HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL
FHO Fachhochschule Ostschweiz

digio

Ticketing System für ein Kundenportal eines IaaS Providers

Studienarbeit

Abteilung Informatik
Hochschule für Technik Rapperswil

Herbstsemester 2017

Autoren: Marco Steiner, Dominik Thamm
Betreuer: Prof. Dr. Markus Stolze
Projektpartner: Digio GmbH, Kilchberg

Zusammenfassung

Die innofield AG mit Sitz in Wallisellen bietet Ihren Kunden seit 2009. Dienstleistungen im Bereich PaaS (Platform as a Service) und IaaS (Infrastructure as a Service) an. Die Firma Digio GmbH entwickelt ein neues Kundenportal für die innofield AG. Die Studenarbeit beschäftigt sich mit dem Support Modul für das Kundenportal.

Bisher wurden die Supportanfragen der innofield AG per Telefon oder Mail entgegengenommen und verarbeitet. Dieser Prozess soll nun überarbeitet werden um dem Kunden das Erfassen von Tickets direkt im Kundenportal zu ermöglichen. Damit soll einerseits die Übersicht und der Komfort des Kunden erhöht, andererseits aber auch der Aufwand seitens innofield AG minimiert werden. In diesem Zuge wurde zugleich die Wahl des Ticketsystems neu evaluiert und entschieden von ZenDesk auf die opensource Lösung Zammad zu wechseln.

Bei der Implementation des Support Moduls wurde darauf geachtet die Erweiterung möglichst gut in die schon bestehende Architektur einzupassen. Die Web-API (Symfony) sowie das Frontend (Angular) wurden entsprechend ausgebaut. Zukünftige Kunden der innofield AG können somit bequem im Kundenportal ihre Supportanfragen erstellen und verwalten.

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Ausgangslage

Die Innofield AG mit Sitz in Wallisellen beschäftigt fünf Mitarbeitende und bietet Ihren Kunden seit 2009 Dienstleistungen im Bereich Paas (Platform as a Service) und Iaas (Infrastructure as a Service) an. Sie verfügt über eine performante, ausfallsichere und mehrfach zertifizierte Infrastruktur in der Schweiz. Das einst aus der Not entstandene Kundenportal (myFlow), wird im Jahr 2017 durch die Firma Digio GmbH von Grund auf neu konzipiert und entwickelt. Die Funktionen des neuen Portals sollen das ganze Angebot der Innofield AG abdecken, so dass die Kunden zukünftig das Management all ihrer Services selber übernehmen können. Aktuell müssen Kunden der Innofield AG zudem das Kundenportal verlassen, um Support Tickets in einem unabhängig betriebenen Ticketsystem zu eröffnen und zu bearbeiten.

1.2 Zweck und Ziel

Die Studienarbeit ist eine Machbarkeitsstudie, welche die Risiken der Einbindung eines Ticketsystems in das myFlow Kundenportal evaluieren und minimieren soll. Ausserdem soll ein Prototyp als ”proof of concept” implementiert werden.

1.3 Umfang

1. Analyse der bestehenden Lösung
2. Analyse der Anforderungen des Kunden
3. Konzept für die Integration
4. Evaluation eines den Anforderungen entsprechenden Ticketsystems
5. Präsentation des Evaluationsentschlusses
6. Aufsetzen und konfigurieren der Plattform
7. Implementierung und Integration in das Kundenportal (Backend und Frontend)
8. Migration der Ticket-Daten

Kapitel 2

Anforderungsanalyse

Da die Arbeit auf ca. 480 Arbeitsstunden begrenzt ist, können nicht alle Anforderungen implementiert werden. Es wurde daher zwischen Projekt- und Produktanforderungen unterschieden.

Die Projektanforderungen sollen während der Studienarbeit umgesetzt werden und sind bereits in der Aufgabenstellung beschrieben. Sie werden in diesem Kapitel noch genauer definiert und überarbeitet.

Die Produktanforderungen werden analysiert, um sie bei Entscheidungen zu Architektur und Umsetzung berücksichtigen zu können und anhand dessen die Weiterentwicklung des Systems zu vereinfachen.

2.1 Projektanforderungen

Die in der Aufgabenstellung definierten Anforderungen werden in den folgenden Unterkapiteln in Bezug auf ihre Umsetzung klarer und zusammengehörig ausgeschrieben.

2.1.1 Anforderungen an das Kundenportal

- Maske um Tickets zu erstellen
- Folgende Daten müssen erfasst werden können:
  - Titel (Textfeld)
  - Nachricht (Textfeld)
  - Gruppe (Auswahlfeld)
  - Priorität (Auswahlfeld)
  - Betroffenes Produkt (Textfeld)
  - Anhänge (Datei)
- Bei Prioritätsauswahl "Sehr Hoch" soll eine Warnung angezeigt werden, dass diese Auswahl kostenpflichtig ist wenn sie missbraucht wird.
- Autocompletefunktion für das Feld "Betroffenes Produkt", welche die Liste von gekauften Produkten durchsucht.
- Übersicht über alle Tickets der Organisation.
- Möglichkeit zum Kommentieren und Schliessen von Tickets.

2.1.2 Anforderungen an das Ticketsystem

- Agent kann Tickets beantworten und bearbeiten.
- Tickets können Gruppen zugewiesen werden.
- Jeder Agent hat einen persönlichen Benutzer
- Ticketsystem kann aus eingehenden E-Mails automatisch Tickets erstellen.
- Es stehen Statistiken z.B. über die Anzahl Tickets oder die Antwortzeiten von Agenten zur Verfügung.
- Konfigurierbare Vorlagen für automatisierte E-Mail antworten.

2.2 Produktanforderungen

Nachfolgend sind die Produktanforderungen aufgelistet. Sie wurden analysiert und ihre Machbarkeit wurde abgeklärt. Wegen der eng begrenzten Zeitdauer konnten sie jedoch teilweise nicht oder nicht vollständig bearbeitet werden.

2.2.1 Produktanforderungen aus der Aufgabenstellung

Anforderungen an das Kundenportal

- Volltextsuche über alle Tickets innerhalb der gleichen Organisation
- Knowledgebase Integration[1]

Anforderungen an das Ticketsystem

- Knowlegebase Integration
- Single Sign On Integration von myFlow

2.2.2 Nichtfunktionale Anforderungen

Nach dem Wikipedia Artikel zu ISO 9126 [12] wurden die nachfolgenden Software Qualitätskriterien betrachtet um eine Bewusstheit bei der Entwicklung des Systems für diese Kriterien zu schaffen. Da für die Studienarbeit vor allem die Frage der Machbarkeit der Funktionalität wichtig zu beweisen war, wurden die NFAs eher generell gehalten. Wichtige Abklärungen zur Einhaltung der NFAs für das Endprodukt wurden jedoch bereits während der Studienarbeit gemacht.

Zuverlässigkeit

Hierbei sind vor allem Fehler, die in Zammad oder bei Übertragung und Parsen der Daten auftreten, interessant. Diese Fehler sollen abgefangen werden und dem Benutzer falls nötig eine generelle Fehlermeldung gezeigt werden, dass etwas nicht funktioniert hat.

Benutzbarkeit

Das Support Modul sollte selbsterklärend sein, das bedeutet, dass jemand der Versteht was ein Ticketsystem ist, sich ohne weitere Einführung zurecht findet und die Informationen findet, die er benötigt.

[1] Die Knowledgebase wurde nach Absprache mit dem Kunden in die Produktanforderungen verschoben. Die Anforderung musste neu bewertet werden, weil Zammad dies noch nicht implementiert hatte oder ansonsten ein anderes Ticketsystem gewählt werden, siehe Sitzungsprotokoll 19.10.2017

Da das Design bereits zu einem grossen Teil vorgegeben ist, kann bei der Wahl von Farbe und Schrift nicht abgewichen werden.

Fokussiert wird die Darstellung der vom Endkunden benötigten Informationen, da dies in der Arbeit Beeinflusst werden kann. Dabei wurde mit den Nachfolgenden Fragen probiert eine möglichst Benutzerfreundliche Implementierung zu erreichen:

- Welche Informationen sind nötig für den Benutzer?
- Wie ist der Prozess aufgebaut, der abgebildet werden soll?
- Welche Informationen sollen vom Benutzer angepasst werden können?

Änderbarkeit und Wartbarkeit

Wo Sinnvoll sollen Abstraktionsschichten eingeführt werden, die gegebenenfalls Abhängigkeiten reduzieren.

2.2.3 Anforderungen als Use Cases

Die Restlichen Funktionalen Anforderungen sind als fully dressed Use Cases aufgeschrieben um einen Eindruck für den Endzustand des Systems zu erhalten. Diese Anforderungen sind unter den Produktanforderungen gelistet, da sie iterativ klarer ausformuliert wurden und während der Studienarbeit nur ihre Machbarkeit bewiesen werden muss, sie aber nicht komplett umgesetzt werden müssen.

Aktoren und Stakeholder

Tabelle 2.1: Aktoren und Stakeholder

Benutzer	Benutzer sind Kunden und erstellen oder bearbeiten Tickets über das myFlow Portal. Zudem informieren sie sich dort über den Status der Supporttätigkeiten.
Agent	Agenten sind Systemadministratoren, welche die Tickets der Benutzer bearbeiten. Sie bearbeiten die Tickets über das Ticketsystem. Im Use Case Diagramm ist daher eine Verbindung zwischen dem Ticketsystem und dem Agenten gezeichnet, da dies als verdeutlichend Empfunden wurde und das Use Case Diagramm hier als Kommunikationsmittel eingesetzt wird.
Ticketsystem	Das Ticketsystem ist Zammad, welches Eingebunden werden soll. Die Tickets und die zugehörigen Daten werden im Ticketsystem gespeichert. Agenten bearbeiten die Tickets nur über das Ticketsystem.

Übersicht

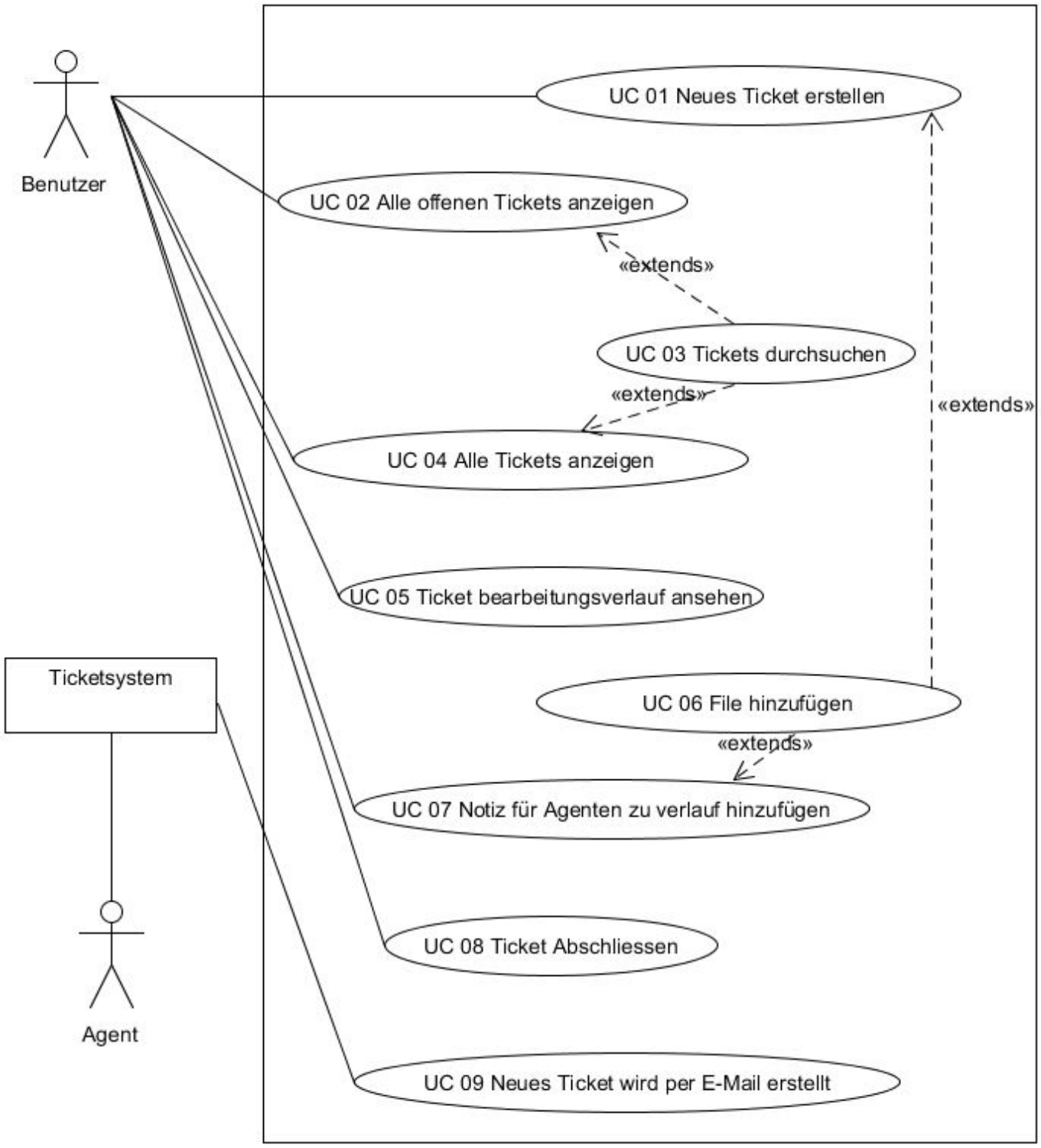

Abbildung 2.1: Use Case Diagramm

UC 01 Neues Ticket erstellen

Tabelle 2.2: UC 01 Neues Ticket erstellen

Goal	Der Benutzer hat ein Problem mit der bereitgestellten Infrastruktur und möchte, dass dies von einem Agenten bearbeitet wird. Dafür erstellt er ein neues Ticket.
Level	User Goal
Primary Actor	Benutzer
Trigger	Es besteht ein Problem mit der Infrastruktur des Kunden
Stakeholders and Interests	Agenten behandeln die Problme der Kunden über das Ticketsystem.
Preconditions	Benutzer besitzt Account
Postconditions	Das Ticket ist erstellt und in Zammad gespeichert. Es wurde ein E-Mail an die Agenenten versendet.
Failure Postconditions	Der Benutzer erhält die generelle Meldung, dass etwas nicht funktioniert hat.
Main Success Scenario	1. Der Benutzer navigiert im myFlow Portal auf Support. 2. Der Benutzer klickt auf den Button zur erstellung eines neuen Tickets 3. Das System zeigt dem Benutzer den Dialog für das erstellen eines neuen Tickets. 4. Der Benutzer gibt die Informationen ein und speichert das Ticket. 5. Das System speichert das Ticket im Ticketsystem und leitet den Benutzer weiter auf die Ticketübersicht.
Extensions	a Zu jeder Zeit wenn der Benutzer abbricht: 1 Das System führt keine Aktionen durch. b Zu jeder Zeit wenn das System fehlschlägt: 1 Das Ticketsystem übermittelt einen Fehler an das myFlow Portal. 2 Das System zeigt dem Benutzer einen generellen Fehler.
Frequency of Occurence	1x pro Woche
Spezielle nicht funktionale Anforderungen	-

UC 02 Alle offenen Tickets anzeigen

Tabelle 2.3: UC 02 Alle offenen Tickets anzeigen

Goal	Der Benutzer möchte sich seine offenen Tickets anzeigen lassen, und sich über deren Status informieren.
Level	User Goal
Primary Actor	Benutzer
Trigger	Benutzer erwartet ein Statusupdate zum Problem.
Stakeholders and Interests	
Preconditions	Benutzer besitzt Account
Postconditions	Tickets werden angzeigt falls vorhanden. Ansonsten eine Meldung, dass keine Tickets vorhanden seien.
Failure Postconditions	Der Benutzer erhält die generelle Meldung, dass etwas nicht funktioniert hat.
Main Success Scenario	1. Der Benutzer navigiert im myFlow Portal auf Support. 2. Das System zeigt dem Benutzer direkt den Ticket dialog an. 3. Der Benutzer schaut seine Tickets and und erhält Übersichtsinformationen.
Extensions	a Zu jeder Zeit wenn der Benutzer abbricht: 1 Das System führt keine Aktionen durch. b Zu jeder Zeit wenn das System fehlschlägt: 1 Das Ticketsystem übermittelt einen Fehler an das myFlow Portal. 2 Das System zeigt dem Benutzer einen generellen Fehler.
Frequency of Occurence	Unterschiedlich nach anzahl der Tickets und deren Dringlichkeit. Schätzung: 5x pro Woche.

UC 03 Tickets durchsuchen

Tabelle 2.4: UC 03 Tickets durchsuchen

Goal	Der Benutzer möchte die Tickets nach gewissen Suchkriterien durchsuchen.
Level	User Goal
Primary Actor	Benutzer
Trigger	Es werden viele Tickets angezeigt und der Benutzer will nicht die ganze Tabelle durchgehen, wenn er ein spezifisches Ticket sucht, sondern die angezeigten Tickets z.B. nach dem Titel des Tickets filtern.
Stakeholders and Interests	-
Preconditions	Benutzer besitzt Account
Postconditions	Tickets werden angzeigt falls die Suche erfolgreich war. Ansonsten eine Meldung, dass die Suche keine Ergebnisse geliefert hat.
Failure Postconditions	Der Benutzer erhält die generelle Meldung, dass etwas nicht funktioniert hat.
Main Success Scenario	1. Der Benutzer navigiert im myFlow Portal auf Support. 2. Das System zeigt dem Benutzer direkt den Ticket dialog an. 3. Der Benutzer klickt auf das Suchfeld und gibt das Suchkriterium ein. 4. Das System liefert dem Benutzer die Tickets, auf welche das Suchkriterium passt oder eine Meldung, dass keine Tickets zum Suchkriterium gefunden worden sind.
Extensions	a Zu jeder Zeit wenn der Benutzer abbricht: 1 Das System führt keine Aktionen durch. b Zu jeder Zeit wenn das System fehlschlägt: 1 Das Ticketsystem übermittelt einen Fehler an das myFlow Portal. 2 Das System zeigt dem Benutzer einen generellen Fehler.
Frequency of Occurence	Unterschiedlich nach Anzahl der Tickets des Benutzers Schätzung: 1x pro Monat.

UC 04 Alle Tickets anzeigen

Tabelle 2.5: UC 04 Alle Tickets anzeigen

Goal	Der Benutzer möchte sich alle je eröffneten Tickets anzeigen lassen, um Informationen zu bereits gelösten Problemen zu finden.
Level	User Goal
Primary Actor	Benutzer
Trigger	Der Benutzer weiss, dass bereits für ein Ähnliches Problem ein Ticket erstellt wurde und er möchte sich über dessen Lösung informieren.
Stakeholders and Interests	Agenenten sind froh, wenn der Benutzer sich selber über ein bereits gelöstes Problem informieren kann.
Preconditions	Benutzer besitzt Account
Postconditions	Alle Tickets werden angzeigt falls vorhanden. Ansonsten eine Meldung, dass keine Tickets vorhanden seien.
Failure Postconditions	Der Benutzer erhält die generelle Meldung, dass etwas nicht funktioniert hat.
Main Success Scenario	1. Der Benutzer navigiert im myFlow Portal auf Support und klickt auf "Alle Tickets". 2. Das System zeigt dem Benutzer direkt den Ticket dialog an. 3. Der Benutzer schaut seine Tickets and und erhält Informationen.
Extensions	a Zu jeder Zeit wenn der Benutzer abbricht: 1 Das System führt keine Aktionen durch. b Zu jeder Zeit wenn das System fehlschlägt: 1 Das Ticketsystem übermittelt einen Fehler an das myFlow Portal. 2 Das System zeigt dem Benutzer einen generellen Fehler.
Frequency of Occurence	1x pro Monat.

UC 05 Ticket Bearbeitungsverlauf ansehen

Tabelle 2.6: UC 05 Ticket Bearbeitungsverlauf ansehen

Goal	Der Benutzer möchte die Aktionen ansehen, die bis jetzt gemacht wurden um das Problem zu beheben
Level	User Goal
Primary Actor	Benutzer
Trigger	Der Benutzer will wissen, welche Aktionen bis jetzt getroffen wurden.
Stakeholders and Interests	-
Preconditions	Benutzer besitzt Account.
Postconditions	-
Failure Postconditions	-
Main Success Scenario	1. Der Benutzer navigiert im myFlow Portal auf Support. 2. Das System zeigt dem Benutzer direkt den Ticket dialog an. 3. Der Benutzer klickt auf das Ticket von welchem er den Bearbeitungsverlauf ansehen will. 4. Das System zeigt Notizen und Emails an.
Extensions	a Zu jeder Zeit wenn der Benutzer abbricht: 1 Das System führt keine Aktionen durch. b Zu jeder Zeit wenn das System fehlschlägt: 1 Es wird eine generelle Fehlermeldung ausgegeben.
Frequency of Occurence	1x pro Woche
Spezielle nicht funktionale Anforderungen	-

UC 06 File hinzufügen

Tabelle 2.7: UC 06 File hinzufügen

Goal	In den Notiz Feldern können Informationen je nach Art nur ungenügend dargestellt werden z.B. Informationen in einer Exceltabelle. Der Benutzer möchte darum Dateien an Tickets anhängen können um dem Agenten weitere Informationen zum Problem zu liefern.
Level	User Goal
Primary Actor	Benutzer
Trigger	Informationen in einer Datei sollen an Ticket angefügt werden.
Stakeholders and Interests	-
Preconditions	Benutzer besitzt Account
Postconditions	Die Datei ist im Ticketsystem gespeichert.
Failure Postconditions	Der Benutzer erhält die generelle Meldung, dass etwas nicht funktioniert hat.
Main Success Scenario	1. Der Benutzer navigiert im myFlow Portal auf Support und zum Ticket, an welches er die Information anfügen will. 2. Das System zeigt dem Benutzer das Ticket an. 3. Der Benutzer erstellt eine neue Notiz und fügt die Datei an. 4. Das System speichert die Notiz und die Datei.
Extensions	a Zu jeder Zeit wenn der Benutzer abbricht: 1 Das System führt keine Aktionen durch. b Zu jeder Zeit wenn das System fehlschlägt: 1 Das Ticketsystem übermittelt einen Fehler an das myFlow Portal. 2 Das System zeigt dem Benutzer einen generellen Fehler.
Frequency of Occurence	1x pro Woche

UC 07 Notiz für Agenten zu Verlauf hinzufügen

Tabelle 2.8: UC 07 Notiz für Agenten zu Verlauf hinzufügen

Goal	Der Benutzer kann mit dem Agenten über Notizen [2]kommunizieren. Zudem kann er weitere Informationen für den Agenten in Notizen erfassen, die helfen das Problem zu beheben.
Level	User Goal
Primary Actor	Benutzer
Trigger	Weitere Informationen zur Lösung des Problems vorhanden oder eine Aktion vom Benutzer gewünscht.
Stakeholders and Interests	
Preconditions	Benutzer besitzt Account
Postconditions	Die Notiz ist im Ticketsystem gespeichert.
Failure Postconditions	Der Benutzer erhält die generelle Meldung, dass etwas nicht funktioniert hat.
Main Success Scenario	1. Der Benutzer navigiert im myFlow Portal auf Support und zum Ticket, an welches er die Notiz anfügen will. 2. Das System zeigt dem Benutzer das Ticket an. 3. Der Benutzer erstellt eine neue Notiz. 4. Das System speichert die Notiz.
Extensions	a Zu jeder Zeit wenn der Benutzer abbricht: 1 Das System führt keine Aktionen durch. b Zu jeder Zeit wenn das System fehlschlägt: 1 Das Ticketsystem übermittelt einen Fehler an das myFlow Portal. 2 Das System zeigt dem Benutzer einen generellen Fehler.
Frequency of Occurence	1x pro Woche

UC 08 Ticket abschliessen

Tabelle 2.9: UC 08 Ticket abschliessen

Goal	Das Problem ist behoben und der Benutzer möchte das Problem als gelöst deklarieren bzw. das Ticket schliessen.
Level	User Goal
Primary Actor	Benutzer
Trigger	Das Problem ist behoben oder das Ticket kann geschlossen werden.
Stakeholders and Interests	-
Preconditions	Benutzer besitzt Account
Postconditions	Das Ticket ist geschlossen.
Failure Postconditions	Der Benutzer erhält die generelle Meldung, dass etwas nicht funktioniert hat.
Main Success Scenario	1. Der Benutzer navigiert im myFlow Portal auf Support und dann zum Ticket, welches er abschliessen will. 2. Das System zeigt dem Benutzer das Ticket an. 3. Der Benutzer fügt einen neue Notiz hinzu und klickt auf den Button Ticket abschliessen.
Extensions	a Zu jeder Zeit wenn der Benutzer abbricht: 1 Das System führt keine Aktionen durch. b Zu jeder Zeit wenn das System fehlschlägt: 1 Das Ticketsystem übermittelt einen Fehler an das myFlow Portal. 2 Das System zeigt dem Benutzer einen generellen Fehler.
Frequency of Occurence	1x pro Woche

UC 09 Neues Ticket wird per E-Mail erstellt

Tabelle 2.10: UC 09 Neues Ticket wird per E-Mail erstellt

Goal	Der Benutzer hat ein Problem und möchte, dass dieses von einem Agenten behandelt wird. Dafür sendet er eine E-Mail an die Support Adresse.
Level	User Goal
Primary Actor	Benutzer
Trigger	Es besteht ein Problem mit der Infrastruktur des Kunden
Stakeholders and Interests	Agenten behandeln die Problme der Kunden über das Ticketsystem.
Preconditions	Benutzer besitzt Account, die E-Mail Adresse im Account stimmt mit der Adresse des E-Mails überein.
Postconditions	Das Ticket ist erstellt und im Ticketsystem gespeichert. Es wurde ein E-Mail an die Support Agenenten versendet.
Failure Postconditions	Der Benutzer erhält eine E-Mail, dass etwas nicht funktioniert hat.
Main Success Scenario	1. Der Benutzer sendet eine E-Mail an die Support Adresse. 2. Das System erstellt ein neues Ticket und informiert die Agenten.
Extensions	a Zu jeder Zeit wenn das System fehlschlägt: 1 Ticketsystem sendet eine E-Mail an den Kunden mit entsprechenden Informationen.
Frequency of Occurence	1x pro Woche
Spezielle nicht funktionale Anforderungen	-
Open Issues	Reaktionen von Ticketsystem auf Fehler abklären.

Kapitel 3

Entscheid Evaluation Ticketsysteme

Im folgenden wird die Evaluation erläutert und der Entscheid für das Ticketsystem der Innofield AG begründet. Die zu analysierenden Ticketsysteme wurden anhand eines Artikels [1] ausgewählt.

Zudem wurden auch Werbungen und weitere Vorschläge für Ticketsysteme die während der Suche im Internet angezeigt wurden, miteinbezogen.

Der Evaluationsprozess und das Resultat ist nachfolgend beschrieben.

3.1 Evaluationsprozess

Der Evaluationsprozess wurde aufgeteilt in die Analyse (Kapitel 3.2) der Produkte anhand welcher ein Vorentscheid (Kapitel 3.3) gemacht wurde, um zwei Empfehlungen (Kapitel 3.4) in einem Meeting mit Damien Vouillamoz besprechen zu können und mit ihm gemeinsam die Entscheidung (Kapitel 3.5) zu treffen.

Produktanalyse

Zu beginn der Analysephase war nicht ganz klar, welche Erwartungen an ein Ticketsystem gestellt werden konnten. Zudem war nicht vollständig ersichtlich was der Kunde schlussendlich wirklich brauchte. Dies konnte nicht bereits zu Beginn festgelegt werden, weil zuerst herausgefunden werden musste, welche Fragen überhaupt an den Kunden gestellt werden mussten. Diese hatten schlussendlich z.B. die Form von "Wie Wichtig ist On Premise Hosting?", was unklar war, weil bis an hin auf einer Cloud Lösung gearbeitet wurde. Die Analysephase diente daher dazu eine Bewusstheit dafür zu schaffen, was die Ticketsysteme können und welche Rückfragen an den Kunden gestellt werden müssen.

Aus Erfahrungen mit Ticketsystemen und den Kriterien, die bereits bekannt waren, wurden Teilbereiche und Tabellen erstellt die in Kapitel 3.2 abgebildet sind. Das einzige Muss-Kriterium war bis zu diesem Zeitpunkt das Vorhandensein einer API, denn ansonsten hätte das Projekt nicht umgesetzt werden können.

Während dem Ausfüllen der Tabelle wurde bekannt, dass jedes Ticketsystem seine Eigenheiten hat, welche als Anmerkungen unterhalb der Tabellen festgehalten sind.

Vorentscheid

In der ersten Entscheidungsphase wurde Rücksprache mit dem Kunden genommen, um herauszufinden, wie Wichtig dem Kunden z.B. On Premises Hosting wäre. Anhand der Ergebnisse der Produktanalyse wurde nun nach Problemen gesucht, die bei Implementation, Datenimport und Integration oder Betrieb, Support und Wartung auftreten konnten. Anhand dieser Probleme und Bedenken wurde nun die Menge der Ticketsysteme auf zwei reduziert. Die Ausschlaggebenden Kriterien für ein Ausscheiden eines Systems sind in den Unterkapiteln wie z.B. in Unterkapitel 3.3.2 dokumentiert.

Entscheid

In der Sitzung mit Damien Vouillamoz wurden die in Kapitel 3.4 beschriebenen Empfehlungen durchgegangen, die Bedürfnisse und Favoriten des Kunden besprochen und anhand dessen die definitive Entscheidung getroffen. Dies ist im Sitzungsprotokoll vom 19.10.2017 festgehalten.

3.2 Produktanalyse

Nachfolgende Ticketsysteme wurden in den Evaluationsprozess miteinbezogen. Dieses Kapitel beinhaltet die Analyse ihrer Funktionen und Eigenschaften. Nachfolgende Nummerierung gilt für das ganze Kapitel.

1. Zendesk
2. osTicket
3. Manage Engine Servicedesk Plus
4. Freshdesk
5. Jira Service Desk
6. desk.com
7. Zoho
8. Zammad
9. ngDesk
10. SpiceWorks Helpdesk

Wie in Kapitel 3.1 erwähnt, war das Vorhandensein einer API bereits zu Beginn ein Kriterium. Es gab einige Ticketsysteme die diese Anforderung nicht erfüllten. Dazu gehören NgDesk und SpiceWorks Helpdesk. NgDesk's API war erst in Entwicklung und zur API von SpiceWorks Helpdesk wurden keine Informationen gefunden. Diese Systeme sind daher aus der Evaluation gefallen und in den nächsten weiteren Tabellen auch nicht mehr aufgeführt.

3.2.1 Allgemeine Anforderungen

Tabelle 3.1: Allgemeine Anforderungen

	Priorität	1	2	3	4	5	6	7	8
Kategorien / Abteilungen	Hoch	j	j	j	j	j	j	j	j
Subkategorien / Titel	Mittel	j	j	j	j	j	j	j	j
Felder: Titel, Text	Hoch	j	j	j	j	j	j	j	j
Benutzer spezifische Felder	Mittel	j	j	j	j	j	j	j	j
Verschiedene SLA	Niedrig	j	j	j	j	j	j	j	j
Priorisieren von Tickets	Hoch	j	j	j	j	j	j	j	j
Volltextsuche	Niedrig	j	j	j	j	j	j	j	j
Anhänge	Mittel	j	j	j	j	j	j	j	j
Tickets aus E-Mails erstellen	Mittel	j	j	j	j	j	j	j	j
Statistiken	Mittel	j	j	j	j	j	j	j	j
Automatische Antworten	Mittel	j	j	j	j	j	j	j	j
Knowledge-Base	Mittel	j	j	j	j	n	j	j	n
Importmöglichkeit	Hoch	j	n	j	j	j	j	j	j

Produkte: (1) Zendesk, (2) osTicket, (3) Manage Engine Servicedesk Plus, (4) Freshdesk , (5) Jira Service Desk, (6) desk.com, (7) Zoho, (8) Zammad

Verwendete Zeichen: (j) Ja, (n) Nein

Anmerkungen:

- Bei der Importmöglichkeit wurde zudem darauf geachtet, ob es eine spezifische Importmöglichkeit für Ticketdaten von Zendesk gibt, da der Kunde bis jetzt Zendesk verwendet. Diese Möglichkeit hatten nur Zendesk, Freshdesk und Zammad, was ihnen einen Vorteil in der späteren Diskussion der Evaluation gab. Die anderen Ticketsysteme konnten nur xls, csv oder andere nicht Zendesk spezifische Formate importieren.
- Die Knowledge-Base von Zammad ist noch in Entwicklung.
- Jira Service Desk benötigt für viele Features oft noch Confluence als Knowledge-Base und es schien eher geeignet als Teil einer grösseren Jira Umgebung aber als eigenständiges Produkt eher unpassend.
- Um Tickets in osTicket importieren zu können müsste man direkt auf die Datenbank zugreifen, da es Open Source ist wurde es aber trozdem noch weiter evaluiert. Das System wirkte allgemein eher etwas unausgereift.

3.2.2 Benachrichtigungen

Tabelle 3.2: Benachrichtigungen

	Priorität	1	2	3	4	5	6	7	8
E-Mail	Hoch	j	j	j	j	j	j	j	j
SMS	Niedrig	j	n	j	j	n	j	j	-
Push	Niedrig	j	n	-	j	j	-	-	-
Benachrichtigung nach Kategorie	Mittel	j	-	j	-	j	-	-	j

Produkte: (1) Zendesk, (2) osTicket, (3) Manage Engine Servicedesk Plus, (4) Freshdesk , (5) Jira Service Desk, (6) desk.com, (7) Zoho, (8) Zammad

Verwendete Zeichen: (j) Ja, (n) Nein, (-) Keine Informationen gefunden

Anmerkungen:

- desk.com, Jira Service Desk und Zendesk können SMS nur über eine externe Applikation verschicken.

3.2.3 Hosting

Tabelle 3.3: Hosting

	Priorität	1	2	3	4	5	6	7	8
On-premise hosting	Hoch	n	j	j	n	j	n	n	j
Betriebssystem Windows (w) Linux (l) OSX (o)	Mittel	-	-	w,l	w,l,o	-	-	-	l

Produkte: (1) Zendesk, (2) osTicket, (3) Manage Engine Servicedesk Plus, (4) Freshdesk , (5) Jira Service Desk, (6) desk.com, (7) Zoho, (8) Zammad

Verwendete Zeichen: (j) Ja, (n) Nein, (-) Keine Informationen gefunden

Anmerkungen:

- Im Laufe der Studienarbeit wurde vom Kunden definiert, dass aus Datenschutzgründen nur On Premise Lösungen in Frage kommen, zu Beginn war dies keine Anforderung.

3.2.4 API

Tabelle 3.4: API

	Priorität	1	2	3	4	5	6	7	8
User	Hoch	j	j	j	j	j	j	j	j
Admin	Mittel	j	n	j	j	j	j	j	j
Qualität	Hoch	gut	genügend	gut	sehr gut	gut	gut	gut	gut
Funktionsumfang	Hoch	gross	gering	mittel	gross	mittel	mittel	gross	gross
JSON / XML	Mittel	JSON	beides	XML	beides	JSON	JSON	JSON	JSON
Access Limit	Mittel	200/min	-	-	-	-	-	-	-

Produkte: (1) Zendesk, (2) osTicket, (3) Manage Engine Servicedesk Plus, (4) Freshdesk , (5) Jira Service Desk, (6) desk.com, (7) Zoho, (8) Zammad

Verwendete Zeichen: (j) Ja, (n) Nein, (-) Keine Informationen gefunden

Anmerkungen:

- JSON war das bevorzugte Format für die API

3.2.5 Kosten

Nachfolgend sind die Kosten nur im Groben dargestellt, da alle Kostenpflichtigen Ticketsysteme bis auf eines bereits während des Vorentscheids herausgefallen sind und dort nur eine Kostenübersicht erstellt wurde.

Tabelle 3.5: Kosten

System	Preis
1	Kostet je nach Funktionsumfang einen gewissen Betrag pro Monat
2	Gratis
3	Gratis bis 5 Mitarbeiter oder kostet je nach Funktionsumfang von 9 Fr. aufsteigend.
4	Gratis mit limitiertem Funktionsumfang oder Preis je nach Funktionsumfang ab 25 Fr. aufsteigend.
5	On Premise 10.- bis 3 Mitarbeiter 1700 bis 5
6	20-100 Fr. pro Jahr pro Mitarbeiter.
7	Gratis bis 10 Agenten. Ab 10 Agenten 25$
8	Gratis

Produkte: (1) Zendesk, (2) osTicket, (3) Manage Engine Servicedesk Plus, (4) Freshdesk , (5) Jira Service Desk, (6) desk.com, (7) Zoho, (8) Zammad

3.2.6 Weiteres

Zendesk, Freshdesk, Service Desk Plus und desk.com haben Mobile Apps für iOS und Android. Auf Anfrage beim Kunden stellte sich heraus, dass dies keine wichtige Anforderung sei, da die Agenten alles über das Webinterface ihre Arbeit erledigten.

3.3 Vorentscheid

Nachdem oben beschrieben wurde, welche Funktionen die Ticketsysteme anbieten, musste nun eingeschätzt werden, auf welches Ticketsystem der Entscheid fallen könnte und dazu musste die Liste Verkleinert werden.

Ziel war es Damien Vouillamoz zwei Ticketsysteme empfehlen zu können und mit ihm nach besprechen der Vorschläge zu einem endgültigen Entschluss zu kommen. In den Nachfolgenden Unterkapiteln ist begründet, warum die einzelnen Ticketsysteme herausgefallen sind.

3.3.1 On Premises Hosting

Gegen ende der Evaluation hat sich das Kriterium On Premises Hosting als sehr wichtig herausgestellt, was zu beginn nicht klar war, da bis jetzt auf einer Cloud Lösung gearbeitet wird. Dadurch sind Zendesk, Freshdesk, desk.com und Zoho herausgefallen.

3.3.2 Reife des Systems

osTicket schien allgemein eher unausgereift und kompliziert zu betreiben, es wurde daher auch aus der Kandidatenliste gestrichen. Zudem bietet es wenig Funktionalität über die API.

3.3.3 Passend für Bedürfnisse des Kunden

Jira Service Desk schien eher dafür gebaut worden zu sein, in einer grösseren Umgebung mit diversen anderen Jira Systemen betrieben zu werden. Da der Kunde aber keine solche Jira Umgebung hat, schien Jira eher unpassend für die Bedürfnisse des Kunden. Zudem wäre es im Vergleich zu Service Desk Plus und Zendesk um einiges teurer gewesen. Es wurde daher Entschieden, auch dieses System von der Kandidaten Liste zu streichen. Verbleibend waren Zendesk und Service Desk Plus.

3.4 Empfehlungen

3.4.1 Zammad

Zammad ist ein webbasierter Opensource Service Desk der Zammad Foundation. Dieser ist als Source, Docker-Image und für alle gängigen Linux Systeme verfügbar.
Zammad Webseite [14]

Pros

- Opensource
- Keine Kosten
- Import direkt von Zendesk

Cons

- Keine Knowledge Base
- Kein offizieller Support (optional)
- Keine App

Kostenpunkt: gratis

3.4.2 Service Desk Plus

Service Desk Plus wird von Manage Engine entwickelt. Die Basisversion ist für bis 5 Agenten kostenlos.
Service Desk Plus Webseite [8]

Pros

- Knowledge Base
- Offizieller Support
- iOS und Android App

Cons

- Import nur durch xls
- Altes Design
- XML API

Kostenpunkt: 995$ pro Jahr (10 Agents)
Demo: Demo Service Desk [3]

3.5 Entscheid

Beim Meeting wurde zusammen mit Damien Entschieden, dass die Vorteile von Zammad überwiegen und die Nachteile für diesen Fall weniger relevant sind. Folgende Punkte sind im Sitzungsprotokoll festgehalten:

- Für die Knowledge-Base gibt es auch andere Tools
- Man hat bereits gute Erfahrungen mit dem Support bei Zammad gemacht.
- innofield AG wird die Supportcases nie über mobile beantworten.
- Direkter import von Zendesk zu Zammad

Kapitel 4

Umsetzung

4.1 Übersicht

Das Kundenportal "myFlow" ist in 2 Komponenten unterteilt. Ein Frontend geschrieben in Angular sowie ein Backend in PHP. Die Angular-Applikation kommuniziert mit dem Backend über eine Web-API. Zudem gibt es im Backend weitere Schnittstellen zu Drittsystemen (Billing, Serverinfrastruktur, IP-Management usw.). Für dieses Projekt ist vor allem die Anbindung an das Helpdesk System zu erwähnen. Diese Schnittstelle wurde in der Studienarbeit implementiert.

Die Architektur des bestehenden abzulösenden Systems:

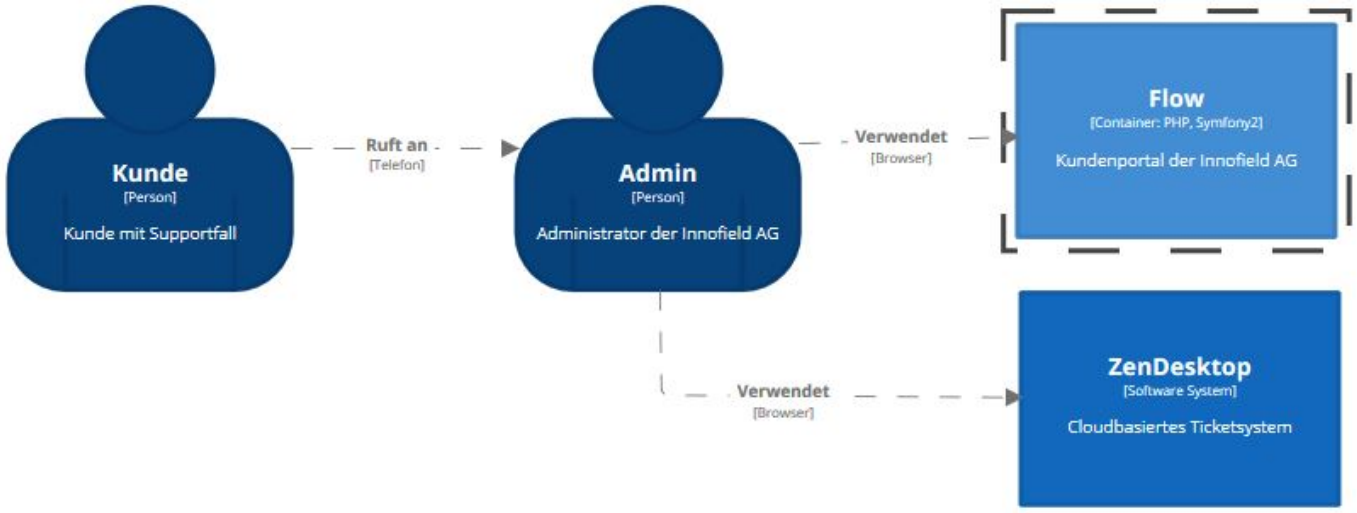

Abbildung 4.1: Architektur des alten Kundenportals

Neu wird die Architektur wie folgt aussehen:

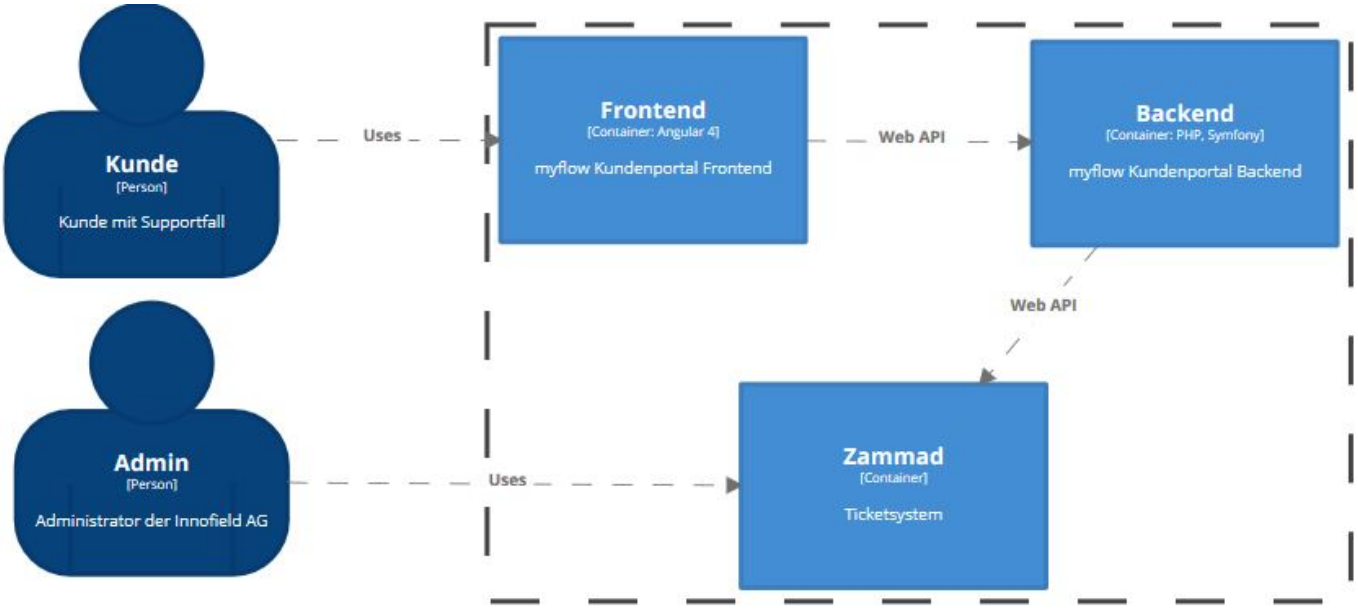

Abbildung 4.2: Architektur des neuen Kundenportals

4.2 Analyse

Das Domain Modell zeigt die für die SA wichtigen Klassen des myFlow-Portals. Es ist ein Auszug aus dem vollständigen Domainmodell des Portals.

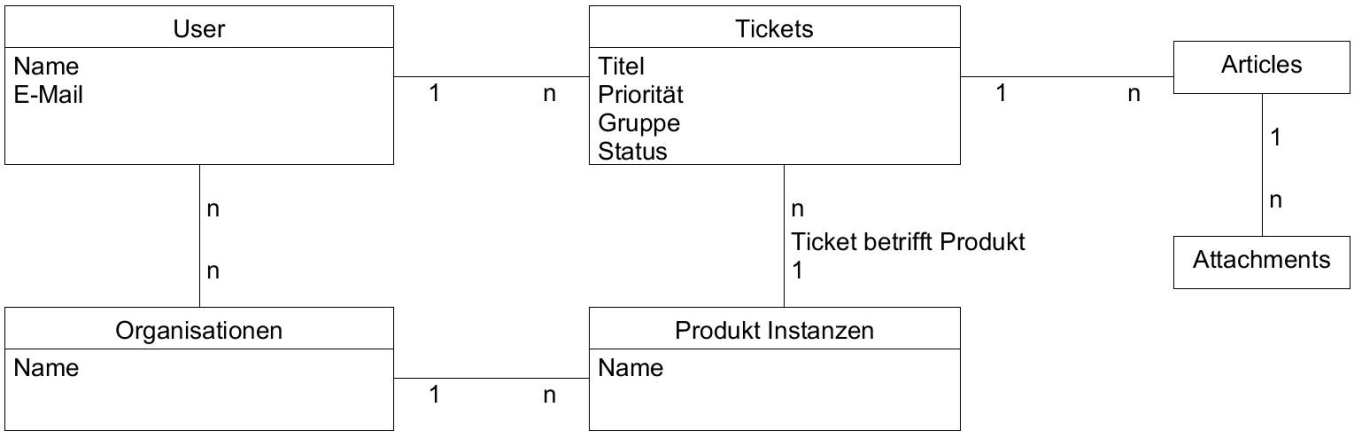

Abbildung 4.3: Auszug Domainmodell myFlow für SA

Auf Seiten Zammad ergibt sich ein ähnliches Bild. Zu beachten gibt es im speziellen die Verbindung zwischen Benutzer und Organisation. Zammad erlaubt es nicht mehrere Organisationen einem Benutzer zuzuweisen, in myFlow kann dies aber der Fall sein.

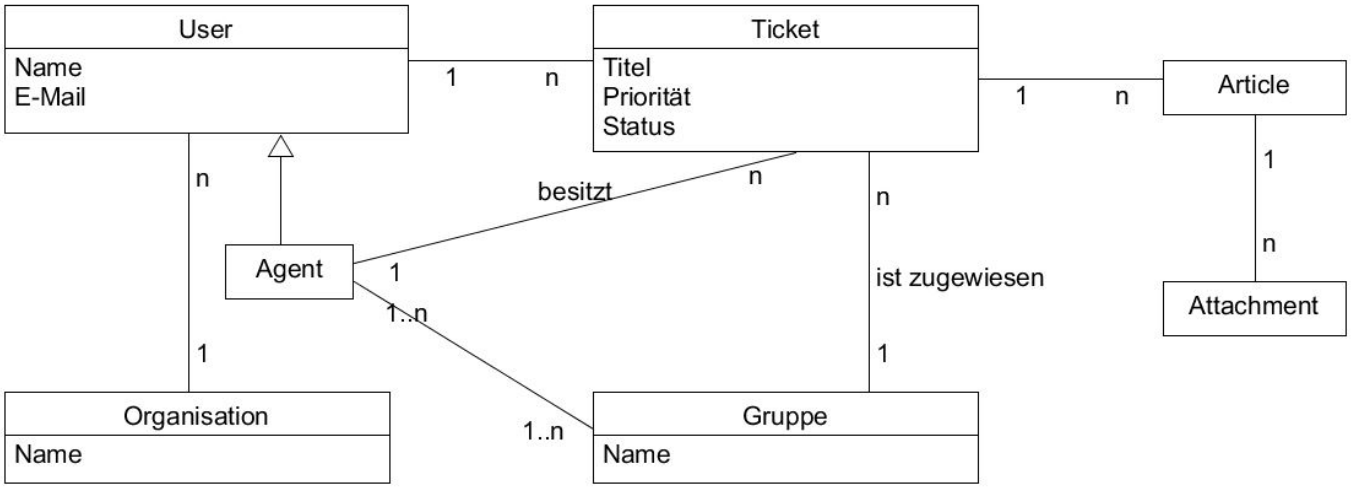

Abbildung 4.4: Domainmodell für Zammad

4.3 Konfiguration von Zammad

Nachfolgend sind die Konfigurationen beschrieben, die an Zammad vorgenommen werden mussten.

4.3.1 Allgemeines

In Zammad wurden folgende kleinere Konfigurationen vorgenommen:

- Erstellen von Benutzern und Zuweisung von Rollen (Agent, Benutzer, Admin)
- Erstellen von Gruppen wie Support und Billing
- Erstellen von Organisationen
- Erstellen von SLAs
- Erstellen von Übersichten zur Verwaltung der Tickets
- Konfiguration des E-Mail Services, was im nächsten Kapitel genauer beschrieben ist.

4.3.2 E-Mail

Das Parsen von E-Mails funktioniert in Zammad indem es zugriff auf eine Mailbox erhält und alle E-Mails per IMAP herunterlädt. Danach erstellt es für jedes einzelne E-Mail in der Inbox einen Benutzer, falls die Senderaddresse nicht bereits in der Datenbank vorhanden ist und ein neues Ticket für jenen Benutzer. Als nächsten Schritt antwortet Zammad auf die E-Mail per SMTP.

Um dies zu konfigurieren wurde eine neue E-Mail Adresse bei Gmail erstellt und danach im Zammad eingetragen. In der Produktiven Umgebung werden dort die Support Mailboxen der innofield AG eingetragen.

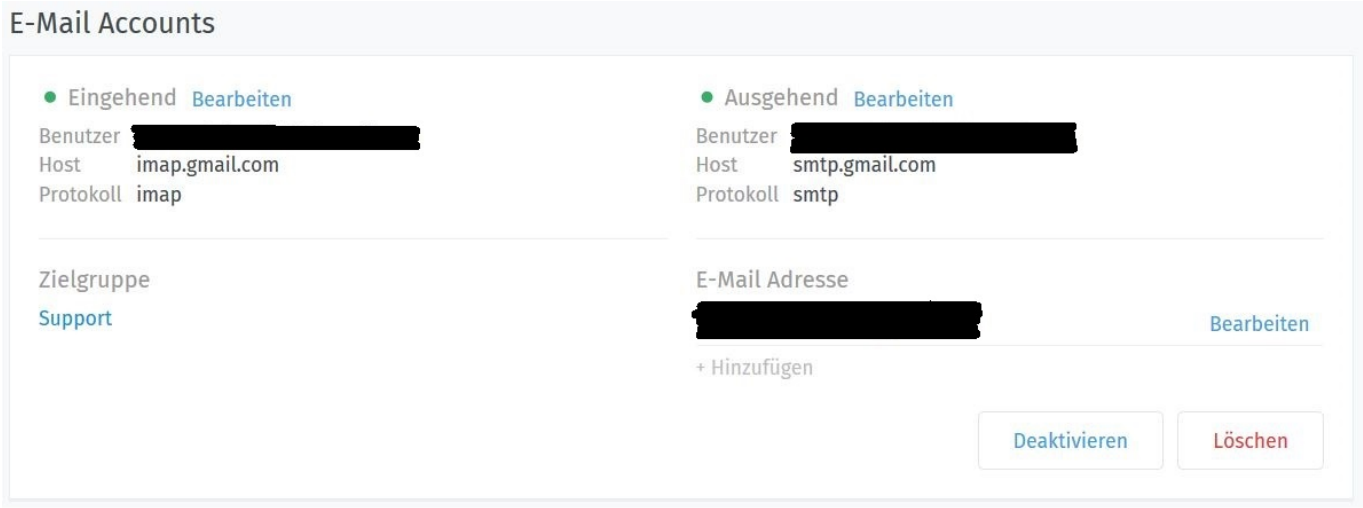

Abbildung 4.5: Konfiguration einer Mailbox

Danach konnten E-Mails an die eingetragene Adresse geschickt werden, für welche dann automatisch ein Ticket erstellt und eine eine Antwort gesendet wurde.

4.3.3 Ticketstatus

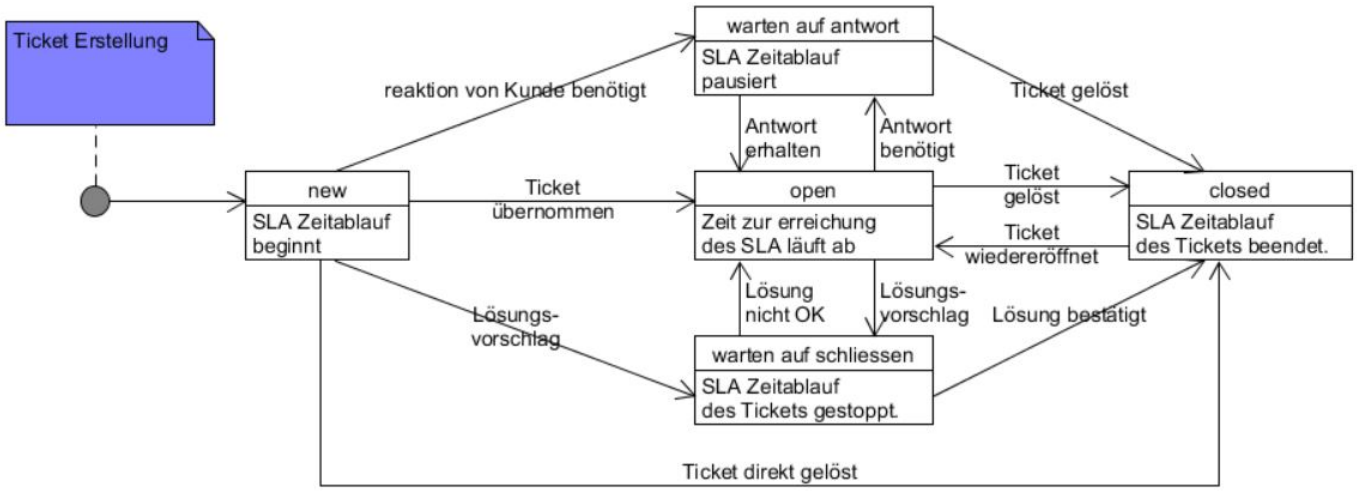

Abbildung 4.6: Zustandsdiagramm zum Ticketstatus

4.3.4 Ticket via E-Mail senden

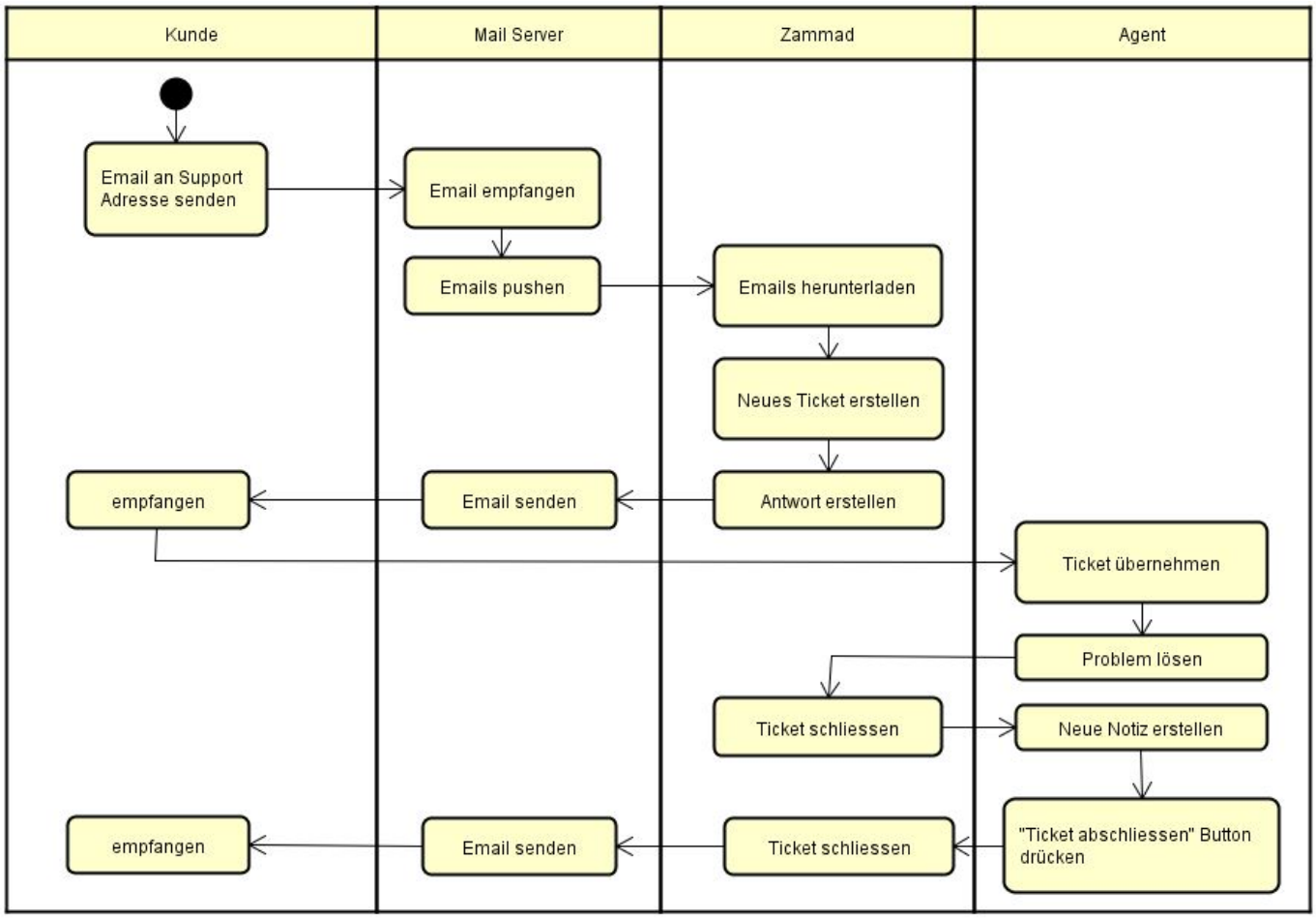

Abbildung 4.7: Activity Diagramm Ticket via E-Mail senden

4.4 Backend

Im folgenden werden Details zur Umsetzung des Backends bzw. der Web-API erläutert.

4.4.1 Bestehende Applikation myFlow

Als Grundlage diente die bestehende Applikation. Diese stellt die grundlegenden Funktionalitäten wie Autorisierung, Konfiguration, Datenbankzugriff, Suchfunktion usw. zur Verfügung und definiert die Struktur des Projekts. Sie ist auf dem PHP Framework Symfony aufgebaut.

4.4.2 Technologiestack myFlow

Die Grafik zeigt die verwendeten Technologien und deren Aufbau.

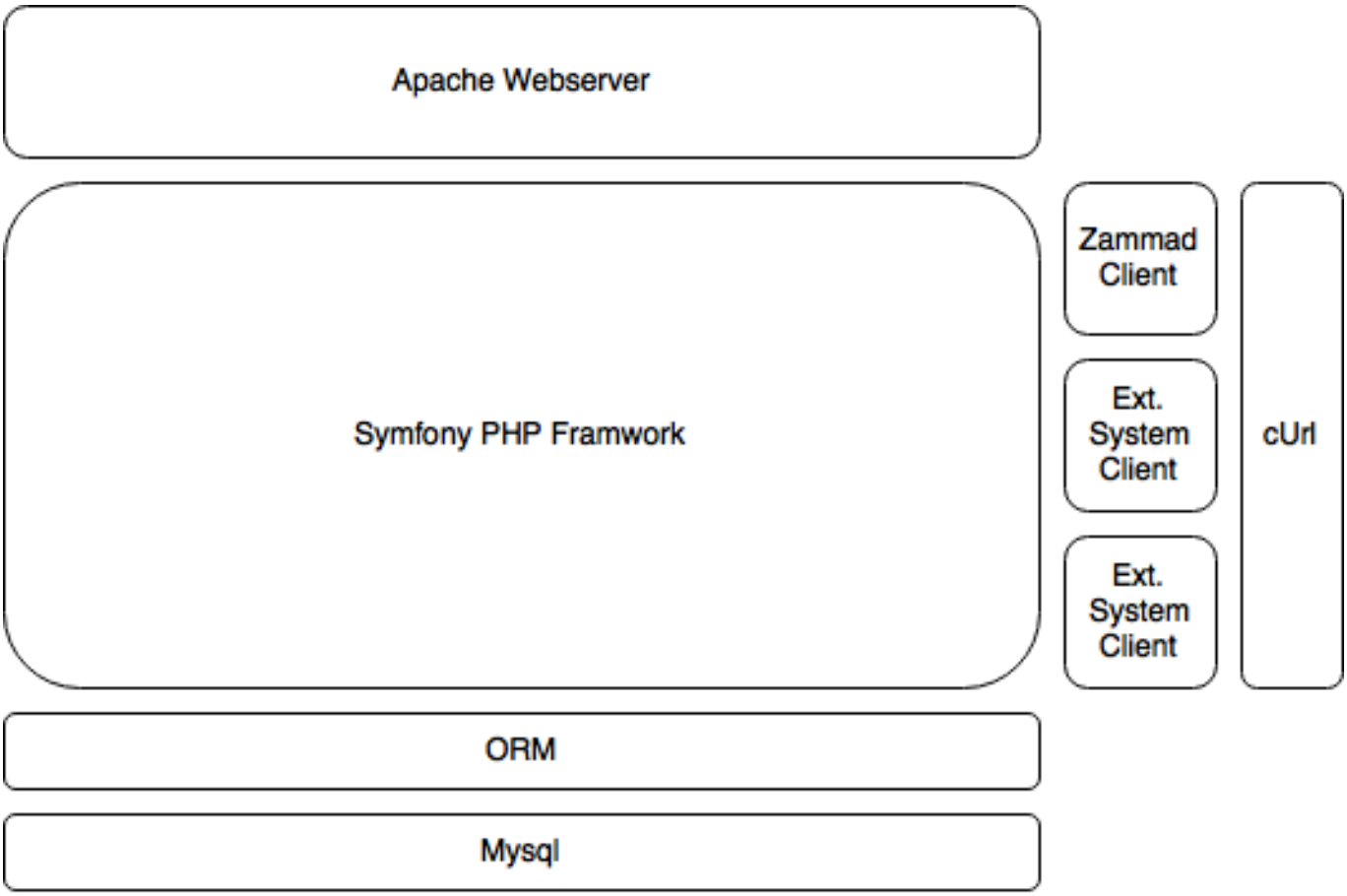

Abbildung 4.8: Technologiestack des Backends

Als Datenbank wird Mysql verwendet, darüber befindet sich ein ORM Layer welcher das Arbeiten mit Entitäten vereinfacht. Die Auslieferung der Daten übernimmt ein Apache2 Webserver. Zur Anbindung externer Systeme per API wird cUrl verwendet.

4.4.3 API

Die API ist das Herzstück des Backends. Sie wird sowohl vom Frontend als auch von externen Drittsystemen verwendet. Alle Daten werden im JSON Format übertragen.

Dokumentation

Zur Dokumentation der Schnittstelle wurde API Blueprint verwendet. Es werden alle verfügbaren Requests und deren Parameter sowie Antwortschemas beschrieben. Mit der Software Aglio[2] wird aus der Spezifikation HTML generiert. Die Spezifikation ist als HTML im Source-Code Archiv zu finden als "api-doc.html".

Urls

Der grundlegende Aufbau von Routes ist wie folgt: "/v2/organisations/orgID/entity/entID/params". Jede Route beginnt mit der Version der API gefolgt von der Organisation für welche die Aktion ausgeführt werden soll. Danach wird der gewünschte Entitättyp angegeben. In unserem Fall war das meistens "Tickets". Je nach Aktion muss auch die ID der Entität angegeben werden. Danach können weitere Parameter folgen. Je nach Aktion wurden GET bzw. POST Requests verwendet.

4.4.4 Implementation

Bei der Implementation wurde viel Wert darauf gelegt bereits bestehende Funktionalitäten zu nutzen statt alles neu zu erfinden. Bereits von der Applikation angebotene Features wie Pagination, globale Suche und Datanbankzugriff sowie auch Caching wurden, wenn immer möglich, nicht neu implementiert. Dies soll auch den weiteren Ausbau des Support Moduls vereinfachen.

4.4.5 Abgrenzung des Codes

Zur genauen Abgrenzung werden hier alle erstellten Dateien aufgelistet. Die mit (m) gekennzeichneten Datein wurden nur modifiziert und nicht komplett erstellt in dieser Arbeit.

- composer.json (m)
- src/AppBundle/DataFixtures/ORM
  - LoadUserData.php (m)
  - LoadKnowledgeArticleData.php
- src/AppBundle/doc
  - documentation.apib (m)
- src/AppBundle/Controller
  - SupportController.php
- src/AppBundle/Handler
  - SupportHandler.php
- src/AppBundle/Helpers
  - ZammadDtoConverter.php
- src/AppBundle/Service
  - SupportDeskService.php
  - ZammadService.php
- src/AppBundle/Normalizer
  - BriefTicketNormalizer.php
  - TicketNormalizer.php
  - ProductNormalizer.php
  - ArticleNormalizer.php
  - AttachmentNormalizer.php
  - KnowledgeArticleNormalizer.php
  - VerboseTicketNormalizer.php
  - VerboseProductNormalizer.php
  - VerboseArticleNormalizer.php
  - VerboseAttachmentNormalizer.php
  - VerboseKnowledgeArticleNormalizer.php
- src/AppBundle/Command

  - ImportZammadUserCommand.php
- src/AppBundle/Security
  - GrantCheck.php (m)
- src/AppBundle/Entity/Support
  - Ticket.php
  - Article.php
  - Attachment.php
  - KnowledgeArticle.php
- src/AppBundle/Resources/config
  - api-routing.yml (m)
  - services.yml (m)
- app/DoctrineMigrations
  - Version20171117114813.php
  - Version20171130092211.php
- app/config
  - parameters.yml.dist (m)
- test/AppBundle/Controller
  - SupportControllerTest.php
- test/AppBundle/Helpers
  - ZammadDtoConverterTest.php
  - ZammadArticleMock.php
  - ZamamdTicketMock.php
- test/AppBundle/Service
  - ZamamdServiceTest.php

Enititäten

Um die Tickets welche von Zammad kommen intern besser verarbeiten zu können, wurden eigene Entitäten für Tickets, Artikel (Kommentare) und Attachments erstellt. Diese wurden dann im System verwendet, so dass nur die Schnittstelle zum Helpdesk die für Zammad spezifischen Entitäten kennen musste. Somit wurde auch ein Wechsel des Helpdesks in der Zukunft vereinfacht.

SupportController

Für die klare Abgrenzung wurde ein neuer Controller erstellt, welcher sich um alle Anfragen bezüglich Helpdesk kümmert. Die URL Parameter werden automatisch beim Methodenaufruf übergeben. Der Controller ist auch für die Überprüfung der Berechtigungen, sowie das Zusammenstellen der Antwort zuständig.

SupportHandler

Damit die Controllerklasse nicht zu gross wird, wurde die Logik in eine Handlerklasse ausgelagert. In dieser Klasse werden Entitäten erstellt, Parameter konvertiert und der SupportDeskService angesprochen.

ZammadDtoConverter

Ist zuständige für die Konvertierung der Dtos (Data Transfer Objects) in die Entitäten des Systems. Der Konverter wurde als abstrakte Klasse implementiert und stellt statische Methoden für jede Entität (Ticket, Artikel, Attachment) zur Verfügung.

Interface SupportDeskService

Um das konkrete System hinter dem Helpdesk zu abstrahieren, wurde ein Interface eingeführt, gegen welches der Handler programmiert ist. Das erlaubt das einfache Austauschen des Services.

ZammadService

Diese Klasse ist die konkrete Implementierung des SupportDeskService Interfaces. Sie beinhaltet die Funktionalität zum Erstellen und Abrufen der Tickets von Zammad. Dort wird auch der ZammadAPIClient instantiiert welcher die eigentlichen Requests der API übernimmt.

ZammadAPIClient

Ist eine externe Dependency welche frei auf Github[1] verfügbar ist und entwickelt wurde, um die Zammad API[13] anzusprechen. Sie generiert die benötigten cUrl Aufrufe für die einzelnen Entitäten.

Normalizer

Die Normalizer übernehmen die Konvertierung von Entitäten (Tickets, Artikeln und Attachments) zu JSON. Sie definieren welche Felder über die API offengelegt werden und wie die Formatierung aussieht. Pro Entität kann es mehrere Normalizer geben die unterschiedliche Ausgaben produzieren. Z.B. werden bei der Auflistung aller Tickets, nicht alle Details zu den einzelnen Artikeln angezeigt. Diese erscheinen erst bei der Abfrage eines spezifischen Tickets.

4.4.6 PHP

Die Applikation basiert auf PHP 7.1, der zur Zeit des Projektes aktuellsten Version von PHP. Es wurde versucht, möglichst robusten PHP-Code zu schreiben. Dies beinhaltet die Aktivierung von ”strict types” und das verwenden von Typehinting für Funktionen und Konstruktoren. PHP ist aber generell nicht ”strongly typed” und garantiert damit keine Typensicherheit. Auch definiert der benutzte API-Client keine klaren Typen für die Dtos.

Coding Guidelines

Als Coding Guidelines wurde der PSR-1[5] und PSR-2[6] Standard festgelegt. Diese Guidelines wurden im PHP-Code der Arbeit durchgehend eingehalten. Dies gilt nicht für bereits bestehenden Code welcher teilweise abweichende Guidelines implementiert.

Coding Metrics

Zur Optimierung der Wartbarkeit wurde PHPMetrics[2] eingesetzt welches eine statische Codeanalyse durchführt und mögliche Fehler hervorhebt. Es hat sich jedoch als schwierig herausgestellt diese Metriken nur auf den Code der Studienarbeit zu begrenzen. Die Analyse wurde daher auf die ganze Applikation ausgeführt. Deshalb sind viele der gefundenen Probleme Teil der Symfony-Architektur. Der genaue Bericht findet sich im abgegeben Code unter ”/report”.

4.4.7 Testing

Um die Qualität und Wartbarkeit weiter zu erhöhen wurden UnitTest’s für alle Features implementiert, basierend auf dem PHPUnit Framework. Das Testing ist auch in den Deploymentprozess integriert.

4.4.8 Performance

Im Laufe der Arbeit wurden Performanceprobleme beim Backend festgestellt. Zuerst wurde unsere Anbindung an Zammad überprüft, dabei aber keine Probleme entdeckt. Daher wurden, nach Absprache mit dem Kunden, weitere Untersuchungen durchgeführt. Diese haben ergeben dass die Probleme auf die Konfiguration der uns zu Verfügung gestellten Server zurückzuführen sind. Die Probleme bestanden bis ans Ende dieser

[1] https://github.com/zammad/zammad-api-client-php
[2] http://www.phpmetrics.org/

Arbeit. Wie auch im Usabilitytest bemängelt, wird dadurch die Userexperience in Mitleidenschaft gezogen. Die von uns gemessenen Zeiten für den Abruf der Tickets von Zammad sind im grünen Bereich. Für den Kunden besteht aber Optimierungsbedarf auf deren Seite.

4.5 Frontend

4.5.1 Wireframes

Als erster konzeptioneller Entwurf des Support Moduls wurde mit Balsamiq ein clickable Wireframe erstellt um herauszufinden, wie die Use Cases umgesetzt werden könnten. Die nachfolgenden Unterkapitel beschreiben die einzelnen Views des konzeptionellen Entwurfs.

Ticket Overview

In der Ticket Overview soll Use Case 02[3] , 03[4] und 04[5] umgesetzt werden. Das bedeutet, diese Komponente soll eine Übersicht über alle offenen und geschlossenen Tickets geben und diese Liste sollte zudem durchsuchbar sein.

Daraus sind folgende zwei Views entstanden.

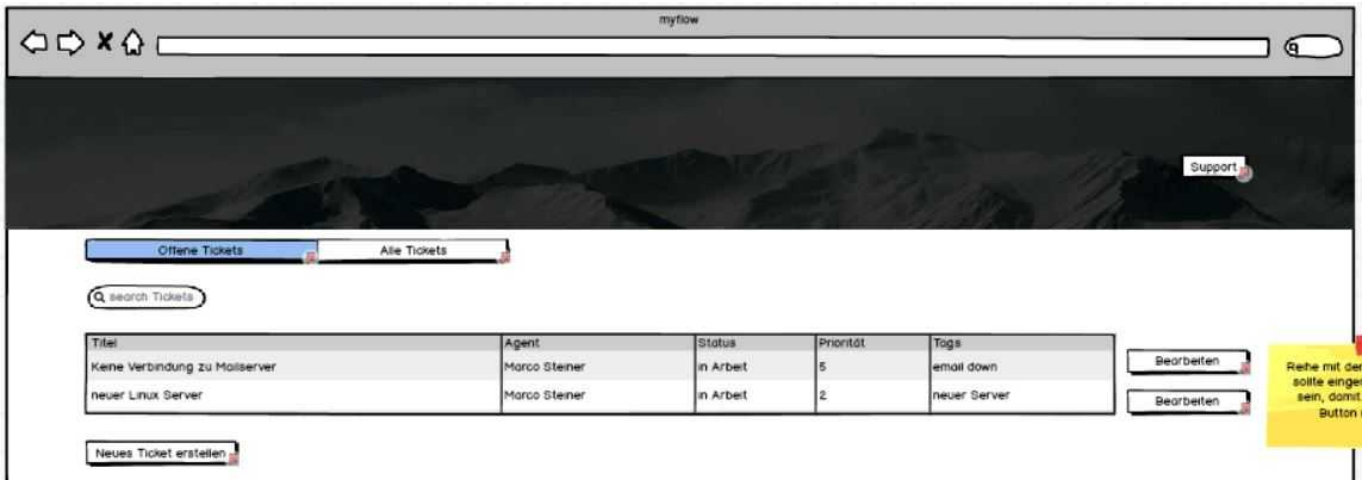

Abbildung 4.9: View zu Open Tickets

Abbildung 4.10: View zu All Tickets

[3]UC 02 Alle offenen Tickets anzeigen
[4]UC 03 Tickets durchsuchen
[5]UC 04 Alle Tickets anzeigen

New Ticket

In New Ticket soll ein neues Ticket erstellt werden können und falls möglich zudem noch Dateien direkt beim erstellen angefügt werden können. Es soll also Use Case 01[6] und 06[7] umsetzen.

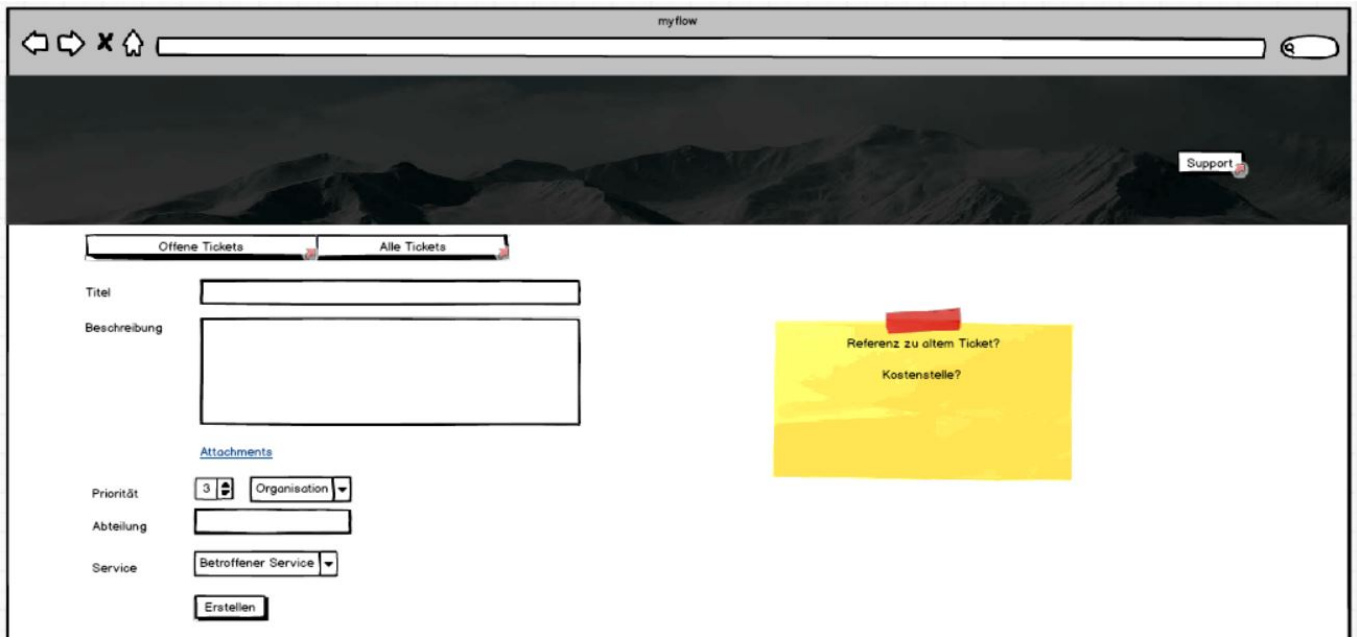

Abbildung 4.11: View zu New Ticket

[6]UC 01 Benutzer erstellt ein neues Ticket
[7]UC 06 File hinzufügen

Ticket

In Ticket soll ein Ticket angezeigt und bearbeitet werden können. Es soll also Use Case 05[8], 06[9], 07[10] und 08[11] umsetzen.

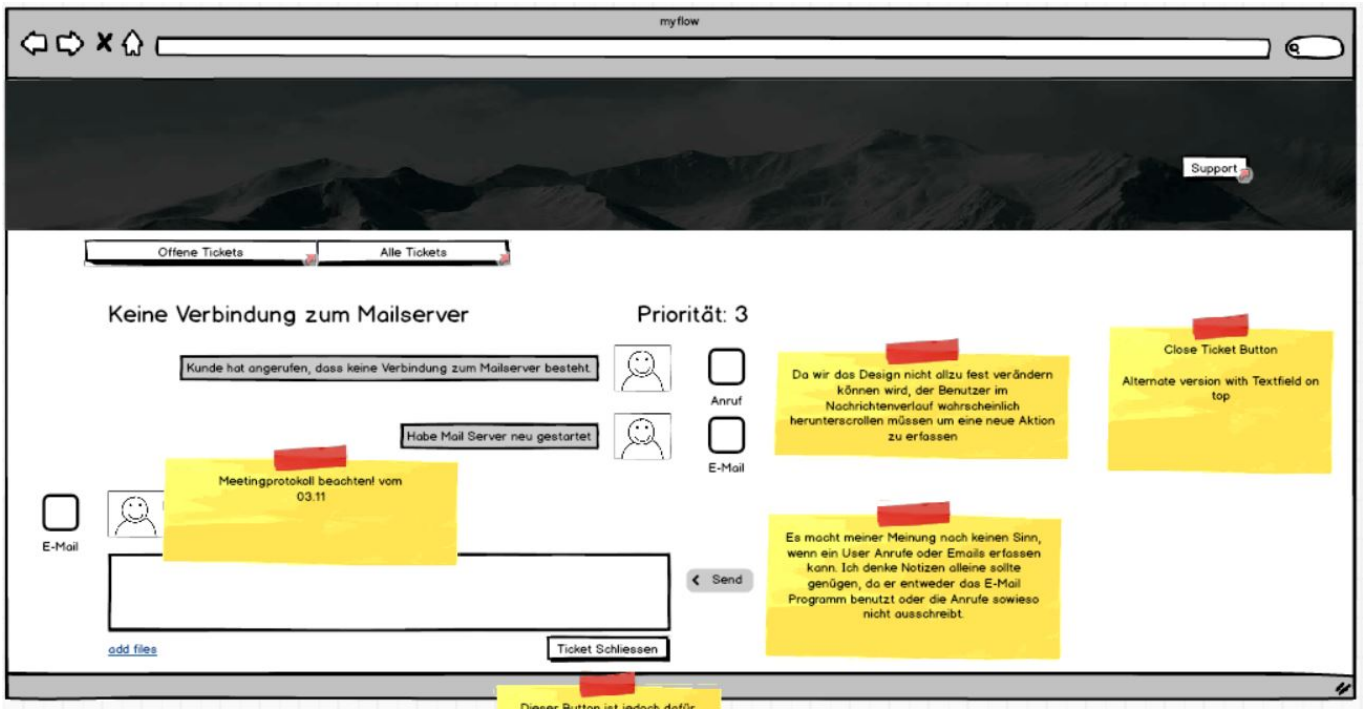

Abbildung 4.12: View zu Ticket Version A

Da der Nachrichtenverlauf eines Tickets sehr lange werden kann, wurde zudem noch eine zweite Version erstellt, bei welcher der Nachrichten verlauf unterhalb des Eingabefeldes ist, damit man schneller neue Eingaben tätigen kann und nicht immer ganz nach unten scrollen muss. Schlussendlich wurde entschieden, die zweite Variante zu verwenden.

[8] UC 05 Ticket Bearbeitungsverlauf ansehen
[9] UC 06 File hinzufügen
[10] UC 07 Notiz für Agenten zu verlauf hinzufügen
[11] UC 08 Ticket abschliessen

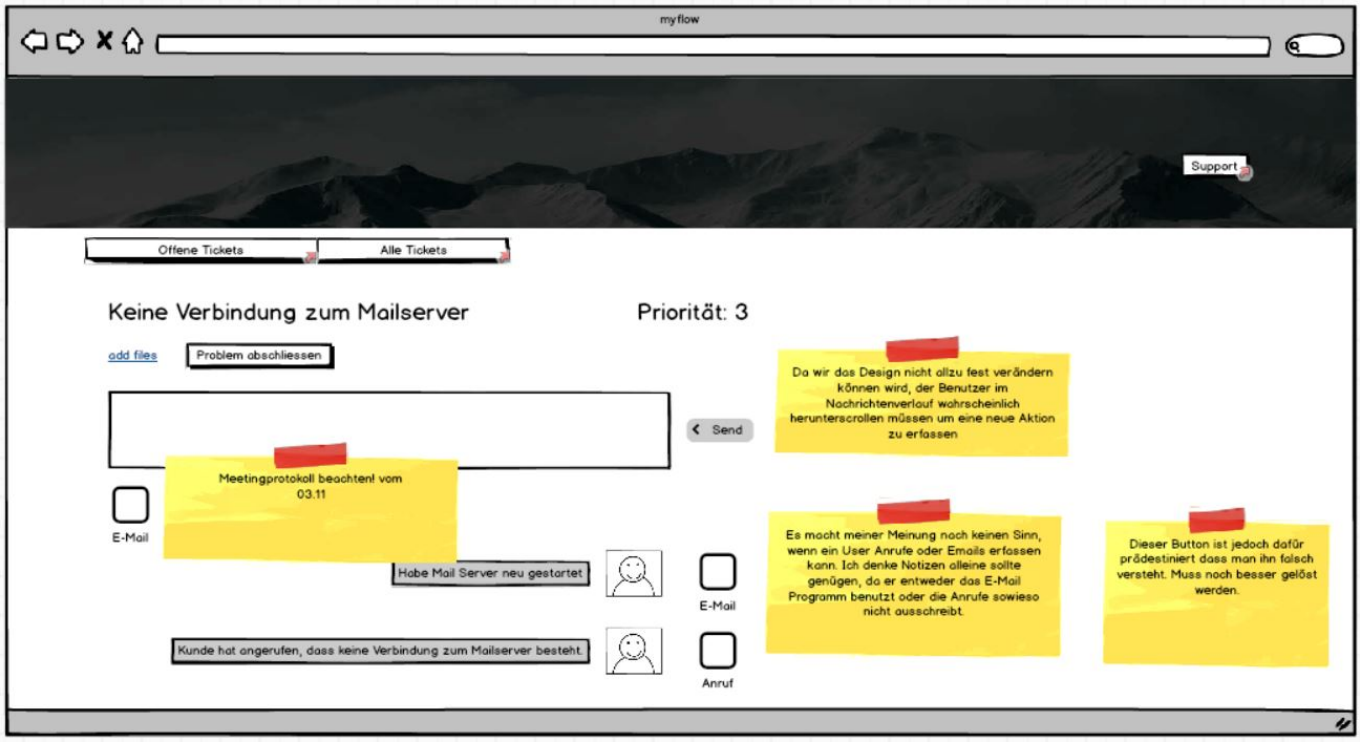

Abbildung 4.13: View zu Ticket Version B

4.5.2 Implementation

Nachfolgend wird die Umsetzung der Wireframes jeweils nur in einigen Sätzen beschrieben, auf interessante Aspekte der Implementation wird jedoch etwas genauer eingegangen.

Das Resultat der Arbeit am Frontend ist in folgendem Komponentendiagramm dargestellt.

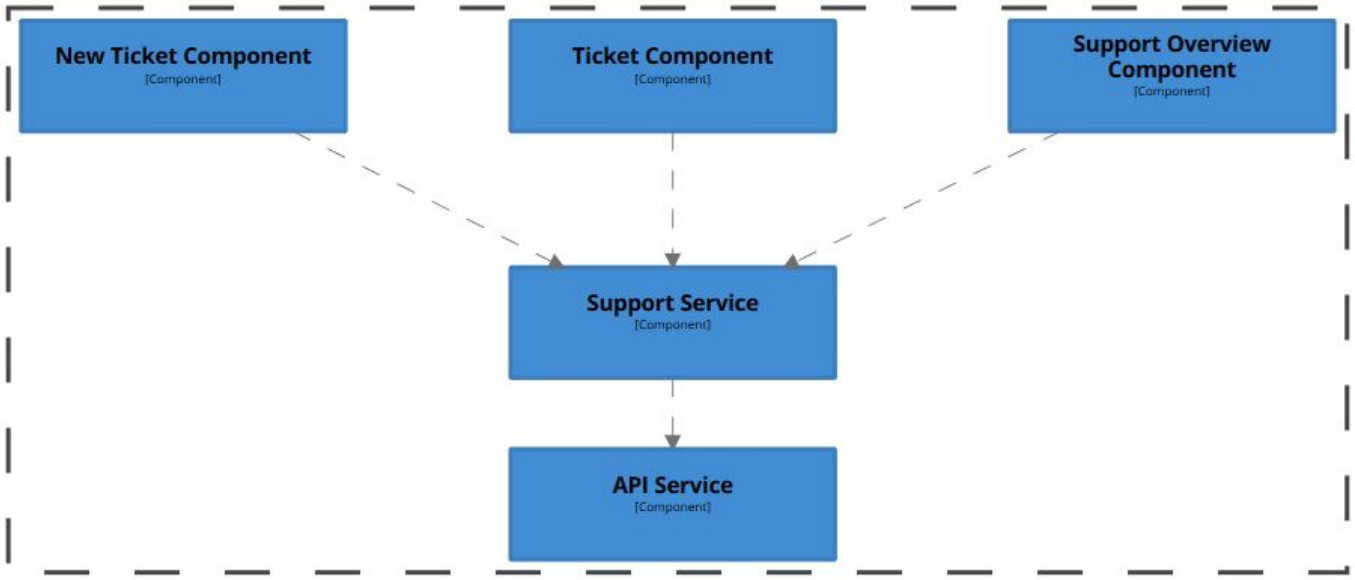

Abbildung 4.14: Komponentendiagramm des Frontends

Support Overview Component

Wie bereits in den Wireframes erwähnt, sollte diese Komponente eine Übersicht über alle offenen und geschlossenen Tickets geben und zudem durchsuchbar sein.

Die Komponente wurde umgesetzt indem eine Tabelle mit den für eine Übersicht wichtigen Informationen, wie Titel, Status etc. angezeigt wird. Um zwischen allen und nur offen Tickets zu unterscheiden, wird die Komponente für zwei unterschiedliche URL instantiiert und anhand der URL eine andere abfrage an das Backend geschickt, wodurch die Komponente dann je nach dem entweder die Informationen zu allen Tickets erhält oder nur diejenigen zu den offenen Tickets. Leider konnte die Suche nicht mehr implementiert werden.

New Ticket Component

In New Ticket Component soll ein neues Ticket erstellt werden können und falls möglich zudem noch Dateien direkt beim erstellen angefügt werden können.

Umgesetzt wurde dies indem die Daten in einem Formular erfasst werden. Je nach gewähltem Wert im Formular, werden dem Benutzer vor dem Senden gewisse Meldungen angezeigt, wie z.B. dass ein Ticket, welches fälschlicherweise mit der höchsten Priorität erstellt wurde zusätzlich 50 Fr. kostet. Beim erstellen eines Tickets kann leider noch nicht direkt eine Datei angefügt werden. Dies ist erst möglich, wenn das Ticket bereits erstellt ist.

Ticket Component

In der Ticket Komponente soll ein Ticket angezeigt und bearbeitet werden können.

Die Implementation kann alle Nachrichten anzeigen, Dateien anfügen und wieder öffnen, neue Nachrichten hinzufügen, das Ticket abschliessen und wieder eröffnen. Für den File up und download wurden folgende zwei Artikel zu Hilfe genommen [7] und [11].

Support Service

Support Service ist ein Abstraktionslayer zwischen API Service und den restlichen Komponenten. Je nach Aufruf reicht er das Resultat direkt weiter oder Validiert die Daten und reicht das Resultat erst dann weiter an den Aufrufer. Die Validierung ist nachfolgend genauer beschrieben.

API Service

Der API Service ist die Schnittstelle zwischen Front- und Backend. Der Service Kommuniziert mit dem Backend über HTTP.

Validierung

Die Daten, welche vom Backend gesendet werden, werden im Support Service validiert. Dabei wird mit der Klasse ArrayParser ein Array des gewünschten Entites erstellt, da die Daten meistens als Array zurück gegeben werden und dieser Code sehr repetitiv war. Danach wird ein Validator Objekt vom Typ des jeweiligen Entities erstellt, welches mit der Methode fromJson() prüft ob alle notwendigen Werte der erhaltenen Daten "truthy" sind. Die Methode fromJson() typifiziert dann jedes Property welche das Entity erwartet und erstellt schlussendlich ein Objekt des Entites aus den erhaltenen Daten. Falls ein Wert nicht 'truthy' ist, dann wird ein Error mit einer entsprechenden Fehlermeldung geworfen.

Testing

Zur einarbeitung wurde für das Testing ein Online Kurs [9] durchgearbeitet. Zudem wurde mit dem Angular Tutorial [10] gearbeitet. Tests existieren nur für die unteren Layer. Die UI Komponenten müssen später noch von einem Designer überarbeitet werden. Leider gibt es jedoch dort zwar wenig aber ungetestete Logik. Aus Zeitmangel hat es nicht mehr für ein weitergehendes Refactoring und Testing dieser Komponenten gereicht. Die Tests zur Validationslogik sind etwas komplizierter und eventuell interessant.

4.5.3 Abgrenzung des Codes

Zur genauen Abgrenzung werden hier alle erstellten und bearbeiteten Dateien aufgelistet:

- **/src/app/support** Alle in diesem Ordner erstellten Dateien sind nur während der Studienarbeit erstellt oder bearbeitet worden. Der gesamte Inhalt des Ordners gehört zur Studienarbeit
- **/src/app/core/api.service.ts** Im API Service wurden folgende Methoden während der Studienarbeit erstellt:
  1. getAssociatedProducts()
  2. getTickets()
  3. getOpenTickets()
  4. getSpecificTicket()
  5. getAttachment()
  6. postNewTicket()
  7. postNewArticle()
  8. postCommand()
- **/src/app/core/api.service.spec.ts** Im API Service Test wurden folgende Testmethoden während der Studienarbeit erstellt, die Reihenfolge der Testmethode stimmt mit der Reihenfolge der oben beschriebenen Methoden überein:
  1. it('it should get all associated Products'...
  2. it('it should get all tickets'...
  3. it('it should get all open tickets'...
  4. it('should get a ticket by its ID'...

5. it('should get an attachment'...
6. it('should post a new ticket'...
7. it('should post a new article to a specific ticket'...
8. it('should post a command for operations like closing a ticket'...

- **/src/app/entity** Die Nachfolgenden Entites wurden während der Studienarbeit erstellt:
  - attachment.ts
  - article.ts
  - ticket.ts
  - ticket-overview-entry.ts

Quellen

Für die Einarbeitung in Angular und auch durchwegs während der Arbeit wurden die angular.io Tutorial Seite [10] und die Fundamentals Section [4] verwendet.

4.6 Migration

Ein Teil der Semesterarbeit war zudem die Portierung der Daten aus dem alten System ins neue. Das betrifft einerseits ZenDesk aber auch die vorhandenen Benutzer von myFlow.

4.6.1 ZenDesk

Zammad unterstütz bereits einen direkten Import aller Daten von ZenDesk. Dies war ein wichtiger Punkt für den Entscheid. Da die Tickets vertrauliche Daten enthalten wurde gemeinsam mit dem Kunden entschieden den Import im Umfang dieser Arbeit nicht mit Live-Daten durchzuführen. Dieser Import wird erst durchgeführt wenn der Server in eine gesicherten Umgebung verschoben wurde.

4.6.2 myFlow

Zusätzlich zum Import von ZenDesk wurde ein eigener Import für myFlow erstellt. Mit diesem Import wird in Zammad für jeden myFlow-Benutzer ein Login erstellt. Pro Benutzer wird jeweils ein Passwort generiert und in der Datenbank gespeichert. Zusätzlich werden die User in Zammad den Organisationen zugewiesen.

Kapitel 5

Resultat

5.1 Produkt und Resultat

Nachfolgend werden die Anforderungen welche in Unterkapitel 2.1.1 definiert wurden einzeln durchgegangen und unsere Resultate dazu beschrieben.

5.1.1 Maske zum erstellen von Tickets

Wie in Unterkapitel 2.1.1 beschrieben, muss das Portal eine Maske zum erstellen neuer Tickets besitzen. Sie wurde, wie in Abb. 5.1 dargestellt, umgesetzt.

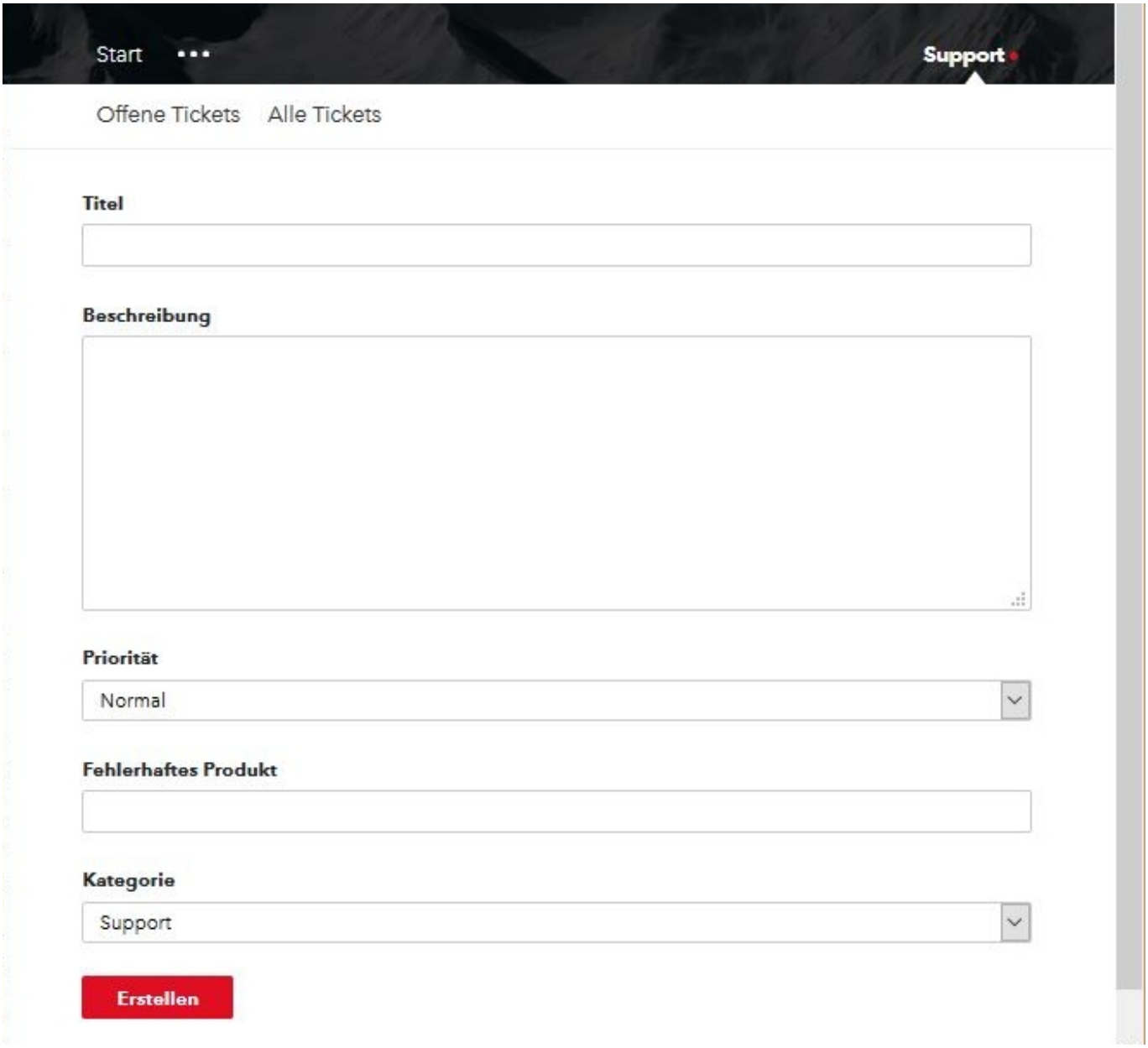

Abbildung 5.1: New Ticket Dialog

Erfassbare Daten

Wie in Abb. 5.1 ersichtlich, können beim erstellen eines Tickets Titel, Nachricht, Gruppe, Priorität und das Betroffene Produkt erfasst werden.

Anhänge können erst erfasst werden, wenn das Ticket bereits erstellt ist. Sie können im Specific Ticket Dialog, der in Abb. 5.2 gezeigt wird, erfasst und wieder geöffnet werden.

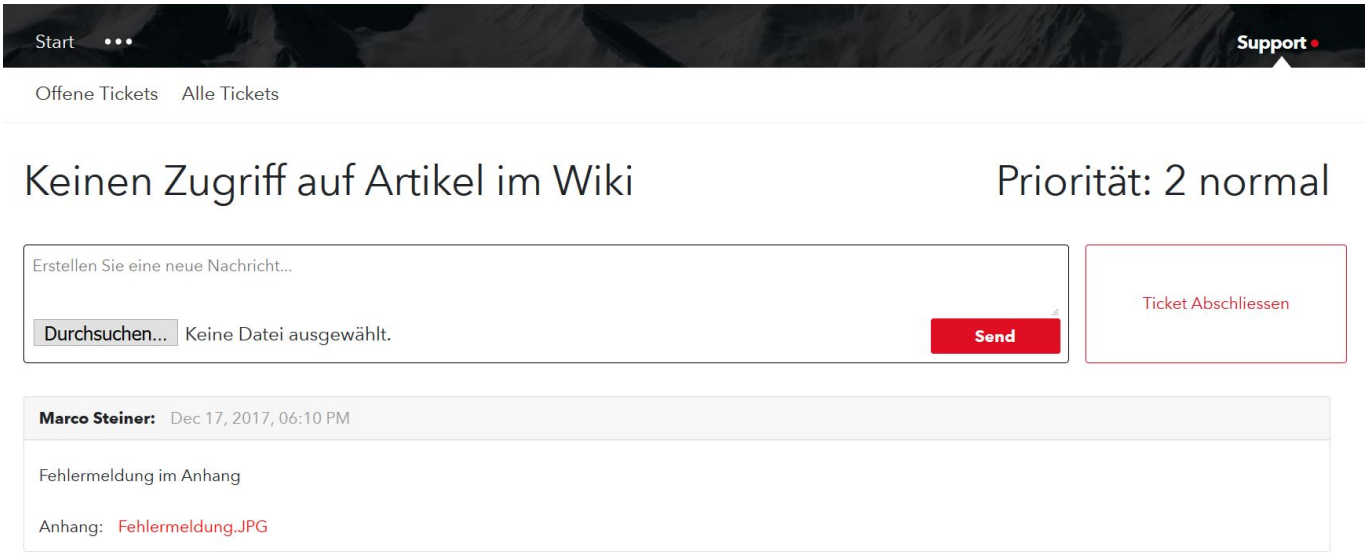

Abbildung 5.2: Specific Ticket Dialog

Warnhinweis für missbräuchliches setzen von Prioritäten

Der Warnhinweis um eine missbräuchliche Nutzung der Priorität "Sehr Hoch" zu verhindern, wurde wie in Abb. 5.3 dargestellt, umgesetzt. Dieser Warnhinweis erscheint, sobald ein Benutzer das Formular um das Ticket zu erstellen mit der Priorität "Sehr Hoch" abschickt.

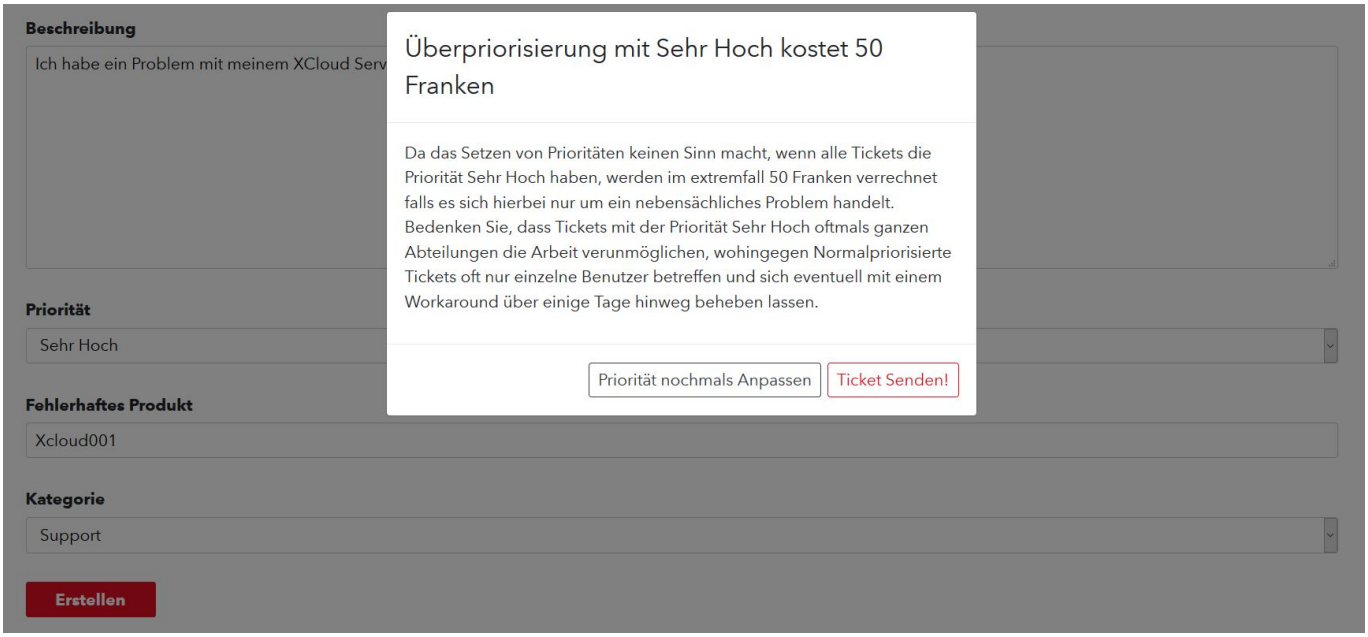

Abbildung 5.3: Warnhinweis für missbräuchliches setzen von Prioritäten

5.1.2 Autocomplete für das Feld Betroffenes Produkt

Abb. 5.4 zeigt das Feld "Betroffenes Produkt" aus dem "New Ticket Dialog" (Siehe Abb. 5.1) mit den Anfangsbuchstaben eines Produktes. Das Feld gibt daraufhin einige Vorschläge aus.

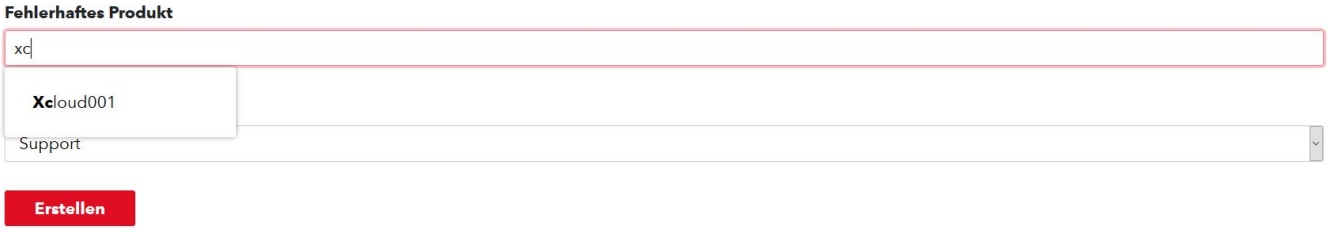

Abbildung 5.4: Autocomplete für das Feld "Betroffenes Produkt"

5.1.3 Übersicht über alle Tickets der Organisation

Die Übersicht über die Tickets wurde wie in Abb. 5.5 dargestellt, umgesetzt. Es wurden zudem die zwei Filter "Offene Tickets" und "Alle Tickets" erstellt, da es sein könnte, dass eine Organisation sehr viele gelöste Tickets hat, jedoch sind oftmals nur die gerade offenen Tickets interessant.

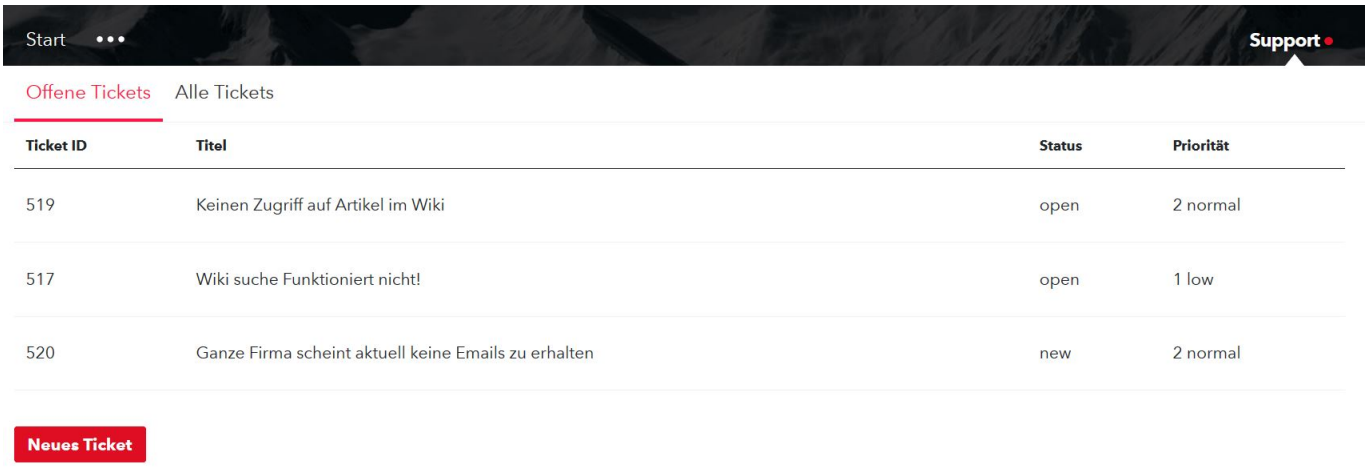

Abbildung 5.5: Übersicht über die Tickets

5.1.4 Möglichkeit zum Kommentieren und Schliessen von Tickets

Im Specific Ticket Dialog, in Abb. 5.2 gezeigt, gibt es ein Feld um neue Kommentare zu erfassen oder einen Button zum schliessen von Tickets. Wird auf den Button "Ticket Abschliessen" geklickt, so erschient ein Warnhinweis, der durch den Benutzer bestätigt werden muss um das Ticket abzuschliessen. Geschlossene Tickets können zudem wiedereröffnet werden, entsprechende Buttons erscheinen jedoch nur bei geschlossenen Tickets.

5.2 Ausblick

Vor dem produktiven Einsatz der Applikation sollten einige Dinge beachtet werden, welche im Weiteren beschrieben sind. Da die Projektzeit begrenzt war, wurde der Fokus auf die Eliminierung der Risiken gelegt. Daher wurden bei der Implementierung einige Features bzw. Details weggelassen welche aber in der Evaluierung berücksichtigt wurden. Es handelt sich hierbei um optionale Produktanforderungen sowie Bugfixes und allgemeine Verbesserungen.

5.2.1 Knowledgebase

Beim Entscheid für Zammad haben wir darauf hingewiesen, dass noch keine Funktionalität für eine Knowledge Base vorhanden ist. Diese ist aber auf der Roadmap des Projekts. Nach Absprache mit dem Kunden wurde diese Anforderung zurückgestellt. Es wurde aber sichergestellt, dass eine Anbindung möglich wäre und auch bereits ein Test mit einer in myFlow integrierten Knowledge Base durchgeführt.

5.2.2 Suche

Es existiert noch keine globale Suche über alle Tickets. Dies wurde als eine optionale Anforderung definiert. Die Zammad API sowie auch die Web-API bieten bereits solche Funktionalitäten an, sind aber noch nicht final implementiert.

5.2.3 Bugfixes

Im Allgemeinen wurde das beste getan um sauberen und wartungsfähigen Code zu schreiben. Der begrenzte Zeitrahmen hat da aber Grenzen gesetzt. Es wurden noch nicht alle Fehler ausgemerzt. Auch die Testcoverage kann noch erhöht werden.

5.2.4 User Interface Design

Für die Applikation wurde bereits ein komplettes Design des Interfaces vorgegeben, welches von einem Designstudio implementiert wird. Daher wurde der Fokus bei der Implementation nicht auf das Styling gelegt. Das Design muss definitiv überarbeitet werden.

5.2.5 Komfort Funktionen

Wir haben uns Gedanken über viele kleine Verbesserungen gemacht welche die Bedienung für den Endkunden einiges vereinfachen kann. Natürlich ist diese Liste nicht abschliessend.

Attachments

Bis jetzt ist es nicht möglich mehrere Attachments gleichzeitig anzuhängen. Grundsätzlich wird dies aber vom Backend (Web-API) bereits unterstützt. Einzig die Implementierung in Angular fehlt. Weiter ist es noch nicht möglich direkt beim Erstellen eines Tickets Attachments mitzugeben. Auch dies bedarf einer kleinen Erweiterung im Frontend. Als zusätzlichen Komfort für den Kunden könnte auch ein ”Drag and Drop” Mechanismus hinzugefügt werden. Derzeit werden die Attachments noch als Blob heruntergeladen. Dies kann bei sehr grossen Files ein Problem darstellen. Zammad stellt die Funktionalität bereits zur Verfügung. Backend und Frontend müssten angepasst werden.

Pagination

Bisher wurde keine Pagination für die Tickets implementiert. Es existiert von Zammad sowie auch von myFlow aber bereits ein Mechanismus um dies zu realisieren.

Anbindung betroffene Produkte

Bei der Erfassung eines neuen Tickets hat der Kunde die Möglichkeit das betroffene Produkt anzugeben. Das Frontend stellt ihm dabei Vorschläge zur Verfügung. Die bestehende Lösung schlägt nur die Server (Compute Engines) des Kunden vor. In Zukunft soll das auf weitere Produkte ausgeweitet werden. Da bereits die interne Suchfunktion verwendet wird, geschieht diese Ausweitung automatisch, sobald neue Produkte in die globale Suche aufgenommen werden.

Einbindung des Infrastrukturstatus

Da viele Tickets von Infrastrukturausfällen ausgehen, könnte bei bekannten Störungen die Information direkt in der Erfassungsmaske angezeigt werden, um so die duplizierte Tickets zum gleichen Vorfall zu verhindern. Die Statusinformation zur Infrastruktur werden bereits von der API zur Verfügung gestellt.

5.2.6 Zammad

Mehrere Organisationen pro Benutzer

Bisher unterstützt Zammad nur eine Organisation pro Benutzer. MyFlow aber erlaubt mehrere. Es wurde auch bereits ein Feature-Request bei Zammad eingereicht um dies zu erweitern. Die Entwickler hinter Zammad stehen dem Vorschlag offen gegenüber. Er wurde in die Pipeline für neue Releases aufgenommen. Der genau Zeithorizont ist aber noch nicht bekannt. Bis zu dessen Implementation wurde eine Übergangslösung gebaut, welche einen einfachen Umstieg erlaubt. Derzeit wird jedem Benutzer in Zammad die Organisation zugeteilt, welche er in myFlow als Standard festgelegt hat. myFlow prüft beim Abruf von Tickets ob der User mit seiner Standardorganisation eingeloggt ist. Wenn nicht, wird eine Fehlermeldung zurückgegeben. Sobald Zammad um die Funktionalität für mehrere Organisationen erweitert wurde, kann der Check in myFlow entfernt werden.

5.3 Reflektion

Im diesem Abschnitt werden die persönlichen Eindrücke und Erfahrungen beschrieben.

5.3.1 Bericht Marco

Zu beginn der Studienarbeit hatte ich eigentlich keine praktische Erfahrung mit der Arbeit an einem grösseren Software System oder mit Angular. Zudem war auch Latex neu für mich und ich musste es zuerst lernen. Dies hat mich zu beginn sehr verunsichert, da ich sehr viel Zeit investieren musste und kaum Output hatte. Mit der Zeit ging dies jedoch immer besser und ich werde in Zukunft auch bei Projekten mit vielen mir Unbekannten Technologien und grosser Komplexität sicherlich bereits von beginn an zuversichtlicher sein können.

Hochinteressant für mich waren zudem die Entscheidungen zu Prioritäten die ich zu setzen hatte. Es war schnell klar, dass ich nicht allem nachgehen konnte wo ich noch Verbesserungspotential gesehen hätte. Oftmals musste ich diese Eindrücke Ignorieren und mich auf diejenigen Dinge fokussieren, die wirklich nötig waren. Ich denke dies ist eine Wichtige Fähigkeit, auch für später und dass wir dies während der Studienarbeit gut gemacht haben.

Ich hätte an vielen Orten gerne noch mehr gemacht, etwas ausgebaut, verbessert und mehr Funktionalität implementiert. Wenn man jedoch bedenkt wie wir uns die Fähigkeiten angeeignet haben die uns Fehlten, wie wir vorgegangen sind und für was wir unsere Zeit genutzt haben, bin ich mit dem Resultat Arbeit sehr zufrieden.

5.3.2 Bericht Dominik

Als gelernter Softwareentwickler habe ich schon an einigen Projekten mit PHP gearbeitet. Da diese Arbeit in Zusammenarbeit mit meinem derzeitigen Arbeitgeber entstanden ist, war ich auch mit der Applikation schon vertraut. Dies hat mir den Aufwand für die Einarbeitung sehr stark reduziert. Ich hatte daher die Aufgabe meinen Partner in das Projekt einzuführen und ihm offene Fragen zu beantworten. Dabei fiel mir auf wie komplex das Projekt eigentlich ist und wurde auch mit Fragen konfrontiert, welche ich mir selbst noch nicht gestellt hatte.

Speziell bei den Architekturentscheidungen empfand ich es als sehr wertvoll eine zweite Herangehensweise zu sehen. Ziemlich neu war für mich auch die Dokumention mit LaTex. Im Nachhinein kann ich sagen dass der Entscheid für diese Technologie richtig war und die Vorteile gegenüber anderen Formaten wie z.B. Word überwiegen.

Die Semesterarbeit war für mich im Allgemeinen ein erfolgreiches Projekt mit einem zufriedenstellendem Ergebnis. Meine Vorstellungen welche ich an das Ergebnis vor dem Start der Arbeit hatte, wurden jedoch nicht komplett erfüllt. Gerne hätte ich noch einige Features mehr im finalen Resultat gesehen. Definitiv habe ich unterschätzt, wie viel Aufwand eine gründliche Evaluation benötigt. Ich bin gespannt wie ich das Gelernte in der Bachelorarbeit umsetzen kann.

Kapitel 6

Projektplan

6.1 Projektorganisation

Dominik Thamm

Dominik hat die Informatikmittelschule besucht und als Informatiker EFZ abgeschlossen. Zurzeit befindet er sich im 7. Semester des Informatikstudiums an der HSR. Nebenbei arbeitet er für die Digio GmbH als Software Engineer. Seine Freizeit verbringt er gerne in den Bergen beim Wandern und Biken.

Marco Steiner

Marco Steiner hat eine Lehre als Informatiker Systemtechnik abgeschlossen. Diese Studienarbeit ist daher das erste mal, dass er an einer grösseren Software Applikation mitarbeitet. Ausserhalb der Schule, liest er gerne oder fährt mit dem Velo durch das Linthgebiet.

6.2 Projektmanagement

6.2.1 Zeitliche Planung

Das Projekt wird als Studienarbeit durchgeführt. Es stehen 14 Wochen zur Evaluation und Implementation zur Verfügung. Der Termin für die Abgabe ist der 22. Dezember 2017. Pro Teammitglied sollen 240 Stunden aufgewendet werden. Das sind in etwa 17 Stunden pro Woche.

6.2.2 Phasen

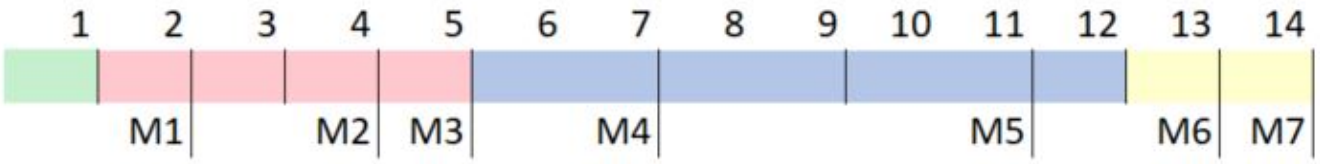

Abbildung 6.1: Projektplan

Das Projekt ist in nach RUP aufgebaut. Iterationen starten bzw. enden jeweils am Freitag.

Inception

In dieser Phase findet der Projekt Kickoff statt. Diese Phase dauert 1 Woche und beinhaltet eine Iteration.

Elaboration

In dieser Phase geht es um die Evaluation der Ticketing Plattform. Sie nimmt 4 Wochen in Anspruch und besteht aus 4 Iterationen. Am Ende dieser Phase soll ein Entscheid vorliegen, welcher das zu verwendende Produkt bestimmt und begründet. Allfällige Anforderungen welche vom Produkt nicht erfüllt werden können müssen ausgewiesen sein.

Construction

Die Construction-Phase dauert 7 Wochen. Sie besteht aus 2 Iteration a 2 Wochen und einer Iteration a 1 Woche. In diesen Iterationen findet die eigentliche Integration statt. Diese umfasst folgende Punkte

- das Aufsetzen des Ticketing Systems
- die Implementation des Frontends (Angular)
- die Implementation des Backends (PHP, Symfony)

Transition

In der Transition-Phase wird das Projekt dem Kunden übergeben. Dies beinhaltet die Finalisierung des Technischen Berichts zur Studienarbeit und die Migration der Kundendaten vom alten System. Diese Phase dauert 2 Wochen und besteht aus 2 Iterationen.

6.2.3 Meilensteine

MS 1: Projekt Kickoff

Bis 29.9.17: Zugangsdaten und Anforderungen erhalten

MS 2: Produktenscheid gefällt (Vorschlag)

Bis 13.10.17: Alle Anforderungen des Kunden sind erfasst und gewichtet. Aufgrund diesen Anforderungen machen wir einen Produktvorschlag.

MS 3: Finaler Entscheid anhand von Kundenfeedback

Bis 20.10.17: Nach Berücksichtigung des Feedbacks von Innofield AG ist der definitive Entscheid gefällt.

MS 4:Architektur-Prototyp

Bis 3.11.17: Das Ticketsystem ist aufgesetzt. Ein funktionaler Architekturprototyp ist zur Präsentation bereit. Es wird ein erster Code-Review durchgeführt.

MS 5 :Feature Implementation beendet

Bis 1.12.17: Die Implementation aller Features ist beendet. Von nun an werden nur noch Bugfixes am Code vorgenommen.

MS 6: Datenmigration und Bugfixes

Bis 15.12.17: Bugfixes und Migration der Daten des alten Ticketsystems (Zendesk). [1]

MS 7: Dokumentation abgeschlossen

Bis 22.12.17: Die Dokumentation ist finalisiert und bereit für die Abgabe an Kunden und HSR.

6.2.4 Arbeitspakete

Die Arbeitspakete werden in Gitlab als Issues verwaltet. Mit Labels werden die einzelen Issues jeweils als ”To Do”, ”In Progress” oder ”To Test” gekennzeichnet.

6.2.5 Besprechungen

Vorerst wird im Wochenrythmus ein Meeting mit Herrn Stolze abgehalten. Nach jeweils ca. 1/3 bzw. 2/3 der Zeit finden Sitzungen mit Damien Vouillamoz statt. Daten der geplanten Meetings:

- 20.10.17
- 14.12.17

[1] Aus Datenschutzgründen, wird die Datenmigration erst auf dem Produktivsystem durchgeführt und somit erst nach dieser Studienarbeit.

6.3 Risikomanagement

Ziel der Arbeit ist es möglichst alle Risiken in der Evaluationsphase zu eliminieren. Falls die Implementierung aller Features nicht vollständig abgeschlossen werden kann, soll der weiteren Entwicklung bzw. dem Ausbau nichts im Weg stehen. Die Risiken werden von Woche zu Woche neu evaluiert und beurteilt.

6.3.1 Risiken

Tabelle 6.1: Risiken

Risiko pro Woche	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
Missverständnisse in der Kommunikation	-	-	35	35	35	35	25	20	18	15	10	5	5	3
Fehler wegen zu wenig Erfahrung mit Angular oder Symphony2	-	-	30	30	30	30	20	19	18	17	12	6	6	6
Produkt lässt sich nicht in Architektur integrieren	-	-	24	24	15	10	5	5	5	2	2	2	2	1
Fehler werden erst nach Ende des Projekts sichtbar	-	-	18	18	18	18	15	15	15	15	15	10	5	5
Produkt lässt sich nicht in Architektur integrieren	-	-	8	8	16	16	16	14	7	5	2	2	2	2
Schnittstellen zu Umgebungssystemen	-	-	15	15	15	15	10	5	5	5	5	5	5	3
Ticketsystem entspricht nicht dem, was der Kunde haben wollte	-	-	20	20	10	5	5	2	2	2	2	2	2	2
Ticket System Qualität ungenügend	-	-	-	-	10	10	10	10	10	10	5	5	5	3
Nicht Funktionale Anforderungen wie Usability und Performance können nicht erfüllt werden	-	-	-	-	10	10	20	20	20	20	10	10	5	5
Ticketsystem API ungenügend	-	-	32	32	8	8	8	4	4	4	4	2	2	2
Ressourcenengpässe wegen Krankheitsausfall	-	-	7.5	7.5	7.5	7.5	7.5	7.5	7.5	7.5	10	5	5	2
Code Qualität ungenügend	-	-	6	6	6	6	6	6	6	6	2	2	2	2
Zeiterfassung nicht aktuell	-	-	5	5	5	5	3	3	3	3	3	3	3	1
Datenverlust	-	-	4	4	4	4	4	2	2	1	1	1	1	1
Datenimport	-	-	30	30	2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5	0
Hardwareausfall	-	-	2	2	2	2	2	2	2	2	2	2	2	2

6.4 Infrastruktur

6.4.1 Hardware

Development Server

Die Development Server werden von der Digio GmbH gestellt. Diese Server dienen vorrangig zum Testen der Applikation während der Entwicklung. Es gibt jeweils einen Server für das Frontend, Backend und den Help Desk. Alle Server basieren auf Linux (Ubuntu) und erlauben den Zugriff per SSH.

Eigene Laptops

Zur Evaluation und Entwicklung werden jeweils die eigenen Laptops verwendet. Zusätzlich stehen in den SA-Zimmern jeder Person ein Computer mit Bildschirm zur Verfügung.

6.4.2 Software

Git / Gitlab

Zur Sourcecodeverwaltung wird Git verwendet. Alle Code- und Dokumenten-Repositories werden auf dem Gitlab-Server von Digio GmbH gehostet. Zugriff erfolgt über ssh oder https.

LaTex

Alle Dokumente sind in LaTex geschrieben und zu PDFs kompiliert.

6.5 Qualitätsmassnahmen

6.5.1 CI

Wir benutzen Gitlab CI um das Deployment einfach zu gestalten. Jeder push auf den Branch ”develop” wird gebuildet, getestet und dann Deployed. Somit ist immer die aktuelle Version der Applikation auf dem Development Server verfügbar.

6.5.2 Feature Branches

Für jedes eigenständige Feature soll ein neuer Branch erstellt werden. Nachdem das Feature getestet wurde, wird der Branch in den Develop-Branch gemerged. Dann kann der Feature-Branch wieder gelöscht werden. Damit sind die einzelnen Features unabhängig und es entstehen weniger Konflikte beim mergen.

6.5.3 Unit Tests

Um die Wartbarkeit und Codequalität zu erhöhen werden für alle wesentlichen funktionen UnitTests geschrieben. Diese werden in den CI Prozess eingebunden.

6.5.4 Dokumentation

Die Dokumentation des Projektes ist in Deutsch gehalten. Die API-Doku ist in API Blueprint geschrieben und wird mit Aglio zu HTML gerendert. Sie ist verfügbar unter ”api-server/v2/”.

6.5.5 Code Reviews

Als weiter Qualitätsmassnahme wurden regelmässig Code Reviews durchgeführt.

6.6 Zeitabrechnung

Während der Arbeit hat jedes Teammitglied seine aufgewendeten Stunden dokumentiert. In der untenstehenden Tabelle wird die finale Auswertung präsentiert. Die Stunden sind jeweils nach Person und Kategorie unterteilt.

Tabelle 6.2: Zeitabrechnung

Kategorie	**Dominik**	**Marco**	**Total**
Projektmanagement und Dokumentation	94.5	82	**176.5**
Konzeption	34.5	40	**74.5**
Besprechungen	17.25	18.5	**35.75**
Implementation	99.5	109	**208.5**
Total	**245.75**	**249.5**	**495.25**

Literaturverzeichnis

[1] The 8 best free and open source help desk software tools. https://blog.capterra.com/the-7-best-free-help-desk-software-tools/. zuletzt abgerufen am 20.10.2017.

[2] Aglio. https://github.com/danielgtaylor/aglio. zuletzt abgerufen am 20.12.2017.

[3] Demo service desk. https://demo.servicedeskplus.com/. zuletzt abgerufen am 01.11.2017.

[4] Fundamentals. https://angular.io/guide/*. zuletzt abgerufen am 01.12.2017, * ist ein platzhalter, da Fundamentals eine Section der Webseite mit vielen Informationen ist. Ein Beispiel für einen gültigen Link wäre https://angular.io/guide/architecture.

[5] Psr-1: Basic coding standard. http://www.php-fig.org/psr/psr-1/. zuletzt abgerufen am 21.12.2017.

[6] Psr-2: Coding style guide. http://www.php-fig.org/psr/psr-2/. zuletzt abgerufen am 21.12.2017.

[7] Receive blob resopnses. https://stackoverflow.com/questions/34149741/how-to-receive-blob-responses-using-angulars-2-angular-http-module. zuletzt abgerufen am 12.11.2017.

[8] Service desk plus webseite. https://www.manageengine.com/products/service-desk/. zuletzt abgerufen am 01.11.2017.

[9] Testing kurs. https://www.udemy.com/testing-angular-apps/learn/v4/overview. zuletzt abgerufen am 22.12.2017.

[10] Tutorial: Tour of heroes. https://angular.io/tutorial. zuletzt abgerufen am 01.12.2017.

[11] Uploading files in angular (2/4) to a rest api. https://nehalist.io/uploading-files-in-angular2/. zuletzt abgerufen am 21.12.2017.

[12] Wikipedia iso 9126. https://de.wikipedia.org/wiki/ISO/IEC_9126. zuletzt abgerufen am 21.12.2017.

[13] Zammad api docs. https://docs.zammad.org/en/latest/api-intro.html. zuletzt abgerufen am 21.12.2017.

[14] Zammad webseite. https://zammad.org. zuletzt abgerufen am 01.11.2017.

Abbildungsverzeichnis

Tabellenverzeichnis