraus, dass sich der User bereits auf der Seite befindet und sich korrekt eingeloggt hat.

#### Erstellen einer neuen Krankmeldung

Der Entwickler Fritz meldet sich initial krank.

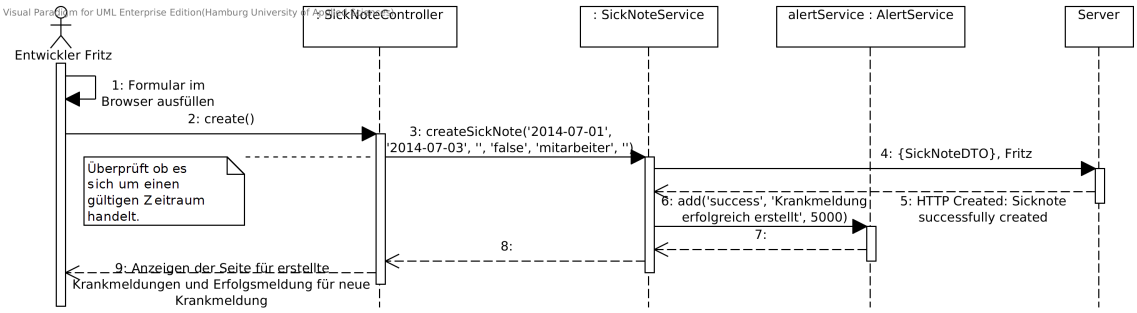


Abbildung 3.12: Komponentendiagramm

1. Der Mitarbeiter ist auf der Seite eingeloggt und gibt die Daten seiner Krankmeldung ein.
2. Das ausgefüllte Formular wird abgeschickt.
3. Der SickNoteController nimmt die Anfrage entgegen, und prüft ob das Enddatum kleiner oder gleich dem Startdatum ist. Der Controller reicht die Anfrage an den SickNoteService weiter.
4. Der Service wandelt die Daten in ein Json String um und schickt sie per Rest an den Server. Die Krankmeldung und der betroffene User werden mitgesendet.
5. Der Server meldet die erfolgreiche Erstellung der Krankmeldung mit einer HTTP Created Response.
6. Aufgrund der Erfolgsmeldung wird mithilfe des AlertServices ein Erfolgs-Alert erstellt, der auf der Seite angezeigt wird.
9. Der Mitarbeiter wird auf die Seite der bereits erzeugten Krankmeldungen umgeleitet und eine Erfolgsmeldung wird angezeigt.

### 3.3.4 AngularJS

AngularJS ist ein Open-Source JavaScript Framework und arbeitet ebenfalls nach dem Model-View-Controller Prinzip. Im Jahr 2009 wurde es von Google unter der MIT Lizenz veröffentlicht. Es ist dafür vorgesehen um Single-Page Anwendungen zu entwickeln. Single-Page Anwendungen bestehen aus einer einzelnen HTML Seite, dessen Inhalte dynamisch nachgeladen werden. Bei einer Änderung wird nicht die komplette Seite aktualisiert, sondern nur die noch benötigten Daten nachgeladen.

Mithilfe von neuen Tags, Attributen und Ausdrücken erweitert AngularJS normale HTML Seiten.

Bei der Entwicklung von AngularJS wurde von Beginn an auf die Testbarkeit geachtet, deswegen ist es einfach und schnell möglich Tests für die WebApp zu schreiben.

Bei AngularJS ist es wie auch bei Symfony möglich weitere Pakete einzubinden.

#### Komponenten

Einen Überblick über die wichtigsten Komponenten ist in Abbildung 3.13 zu sehen. Eine Auswahl der Komponenten wird im folgendem Abschnitt vorgestellt.

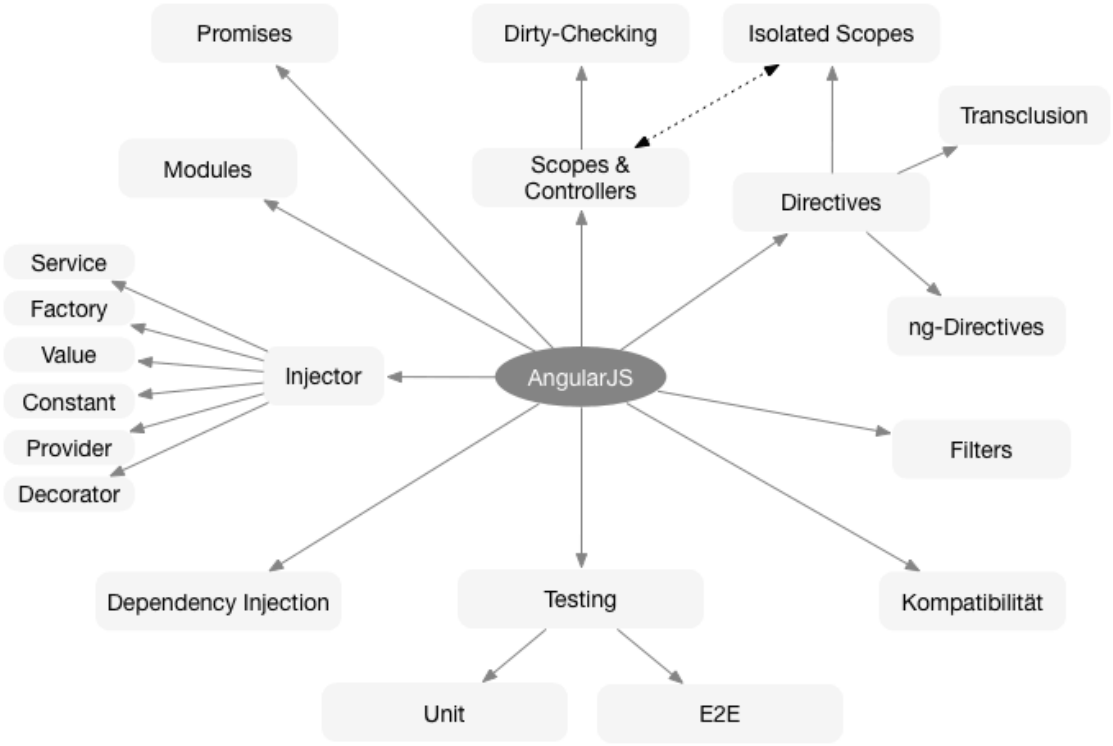


Abbildung 3.13: Komponenten von AngularJS (angularjs.de, 2014)

**Directives:**

Direktiven (engl. directives) werden dazu verwendet um, DOM (Document Object Model) Manipulation vorzunehmen. DOM ist eine Plattform und Sprachen unabhängige Schnittstelle, die Programmen und Skripten dynamischen Zugriff auf die Webseite gestattet. Elemente wie z. B. *ng-model* oder *ng-click* sind Direktiven. Mithilfe von ng-click kann eine Methode eines Controllers ausgeführt werden, wenn darauf geklickt wird. Direktiven erweitern das Standard HTML. Eine spezielle Direktive gestattet es z.B. eine Schleife in dem HTML Dokument auszuführen.

**Scopes:**

Durch die Scopes ist es möglich, das unterschiedliche Programmteile miteinander kommunizieren. Sind sind wie eine globale Variable. Durch sie kommunizieren die Views mit den Controllern. Komplette Methoden können dadurch aufgerufen werden.

**Dependency Injection:**

Bei der Dependency Injection werden alle zusätzlich benötigten Objekte per Parameter übergeben. Dies hat den Vorteil, dass man flexibel bleibt und keine Abhängigkeiten innerhalb der Objekte erzeugt werden. Dadurch können sie bei Bedarf einfacher ausgetauscht werden. Die Dependency Injection ist sehr wichtig, um die einzelnen Komponenten einfacher testbar zu machen.

**Filter:** Mithilfe der Filter (engl. filters) können Daten modifiziert werden. Die Anzeige der Informationen für die User auf der GUI wird mit ihnen beeinflusst. Sie können in den Templates, Controllern und Services verwendet werden. Filter wie z.B. das Umwandeln von Zeichen in Großbuchstaben sind bereits vorhanden, es ist aber auch möglich eigene Filter zu definieren.

**Modules:**

Module (engl. modules) fassen verschiedene Teile der App zusammen. Module können z.B. Controller, Services oder Filter sein.

**Testing:**

AngularJS kann durch Unit und End-To-End (E2E) Tests überprüft werden. Durch die E2E Tests wird die Zusammenarbeit unterschiedlicher Module überprüft. Diese Tests überprüfen vollständige Szenarios. Es handelt sich dabei um Black Box Tests, da die inneren Vorgänge unbekannt sind. Es werden automatisch verschiedene Schritte durchlaufen, die ein Benutzer auch bei Bedienung des Systems ausführen kann.

### Externe Komponenten

AngularJS bietet ebenfalls die Möglichkeit weitere Komponenten von Drittanbietern einzubinden. Folgende Komponenten wurde im Rahmen dieser Arbeit eingesetzt.

**Restangular:**

Diese Erweiterung bietet die Möglichkeit einen Restservice umzusetzen. Restangular vereinfacht die HTTP Rest Anfragen an das Backend. Die Menge des zu schreibenden Quellcodes wird verringert.

Im Vergleich zum integrierten Restservice wird der Aufwand zum Senden und Empfangen von Anfragen und Antworten verringert. Restangular beinhaltet im Gegensatz zum integrierten Service Promises. Promises dienen dazu, asynchrone Anfragen zu behandeln. Sie verzögern die Ausführung eines Codeabschnittes solange, bis eine Antwort eingetroffen ist. Dies ermöglicht es, je nach Antwort Status, entweder eine Fehlerbehandlung oder eine Aktion bei Erfolg durchzuführen.

Diese Erweiterung kann mithilfe von eigenen Methoden ergänzt werden.[31]

**NgTable:**

Mit NgTable ist es möglich Tabellen mit vielen mitgelieferten Funktionalitäten zu erzeugen. So ist es sehr einfach möglich eine Sortierbarkeit umzusetzen. Außerdem wird direkt eine Paginierung[32] und Filter[33] unterstützt.[34]

**Twitter Bootstrap:**

Twitter Bootstrap ist ein Framework für responsive Web-Entwicklung. Es stellt zum Beispiel CSS-Klassen für Glyphicons[35], Positionierung von Objekten, Formulare und Navigationsleisten zur Verfügung. Die Positionierung und die Größen von Objekten werden automatisch, je nach Bildschirmgröße des besuchenden Clients angepasst. Eine breite Navigationsleiste wird ausgeblendet und zu einem manuell öffenbares Dropdown Menü umgewandelt.[36]

Ein responsives Web-Design ist sehr wichtig, damit die Webseite auf Desktop PCs, sowie auf mobilen Geräten ohne Probleme einfach bedient werden kann. Dies kann auch durch mehrere eigenständige Webseiten realisiert werden, diese würden aber erheblich mehr Aufwand bei der Umsetzung und Wartung in Anspruch nehmen.

[31] https://github.com/mgonto/restangular, Abgerufen am 02.08.2014
[32] Aufteilung der Daten auf mehrere Seite für eine bessere Übersichtlichkeit
[33] Anzeige von Daten, die speziellen Kriterien entsprechen
[34] http://bazalt-cms.com/ng-table/, Abgerufen am 02.08.2014
[35] Icons und Symbole
[36] http://getbootstrap.com/2.3.2/, Abgerufen am 02.08.2014

Neben dem Standard Bootstrap von Twitter gibt es eine Version, die mithilfe von AngularJS entwickelt wurde.[37] Beide Versionen ergänzen sich gegenseitig.

[37] http://angular-ui.github.io/bootstrap/, Abgerufen am 02.08.2014

# 4 Realisierung

In diesem Kapitel wird die Umsetzung der Software beschrieben. Um die Entwicklung nicht durch zusätzlichen Aufwand unnötig zu beeinflussen, wird ein geeignetes Entwicklungsvorgehen gewählt.

Für eine Steigerung der Qualität der Software werden Tests verwendet, die genutzten Verfahren werden beschrieben. Neben den notwendigen Komponenten, die für die Ausführung nötig sind, werden hier auch Tools vorgestellt, die die Entwicklung unterstützt haben.

Wie auch beim Design Kapitel, wird vollständig zwischen dem Client und dem Server unterschieden.

## 4.1 Entwicklungsvorgehen

Um eine strukturierte Entwicklung zu gewährleisten, ist ein geeignetes Vorgehen sinnvoll. Damit dieses möglichst effektiv ist, ist ein Verfahren zu wählen, welches verhindert, dass zu viel administrativer Aufwand notwendig ist. Nebensächliche administrative Tätigkeiten, die die eigentliche Entwicklung verlangsamen, sollen minimiert werden. Zu diesen Tätigkeiten gehört zum Beispiel das Schreiben eines Pflichtenheftes, wie es in vielen traditionellen Vorgehen üblich ist.

Da die Entwicklung dieser Anwendung nur durch eine Person durchgeführt wird, ist es notwendig, bekannte Verfahren daran anzupassen. Innerhalb von Silpion, werden die meisten Projekte mit dem agilen Vorgehen Scrum entwickelt. Deshalb bietet es sich an, die Anwendung ebenfalls mithilfe eines agilen Verfahrens zu entwickeln. Mit der Wahl solch eines Vorgehens, kann die Entwicklung flexibler gestaltet werden. Wie eine Anforderung umgesetzt wird, wird erst entschieden, sobald diese wirklich realisiert wird. Die einzelnen Anforderungen müssen dadurch erst genauer betrachtet und vollständig verstanden werden, sobald diese umgesetzt werden. Das Verständnis der einzelnen Bestandteile wird dadurch erheblich erleichtert, da nicht mehr alle Teile auf einmal bekannt sein müssen. Mithilfe der Unterteilung in kleine

Aufgabenpakete, soll sichergestellt werden, dass zu jedem Zeitpunkt ein lauffähiges System vorhanden ist. Die gewünschten Funktionen werden nach und nach hinzugefügt.

Bei Scrum wird die Entwicklungszeit in regelmäßige Zeitabschnitte unterteilt, sogenannte Sprints[1]. Das Vorgehen beinhaltet eine feste Regelung, für eine konkrete Rollenverteilung in Scrum Master, Product Owner und Entwickler. Der Scrum Master ist für die korrekte Verwendung von Scrum verantwortlich. Bei Fragen zum Prozess können die Entwickler sich an ihn richten. Eine weitere wichtige Aufgabe ist es, dem Team bei Problemen, die die eigentliche Arbeit behindern, zu helfen und diese zu lösen. Der Product Owner definiert und priorisiert die einzelnen, zu lösenden Aufgaben. Er ist für das Projekt verantwortlich und hält den Kontakt zum Kunden und anderen, nicht direkt an der Entwicklung beteiligten, Interessensgruppen.[2]

Neben der eindeutigen Rollenverteilung existieren klare Vorschriften für regelmäßige Meetings. Das Daily Scrum Meeting ist ein tägliches, kurzes Meeting der Entwickler, um dem Team bekannt zu machen, was sie als letztes getan haben, was sie vorhaben zu tun und was sie daran hindern könnte. Sprint Review ist ein Meeting, das am Ende jedes Sprints gehalten wird. Die Fortschritte des aktuellen Sprints werden vorgestellt. Während des Sprint aufgetretene Probleme, aber auch gut gelaufene Ereignisse werden besprochen. Weitere Meetings sind das Sprint Planing[3] und die Sprint Retrospektive[4].[5]

Ein Vorgehen wie Scrum ist für diese Arbeit ungeeignet. Die Eigenschaften des Vorgangs müssten stark angepasst werden, dadurch würde das Vorgehen nur noch sehr entfernt Scrum ähneln. Bei der Rollenverteilung kann der Betreuer innerhalb der Firma als Product Owner eingesetzt werden. Für den Scrum Master gibt es keine sinnvolle Person bei dieses Projekt. Das gesamte Entwicklerteam würde ebenfalls nur aus einer Person bestehen.

Da das Team nur aus einer Person besteht, sind die Meetings ebenfalls nicht möglich. Das Daily Scrum macht keinen Sinn, da kein Wissensaustausch stattfindet und nicht täglich an dem Produkt gearbeitet werden kann. Beim Sprint Review müsste der Entwickler sich vor dem Scrum Master und Product Owner für seine geleistete Arbeit rechtfertigen. Es sind keine Personen anwesend, die in der gleichen Position wie er sind.

Wegen der zuvor genannten Gründe müsste Scrum zu sehr angepasst werden, das resultierende Verfahren, wäre ein vollständig anderes. Stattdessen wurde eine angepasste Form von Kanban für diese Arbeit gewählt. Kanban wurde ursprünglich von Toyota entwickelt, um die Effizienz

[1] Dauer von einem Monat oder kürzer
[2] vgl. http://www.agile42.com/en/agile-info-center/scrum-roles/, Abgerufen am 10.08.2014
[3] Die Arbeit während des kommenden Sprints wird festgelegt
[4] Die eigentliche Arbeit wird in Bezug auf Mitarbeitern, Beziehungen, Prozesse und Werkzeugen analysiert
[5] vgl. Scrum Guide, Vorlesung Agile Softwareentwicklung SS2013

beim Bauen von Autos zu erhöhen.[6] Im Vergleich zu anderen Vorgehen existieren weniger Vorschriften, die erfüllt werden müssen. Es ist vorgesehen, dass das Tool an die eigenen Bedürfnisse anpasst wird, um es für das aktuelle Projekt und Team zu optimieren.

Bei Kanban ist es wichtig den Workflow zu visualisieren, die angefangene Arbeit zu begrenzen, den Vorgang zu beobachten und stetig zu verbessern.

Für die Visualisierung wird oft ein Taskboard verwendet, auf diesem werden die noch zu erledigenden Aufgaben, die die sich bereits in Bearbeitung befinden und die abgeschlossenen Aufgaben eingeordnet. Die Felder des Boards sind individuell wählbar und sollten an die einzelnen Abschnitte des Arbeitsablaufes angepasst werden. Felder für aktuell im Test befindliche Aufgaben und Fehlerbehebungen sollten ebenfalls hinzugefügt werden. Im Verlauf dieser Arbeit sind die einzelnen Aufgaben innerhalb der Anwendung Redmine[7] angelegt worden und dessen entsprechender Status gesetzt worden.

Die Anforderungen werden immer weiter aufgeteilt, bis sie schrittweise in einer überschaulichen Größe sind. Dadurch kann genau festgestellt werden, welche Aufgaben bereits erledigt sind und welche nicht. Die Aufteilung hat den Vorteil, dass die Aufgaben einfacher umsetzbar sind und immer ein lauffähiges Produkt vorhanden ist. Bei mehrpersonen Teams unterstützt dies ebenfalls die Aufgabenteilung.

Der Standard Workflow von Kanban sieht vor, die aktuell angefangene Arbeit zu limitieren. Durch die Limitierung soll sichergestellt werden, dass auch ältere, unbeliebtere Aufgaben erledigt werden, bevor eine neue ausgewählt wird. Aufgaben, die bereits in den Workflow aufgenommen sind, sollen vollständig abgearbeitet werden und nicht innerhalb eines Status längere Zeit verweilen. Die Limitierung kann die Konzentration der Entwickler steigern, da diese nicht zu viele zukünftige Aufgaben im Kopf haben, die sie indirekt versuchen mit zu lösen. Dies ist in diesem Projekt nur indirekt geschehen. Da es sich um ein Ein-Mann Projekt handelt, wurde automatisch immer nur eine Aufgabenstellung vollständig abgearbeitet.

Dieses Verfahren muss nur gering an die eigenen Bedürfnisse angepasst werden, dadurch eignet es sich für dieses Projekt gut.

Da es sich bei diesem Projekt um ein Ein-Mann Projekt mit unregelmäßigen Entwicklungszeiten handelt, kann der Entwicklungsprozess nicht in allen Bereichen vollständig festgelegt werden. Bei Projekten unter anderen Gegebenheiten sollten die Abläufe fester definiert sein, damit jedem Team Mitglied klar ist, was dessen Aufgaben sind.

[6] vgl. http://leankit.com/kanban/what-is-kanban/, Abgerufen am 02.07.2014
[7] Kostenloses, Open Source Projekt Management Tool, http://www.redmine.org/

Der schriftliche Teil dieser Bacherlorarbeit wurde ebenfalls nach diesen Prinzipien entwickelt. Die Problembereiche sind einzeln betracht worden, um anschließend nach und nach Lösungen für sie zu finden. Ein Beispiel dafür sind die Sequenzdiagramme (siehe Kapitel 3.2.3 und Kapitel 3.3.3). Diese sind zeitgleich zur Realisierung entstanden.

## 4.2 Server

Dieser Abschnitt beschäftigt sich mit der Entwicklung des Servers. Der Server beinhaltet die gesamte Logik und ist die Schnittstelle zur Datenbank. Er wird auch als Backend bezeichnet. Systeme, die für die Lauffähigkeit der Anwendung zwingend erforderlich sind, aber nicht direkt Bestandteil der Applikation sind, werden vorgestellt. Die Art der Überprüfung der Funktionalitäten und Dependency Injection wird dargestellt.

### 4.2.1 Verwendete Komponenten

Die Komponenten, die hier vorgestellt werden, sind nicht direkt Teil der Anwendung. Der Webserver und SQL Server können leicht ausgetauscht werden. Sie sind für die Ausführung der Anwendung allerdings zwingend erforderlich. PHP, als verwendete Programmiersprache wird vorgestellt.

#### Webserver - Apache2

Der Webserver führt die Anwendung aus. Alle von außen empfangenen Anfragen leitet er an die Anwendung weiter und sendet die Antworten der Anwendung zurück an den Client. Die Kommunikation mit dem Client läuft über HTTP-Nachrichten.

Apache 2.4.7[8] wird als Webserver verwendet. Dies ist die aktuellste Version des Servers, zum Zeitpunkt der Entwicklung.

Dieser Server wurde ausgewählt, weil er auch im kommerziellen Bereich kostenfrei genutzt werden kann. Dadurch entstehen keine zusätzlichen Kosten. Innerhalb der Firma wird der Server für einen Großteil der Projekte eingesetzt. Dadurch ist bereits das notwendige Fachwissen über den Server innerhalb der Firma vorhanden. Dies verkürzt den Aufwand für die Konfiguration und Wartung erheblich. Möglicherweise auftretende Fehler können schneller gefunden und behoben werden.

[8] http://httpd.apache.org/, Abgerufen am 11.08.2014

#### Datenbankserver - MySQL

Der Datenbankserver beinhaltet die Datenbank der Anwendung. In ihr sind alle Krankmeldungen, AUs, Benutzer und sonstige dauerhaft gespeicherte Informationen gesichert.

MySQL wurde zur Realisierung in der Version 5.5.38[9] betrieben. Dieser Datenbankserver wird bei den meisten Projekten mit relationaler Datenbank innerhalb der Firma verwendet.

In einer eingeschränkten Version ist er kommerziell kostenfrei nutzbar. Falls diese Anwendung irgendwann inklusive der MySQL Datenbank verkauft wird, muss eine Lizenzvereinbarung getroffen werden.[10] Durch das integrierte DBAL System (siehe Kapitel 3.2.5), ist es zusätzlich möglich, bei eventuellen Verkäufen, die zugrundeliegende Datenbank auszuwählen und auf einen beim Kunden vorhandenen Datenbankserver zu wechseln.

#### Programmiersprache - PHP

Der Server ist in der Programmiersprache PHP mit der Version 5.5.9 entwickelt worden. PHP ist eine Skriptsprache, sie wird nicht in Maschinencode compiliert. Die Sprache wird zur Erzeugung von dynamischen Webseiten verwendet.[11] Die Ausführung geschieht auf dem Server.

### 4.2.2 Tests

Um Fehler zu verhindern wird das Backend mithilfe von Tests überprüft. Sie tragen im wesentlichen Sinne dazu bei, die Qualität des Codes zu verbessern.

Bei der Realisierung wurde sich auf funktionale Tests beschränkt. Bei dieser Art von Tests handelt es sich um Black Box Tests, da keine Informationen über die innere Abarbeitung vorhanden sein muss. Es werden nur über externe Schnittstellen Anfragen gestellt und Antworten empfangen.

Beispielhaft dafür ist das Testen der Rest-Api. So wird mit Test-Anfragen überprüft, ob das zurückgegebene und erwartete Ergebnis übereinstimmt. Mithilfe der funktionalen Tests werden alle Schichten der Software geprüft, jeder Aufruf wird von den Controllern an einen Service zum Repository bis zur Speicherung, bzw. Abfrage an die Datenbank durchgereicht. Es werden

[9] http://dev.mysql.com/downloads/repo/apt/, Abgerufen am 11.08.2014
[10] vgl. http://www.mysql.de/about/legal/licensing/oem/, Abgerufen am 13.08.2014
[11] vgl. http://www.w3schools.com/php/default.asp, Abgerufen am 13.08.2014

positive Tests und negative Tests mit falschen Berechtigungen oder fehlerhaften Anfragen ausgeführt.

Für diese Tests ist es wichtig, dass bei Beginn immer der gleiche Stand in der Datenbank vorhanden ist, deshalb wird vor jeder Ausführung die vollständige Datenbank geleert und mit vordefinierten Werten befüllt.

Durch diese Tests kann sichergestellt werden, dass bei Änderungen die Funktionalität gleich geblieben ist. Dies ist bei einer Rest Api wichtig, damit alte Clients funktionsfähig bleiben und bei Änderungen nicht angepasst werden müssen.

Zusätzliche Unit Tests sollten hinzugefügt werden, sobald komplexere Berechnungen innerhalb der Anwendungslogik hinzukommen, um sicherzustellen, dass diese in allen Fällen korrekt arbeiten.

### 4.2.3 Dependency Manager

Der Dependency Manger (dt. Abhängigkeits Manger) hilft bei der Installation von Fremdpaketen. Er sorgt dafür, dass alle Pakete installiert werden, von denen das gewünschte Paket abhängig ist. Außerdem sorgt er dafür, dass die richtigen Versionen installiert werden. Durch den Depencency Manager erleichtert sich das Herunterladen, Installieren, Aktualisieren und Entfernen von Paketen erheblich. Die Pakete werden nicht in die Versionsverwaltung geladen, dies hat den Vorteil, dass sie nicht gepflegt werden müssen und keinen unnötigen Speicherplatz auf den Enwicklungsservern benötigen.

Die benötigten Pakete werden nicht global installiert, sondern pro Projekt in einem Unterordner gespeichert. Dadurch entstehen keine Abhängigkeiten zwischen den Projekten und es kann mit verschiedenen Versionen und Konfigurationen gearbeitet werden.

In dieser Arbeit wurde Composer[12] verwendet. Composer ist für das Abhängigkeitsmanagement in PHP entwickelt worden und wurde unter der MIT Lizenz veröffentlicht.

Mithilfe einer Json Datei (composer.json) werden die Abhängigkeiten definiert. Diese Datei sollte mit in die Versionsverwaltung geladen werden, damit alle Projektmitarbeiter/innen mit den gleichen Versionen arbeiten. Das Installieren der Pakete mithilfe einer Datei, die die Pakete sammelt, unterstützt die Zusammenarbeit und das Einrichten eines Projektes auf mehreren Geräten, mit mehreren Personen.

[12] https://getcomposer.org/, Abgerufen am 04.08.2014

```
{
    "require": {
        "phpunit/phpunit": "3.7.*",
        "nelmio/api-doc-bundle": "@stable",
    }
}
```

Auflistung 4.1: Hinzufügen von Paketen zur composer.json

Hier werden zwei Pakete hinzugefügt. Für das Testframework PHPUnit ist eine konkrete Version angegeben. Das zweite Paket hilft bei der Dokumentation der Api. Dieses Paket ermöglicht es die Dokumentation als Webseite mithilfe von Annotationen zu erzeugen. Anhand des zweiten Paketes ist sichtbar, dass nicht die spezielle Versionen angegeben werden muss. Bei dem Api-Doc Bundle wird immer die letzte stabile Version installiert. Für Komponenten, ohne die die Anwendung nicht funktionsfähig ist, sollten immer genaue Versionen angegeben werden, da es durchaus vorkommen kann, dass sich bei größeren Versionssprüngen einige Funktionen ändern oder entfernt werden können.

Innerhalb der composer.json Datei kann eine Konfiguration für das Management angeben werden. Dies können z.B. der Installationspfad für die Pakete oder andere Repositories, die als Quelle dienen sollen, sein.

Über folgenden Befehl werden alle angegebenen Pakete installiert.

```
        php composer.phar install
```

Auflistung 4.2: Installation aller angegebenen Pakete

Als Standard Repository für Composer dient Packagist[13]. Hier sind sehr viele Pakete erhältlich, die mit Composer heruntergeladen und installiert werden können. Auf der Webseite sind die aktuellen Versionen, die Abhängigkeiten und Konflikte zu anderen Paketen sichtbar. Dort ist auch die genaue Bezeichnung zum Einbinden innerhalb von Composer zu finden.

[13] https://packagist.org/, Abgerufen am 15.07.2014

## 4.3 Client

Im Rahmen der Bachelorarbeit wurde ein Client als Webservice entwickelt. Er wird auch als Frontend bezeichnet. Dieser Abschnitt beschäftigt sich mit den in der Entwicklung verwendeten Technologien und Techniken.

Neben dem hier vorgestellten Client kann es beliebig viel andere andere geben, die auf unterschiedliche Art und Weise umgesetzt sind. Die Clients müssen die Schnittstelle zum Server implementieren, damit sie mit dem System kompatibel sind.

Abbildung 4.1 zeigt das Frontend des realisierten Clients um eine Krankmeldung zu erstellen.


Abbildung 4.1: Screenshot: Erstellen einer neuen Krankmeldung

### 4.3.1 Verwendete Komponenten und Tools

Für den Client relevante Komponenten werden hier genannt und wenn nicht bereits im Abschnitt über den Server geschehen, genauer erläutert.

Tools, die die Entwicklung unterstützt haben, werden vorgestellt.

Im Gegensatz zum Server benötigt der Client keine Datenbank. Auf dem Client werden keine Daten persistent gespeichert. Er erhält alle Daten über die Rest-Schnittstelle vom Backend.

### Webserver - Apache2

Die Webapplikation läuft ebenfalls auf dem gleichen Server wie das Backend (siehe Kapitel 4.2.1).

Wenn erforderlich kann er aber auch auf einem anderen Webserver ausgeführt werden (siehe Kapitel 3.1.1). Das Frontend hat keine Kenntnisse darüber, dass es sich auf dem gleichen Webserver wie das Backend befindet. Es greift lediglich über die URL auf ihn zu.

### Programmiersprache - JavaScript

JavaScript ist eine Skriptsprache. Sie wurde für dynamische Webseiten entwickelt und 1995 veröffentlicht. JavaScript ermöglicht es dynamisch Inhalte nachzuladen und das Verhalten von Webseiten zu beeinflussen, dies ermöglich unter anderem eine clientseitige Prüfung von Formularen. Der Quellcode wird vollständig zum Client übertragen und auf ihm ausgeführt. Dies macht schnelle Berechnungen ohne viel Kommunikation möglich. Da der gesamte Quellcode übermittelt wird, ist es wichtig, dass dieser nicht zu groß wird. Die Sprache ist dynamisch typisiert.

### Yeoman

Yeoman[14] ist ein Code Generator. Mit Yeoman ist es möglich neue Projekte aufzusetzen und eine Grundstruktur mit Basis Sourcecode zu generieren.

Bei dem Generieren eines neuen Projektes wird automatisch eine package.json und bower.json Datei angelegt. Diese Dateien dienen NPM (siehe unten) und Bower zur Information, welche Pakete für dieses Projekt benötigt werden. Die Dateien sind mit der composer.json Datei des Dependency Managers des Backends vergleichbar (Kapitel 4.2.3).

Eine weitere, wichtige Datei, die generiert wird, ist Gruntfile.js. Diese beinhaltet ein Set an Aufgaben für Grunt (siehe unten).

Im Laufe der Entwicklung kann die Anwendung dazu genutzt werden, weitere Services, Routen, Filter oder andere Bestandteile von AngularJS zu generieren.

[14] http://yeoman.io/, Abgerufen am 04.08.2014

### Grunt

Grunt[15] ist ein JavaScript Task Runner und Build Tool. Er wird verwendet, um automatisiert Aufgaben durchzuführen. Dies kann die Testausführung, Minifizierung von JavaScript und CSS Dateien [16], Codeanalyse[17] und vieles mehr sein. Neben der Minifizierung der Dateien, kann es auch mehrere Dateien zu einer zusammenfassen.

Mithilfe von Grunt kann ein kleiner Server gestartet werden, der die Entwicklung erleichtert. Die Webseite auf dem Server aktualisiert sich automatisch bei Änderungen im Code. Der Server ist ein Bestandteil von NodeJS (siehe unten).

Durch selbst definierte Aufgaben, innerhalb der Gruntfile.js Datei, können beliebig komplexe Aufgaben durchgeführt werden. Die selbst definierten Aufgaben ermöglichen es, das Tool vollständig auf die eigenen Bedürfnisse anzupassen.

### Bower

Bower[18] ist ein Paketmanager für Frontend-Module. Mit Bower werden zusätzliche JavaScript Bibliotheken, CSS Frameworks oder sonstige Frontend Module installiert. Bower installiert, wie auch Composer beim Server (Kapitel 4.2.3), alle benötigten Abhängigkeiten. Die Pakete werden nicht mit in die Versionskontrolle geladen.

Das Tool bietet die Möglichkeit zwischen Bibliotheken für die Entwicklung und den produktiven Betrieb zu unterscheiden. Dadurch ist es möglich, dass zum Beispiel Tools für Unit Tests bei einem produktiv System ausgeschlossen werden.

Die mit AngularJS verwendeten Pakete, wie Bootstrap, werden mithilfe von Bower installiert.

### NodeJS - Node Package Mangager

NodeJS ist eine Laufzeitumgebung für JavaScript Module außerhalb des Browsers. Sie ist für serverseitiges JavaScript gedacht. Normalerweise wird JavaScript beim Client im Browser ausgeführt. Der Node Package Mangager (NPM)[19] ist ein in NodeJS integrierter Paketmanager für JavaScript Module.

[15] http://gruntjs.com/, Abgerufen am 01.08.2014
[16] Entfernen von nicht relevanten Zeichen, um die zu übertragende Datenmenge zu verringern. Der Code wird dadurch schwer lesbar.
[17] z.B. Testabdeckung, Codestyle, uvm.
[18] http://bower.io/, Abgerufen am 03.08.2014
[19] https://www.npmjs.org/, Abgerufen am 02.08.2014

Die zuvor genannten Tools, Yeoman, Grunt und Bower müssen über NPM installiert werden. Alle Pakete werden pro Projekt installiert und können so in verschiedenen Versionen je Projekt verwendet werden.

Bei einem bereits existierenden Projekt, haben diese Tools den Vorteil, dass nur die Installationsanweisungen „npm install“ und „bower install“ ausgeführt werden müssen, um das Projekt zum Laufen zu bringen. Alle benötigten Pakete und Tools werden automatisch installiert, einzig NodeJS mit NPM muss auf dem Zielrechner vorhanden sein.

### 4.3.2 Tests

Der Client wird durch End-To-End (E2E) Tests überprüft. Diese Tests können ein gesamtes Szenario prüfen. E2E Tests sind ebenfalls BlackBox Tests. Sie überprüfen gleichzeitig das Frontend und die Zusammenarbeit mit dem Backend. Die internen Vorgänge sind dabei unbekannt.

Zur Ausführung dieser Tests wird die Software Protractor[20] verwendet. Protractor ist ein NodeJS[21] Programm. Die Tests werden ebenfalls in JavaScript geschrieben.

In der Abbildung 4.2 ist der Aufbau eines Tests dargestellt. Tests werden in der Jasmin[22] Syntax geschrieben. Die *it* Blöcke definieren einzelne, zu testende, Szenarien. Wenn ein *it* Block fehlschlägt, wird dieser als fehlgeschlagen markiert und mit der Ausführung der anderen Blöcke fortgefahren.

[20] https://docs.angularjs.org/guide/e2e-testing, Abgerufen am 04.08.2014
[21] NodeJS ist eine Laufzeitumgebung für JavaScript Programme
[22] Behavior-Driven development Framework um JavaScript zu testen

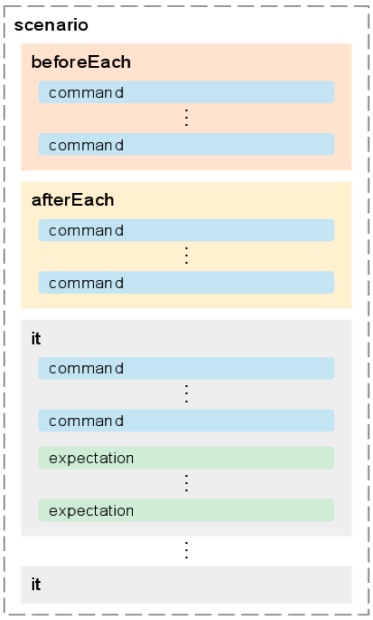


Abbildung 4.2: Aufbau eines E2E Tests in Jasmine Syntax (https://docs.angularjs.org/guide/e2e-testing)

Einen einfachen Login Test zeigt Auflistung 4.3. Vor dem Test wird geprüft, ob der Browser auf die Login Seite umgeleitet wird. Der eigentliche Test identifiziert das Feld für den Benutzernamen und für das Passwort anhand ihrer HTML IDs und trägt die Userdaten ein. Anschließend wird der Login Knopf gedrückt und geprüft, ob auf die Startseite zum Erstellen einer neuen Krankmeldung weitergeleitet wird. Nach Durchführung des Tests wird der Logout geprüft, indem der dazugehörige Button gedrückt und geprüft wird, ob die Login Seite wieder angezeigt wird.

```
describe('Login Test', function () {
    browser.get('http://krankmeldung.local/frontend/app/');
    beforeEach(function () {
        expect(browser.getLocationAbsUrl()).toMatch("/login");
    });
    it('should login the user leon', function () {
        element(by.id('username')).sendKeys('leon');
        element(by.id('password')).sendKeys('123');
        element(by.id('btn-login')).click().then(function () {
            expect(browser.getLocationAbsUrl()).toMatch("http:
                //krankmeldung.local/frontend/app/#/sicknote/
                create");
        });
    });
    afterEach(function () {
        element(by.id('btn-logout')).click().then(function ()
            {
            expect(browser.getLocationAbsUrl()).toMatch("/
                login");
        });
    });
});
```

Auflistung 4.3: Login Test

Die Tests werden auf einem Selenium Server[23] ausgeführt. Selenium kann Tests mit unterschiedlichen Browsern durchführen. Bei den Tests wird der gewünschte Browser geöffnet und sich automatisch durch das erstellte Szenario geklickt. Der Test wird bei der Ausführung an den Selenium Server geschickt, dieser kann auch auf einem anderen Rechner laufen.

Da diese Art von Test, je nach Umfang, sehr lange dauern können ist es nicht möglich sie bei jeder Änderung vollständig durchlaufen zu lassen. Sie sollten aber bei größeren Änderungen und vor der Veröffentlichung einer neuen Version der Anwendung auf jeden Fall ausgeführt werden, um sicherzustellen, dass alle Szenarien korrekt ablaufen. Bei großen Projekt empfiehlt es sich alle Tests in regelmäßigen Zeitabschnitten automatisch auszuführen.

[23] http://docs.seleniumhq.org/, Abgerufen 05.08.2014

Wie auch beim Server, ist ein gleichbleibender Kontext bei den Tests notwendig, deswegen wird bei der Ausführung ebenfalls die Datenbank erneuert.

### 4.3.3 Responsive Webdesign

Heutzutage haben alle Firmen, Institutionen und weitere Organisationen eine Webseite. Um möglichst viele Leute mit diesen Webseiten erreichen zu können, ist es wichtig, dass diese Seiten ebenfalls gut auf mobilen Geräten verwendet werden können.

Responsive Webdesign bezeichnet ein Design, das sich automatisch an die Auflösung des Endgerätes anpasst. Es werden unterschiedliche Designs für ein Desktop Gerät, sowie ein Smartphone und Tablet definiert. Die Darstellungen müssen so angepasst werden, dass sie sowohl mit einer Maus wie auch mit Touchgesten komfortabel bedient werden können.

| **Geräte Typ** | **2012** | **2013** | **2014** | **2017** |
|---|---|---|---|---|
| PC (Desktop PCs und Notebooks) | 341,263 | 315,229 | 302,315 | 271,612 |
| Tablets | 116,113 | 197,202 | 265,731 | 467,951 |
| Smartphones | 1,746,176 | 1,875,774 | 1,949,722 | 2,128,871 |

Tabelle 4.1: Verkauf von Mobilen Geräten (Gartner, 2013)

Tabelle 4.1 zeigt einen Ausschnitt einer Studie von Gartner, die analysiert wie viele mobile Geräte in den letzten Jahren verkauft wurden, inklusive einer Hochrechnung für 2014 und 2017. In der Darstellung ist sichtbar, dass es immer mehr mobile Geräte geben wird. Im Gegensatz zu den mobilen Geräten, geht der Verkauf von Desktop PCs und Notebooks zurück. Durch die Hochrechnungen wird deutlich, dass Webseiten dieser Art immer wichtiger werden.

Es gibt mehrere Möglichkeiten eine Webseite für mehrere Geräte zu optimieren. Eine Möglichkeit ist es mehrere unabhängig voneinander existierende Webseiten zu erstellen. Jede dieser Webseiten ist durch eine eigene URL erreichbar. Bei dieser Lösung entstehen sehr leicht Fehler, weil mehrere Webseiten mit dem gleichen Inhalt gepflegt werden müssen. Updates und Wartungsarbeiten müssen bei jeder Version durchgeführt werden. Dadurch ist das Fehlerpotenzial erheblich erhöht.

Eine Webseite, die die Prinzipien des responsiven Webdesigns befolgt, ist für diese Aufgabe erheblich besser geeignet. Bei responsiven Webseiten gibt es nur noch eine Version der Seite, mit einer URL und nur einer Codebasis. Durch die Prinzipien kann die Webseite für alle aktuellen Auflösungen der Endgeräte und auch für zukünftige optimiert werden.

Die wichtigsten responsiven Design Prinzipien[24]:

1. Statt feste Größen, wie Pixel werden nur relative Größenangaben, wie Prozent verwenden.
2. Bilder und Videos werden je nach Displaygröße des Endgerätes skaliert.
3. CSS Media Querys können verwendet werden, um verschiedene Stylesheets für unterschiedliche Displaygrößen anzuwenden.
4. Nicht relevante Informationen sollen bei kleineren Displaygrößen ausgeblendet werden.
5. Navigationsleisten, wenn erforderlich, sollen ebenfalls für kleinere Displaygrößen optimiert werden.

Die Prinzipien sollen die Lesbarkeit und Bedienbarkeit der Webseiten auf verschiedenen Endgeräten vereinfachen.


Abbildung 4.3: Screenshot: Erstellen einer neuen Krankmeldung auf einem Gerät mit geringer Bildschirmgröße

[24] vgl. http://www.echosurgemarketing.com/key-responsive-web-design-principles/, Abgerufen am 06.08.2014

Abbildung 4.3 zeigt die gleiche Webseite wie Abbildung 4.1. Durch die unterschiedlichen Darstellungsgrößen, wird die Navigationsleiste bearbeitet. Bei Fenstern mit geringer Größe, werden die Menüpunkte ausgeblendet und lassen sich stattdessen mithilfe des Buttons (oben rechts im Screenshot) ein und ausblenden.

Eine Geräte unabhängig gestaltete Webseite benötigt weniger Aufwand als mehrere Webseiten für unterschiedliche Geräte, dadurch kann der Fokus mehr auf die Inhalte als die Entwicklung der Webseiten gesetzt werden. Durch mehrere Webseiten kann es passieren, dass die unterschiedlichen Seiten inkonsistent zueinander werden. Die Aufwände für die Wartung und Pflege werden verringert.

Responsive Webseiten sind teurer und benötigen mehr Zeit in der Entwicklung als auf ein Endgerät angepasste Seiten. Allerdings ist für eine zweite mobile Version keine weitere Codebasis erforderlich. Deshalb ist der Gesamtaufwand geringer, da nur eine Seite entwickelt und gewartet werden muss.

Für aktuelle Webseiten sollte es immer eine mobile Version geben. Der Ansatz der responsiven Prinzipien mithilfe von CSS3[25] eignet sich dafür sehr gut. CSS3 ist der aktuelle Standard von CSS. CSS ermöglicht es die Seite zu stylen.

[25] http://www.w3schools.com/css/css3_intro.asp, Abgerufen am 05.08.2014

# 5 Schluss

Dieses Kapitel fasst die Ergebnisse und gesammelten Erkenntnisse zusammen. Die noch notwendigen Schritte, bis zur Einführung der Anwendung, werden im Ausblick beschrieben. Außerdem werden denkbare Weiterentwicklungsmöglichkeiten vorgestellt.

## 5.1 Zusammenfassung

Diese Arbeit hat sich mit der Entwicklung einer individuellen Softwarelösung, um Krankmeldungen innerhalb des Unternehmens Silpion IT Solutions durchzuführen, beschäftigt.

Der vorhandene, manuelle Vorgang wurde analysiert, dadurch gefundene Schwachstellen wurden hervorgehoben. Ein neuer Vorgang ist aus dem alten entwickelt worden. Die grundlegende Vorgehensweise ist die gleiche geblieben, sie wurde allerdings durch eine Software automatisiert und dadurch ebenfalls vereinfacht.

Anhand des Anforderungsdokuments wurden am Anfang dieser Arbeit die benötigten Funktionen des Systems erarbeitet und in funktionale und nichtfunktionale Anforderungen aufgeteilt.

Das Analyse Kapitel hat den aktuellen Zustand analysiert und mit dem Soll-Zustand verglichen. Die Anforderungen an die neue Software wurden erfasst und analysiert. Bereits existierende Softwarelösung wurden untersucht und bewertet, wie gut sie innerhalb der Firma eingesetzt werden können.

Im Laufe dieser Arbeit wurde ein Softwaredesign für die Realisierung der Anwendung entwickelt. Das Konzept geht auf die strikte Aufteilung des Systems in Client und Server ein, sowie deren Kommunikation mithilfe einer Rest-Schnittstelle. Bis auf die Kommunikationsschnittstelle wurden der Client und der Server separat entwickelt. Der Client und der Server sind in Komponenten aufgeteilt worden und der interne Ablauf beispielhafter Szenarien wurden mithilfe von Sequenzdiagrammen gezeigt. Die verwendeten Frameworks wurden vorgestellt. Ein relationales Datenmodell für den Server ist entstanden.

Für die Umsetzung relevante Werkzeuge, wie der Dependency Manager oder die Möglichkeit Webseiten zu gestalten, die sowohl auf Desktop PCs, wie auch mobilen Clients mit kleinen Displays gut bedienbar sind, wurden gefunden. Unterschiedliche Arten zum Testen, mithilfe von behavior und funktionalen Tests, wurden beschrieben.

## 5.2 Fazit

Im Rahmen dieser Arbeit ist ein System entstanden, welches einen Prozess zur Krankmeldung zur Verfügung stellt. Das System führt durch den Prozess, sodass keine Schritte übersehen werden können. Die zentrale Haltung der Daten hat den Vorteil, dass die Informationen allen berechtigten Personen schnell zur Verfügung stehen, ohne in manuell geführten Listen und alten E-Mails suchen zu müssen. Fragen der Krankenkassen oder des Steuerberaters können dadurch schneller beantwortet werden. Ob die allgemein gewünschte Steigerung der Effizienz wirklich merkbar ist, muss sich im Laufe der Verwendung zeigen.

Trotz der Führung durch den Ablauf, können weiterhin Fehler entstehen, da wie zuvor ein manuelles Handeln der Abteilungsleitung notwendig ist. Deshalb kann es immer noch zu menschlichen Fehlern führen, wenn die entsprechenden Personen die Benachrichtigungen nicht rechtzeitig bearbeiten. Falls Benachrichtigungen nur innerhalb des Systems angezeigt werden, ist es erforderlich, dass regelmäßig nachgesehen werden muss, ob neue Ereignisse vorliegen. Dies ist noch aufwendiger, als auf entsprechende E-Mails zu reagieren. Da die Abteilungsleitung auch noch bei weiteren Systemen, wie der Urlaubsplanung nachsehen müssen, ist wahrscheinlich, dass dies an ereignisreichen Tagen schnell vernachlässigt wird.

Bei Benachrichtigungen per E-Mail werden die betroffenen Personen automatisch darauf aufmerksam gemacht, dass ein Handeln von ihnen erforderlich ist. Um eine zeitnahe Bearbeitung zu erwarten, müssten die Benachrichtigungen vom Smartphone empfangen werden. Es kann und sollte allerdings nicht davon ausgegangen werden, dass alle Mitarbeiter ihre beruflichen Mails auch privat empfangen. Das Problem der massenhaft empfangenen E-Mails wird dadurch ebenfalls nicht gelöst, dadurch können sie weiterhin übersehen werden.

## 5.3 Ausblick

Dieser Abschnitt beschreibt noch offene Fragen und die nächsten Schritte im Anschluss dieser Arbeit.

Die Entwicklung dieser Arbeit hat gezeigt, dass ein neuer, softwaregestützter Vorgang sinnvoll ist. Es ist allerdings nicht untersucht worden, inwieweit der neue Vorgang den Prozess wirklich vereinfacht und optimiert. Der gesamte Vorgang sollte nach Ablauf einer festgelegten Zeitspanne kritisch betrachtet werden und eventuell erneut angepasst werden.

Die Arbeit behandelt hauptsächlich die Erfassung von neuen Krankmeldungen, die interne Weiterverarbeitung in der Abteilungsleitung und der Personalabteilung wird nur am Rande betrachtet. Eine Untersuchung, wie die interne Weiterverarbeitung abläuft, ist sinnvoll, um diese ebenfalls zu optimieren und den entstandenen Vorgang daran anzupassen und zu erweitern.

Die Anwendung ist für einen produktiven Einsatz bei Silpion gedacht. Bevor die Anwendung eingesetzt werden kann, muss sie einen Testlauf überstehen. Dazu wird die Software vorerst nur innerhalb einer Abteilung im Rahmen einer Testphase eingeführt. Auftretende Fehler und kommende Änderungswünsche, während dieser Phase, werden ausgebessert und eingepflegt. Sobald die Änderungen erneut getestet wurden, wird die Anwendung nach und nach für weitere Abteilungen freigeschaltet, bis die alte Vorgehensweise vollständig ersetzt wird.

Damit eine aussagefähige Testphase beginnen kann, müssen noch fehlende Anforderungen umgesetzt werden. Dazu gehören unter anderem die Benachrichtigung der User, per E-Mail und innerhalb des Systems. Eine Möglichkeit um Krankmeldungen zu bearbeiten, wie z.B. um den Erhalt der original AU zu bestätigen, muss noch umgesetzt werden. Für die Bearbeitung des Zeitraumes, sowie der hochgeladenen AUs müssen zuvor genaue Regelungen beschlossen werden, unter welchen Umständen dies von welchen Personengruppen erlaubt ist.

Das Herunterladen einer Zusammenfassung von Krankmeldungen im PDF Format ist ebenfalls noch in der Planung. Die PDFs sollen bei Bedarf an die Krankenkassen oder dem Steuerberater weitergereicht werden. Da sich die realen Benutzer der Testgruppe anmelden müssen, muss die Anmeldung auf das existierende LDAP System umgestellt werden. Für die Umstellung auf das LDAP System ist eine gesicherte Verbindung erforderlich. Die vollständige Kommunikation zwischen den Benutzern, den Clients und dem Server werden auf HTTPS, mit einem für Silpion signiertem Zertifikat, umgestellt. Dies sichert die Mitarbeiter vor Phisingangriffen ab, weil sie dadurch sicher sein können, dass diese Webseite wirklich zu Silpion gehört.

Im Laufe der Testphase oder im anschließenden produktiven Betrieb ist es sehr wahrscheinlich, dass der Wunsch nach weiteren Funktionalitäten entsteht. Diese werden je nach Relevanz und Aufwand anschließend umgesetzt. Statistiken, unterstützt von übersichtlichen Grafiken, können sinnvoll sein, um den Verlauf von Krankmeldungen über einen Zeitraum zu erkennen. Bei jeder Art von Statistiken oder Auswertungen, ist die Frage offen, inwieweit rechtliche Vorgaben beachtet werden müssen. Da es hierbei um personenbezogene Daten geht, muss

eindeutig geklärt werden, wer welche Auswertungen sehen darf und welche firmenintern veröffentlicht werden dürfen. Die Möglichkeit bei einer Krankmeldung optional weitere Personen per E-Mail zu informieren ist ebenfalls nützlich. Mitarbeiter, die auswärtig arbeiten, können Personen vor Ort so direkt informieren.

Die Rest-Schnittstelle bereitet die Anwendung auf eine Interaktion mit anderen Systemen vor. Durch eine Verbindung zum Urlaubstool, kann eine entsprechende Aktion gestartet werden, wenn jemand innerhalb eines geplanten Urlaubes krank wird. Eine Verbindung zum Tool für die Ressourcenplanung ist ebenfalls denkbar, dadurch kann verhindert werden, dass Mitarbeiter für Projekte eingeplant werden, obwohl sie zu dem Zeitpunkt nicht arbeitsfähig sind.

Die Unterstützung von weiteren Clients ist durch die allgemeine Kommunikationsschnittstelle ebenfalls gegeben. Bei allen weiteren Anwendungen, die für die interne Verwendung bei Silpion vorgesehen sind, wird auf dieses Feature ebenfalls geachtet und von vornherein mit entwickelt. Deswegen ist es wahrscheinlich, dass eine mobile App für Smartphones entwickelt wird. So kann eine allgemeine Silpion App entstehen, die mehrere Dienste miteinander vereint.

# Literaturverzeichnis


[packagist 2014] *Packagist*. https://packagist.org/. 2014. – Abgerufen am 15.07.2014

[Adermann und Boggiano 2014] Adermann, Nils ; Boggiano, Jordi: *Composer Dokumentation*. https://getcomposer.org/doc/00-intro.md. 2014. – Abgerufen am 15.07.2014

[Agile42 2014] Agile42: *Scrum Roles*. http://www.agile42.com/en/agile-info-center/scrum-roles/. 2014. – Abgerufen am 10.08.2014

[AngularJS 2014a] AngularJS: *AngularJS Blog*. http://angularjs.de/. 2014. – Abgerufen am 26.04.2014

[AngularJS 2014b] AngularJS: *AngularJS Github Wiki*. https://github.com/angular/angular.js/wiki. 2014. – Abgerufen am 26.04.2014

[Apache 2014] Apache: *Apache Webserver*. http://httpd.apache.org/. 2014. – Abgerufen am 06.08.2014

[Arbeitsrecht.org 2014] Arbeitsrecht.org: *Voraussetzungen für die entgeltliche Freistellung zur Pflege von kranken Kindern*. http://www.arbeitsrecht.org/arbeitnehmer/krankheit/voraussetzungen-fuer-die-entgeltliche-freistellung-zur-pflege-von-kr 2014. – Abgerufen am 24.07.2014

[Bergmann 2014] Bergmann, Sebastian: *PHPUnit*. http://phpunit.de/. 2014. – Abgerufen am 09.08.2014

[Bower 2014] Bower: *Bower*. http://bower.io/. 2014. – Abgerufen am 03.08.2014

[Bégaudeau 2013] Bégaudeau, Stéphane: *Everything you need to understand to start with AngularJS*. http://

stephanebegaudeau.tumblr.com/post/48776908163/everything-you-need-to-understand-to-start-with. 2013. – Abgerufen am 18.07.2014

[Corbyn 2009] Corbyn, Chris: *Swiftmailer*. http://swiftmailer.org/. 2009. – Abgerufen am 09.08.2014

[Doctrine 2014] Doctrine: *Doctrine*. http://www.doctrine-project.org. 2014. – Abgerufen am 09.08.2014

[Egham 2013] Egham: *Gartner Says Worldwide PC, Tablet and Mobile Phone Combined Shipments to Reach 2.4 Billion Units in 2013*. http://www.gartner.com/newsroom/id/2408515. 2013. – Abgerufen am 20.07.2014

[Eilebrecht und Starke 2014] Eilebrecht, Karl ; Starke, Gernot: *Patterns kompakt - Entwurfsmuster für effektive Software-Entwicklung*. 4. Auflage. Springer Verlag, 2014. – ISBN 978-3-642-34717-7

[Elsner 2014] Elsner, Herbert: *ESIS - Elektronisches Schüler Informations System*. http://www.esis.de/krankmeldung.html. 2014. – Abgerufen am 26.03.2014

[Esposito 2014] Esposito, Dino: *Pros and Cons of Data Transfer Objects*. http://msdn.microsoft.com/en-us/magazine/ee236638.aspx. 2014. – Abgerufen am 15.07.2014

[FriendsOfSymfony 2014a] FriendsOfSymfony: *FOSRestBundle*. http://knpbundles.com/FriendsOfSymfony/FOSRestBundle. 2014. – Abgerufen am 26.05.2014

[FriendsOfSymfony 2014b] FriendsOfSymfony: *FOSUserBundle*. http://knpbundles.com/FriendsOfSymfony/FOSUserBundle. 2014. – Abgerufen am 26.05.2014

[Förstermann 2014] Förstermann, Michael: *Blaumachen oder krank zur Arbeit*. https://www.ikk-classic.de/presse/pressemitteilungen/bundesweit/aktuell/19032014-blaumachen-oder-krank-zur-arbeit.html. 2014. – Abgerufen am 29.07.2014

[Gontovnikas 2014] Gontovnikas, Martin: *Restangular*. https://github.com/mgonto/restangular. 2014. – Abgerufen am 02.08.2014

[Groll 2014] Groll, Tina: *Immer mehr Berufstätige sind lange krank*. http://www.zeit.de/karriere/beruf/2014-04/infografik-zunahme-fehltage-arbeitslosenquote. 2014. – Abgerufen am 29.07.2014

[Göschka 2012] Göschka: *TU Wien:Verteilte Systeme VO (Göschka)/Prüfung 2012-10-08*. https://vowi.fsinf.at/wiki/TU_Wien:Verteilte_Systeme_VO_(G%C3%B6schka)/Pr%C3%BCfung_2012-10-08. 2012. – Abgerufen am 17.07.2014

[HCM 2014] HCM: *HCM Fehlzeitenverwaltung*. http://hcm-infosys.com/vdoc/easysite/hcm/losungen/urlaub-krank-dienstreise. 2014. – Abgerufen am 26.03.2014

[Joyent 2014a] Joyent, Inc: *NodeJs*. http://nodejs.org/. 2014. – Abgerufen am 02.08.2014

[Joyent 2014b] Joyent, Inc: *NodeJs*. 2014

[Lindley 2013] Lindley, Cody: *Package Managers: An Introductory Guide For The Uninitiated Front-End Developer*. http://tech.pro/tutorial/1190/package-managers-an-introductory-guide-for-the-uninitiated-front-end-what_is_a_package_manager. 2013. – Abgerufen am 21.07.2014

[Mohorovicic 2013] Mohorovicic, S.: *Implementing Responsive Web Design for Enhanced Web Presence*. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6596440. 2013. – Abgerufen am 10.03.2014

[Munsch 2013] Munsch, John: *Yeoman, Grunt, and Bower*. http://johnmunsch.com/2013/07/07/yeoman-grunt-and-bower/. 2013. – Abgerufen am 13.08.2014

[Nelmio 2014a] Nelmio: *NelmioApiDocBundle*. http://knpbundles.com/nelmio/NelmioApiDocBundle. 2014. – Abgerufen am 26.05.2014

[Nelmio 2014b] Nelmio: *NelmioApiDocBundle*. https://github.com/nelmio/NelmioApiDocBundle. 2014. – Abgerufen am 08.08.2014

[Oracle 2014] Oracle: *MySQL Datenbankserver*. http://www.mysql.com/. 2014. – Abgerufen am 06.08.2014

[Potencier 2011] Potencier, Fabien: *What is Symfony2?* http://fabien.potencier.org/article/49/what-is-symfony2. 2011. – Abgerufen am 18.07.2014

[SAP 2014a] SAP: *Personalzeitwirtschaft (PT).* http://help.sap.com/saphelp_470/helpdata/de/8a/98459c46c411d189470000e829fbbd/content.htm. 2014. – Abgerufen am 27.07.2014

[SAP 2014b] SAP: *SAP Hilfe - Funktionen zur Konfliktlösung.* http://help.sap.com/saphelp_erp60_sp/helpdata/de/99/78dfde90f44ecca9c06b196e88fa36/content.htm. 2014. – Abgerufen am 26.03.2014

[Savchuk 2014] Savchuk, Vitalii: *ngTable.* http://bazalt-cms.com/ng-table/. 2014. – Abgerufen am 02.08.2014

[Schmitt 2014] Schmitt, Johannes: *JMSSecurityExtraBundle.* http://jmsyst.com/bundles/JMSSecurityExtraBundle/master,. 2014. – Abgerufen am 17.07.2014

[Schwaber und Sutherland 2011] Schwaber, Ken ; Sutherland, Jeff: *Scrum Guide - Der gültige Leitfaden für Scrum: Die Spielregeln.* Vorlesung Agile Softwareentwicklung SS2013, HAW Hamburg. 2011. – Abgerufen am 10.08.2014

[SensioLabs 2014] SensioLabs: *Why should I use a framework?* http://symfony.com/why-use-a-framework. 2014. – Abgerufen am 18.07.2014

[Siemann 2012] Siemann, Christiane: *Wie das Web 2.0 Machtstrukturen ändert.* http://www.sueddeutsche.de/karriere/social-media-in-unternehmen-wie-das-web-machtstrukturen-aendert-1.1424446. 2012. – Abgerufen am 23.07.2014

[Silver 2009] Silver, Bruce: *BPMN Method and Style.* 1. Auflage. Cody-Cassidy Press, 2009. – ISBN 978-0-9823681-0-7

[Simmet 2012] Simmet, Prof. Dr. H.: *Enterprise 2.0: Nutzung sozialer Software im Unternehmen – Konsequenzen für den Kundenservice.* http://hsimmet.com/2012/02/19/enterprise-2-0-nutzung-sozialer-software-im-unternehmen-konsequenzen 2012. – Abgerufen am 05.08.2014

[Systems 2014] Systems, Softed: *Wie funktioniert HTTPS?* http://www.softed.de/fachthema/https.aspx. 2014. – Abgerufen am 06.08.2014

[Tanenbaum und van Steen 2007] Tanenbaum, Andrew S. ; Steen, Maarten van: *Verteilte Systeme Prinzipien und Paradigmen.* 2. Auflage. Pearson, 2007. – ISBN 978-3-8273-7293-2

[Techniker Krankenkasse 2014] Techniker Krankenkasse, Hamburg: *Gesundheitsreport 2014 - Risiko Rücken.* http://www.tk.de/centaurus/servlet/contentblob/644772/Datei/124007/Gesundheitsreport-2014.pdf. 2014. – Abgerufen am 29.07.2014

[Twitter 2014] Twitter: *Bootstrap.* http://getbootstrap.com/. 2014. – Abgerufen am 02.08.2014

[W3Schools 2014] W3Schools: *PHP.* http://www.w3schools.com/php/default.asp. 2014. – Abgerufen am 13.08.2014

[Webber u. a. 2010] Webber, Jim ; Parastatidis, Savas ; Robinson, Ian ; Fowler, Martin: *REST in Practice - Hypermedia and Systems Architecture.* 1. Auflage. O'Reilly Media, 2010. – ISBN 978-0-596-80582-1

[Wissen 2014] Wissen, IT: *Drei-Schichten-Architektur.* http://www.itwissen.info/definition/lexikon/Drei-Schichten-Architektur-three-tier-architecture.html. 2014. – Abgerufen am 01.08.2014

[Yeoman 2014] Yeoman: *Yeoman.* http://yeoman.io/. 2014. – Abgerufen am 04.08.2014

# API documentation

body format: Form Data ▾ request format: json ▾

GET /api/findauthusername.{_format} get username of authenticated user

**Documentation** Sandbox

## Requirements

| Name | Requirement | Type | Description |
|---|---|---|---|
| _format | xml\|json | string | return data format |

## Status Codes

| Status Code | Description |
|---|---|
| 200 | request okay, returns username |
| 401 | no user is authenticated |

GET /api/finddepartments.{_format} find all departments

**Documentation** Sandbox

## Requirements

| Name | Requirement | Type | Description |
|---|---|---|---|
| _format | xml\|json\|html | | |

## Return

| Parameter | Type | Versions | Description |
|---|---|---|---|
| name | string | * | |

## Status Codes

| Status Code | Description |
|---|---|
| 200 | request okay, returns departments array |
| 401 | no user is authenticated |
| 403 | forbidden for authenticated user |

GET /api/finduser.{_format} find all user

**Documentation** Sandbox

### Requirements

| Name | Requirement | Type | Description |
|---|---|---|---|
| _format | xml\|json | string | return data format |

### Return

| Parameter | Type | Versions | Description |
|---|---|---|---|
| firstname | string | * | |
| lastname | string | * | |
| username | string | * | |
| groups | array | * | |
| department | string | * | |

### Status Codes

| Status Code | Description |
|---|---|
| 200 | request okay, returns user array |
| 401 | no user is authenticated |
| 403 | forbidden for authenticated user |

POST /api/sicknote/create/{username}.{_format} create a new initial sicknote

**Documentation** Sandbox

### Requirements

| Name | Requirement | Type | Description |
|---|---|---|---|
| username | | String | username of disabled person |
| _format | xml\|json | string | return data format |

### Parameters

| Parameter | Type | Required? | Format | Description |
|---|---|---|---|---|
| original_submitted | boolean | true | | check if original au is submitted |
| first_date | dateTime | true | yyyy-MM-dd , z.B. 2014-07-08 | first date of disability, must be smaller or equal than last_date |
| last_date | dateTime | true | yyyy-MM-dd , z.B. 2014-07-08 | last date of disability |

| | | | | |
|---|---|---|---|---|
| comment | string | true | | add a comment |
| type | array | true | array with name and description of type, musst exist in Database | sicknote for the user himself or an person he has to take care of |
| au | string | true | | id of uploaded and converted au |

### Status Codes

| Status Code | Description |
|---|---|
| 201 | successfully created |
| 401 | no user is authenticated |
| 403 | not allowed for user |
| 404 | user not found |
| 409 | even one conflicting sicknote exists, or illegal date area |

GET /api/sicknote/downloadau/{id}.{_format} download an au

**Documentation** Sandbox

### Requirements

| Name | Requirement | Type | Description |
|---|---|---|---|
| _format | xml\|json | string | return data format |
| id | | string | id of au |

### Status Codes

| Status Code | Description |
|---|---|
| 200 | request okay, download au |
| 401 | no user is authenticated |
| 403 | forbidden for authenticated user |
| 404 | au not found |

POST /api/sicknote/extend/{id}.{_format} create an extension

**Documentation** Sandbox

### Requirements

| Name | Requirement | Type | Description |
|---|---|---|---|
| id | \d+ | integer | id of sicknote or extension to extend, musst be last extension of sicknote |
| _format | xml\|json | string | return data format |

### Parameters

| Parameter | Type | Required? | Format | Description |
|---|---|---|---|---|
| original_submitted | boolean | true | | check if original au is submitted |
| last_date | dateTime | true | yyyy-MM-dd , z.B. 2014-07-08 | last date of disability, must be greater than the last date of the last extension |
| comment | string | true | | add a comment |
| type | array | true | array with name key nad value, musst exist in Database | sicknote for the user himself or an person he has to take care of |
| au | string | true | | id of uploaded and converted au |

### Status Codes

| Status Code | Description |
|---|---|
| 201 | successfully created |
| 401 | no user is authenticated |
| 403 | not allowed for user |
| 409 | even one conflicting sicknote exists |

GET /api/sicknote/id/{noteid}.{_format} find sicknotes by id

**Documentation** Sandbox

### Requirements

| Name | Requirement | Type | Description |
|---|---|---|---|
| id | | \d+ | id of sickNote |
| _format | xml\|json | String | return data format |
| noteid | | | |

### Return

| Parameter | Type | Versions | Description |
|---|---|---|---|
| id | integer | * | |
| user | array | * | |
| original_submitted | boolean | * | |

| | | |
|---|---|---|
| first_date | DateTime | * |
| last_date | DateTime | * |
| created | DateTime | * |
| comment | string | * |
| type | array | * |
| au | string | * |
| created_by | array | * |

### Status Codes

| Status Code | Description |
|---|---|
| 200 | request okay, returns sicknotes array |
| 401 | no user is authenticated |
| 403 | not allowed for user |
| 404 | user not found |

GET /api/sicknote/types.{_format} finds all possible types of sicknotes

**Documentation** Sandbox

### Requirements

| Name | Requirement | Type | Description |
|---|---|---|---|
| _format | xml\|json | String | return data format |

### Return

| Parameter | Type | Versions | Description |
|---|---|---|---|
| id | integer | * | |
| user | array | * | |
| original_submitted | boolean | * | |
| first_date | DateTime | * | |
| last_date | DateTime | * | |
| created | DateTime | * | |
| comment | string | * | |
| type | array | * | |
| au | string | * | |
| created_by | array | * | |

### Status Codes

| Status Code | Description |
|---|---|
| 200 | request okay, returns sicknote types in an array |
| 401 | no user is authenticated |

POST /api/sicknote/uploadau/sicknote/{noteid}.{_format} upload an au

**Documentation** Sandbox

### Requirements

| Name | Requirement | Type | Description |
|---|---|---|---|
| _format | xml\|json | string | return data format |
| noteid | | string | id of note to add au |

### Parameters

| Parameter | Type | Required? | Format | Description |
|---|---|---|---|---|
| frontpage | file/image | true | | frontpage of au |
| backpage | file/image | true | | backpage of au |

### Status Codes

| Status Code | Description |
|---|---|
| 201 | successfully uploaded |
| 401 | no user is authenticated |
| 403 | forbidden for authenticated user |
| 404 | user not found |

POST /api/sicknote/uploadau/{username}.{_format} upload an au

**Documentation** Sandbox

### Requirements

| Name | Requirement | Type | Description |
|---|---|---|---|
| _format | xml\|json | string | return data format |
| username | | string | username of disabled person |

### Parameters

| Parameter | Type | Required? | Format | Description |
|---|---|---|---|---|
| frontpage | file/image | true | | frontpage of au |
| backpage | file/image | true | | backpage of au |

### Status Codes

| Status Code | Description |
|---|---|
| 201 | successfully uploaded |
| 401 | no user is authenticated |
| 403 | forbidden for authenticated user |
| 404 | user not found |

GET /api/sicknotes/department/{department}.{_format}

find all sicknotes for an single department

**Documentation** Sandbox

### Requirements

| Name | Requirement | Type | Description |
|---|---|---|---|
| department | | string | department name |
| _format | xml\|json | String | return data format |

### Return

| Parameter | Type | Versions | Description |
|---|---|---|---|
| id | integer | * | |
| user | array | * | |
| original_submitted | boolean | * | |
| first_date | DateTime | * | |
| last_date | DateTime | * | |
| created | DateTime | * | |
| comment | string | * | |
| type | array | * | |
| au | string | * | |
| created_by | array | * | |

### Status Codes

| Status Code | Description |
|---|---|
| 200 | request okay, returns sicknotes array |
| 401 | no user is authenticated |
| 403 | not allowed for user |
| 404 | department not found |

GET /api/sicknotes/department/{department}/period/{firstDate}/{lastDate}.{_format}

find all sicknotes for an single department in an period

**Documentation** Sandbox

## Requirements

| Name | Requirement | Type | Description |
|---|---|---|---|
| department | | string | department name |
| _format | xml\|json | String | return data format |
| firstDate | yyyy-MM-dd , z.B. 2014-07-08 | dateTime | minimum first date of disability |
| lastDate | yyyy-MM-dd , z.B. 2014-07-08 | dateTime | maximum last date of disability |

## Return

| Parameter | Type | Versions | Description |
|---|---|---|---|
| id | integer | * | |
| user | array | * | |
| original_submitted | boolean | * | |
| first_date | DateTime | * | |
| last_date | DateTime | * | |
| created | DateTime | * | |
| comment | string | * | |
| type | array | * | |
| au | string | * | |
| created_by | array | * | |

## Status Codes

| Status Code | Description |
|---|---|
| 200 | request okay, returns sicknotes array |
| 401 | no user is authenticated |
| 403 | not allowed for user |

| | |
|---|---|
| 404 | department not found, or invalid date given |

GET /api/sicknotes/period/{firstDate}/{lastDate}.{_format} find all sicknotes in an period

**Documentation** Sandbox

## Requirements

| Name | Requirement | Type | Description |
|---|---|---|---|
| _format | xml\|json | String | return data format |
| firstDate | yyyy-MM-dd , z.B. 2014-07-08 | dateTime | minimum first date of disability |
| lastDate | yyyy-MM-dd , z.B. 2014-07-08 | dateTime | maximum last date of disability |

## Return

| Parameter | Type | Versions | Description |
|---|---|---|---|
| id | integer | * | |
| user | array | * | |
| original_submitted | boolean | * | |
| first_date | DateTime | * | |
| last_date | DateTime | * | |
| created | DateTime | * | |
| comment | string | * | |
| type | array | * | |
| au | string | * | |
| created_by | array | * | |

## Status Codes

| Status Code | Description |
|---|---|
| 200 | request okay, returns sicknotes array |
| 401 | no user is authenticated |
| 403 | not allowed for user |
| 404 | invalid date given |

GET /api/sicknotes/user/{username}/period/{firstDate}/{lastDate}.{_format}

finds all sicknotes for an single user in an period

**Documentation** Sandbox

## Requirements

| Name | Requirement | Type | Description |
|---|---|---|---|
| username | | string | username of disabled person |
| _format | xml\|json | String | return data format |
| firstDate | yyyy-MM-dd , z.B. 2014-07-08 | dateTime | minimum first date of disability |
| lastDate | yyyy-MM-dd , z.B. 2014-07-08 | dateTime | maximum last date of disability |

## Return

| Parameter | Type | Versions | Description |
|---|---|---|---|
| id | integer | * | |
| user | array | * | |
| original_submitted | boolean | * | |
| first_date | DateTime | * | |
| last_date | DateTime | * | |
| created | DateTime | * | |
| comment | string | * | |
| type | array | * | |
| au | string | * | |
| created_by | array | * | |

## Status Codes

| Status Code | Description |
|---|---|
| 200 | request okay, returns sicknotes array |
| 401 | no user is authenticated |
| 403 | not allowed for user |
| 404 | user not found, or invalid date given |

GET /api/user/department/{name}.{_format}

find all user for single department, user with the role * manager are only allowed to find user for their department

**Documentation** Sandbox

### Requirements

| Name | Requirement | Type | Description |
|---|---|---|---|
| name | | string | name of department |
| _format | xml\|json | string | return data format |

### Return

| Parameter | Type | Versions | Description |
|---|---|---|---|
| firstname | string | * | |
| lastname | string | * | |
| username | string | * | |
| groups | array | * | |
| department | string | * | |

### Status Codes

| Status Code | Description |
|---|---|
| 200 | request okay, returns user array |
| 401 | no user is authenticated |
| 403 | not allowed for user |
| 404 | department not found |

GET /api/user/{username}.{_format} get all userdata for single user

**Documentation** Sandbox

### Requirements

| Name | Requirement | Type | Description |
|---|---|---|---|
| username | | string | username of disabled person |
| _format | xml\|json | String | return data format |

### Return

| Parameter | Type | Versions | Description |
|---|---|---|---|
| firstname | string | * | |
| lastname | string | * | |
| username | string | * | |
| groups | array | * | |
| department | string | * | |

### Status Codes

| Status Code | Description |
|---|---|
| 200 | request okay, returns userdata |
| 401 | no user is authenticated |
| 403 | not allowed for user |
| 404 | user not found |

*Documentation auto-generated on Mon, 11 Aug 14 17:19:19 +0200*


# Anforderungen

an einen softwareunterstützten Krankmeldungs- und Urlaubsantragsprozess bei der Silpion IT-Solutions GmbH

| | |
|---|---|
| **Kunde** | Silpion IT-Solutions GmbH<br>Brandshofer Deich 48<br>20539 Hamburg |
| **Ansprechpartner** | Volkmar Nossenheim, Michael Pohlers |

**Silpion IT-Solutions GmbH**
Brandshofer Deich 48
20539 Hamburg

Patrick Postel
Geschäftsführer
postel@silpion.de

tel +49 (40) 39 99 76 55
fax +49 (40) 39 99 76 40
mobil +49 (178) 27 87 133
web www.silpion.de

Commerzbank Hamburg
Konto 1307222
BLZ 200 400 00

St.Nr. 02/864/00965

Amtsgericht Hamburg
HRB 78585


# 1 Inhaltsverzeichnis



**Silpion IT-Solutions GmbH**
Brandshofer Deich 48
20539 Hamburg

Patrick Postel
Geschäftsführer
postel@silpion.de

tel +49 (40) 39 99 76 55
fax +49 (40) 39 99 76 40
mobil +49 (178) 27 87 133
web www.silpion.de

Commerzbank Hamburg
Konto 1307222
BLZ 200 400 00

St.Nr. 02/864/00965

Amtsgericht Hamburg
HRB 78585

## 2 Versionshistorie

| Version | Autor | Datum | Kommentar |
|---|---|---|---|
| 0.1 | Michael Pohlers | 31.10.13 | Initiale Anforderungen nach Absprache mit Abteilungsleitern und der Buchführung |
| 0.2 | Michael Pohlers | 05.11.13 | Anforderung der Vertretung bei Urlauben aufgrund von Abteilungsleiterfeedback hinzugefügt; Buchhaltung durch Personalabteilung ersetzt |

## 3 Zielsetzung

Die Silpion IT-Solutions GmbH ist ein IT-Ingenieurbüro, dass nach agilen Software-Entwicklungsprinzipien arbeitet. Genau nach diesen Prinzipien sollen auch Prozesse und Strukturen bei Silpion etabliert werden.

Silpion ist in den letzten Jahren schnell gewachsen. Die Prozesse und Strukturen erinnern aber noch teilweise an ein Unternehmen mit 10 Angestellten. Hier besteht akuter Handlungsbedarf.

Silpion hat einen bestehenden Prozess wie Krankmeldungen und Urlaubsanträge verarbeitet werden. Silpion ist klar geworden, dass dieser bestehende Prozess durch eine Softwarelösung verbessert werden kann, wodurch kein künstlicher "Wasserkopf" aufgebaut werden soll und hingehen Zeit und Aufwände eingespart werden sollen.

Wir wissen, dass eine Verbesserung durch eine Software herbeigeführt werden kann. Deshalb gilt für uns der agile Grundsatz, wir brauchen hier eine Software, die genau den Anforderungen im Kapitel 5 entspricht und nicht mehr oder weniger für Silpion erledigt. Diese Lösung soll also schlank sein und zeitnah eingeführt werden können und lediglich die derzeitigen Minimalanforderungen erfüllen. Sollten sich dann in den

**Silpion IT-Solutions GmbH**
Brandshofer Deich 48
20539 Hamburg

Patrick Postel
Geschäftsführer
postel@silpion.de

tel +49 (40) 39 99 76 55
fax +49 (40) 39 99 76 40
mobil +49 (178) 27 87 133
web www.silpion.de

Commerzbank Hamburg
Konto 1307222
BLZ 200 400 00

St.Nr. 02/864/00965

Amtsgericht Hamburg
HRB 78585

Reviews oder bei der täglichen Arbeit neue Anforderungen ergeben, soll die Software einfach und flexibel um diese Anforderungen und/oder Änderungen erweitert bzw. geändert werden können.

Die Software-Lösung soll auf neusten Technologien (Frameworks) mit den serverseitigen Programmiersprachen Java oder PHP und mit HTML/CSS/Js im Frontend (clientseitig) realisiert sein, da Silpion hier das größte Know-How hat um die Lösung ggf. weiter an die Bedürfnisse von Silpion anzupassen. Gewünscht wäre es für das Frontend das sogenannte Twitter Bootstrap im Silpion CI zu verwenden, welches gebrandet bereits existiert.

Um diese Lösung zu realisieren muss natürlich der Markt im Vorwege nach geeigneten Software-Lösungen evaluiert werden, die unseren Anforderungen entsprechen. Erst nach dieser Evaluierungsphase sollte entschieden werden, ob eine fertige Software-Lösung integriert wird oder eine Software selbst entwickelt wird.

Natürlich ist es nach dem agilen Ansatz auch denkbar, dass diese „günstige" „Quick-Win-Lösung" nur eine begrenzte Zeit Vorteile für Silpion bringt und diese, nach dem sich die Anforderungen von Silpion noch weiter ändern wieder komplett gegen eine andere Lösung ausgetauscht wird.

# 4 Ist-Zustand

Aktuell ist es Silpion klar, dass Krankmeldungen und Urlaubsanträge gewisse Aufwände bei den Mitarbeitern, den Abteilungsleitern, der Personalabteilung und dem Steuerberater einfordern.

Zurzeit finden verschiedene manuelle Schritte statt um den Krankmeldungs- und Urlaubsantragsprozess abzuwickeln. Das geschieht meistens per E-Mail.

## 4.1 Krankmeldungen

Der Mitarbeiter informiert am morgen per E-Mail die Abteilungsleitung und die Personalabteilung.

Am dritten Tag der Krankheit wird per E-Mail die Arbeitsunfähigkeitsbescheinigung (AU) vom Mitarbeiter eingefordert, der diese manchmal einscannt oder abfotografiert

und per E-Mail an die Personalabteilung oder die Abteilungsleiter schickt. Ein anderer Weg ist es die AU bei der Personalabteilung oder den Arbeitsleitern persönlich vorbei zu bringen oder per Post zu schicken. Sollte die AU bei den Abteilungsleitern landen, müssen diese die AU an die Personalabteilung weitergegeben. (Theoretisch sollte die AU immer direkt an die Personalabteilung gehen. Die Abteilungsleiter sollten nicht zuständig für die AU sein. Die Praxis zeigt aber häufig Anderes.)

In der Personalabteilung wird die AU der Mitarbeitermappe beigelegt und ggf. kopiert, wenn Fragen von der Krankenkasse oder des Steuerberaters beantwortet werden müssen. In diesem Fall wird die AU per Kopie oder im Original an den Steuerberater geschickt und parallel per E-Mail beantwortet. Die Fragen der Krankenkassen werden traditionell per Briefpost beantwortet.

## 4.2 Urlaubsanträge

Mitarbeiter besprechen ihren Urlaubswunschtermin grundsätzlich immer zuerst mit dem Team oder ggf. mit dem Kunden. Sollte von dieser Seite nichts gegen einen Urlaub im gewünschten Zeitraum sprechen, schreibt der Mitarbeiter eine E-Mail an den Abteilungsleiter mit der Personalabteilung in CC. Der Abteilungsleiter bewilligt den Urlaub abschließend oder lehnt den Urlaubswunsch mit Begründung ab. Der Mitarbeiter trägt bei Bewilligung des Urlaubs zurzeit manuell den Urlaub in den „Zimbra" (http://www.zimbra.com/products/index.html) Urlaubskalender ein. Die Abteilungsleitung trägt den Urlaub teils in eigene Tabellen ein und die Personalabteilung trägt dies ebenfalls teils in Tabellen ein. Jedes Jahr im Februar müssen entweder alle Mitarbeiter, Abteilungsleiter und die Personalabteilung dem Steuerberater mitteilen wie viel Resturlaub jeder Mitarbeiter noch hat. Dies wird durch einen Abgleich von verschiedenen Tabellen erreicht:

- Die Abteilungsleitung hat keine Tabelle geführt:
  - Die Mitarbeiter dieser Abteilung schreiben separat E-Mails an die Personalabteilung
  - Die Personalabteilung gleicht diese Meldungen mit der eigenen Tabelle ab
- Die Abteilungsleitung hat eine Tabelle der Urlaubstage geführt
  - Die Abteilungsleitung schickt eine Auswertung pro Mitarbeiter in einer Tabelle an die Personalabteilung
  - Die Personalabteilung gleicht diese Tabelle mit der eigenen Tabelle ab

**Silpion IT-Solutions GmbH**
Brandshofer Deich 48
20539 Hamburg

Patrick Postel
Geschäftsführer
postel@silpion.de

tel +49 (40) 39 99 76 55
fax +49 (40) 39 99 76 40
mobil +49 (178) 27 87 133
web www.silpion.de

Commerzbank Hamburg
Konto 1307222
BLZ 200 400 00

St.Nr. 02/864/00965

Amtsgericht Hamburg
HRB 78585

Im Anschluss daran, wenn alle Daten konsolidiert und kumuliert wurden, wird eine Liste mit dem Resturlaub aller Mitarbeiter an den Steuerberater per E-Mail geschickt.

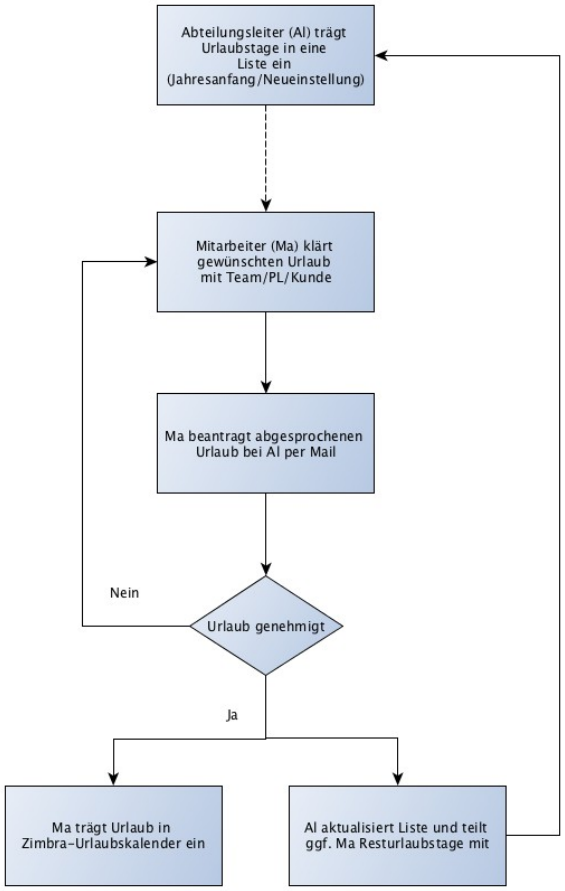


*Abbildung 1: Kernurlaubsantragsprozess*

## 5 Ziele für die Zukunft

Im Folgenden wird beschrieben, wie dieser Prozess verschlankt und vereinfacht werden kann um den Mitarbeitern, der Personalabteilung und der Abteilungsleitung kurzfristig Aufwände zu ersparen.

Für beide Prozesse gilt:

- Es gibt nur einen Login für beide Prozesse für die Rollen
  - Mitarbeiter
  - Abteilungsleiter
  - Personalabteilung
- Der Login soll gegen, das bei Silpion bestehende, LDAP-System authentifizieren
  - Hierzu muss nach jetzigem Wissensstand für jeden Mitarbeiter einmal initial der Abteilungsleiter als Vorgesetzter im LDAP-System hinterlegt werden. (Bereits geschehen)
- In einem anderen Schritt (andere Bachelorarbeit) soll über diesen Login auch ein Ressourcenplanungstool erreichbar sein und die gleiche Grundarchitektur verwendet werden. Das Ressourcenplanungstool ist nicht Teil dieser Unterlage.
- Es sollen speziell für Tablet und Smartphone optimierte User Interfaces vorhanden sein, um Urlaubsanträge zu stellen und Krankmeldungen durchzuführen.
- Um später ggf. eine native iOS oder Android-App. leichter anbinden zu können sollen User-Interface und serverseitige Programmlogik entkoppelt sein z.B. über einen Rest-Service.

Ein Nebenziel ist es eine zentrale Webseite im internen Netzwerk der Silpion IT-Solutions GmbH zu haben, auf der es Verlinkungen zu den folgenden Services gibt:

- Urlaubsanträge
- Krankmeldungen
- Zimbra Webmail
- Wiki
- Redmine
- Openfiler
- Alfreso
- Trac
- Später ggf. Socialcast oder Socialcast wird die Intranet-Startseite

## 5.1 Krankmeldungen

- Mitarbeiter:

- Die Mitarbeiter haben nach Login eine Eingabemaske mit der sie sich krank melden können:
  - Datum von bis
  - Upload der AU
- Jeder Mitarbeiter hat eine Übersicht seiner eigenen Krankheitstage filterbar nach
  - Jahr
  - Monat

- Personalabteilung
  - Kann alle Krankmeldungen aller Mitarbeiter einsehen
  - Sie kann filtern nach
    - Mitarbeiter
    - Monat
    - Jahr
  - Die Personalabteilung soll bei Nachfragen der Krankenkasse auf einen Blick Auskunft geben können wann für wie lange mit gültiger AU ein Mitarbeiter krankgeschrieben war.
  - Es ist möglich aus jeder gefilterten Ansicht ein PDF zu genieren, was ggf. der Krankenkasse und dem Steuerberater zugeschickt werden kann.
  - Die Personalabteilung kann konfigurieren ob E-Mailbenachrichtigungen und/oder im System Benachrichtigungen gewünscht sind.
  - Die Personalabteilung (und Abteilungsleitung) hat ebenfalls die Möglichkeit im Namen eines Mitarbeiters eine Krankmeldung im System durchzuführen (z.B. wenn der Mitarbeiter dies aus verschiedenen Gründen nicht kann)
  - Die Personalabteilung hat eine zusätzliche Checkbox pro Krankmeldung wenn die AU im Original bei der Personalabteilung angekommen ist.
- Die Abteilungsleitung hat die gleichen Möglichkeiten wie die Personalabteilung aber nur für Mitarbeiter der jeweiligen Abteilung.
  - Die Abteilungsleitung kann nicht die Checkbox setzen, wenn eine AU im Original angekommen ist.
- Allgemein:
  - Der Mitarbeiter hat optional vorläufig die Möglichkeit ein lesbares Bild der AU (Vor- und Rückseite) hochzuladen. Diese muss auch immer noch im Original abgegeben werden.
    - Die zwei Bilder werden automatisiert zu einem PDF zusammengefasst.
    - Es findet eine Validierung statt, ob die Qualität des Bildes ausreichend ist um theoretisch Schrift darauf zu erkennen.

        - Die Personalabteilung und der Abteilungsleiter kann bei einer Krankmeldung diese hochgeladene AU einsehen.
    - Wenn eine Krankmeldung über das System erfolgt, werden immer Abteilungsleiter und Personalabteilung informiert.
        - Je nach eigener Konfiguration entweder per E-Mail
        - oder per Benachrichtigung im System
    - Außerdem wird automatisch ein Abwesenheitseintrag für den Mitarbeiter im Zimbra Urlaubskalender vorgenommen.
    - Benachrichtigungen gibt es wenn:
        - sich ein Mitarbeiter initial krank meldet.
        - sich eine Krankmeldung verlängert.
        - ein Mitarbeiter inzwischen 3 Tage krank ist und keine AU hochgeladen wurde.
        - ein Mitarbeiter inzwischen mehr als 4 Wochen krank ist
        - ein Mitarbeiter inzwischen mehr als 6 Wochen krank ist
    - Der Personalabteilung müssen nach spätestens 4 Wochen Krankheit alle AU im Original vorliegen. Sollte dies nicht der Fall sein, wird der Mitarbeiter per E-Mail benachrichtigt.
    - Auch wird der Mitarbeiter per E-Mail benachrichtigt, wenn am dritten Tag der Krankheit noch keine Form der AU eingereicht wurde, weder digital noch in Papierform (Checkbox im System)

## 5.2 Urlaubsanträge

- Mitarbeiter
    - Jeder Mitarbeiter spricht weiterhin seinen Urlaubswunsch zuerst im Team und ggf. mit dem Kunden ab.
    - Hiernach kann er offiziell seinen Urlaub nach Login beantragen
        - Mandatory:
            - Datum Von Bis
            - Vertretung: Textfeld (am liebsten mit Autocomplete durch LDAP)
            - Checkbox, dass der Urlaub mit Team und Kunden abgesprochen ist.
        - Optional:
            - Bemerkung
    - Der Mitarbeiter hat immer einen Überblick wie viel Urlaub ihm noch zusteht und wie viel Urlaub er wann schon genommen hat.
    - Ein Übertrag des Resturlaubs wird in das nächste Kalenderjahr automatisch vorgenommen.

- Ein Verfall des Resturlaubs zum 31.03. wird derzeit nicht aktiviert sein, aber soll vorgesehen sein (Konfiguration für Entwickler oder Administratoren)
- Es ist möglich mehr Urlaub anzufragen, als dem Mitarbeiter zu steht. (Ggf. Sondervereinbarungen). Hier sieht der Mitarbeiter lediglich seinen verfügbaren Resturlaub mit einer negativen roten Zahl.
- Der Mitarbeiter bekommt später eine Benachrichtigung im System und/oder konfigurierbar per E-Mail, dass seine Urlaubsanfrage genehmigt oder abgelehnt wurde.

- Abteilungsleiter
  - Der Abteilungsleiter bekommt eine Benachrichtigung im System und/oder konfigurierbar per E-Mail, dass eine Urlaubsanfrage vorliegt.
    - Wenn per Mail, dann mit Link auf den entsprechenden View im System.
    - Der Abteilungsleiter sieht im System:
      - Wie viel Urlaub der Mitarbeiter schon genommen hat
      - Wie viel Urlaub dem Mitarbeiter noch zu steht
      - Den gewünschten Zeitraum und ggf. eine abgegebene Bemerkung des Mitarbeiters
    - Der Abteilungsleiter kann den Urlaub mit einem Klick genehmigen oder ihn mit einer optionalen Bemerkung ablehnen.
    - Wird der Urlaub bewilligt, wird automatisch ein Urlaubseintrag für den Mitarbeiter im Zimbra Urlaubskalender vorgenommen.
  - Der Abteilungsleiter kann außerdem per Filter die genommenen und offenen Urlaubstage einsehen und als Exceldatei exportieren. Und zwar wie folgt:
    - Pro Mitarbeiter seiner Abteilung oder der ganzen Abteilung
    - pro Monat
    - pro Jahr
  - Der Abteilungsleiter bekommt ebenfalls Benachrichtigungen (konfigurierbar per E-Mail oder im System) wenn ein Mitarbeiter nicht bis 31.03. seinen Resturlaub des letzten Jahres verbraucht hat und wenn bis 30.09. nicht die Hälfte des Jahresurlaubs gebucht wurde.
- Personalabteilung
  - Die Personalabteilung hat fast alle Rechte der Mitarbeiter und der Abteilungsleiter. Allerdings darüber hinaus:
    - Kann die Personalabteilung alle Urlaubstage ALLER Mitarbeiter Einsehen, Filtern und Exportieren.

- Bekommt eine Benachrichtigung, dass im Februar die Gesamtliste des Resturlaubs für den Steuerberater zum Download als Exceltabelle bereit steht.
- Die Personalabteilung muss einmal pro Mitarbeiter initial einpflegen, wie viel Tage Urlaub jeder Mitarbeiter pro Jahr zur Verfügung hat.

- Die Personalabteilung kann hingegen nicht:
  - Urlaubsanträge bewilligen. Dies kann nur der Abteilungsleiter.

*Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.*

Hamburg, 28. August 2014 Leon Fausten