

SPA für das Management von Produktlebenszyklen

Bachelorarbeit

Abteilung Informatik
Hochschule für Technik Rapperswil

Frühjahrssemester 2018

Autoren: Fabian Gübeli
Luca Salzani

Betreuer: Prof. Dr. Daniel P. Politze

Experte: Ramon Schildknecht

Gegenleser: Mirko Stocker

Inhaltsverzeichnis

Abstract	iii
Management Summary.....	iv
1 Einleitung.....	1
1.1 Beschreibung der Ausgangslage.....	1
1.2 Projektscope.....	1
1.3 Vision.....	1
2 Problemanalyse.....	2
2.1 Umfeldanalyse.....	2
2.2 Begriffs- und Rollendefinition	2
2.3 Schnittstellenanalyse.....	3
2.4 User Stories	4
2.5 Funktionale Anforderungen	5
2.5.1 Übersicht	5
2.5.2 Use Cases.....	6
2.6 Nicht-funktionale Anforderungen	8
2.6.1 Bachelorarbeit/Prototyp	8
2.6.2 Produktives Umfeld.....	9
3 Lösungskonzept.....	10
3.1 Konzeption Frontend.....	10
3.2 Cockpit Evaluation.....	10
3.2.1 Nebular	10
3.2.2 PrimeNG	11
3.2.3 Metronic.....	12
3.3 Konzeption Backend API.....	14
3.3.1 Architektur.....	15
3.3.2 Error Handling	16
3.3.3 Asynchronität	17
3.3.4 Testing	17
3.3.5 Authentifizierung.....	17
3.3.6 Request Pipeline.....	18
3.4 Product Backbone	19
3.5 Werkzeuge und Technologien.....	19
3.6 Deploymentkonzept.....	20
3.7 Schnittstellenkonzept.....	20
3.8 Berechtigungskonzept.....	21

3.9	Funktionsmanagement.....	22
3.10	Testkonzept	24
4	Umsetzung und Testing.....	26
4.1	Systemübersicht	26
4.2	Usability-Test.....	27
4.2.1	Aufgaben	27
4.2.2	Ergebnisse.....	28
4.3	Systemtest	29
4.4	Erweiterung der Applikation	32
4.5	Code Metriken.....	33
4.5.1	Backend	33
4.5.2	Frontend	34
4.6	Showcases	35
4.6.1	Showcase „Häufig auftretendes Problem“	35
4.6.2	Showcase „Upgrade“	35
4.6.3	Showcase „Einsatz des Servicetechnikers“	35
5	Ergebnis und Ausblick.....	36
5.1	Zielerreichung.....	36
5.2	Übergang zu produktivem System	37
5.3	Ausblick und Erweiterungen.....	37
5.3.1	Azure Cloud	38
5.3.2	Sicherheitserweiterungen	39
5.3.3	Offline Verfügbarkeit.....	39
6	Literaturverzeichnis.....	40
7	Abbildungsverzeichnis.....	42
8	Tabellenverzeichnis	43
Anhang	44
Anhang A	– Projektorganisation.....	44
Anhang B	– Diagramme.....	49
Anhang C	– Test Protokolle	54
Anhang D	– Installationsanleitung	61
Anhang E	– Sitzungsprotokolle	68
Anhang F	– Zugangsdaten	72
Anhang G	– Persönliche Reflexion.....	73
Anhang H	– Administrative Anhänge	75

Abstract

Nach dem Vertrieb einer Maschine fallen im Produktionsbetrieb fortwährend Arbeiten an. Einzelne Systeme wie PLM, ERP oder Projektmanagement arbeiten isoliert. Ein System, welches die Daten dieser Systeme in Verbindung setzt und die Abbildung eines Workflows erlaubt, fehlt. Die Servicetechniker, Verkäufer, Produktmanager und Kunden können ihre gewünschten Informationen nicht auf einen Blick einsehen.

Aufbauend auf der im Herbstsemester 17/18 durchgeführten Studienarbeit sollte für die verschiedenen Benutzerrollen eine Anwendung entwickelt werden, welche die Daten der Umsysteme miteinander in Verbindung bringt und übersichtlich in einem Dashboard darstellt. Da die Umsysteme keine Unterstützung für die funktionsbasierte Weiterentwicklung und das Servicemanagement bieten, werden diese Informationen in der Applikation erfasst und verwaltet.

Die entstandene Applikation wurde im Frontend mit Angular, Bootstrap und Modulen von Nebular entwickelt. Somit entstand ein responsives und modernes Design. Auch die Benutzerfreundlichkeit hatte im Hinblick auf die späteren Bediener einen grossen Stellenwert. Das Frontend interagiert mit einer in ASP.NET Core programmierten REST-Schnittstelle. In diesem Backend wurde vor allem Wert auf eine saubere Codebasis und Architektur sowie Erweiterbarkeit und Testbarkeit gelegt. So kann eine gute Grundlage für aufbauende Arbeiten auf diesem Gebiet weitergegeben werden. Das Backend wiederum bezieht seine Daten aus dem ASP.NET Core Backbone, welches diese aus den Umsystemen zusammenbringt und über eine REST-Schnittstelle zur Verfügung stellt. Das Backbone war nicht Teil der Arbeit und wird zu einem späteren Zeitpunkt von der HSR weiterentwickelt. Aus diesem Grund wurden die Daten während der Arbeit im Backbone mittels einer NoSQL Datenbank gemockt.

Das Endprodukt ist in der Lage anhand definierter Showcases die Funktionalitäten und eingeführten Konzepte aufzuzeigen. Das System ist mit weiteren Anforderungen und Modulen sowie Rollen erweiterbar. Daher eignet es sich für weitere Entwicklungen und Arbeiten auf dem Gebiet der Maschinenlebenszyklen.

Management Summary

Ausgangslage

Nach dem Vertrieb einer Maschine fallen im Produktionsbetrieb fortwährend Arbeiten an. Einzelne Umsysteme für die Entwicklung der Produkte oder das ERP arbeiten isoliert. Ein System, welches die Daten dieser Systeme in Verbindung setzt und die Abbildung eines Workflows erlaubt, fehlt. Die Servicetechniker, Verkäufer, Produktmanager und Kunden können ihre gewünschten Informationen nicht auf einen Blick einsehen.

Aufbauend auf der im Herbstsemester 17/18 durchgeführten Studienarbeit sollte für die verschiedenen Benutzerrollen eine Anwendung entwickelt werden, welche die Daten der Umsysteme miteinander in Verbindung bringt und übersichtlich in einem Dashboard darstellt. Da die Umsysteme keine Unterstützung für die funktionsbasierte Weiterentwicklung und das Servicemanagement bieten, sollen diese Informationen in der Applikation erfasst und verwaltet werden.

Vorgehen

In einem Workshop wurden die wichtigsten Bedürfnisse und Funktionalitäten aufgenommen. Anhand dieser liessen sich die funktionalen Anforderungen ableiten. Nach der Evaluation der zu verwendenden Technologien wurde in einem iterativen Prozess die Applikation entwickelt. Dabei wurde in der Benutzeroberfläche auf eine einfache und intuitive Bedienung geachtet. Durch die saubere Codebasis und gute Wartbarkeit sowie Erweiterbarkeit kann sichergestellt werden, dass die Applikation auch für projektfremde Personen gut verständlich ist und weitergeführt werden kann.

Ergebnis

Das Endprodukt ist in der Lage anhand definierter Showcases die Funktionalitäten und eingeführten Konzepte aufzuzeigen. Das System ist mit weiteren Anforderungen und Modulen sowie Rollen erweiterbar. Daher eignet es sich für weitere Entwicklungen und Arbeiten auf dem Gebiet der Maschinenlebenszyklen.

Die verschiedenen Benutzer werden anhand ihrer Funktion in einem Unternehmen individuell in ihrer Tätigkeit unterstützt. Mit weiteren Arbeiten an der Applikation kann das Ziel, das Management von Maschinen nach ihrer Instandsetzung und während der Entwicklung deutlich effizienter zu gestalten, erreicht werden.

Ausblick

Aufgrund der Komplexität und dem Umfang des Gebiets sind viele verschiedene Erweiterungen denkbar. So wäre es zum Beispiel möglich, dass ein Servicetechniker die Teile nicht mehr selber auswählen muss, weil er diese zum Beispiel mit einem Scanner einfach erfassen kann. Auch vorstellbar ist die Einbindung von Sensordaten der einzelnen Maschinen zur Auswertung und Fehlererkennung.

Diese Anforderungen können zum Beispiel mittels einer Cloud-Lösung umgesetzt werden. Ein mögliches Deployment der Applikation in die Cloud wurde am Ende der Arbeit evaluiert und getestet.

Ein weiteres Thema ist die Verfügbarkeit des Systems ohne Internetzugang. Dabei kann der Servicetechniker in einer Produktionshalle ohne Internet seine Arbeiten erfassen und sobald die Verbindung wiederhergestellt ist, die Daten übertragen.

Grosses Potential birgt die Einführung einer Tagesplanung. So können effiziente Routen für Serviceeinsätze geplant werden, sodass weniger Zeit für die Fahrten zu den Kunden verloren geht.

1 Einleitung

1.1 Beschreibung der Ausgangslage

Nach dem Vertrieb einer Maschine fallen im Produktionsbetrieb fortwährend Arbeiten an. Einzelne Umsysteme für die Entwicklung der Produkte oder das ERP arbeiten isoliert. Ein System, welches die Daten dieser Systeme in Verbindung setzt und die Abbildung eines Workflows erlaubt, fehlt. Die Servicetechniker, Verkäufer, Produktmanager und Kunden können ihre gewünschten Informationen nicht auf einen Blick einsehen.

Aufbauend auf der Studienarbeit HS 17/18 sollen ein Rollenkonzept und Workflows erarbeitet und implementiert werden, um die Situation für alle Anspruchsgruppen zu verbessern.

Zur erfolgreichen Bearbeitung dieser Aufgabenstellung soll zunächst eine Anforderungsanalyse durchgeführt sowie die Risiken identifiziert werden. Anschliessend wird das Backend gemäss den Anforderungen entwickelt. Dieses stellt eine Schnittstelle bereit, welche von einer gleichzeitig zu entwickelnden Frontend Applikation genutzt werden kann. Funktionen, welche von einem Benutzer unterwegs ausgeführt werden müssen, sind responsive zu designen. Während, sowie nach der Entwicklung wird die Software mittels verschiedener Tests auf ihre Qualität geprüft. Dabei liegt der Fokus auf einigen festgelegten Showcases, welche die Möglichkeiten der Lösung aufzeigen.

1.2 Projektscope

Die Schnittstellen der umliegenden Systeme – wie PLM, ERP und JIRA – sind nicht Teil der Bachelorarbeit. Diese werden während der Arbeit parallel durch andere HSR Mitarbeiter entworfen und implementiert.

Ausserdem wurde im Herbstsemester 17/18 eine Studienarbeit erstellt, welche die Basisfunktionalitäten von Serviceaufträgen sowie dem Stücklistenmanagement implementiert. Darauf baut die Bachelorarbeit auf.

1.3 Vision

Ein Unternehmen, welches Maschinen vertreibt und wartet, soll bei ihren Geschäftsprozessen unterstützt werden. Durch den Zusammenzug von Daten aus isolierten Systemen wie PLM oder ERP ist es möglich ein Dashboard mit Informationen für jede Anspruchsgruppe darzustellen. Durch die zusätzlichen Informationen ist es auch denkbar, verfügbare Maschinenupgrades vorzuschlagen, zu verkaufen und deren Einbau zu dokumentieren. Ein Service Case Management, im Stile von bekannten Ticketingsystemen, kann dann dabei helfen Serviceaufträge zuzuordnen und nachzuvollziehen. Die Daten aus all diesen Funktionen können dazu genutzt werden umfassende Möglichkeiten für das Reporting anzubieten. So können zum Beispiel Trends in der Auftragslage erkannt oder Serienfehler gewisser Maschinentypen effizient gefunden werden.

Die Integration von weiteren Systemen, wie zum Beispiel einem Projektmanagementsystem oder Active Directory, kann in einem weiteren Schritt aufgrund der gut wartbaren Projektstruktur umgesetzt werden.

2 Problemanalyse

2.1 Umfeldanalyse

Das Projekt ist in einem sehr komplexen Umfeld angesiedelt. In diesem geht es um das Zusammenfügen, die Gewinnung und das Anreichern von Daten nach der Inbetriebnahme der Maschine und der Dateninitialisierung bei der Auslieferung.

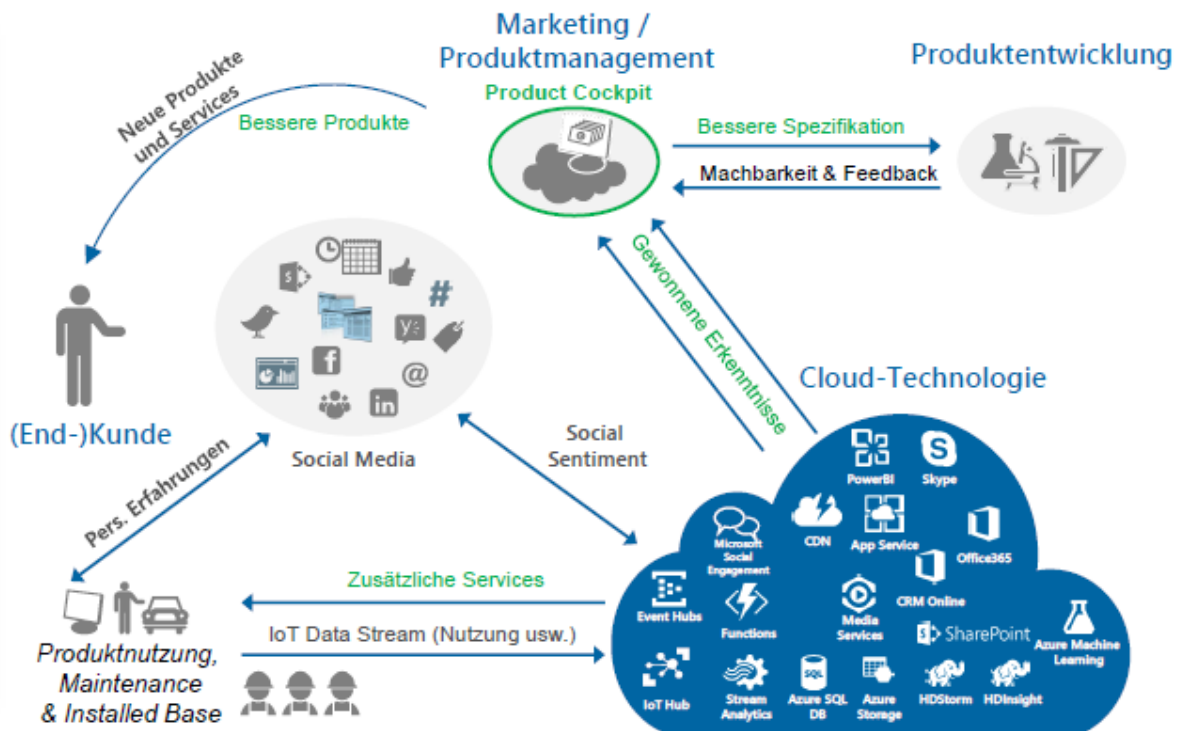


Abbildung 1 – Umfeld und Projektabgrenzung (Bildherkunft: HSR/Microsoft)

Die gewonnenen Daten können von verschiedenen Personengruppen genutzt werden, um einen möglichst guten Service bereitzustellen. Vor allem die Produktentwicklung interessiert sich für den Zustand und die Probleme „im Feld“. Auch für den Service können Daten aus der Produktion relevant sein, um sich besser auf Serviceeinsätze vorzubereiten und somit effizienter zu arbeiten.

2.2 Begriffs- und Rollendefinition

Benutzerrollen

In diesem Dokument sowie in der Software werden verschiedene Rollen verwendet, um die Bedürfnisse der verschiedenen Anspruchsgruppen abzudecken.

Als *Servicetechniker* wird ein Mitarbeiter beschrieben, welcher Arbeiten an den Maschinen durchführt. Er möchte vor allem möglichst wenig zusätzlichen Aufwand durch die Datenpflege haben. Falls dies nicht gegeben ist, ist die Wahrscheinlichkeit gross, dass er aufgrund des Zeitdrucks gar keine Daten erfassen wird. Er kann auch erwarten, dass er mit der Applikation seine Arbeit effizienter erledigt als bisher. Dies zum Beispiel mittels einer Analyse der Standzeiten oder der Konsultation bisheriger Arbeitseinsätze an dieser Maschine.

Der *Verkäufer* arbeitet im Büro und selten unterwegs. Er pflegt den Kundenkontakt und ist für die Kundenzufriedenheit sowie die Auftragsbeschaffung zuständig. Ihn interessiert primär den erzielbaren Umsatz. Aus dieser Motivation heraus stellt er sich die Frage wem er was verkaufen kann. Dafür ist es wichtig zu wissen, welche Maschinen mit welcher Konfiguration bei welchem Kunden stehen und wie die Vergangenheit aussieht. So kann er vorbeugende Wartungsarbeiten und Maschinenupgrades

empfehlen und verkaufen. Diese soll er direkt im System erfassen können. Ein Servicetechniker wird die Arbeiten dann ausführen. Auch ist er die Kontaktperson für den Kunden und muss auch in der Lage sein, selbst Serviceaufträge zu erfassen.

Diese Probleme kommen vom *Kunden*. Er benutzt die Maschinen des Herstellers und möchte, dass diese so effizient wie möglich funktionieren. Er hat Interesse an langlebigen und zuverlässigen Maschinen. Um die Verfügbarkeit hoch zu halten ist es essentiell, dass Probleme schnellstmöglich behoben werden. Daher interagiert er direkt über das System mit der Maschinenvertriebsfirma. Diese bietet ihm die Möglichkeit, selbst Probleme zu erfassen und diese dann nachzuverfolgen.

Der *Produktmanager* ist zuständig für das Management der Produktentwicklungen und ist auch verantwortlich für den Markt. Er hat das letzte Wort bei der Einführung von neuen Produktpaketen und entscheidet wie diese ausgestaltet sind. Er arbeitet im Büro und ist das Bindeglied zwischen dem Verkauf und der Entwicklung. Ihn interessieren aktuelle Probleme der Kunden, worauf seine Analysen hauptsächlich basieren. Er möchte dabei möglichst genaue Modelle betrachten können, um fundierte Entscheidungen zu treffen.

Begriffsdefinitionen

Service Cases beschreiben eine auszuführende Arbeit an einer Anlage. Diese können präventiv oder auch reaktiv sein. Service Cases werden von Servicetechnikern bearbeitet und daher ist auch jedem Service Case ein Servicetechniker als Verantwortlicher zugeordnet. Kunden, Servicetechniker und Verkäufer können Service Cases erfassen und damit gleich einen Auftrag an die Firma erteilen. Die ausgeführten Arbeiten werden in Aktivitäten erfasst.

Ein *Marketing Kit* ist ein Feature, welches für mehrere Maschinen verfügbar gemacht werden kann. Diese werden vom Produktmanager erfasst und von der Entwicklung realisiert. Nach der nicht im Scope des Projekts enthaltenen Testphase, werden Features für gewisse Maschinentypen freigegeben. Auf die Freigabe erfolgt der Verkauf an die Kunden und die Installation durch Servicetechniker.

2.3 Schnittstellenanalyse

Da die Integration der Umsysteme nicht Teil der Arbeit ist, wurde ein Backbone zur Verfügung gestellt. Dieses greift über Adapter und das Repository-Pattern auf die Umsysteme zu und stellt eine API zur Verfügung. Ein Teil der Architektur des Backbones ist in folgender Abbildung ersichtlich. Das vollständige Diagramm befindet sich im Anhang.

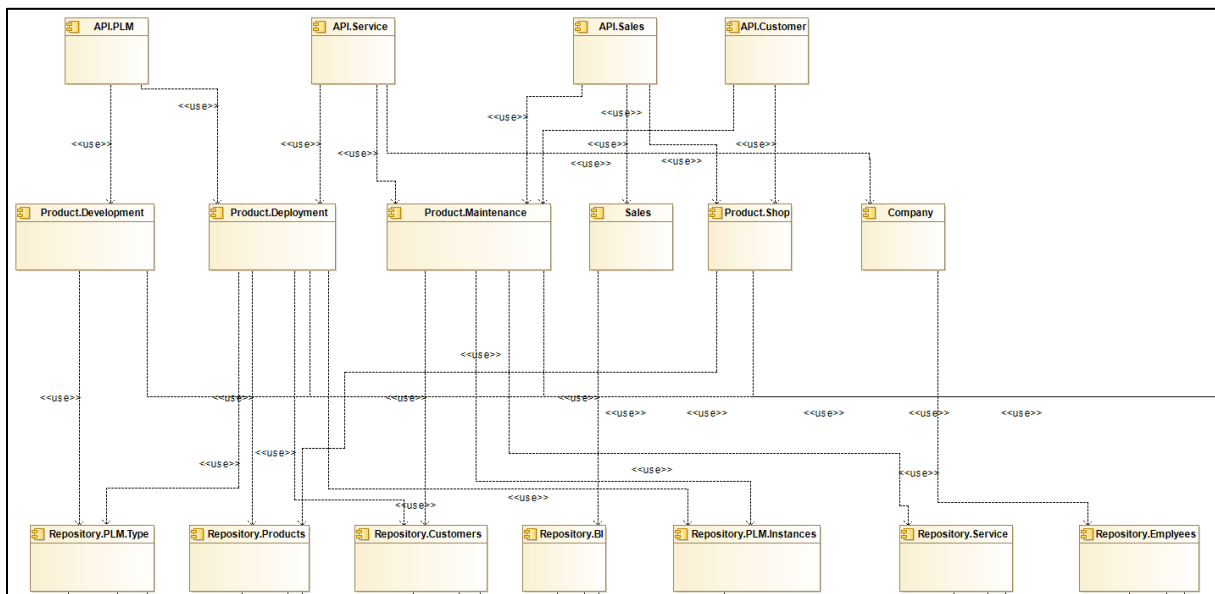


Abbildung 2 – Backbone Architektur

Wie im vollständigen Diagramm zu sehen ist, werden Daten aus verschiedenen Systemen, wie dem PLM System Aras und SAP, zusammengetragen und bereitgestellt. Auf der rechten Seite ist auch eine relationale Datenbank abgebildet welche, zur Verwaltung eigener Daten dient.

Über das Repository-Pattern wurden die Schichten voneinander getrennt. Dies erhöht die Austauschbarkeit und die Testbarkeit erheblich.

Die API wurde auf die vier Teile PLM, Service, Sales und Customer aufgeteilt. Durch Zwischenebenen können die dazugehörigen Daten zugeordnet werden. Die API Teile sind lose den Benutzerrollen Produktmanager, Servicetechniker, Verkäufer und Kunde zugeordnet. Diese Zuteilung ist jedoch nicht immer genau, da dieselben Daten in vielen Fällen von mehreren Benutzergruppen benötigt werden.

Nicht ersichtlich sind weitere Funktionen der Schnittstelle. Dazu gehören ein globales Exceptionmanagement, ein Loggingsystem sowie eine Validierungsmiddleware für die Objekte. Diese Middleware verhindert Under- und Overposting in der REST-Schnittstelle. Dadurch entsteht eine stabile und wartbare Umgebung.

Durch die Einführung eines Datenmockinglayers wurde bei der Modellierung der Klassen und Attribute grosse Freiheiten gewährt. Dies ermöglichte es, benötigte Eigenschaften und Klassen direkt in den Testdaten zu definieren und die Datenbeschaffung aus Umsystemen nicht zu berücksichtigen. Diese Daten werden direkt über einen statischen Service von den Controllern im API Layer verwendet.

Die Datengrundlage stellt sicher, dass alle gewünschten Funktionen der Applikation umgesetzt und dargestellt werden können. Im Backbone wurden keine Businesslogik, Berechnungen und Auswertungen erledigt. Dies muss durch einen weiteren Layer sichergestellt werden.

2.4 User Stories

- US01** Als Verkäufer möchte ich im Dashboard erkennen, bei welchen Kunden die meisten Marketing-Kits verfügbar sind, um den Umsatz zu steigern.
- US02** Als Verkäufer habe ich in der Kundenansicht alle Informationen, wie Kontaktdaten, Standorte, Service Cases, Maschinen sowie verfügbare Marketing-Kits auf einen Blick, um mir einen schnellen Überblick über den Kunden zu beschaffen.
- US03** **OPTIONAL:** Als Verkäufer kann ich eine Bestellung auslösen. Daraus wird direkt ein Service Case erstellt, um eine zeitnahe Bearbeitung durch den Servicetechniker zu gewährleisten und den Fortschritt zu verfolgen.
- US04** Als Verkäufer, Servicetechniker oder Kunde will ich für eine Maschine einen neuen Service Case eröffnen, um Probleme einfach zu erfassen, nachzuverfolgen oder Auswertungen zu erstellen.
- US05** **Vorarbeit:** Als Servicetechniker möchte ich meine Arbeiten pro Service Case dokumentieren, um eine vollständige Änderungshistorie der betreffenden Maschine zu unterhalten.
- US06** **Vorarbeit:** Als Servicetechniker möchte ich als Vorbereitung auf einen Serviceeinsatz die bisherigen Service Cases der Maschine auswerten, um mögliche absehbare Lösungen sowie Ersatzteile und Werkzeuge bereits im Voraus evaluieren zu können.
- US07** Als Verkäufer möchte ich bei der Kundenansicht Vorschläge für vorbeugende Wartungsarbeiten an einer Maschine erhalten, um die Kundenzufriedenheit sowie Verfügbarkeit zu steigern.
- US08** Als Kunde möchte ich eine Übersicht über meinen Maschinenpark. Diese beinhaltet Informationen, wie offene Service Cases und verfügbare Features um die Maschinen zentral zu verwalten.

- US09** Als Produktmanager möchte ich detailliert auslesen können, was meine Kunden im Moment beschäftigt und was für Anforderungen und Probleme sie haben, um meine Produkt- und Marktstrategie dementsprechend anzupassen oder voranzutreiben.
- US10** Als Produktmanager möchte ich ein Marketingpaket für einen oder mehrere Maschinentypen erstellen können. Dieses beinhaltet eine Beschreibung der neu gewünschten Funktionalitäten sowie einen Preis. Nach erfolgreichem Engineering durch die Entwicklung kann ich den Status entsprechend verändern, um Testläufe und Verfügbarkeiten zu steuern.
- US11** Als Anwender der Applikation möchte ich mich mit einem Login in der Applikation authentifizieren, um für meine Rolle zugeschnittene Ansichten präsentiert zu bekommen.

2.5 Funktionale Anforderungen

In der folgenden Übersicht sind die funktionalen Use Cases aufgeführt. Im nächsten Abschnitt werden diese genauer spezifiziert. Die Titel sind jeweils in Englisch, um sie in das Ticketingsystem einzubinden. Die Use Cases sind in drei verschiedene Ausbaustufen aufgeteilt. Diese sollen der Priorisierung dienen. Zuerst müssen die mit Bronze bezeichneten Use Cases umgesetzt werden. So entsteht ein Produkt welches schon benutzbar ist. Erst wenn alle Bronze Use Cases umgesetzt sind kann an den Silber Use Cases gearbeitet werden. Analog verhält es sich mit denen welche als Gold eingestuft sind.

2.5.1 Übersicht

Use Case	Ausbaustufe
UC 100 – Create marketing kit	Bronze
Extension UC 100a – Set price independent per machine type	Gold
UC 110 – Show marketing kit per machine type	Bronze
UC 120 – Show marketing kit per machine	Silber
UC 130 – Update marketing kit instance	Gold
Extension UC 130a – Update state independent per machine type	Gold
UC 140 – Show customer feedback and suggestions	Gold
Extension UC 140a – Filter feedback and suggestions	Gold
UC 150 – Update customer feedback and suggestions for salesman	Gold
UC 200 – Show machines by customer	Bronze
UC 210 – Show machine properties	Bronze
Extension UC 210a – Show service cases of machine	Bronze
Extension UC 210b – Show history of activities	Silber
Extension UC 210c – Show maintenance plan	Gold
Extension UC 210d – Show bill of materials	Silber
UC 300 – Show maintenance plan by customer	Gold
UC 310 – Show marketing kits by customer	Gold
UC 320 – Show customer locations	Bronze
UC 330 – Show customer contact by location	Bronze
UC 340 – Show service cases by customer	Bronze
UC 400 – Create service case	Bronze
UC 410 – Show service case with all activities	Bronze
UC 420 – Update service case	Silber
UC 430 – Create activity from service case	Bronze
Extension UC 430a – Add part	Bronze
Extension UC 430b – Remove part	Bronze
Extension UC 430c – Replace part	Bronze
Extension UC 430d – Maintenance	Silber
UC 440 – Read activity from service case view	Bronze

UC 450 – Update activity from service case view	Bronze
UC 500 – Login per role	Silber
UC 510 – Logout	Silber

Tabella 1 – Übersicht der funktionalen Anforderungen und deren Ausbaustufe

2.5.2 Use Cases

UC 100 – Create marketing kit

Bevor ein Marketingpaket verkauft werden kann, muss der Produktmanager dieses anfordern. Dazu legt er fest für welche Maschinentypen dieses Paket kompatibel ist. Zusätzlich legt er eine Beschreibung der gewünschten Funktionalität sowie den Verkaufspreis, falls bereits verfügbar, an. Das neu eröffnete Marketingpaket wird an die Entwickler weitergeleitet. Für die einzelnen Maschinentypen kann zusätzlich eine Installationsanleitung hinterlegt werden.

Extension UC 100a – Set price independent per machine type

Der Preis kann für jeden kompatiblen Maschinentyp separat festgelegt werden.

UC 110 – Show marketing kits per machine type

Der Anwender kann einsehen, welche Marketingpakete für einen Maschinentyp verfügbar und geplant sind und wann diese veröffentlicht werden.

UC 120 – Show marketing kits per machine

Der Anwender kann einsehen, welche Marketingpakete auf der entsprechenden Maschine verfügbar sind und welche bereits verbaut wurden. Auch ist ersichtlich, welche Marketingpakete sich in der Planung befinden und wann diese veröffentlicht werden.

UC 130 – Update marketing kit instance

Der Status, Preis oder die Installationsanleitung einer Marketingpaketinstanz kann angepasst werden. Dies ist denkbar, wenn ein solches fertiggestellt ist und freigegeben werden soll.

Extension UC 130a – Update properties independent per machine type

Die Eigenschaften können für jeden Maschinentyp separat festgelegt werden, da das Engineering oder Testing nicht immer gleich lange dauern.

UC 140 – Show customer feedback and suggestions

Der Informationsrückfluss des Kunden soll in den Kategorien „Feedback“ und „Vorschläge“ dargestellt werden. Anhand dieser Informationen kann das aktuelle Bedürfnis der Kunden evaluiert werden.

Extension UC 140a – Filter feedback and suggestions

Es kann ein Filter „Aktiv“ und „Archiviert“ gesetzt werden, um stets die richtigen Informationen zu erhalten.

UC 150 – Update customer feedback and suggestions for salesman

Ein ausgewähltes Feedback oder Verbesserungsvorschlag kann nach erfolgreichem Engineering archiviert werden.

UC 200 – Show machines by customer

Es soll eine Übersicht über alle Maschinen im Maschinenpark des Kunden, filterbar nach Niederlassungen angezeigt werden. Der Kunde sieht nur seine eigenen Maschinen.

UC 210 – Show machine properties

Nach der Auswahl einer Maschine werden alle Metainformationen der Maschine dargestellt. Dies umfasst auch eine Anzeige des Standorts sowie ein Bild des Maschinentyps.

Extension UC 210a – Show service cases of machine

In der Maschinendetailansicht können alle Service Cases einer Maschine angezeigt werden.

Extension UC 210b – Show history of activities

In der Maschinendetailansicht wird eine Historie aller jemals ausgeführten Arbeiten an der Maschine, zusammengefasst in Service Cases, angezeigt.

Extension UC 210c – Show maintenance plan

In der Maschinendetailansicht wird angezeigt, wann die nächste Wartung für die Maschine fällig ist.

Extension UC 210d – Show bill of materials

In der Maschinendetailansicht wird in einer Baumstruktur angezeigt, welche Teile in der Maschine verbaut sind. Ausserdem können die Standzeiten sowie das Datum der letzten Wartung pro Maschinenteil angezeigt werden.

UC 300 – Show maintenance plan by customer

Im Dashboard sind alle überfälligen sowie zukünftigen Wartungen für einen Kunden aufgelistet.

UC 310 – Show marketing kits by customer

Im Dashboard sind alle verfügbaren Marketingpakete ersichtlich, die noch nicht verkauft bzw. verbaut wurden.

UC 320 – Show customer locations

Es soll auf einen Blick ersichtlich sein, wo der ausgewählte Kunde Standorte besitzt.

UC 330 – Show customer contact by location

Wird in der Standortübersicht ein Standort ausgewählt, so soll der zuständige Kontakt des Kunden mit seinen Kontaktinformationen angezeigt werden.

UC 340 – Show service cases by customer

Für den ausgewählten Kunden sollen alle Service Cases angezeigt werden. Diese sollten nach den dargestellten Eigenschaften gefiltert werden können.

UC 400 – Create service case

In der Maschinendetailansicht kann ein neuer Service Case erstellt werden. Dabei wird die auszuführende Arbeit oder das Problem erfasst.

UC 410 – Show service case with all activities

In einer Service Case Übersicht wählt der Anwender den Service Case aus der ihn interessiert und öffnet ihn. Das System präsentiert eine Detailansicht mit allen Metadaten und allen Activities in einer Liste.

UC 420 – Update service case

In der Service Case Detailansicht bearbeitet der Anwender die Metadaten oder den Status des Service Cases und speichert diese. Sollten in einem abzuschliessenden Service Case ein Marketingpaket verbaut worden sein, wird dieses ebenfalls gespeichert.

UC 430 – Create activity from service case

Der Servicetechniker ist vor Ort und erstellt eine Aktivität in der Service Case Detailansicht. Er kann dort seine Tätigkeiten erfassen.

Extension UC 430a – Add part

In der Aktivität fügt der Servicetechniker neue Teile in die Maschine ein. Er dokumentiert das in einer Modifikation. Nach dem Bestätigen wird die Stückliste der Maschine entsprechend angepasst.

Extension UC 430b – Remove part

In der Aktivität dokumentiert ein Servicetechniker das Entfernen eines Maschinenteils. Nach dem Bestätigen wird die Stückliste der Maschine entsprechend angepasst.

Extension UC 430c – Replace part

In der Aktivität dokumentiert ein Servicetechniker das Austauschen eines Maschinenteils. Nach dem Bestätigen wird die Stückliste der Maschine entsprechend angepasst.

Extension UC 430d – Maintenance

In der Aktivität dokumentiert ein Servicetechniker das Warten eines Maschinenteils. Nach dem Bestätigen wird die Stückliste der Maschine entsprechend angepasst.

UC 440 – Read activity from service case view

In der Service Case Detailansicht kann eine Aktivität genauer betrachtet werden. Das System zeigt alle Metainformationen der Aktivität und alle Modifikationen an.

UC 450 – Update activity from service case view

In der Detailansicht der Aktivität können die Metadaten und der Status angepasst werden.

UC 500 – Login per role

Es besteht die Möglichkeit sich anzumelden, um sich zu authentifizieren und die passende Rolle im System zu erhalten.

UC 510 – Logout

Es besteht die Möglichkeit sich abzumelden, um sich vor unberechtigtem Zugang und Änderungen zu schützen.

2.6 Nicht-funktionale Anforderungen

In der nachfolgenden Tabelle sind die möglichen Betriebsumgebungen definiert.

Name	Beschreibung des Zustandes
Normal Condition	Alle Services laufen auf dem Server. Der Client verfügt über eine aktive Internetverbindung mit einer Geschwindigkeit grösser als 5 Mbit/s zum Server.
Offline Condition	Alle Services laufen auf dem Server. Der Client hat keine Internetverbindung zum Server. Dies ist zum Beispiel denkbar, wenn der Servicetechniker vor Ort in einer Produktionshalle mit schlechtem Empfang ist.

Tabelle 2 – Betriebsumgebungen

2.6.1 Bachelorarbeit/Prototyp

Diese Anforderungen gelten für die Bachelorarbeit und sind zu erfüllen.

Kompatibilität

Die Anwendung ist mit Google Chrome ab Version 64.0.3325.x verwendbar. Es werden alle UI Elemente auf dem Bildschirm vollständig angezeigt.

Erweiterbarkeit

Die Architektur lässt zu, dass neue Module mit ähnlichem Funktionsumfang, wie bestehende Module, hinzugefügt werden können.

Änderungssensivität

Alle Features des Servicetechnikers sind responsive designed, sodass diese auf Tablets ab 640 Pixel Displaybreite sowie auf dem Desktop bedienbar sind.

Datenintegrität

Das Backend befindet sich jederzeit in einem konsistenten Zustand. Transaktionen unterliegen dem ACID-Prinzip¹.

Bedienbarkeit

Die Applikation soll von allen Anwendern nach einer einstündigen Einführung, durch eine mit der App vertrauten Person, selbständig korrekt bedient werden können.

Sicherheit

Nur privilegierte Personen dürfen über das Frontend auf die geschützte API zugreifen.

2.6.2 Produktives Umfeld

Folgende Anforderungen sind zusätzlich für den produktiven Einsatz zu beachten. Die Werte können dabei variieren und sollten von einem Systemarchitekten vor der Inbetriebnahme neu definiert werden.

Verfügbarkeit 1

Alle Features sind unter Offline Condition zu 90% der Zeit innerhalb von 5 Sekunden erreichbar.

Verfügbarkeit 2

Alle API Anfragen werden unter Normal Condition zu 90% der Zeit innerhalb von drei Sekunden beantwortet. Unter Offline Condition sind die Features per Definition nicht erreichbar. Dabei darf kein Datenverlust eintreten. Der Benutzer wird über die nicht vorhandene Verbindung informiert und kann durch eine Aktion seine Arbeit wieder fortsetzen.

Migration

Es ist einem Systemadministrator möglich, die Datenbasis mit maximal einer Million Datensätzen innerhalb von acht Stunden auf einen anderen Server zu migrieren.

Sicherheit

Nur privilegierte Personen dürfen auf die API zugreifen. Dies gilt für das Frontend, das Backend sowie die Datengrundlage und Umsysteme.

¹ ACID: Atomicity, Consistency, Isolation, Durability

3 Lösungskonzept

3.1 Konzeption Frontend

In der Vorarbeit wurde bereits Angular verwendet. Daher können wichtige Komponenten wie der Stepper für das Erstellen einer Modifikation oder die Baumkomponente für die Darstellung der Stückliste wiederverwendet werden. Auch kleinere Dinge, wie Direktiven oder Pipes, müssen nicht von Grund auf neu entwickelt werden.

Ein weiterer Grund für Angular war die intuitive Aufteilung der Applikation in Module welche auf die verschiedenen Routen verteilt werden können. So ist es möglich, die Module mittels eines Sicherheitslayers aufzuteilen. Dies erlaubt das Rollenkonzept so umzusetzen, dass dieses jederzeit sehr einfach angepasst werden kann.

In der Studienarbeit wurde neben Angular auch Komponenten von Material Design eingesetzt. Um zusätzliche Layout-Möglichkeiten zu erhalten, wurde auf Bootstrap gewechselt. Diese Entscheidung ist im folgenden Kapitel genauer erläutert.

3.2 Cockpit Evaluation

Für das Angular Frontend werden einige Elemente wie die Baumstruktur für die Stückliste oder Graphen benötigt. Ausserdem ist das Frontend als Dashboard konzipiert. In diesem Abschnitt werden drei verschiedene Dashboard-Lösungen evaluiert, sodass eine fundierte Entscheidung für den Einsatz in diesem Projekt getroffen werden kann. Bei der Evaluation wurde vor allem darauf geachtet, dass die Formelemente und Tabellen die Anforderungen an das Frontend ohne grossen Mehraufwand erfüllen. Die Dashboard-Lösung soll zusätzlich gut mit Bootstrap funktionieren. Die verschiedenen Punkte wurden von 1 (nicht vorhanden/unbrauchbar) bis 5 (sehr gut) bewertet.

3.2.1 Nebular

Nebular bietet ein Set an Modulen für eine Angular Applikation, um die Entwicklungszeit zu verkürzen. Nebular ist mit der MIT Lizenz kostenlos und bietet folgende Kernmodule an [1].

UI Kit

In diesem Kit sind sämtliche gängigen UI Komponenten vorhanden. Zudem wird von den Entwicklern explizit empfohlen weitere CSS-Frameworks wie Bootstrap oder Angular Material, welches in der Vorarbeit eingesetzt wurde, zu verwenden.

Theme System

Standardmässig werden zwei Themes zur Verfügung gestellt. Diese Themes lassen sich ohne Page reload ändern. Anhand dieser Beispiele können eigene Themes mit vordefinierten Variablen erstellt werden.

Auth Module

Inbegriffen ist ein Modul für die Authentifizierung. Es wird ein Zwischenlayer erstellt, welcher es ermöglicht, die Authentifizierung zum Beispiel über JWT zu lösen. Dabei muss im Backend eine entsprechende API zur Verfügung gestellt werden.

Security Module

Dieses Modul ermöglicht, es ein Rollen- sowie Zugriffskonzept zu erstellen und zu verwalten.

Show Cases

Das ngx-admin Projekt zeigt auf, wie ein Admin Template beispielhaft mit allen verfügbaren Komponenten aus dem UI Kit aussehen kann [2].

Nebular baut auf nativen Angular Komponenten auf. Die Show Cases wurden unter Verwendung von Angular 5+, Bootstrap 4 und Nebular erstellt.

Komponenten	Bewertung	Begründung
Tree	5	Verwendet ng2-tree ² , welcher Drag & Drop, Context Menu, Checkable, Custom Icons sowie Lazy loading unterstützt. Als Alternative könnte auch MatTree von Angular Material verwendet werden. Dieser ist jedoch momentan noch in der Beta.
Inputs, Forms	3	Übliche Inputs inklusive Validierung vorhanden. Ein Datepicker ist nicht vorhanden.
Tables	4	Es ist eine Smart Table mit Sortierung und Filterung vorhanden.
Charts	5	Verwendung von Echarts, Charts.js sowie D3 sind möglich.
Auth / Security	4	Wird direkt mitgeliefert und ist nach der Anpassung an die API verwendbar. Erklärung in der Dokumentation ist vorhanden.

Tabelle 3 – Auswertung Komponenten Nebular

3.2.2 PrimeNG

PrimeNG ist eine Bibliothek von nativen UI Komponenten für Angular [3]. Sämtliche Komponenten sind Open Source und mit der MIT Lizenz verwendbar. Es ist eine Auswahl von gratis verfügbaren Standard-Themes vorhanden. Es besteht zusätzlich die Möglichkeit eine Lizenz für professionell vorgefertigte Themes zu kaufen [4]. Die Lizenzkosten belaufen sich je nach Theme zwischen \$200-\$1'000.

Komponenten	Bewertung	Begründung
Tree	5	Unterstützt Drag & Drop, Context Menu, Checkable, Custom Icons sowie Lazy loading.
Inputs, Forms	4	Alle in der Vorarbeit verwendeten Komponenten sind vorhanden.
Tables	5	Diverse Beispiele mit JSON werden zur Verfügung gestellt. Die Filterung, Sortierung und Pagination sind ebenfalls möglich. Zudem gibt es noch DataView, die die Daten in einem Grid oder List Layout anzeigt.
Charts	3	Basierend auf Chart.Js als einzige Komponente.
Auth / Security	1	Nicht vorhanden im Open Source Projekt.

Tabelle 4 – Auswertung Komponenten PrimeNG

² <https://github.com/valor-software/ng2-tree>

3.2.3 Metronic

Metronic bietet ein Template für die Erstellung von Backend Systemen, SASS Applications, Modern Dashboards und vieles mehr [5]. Metronic ist aktuell auf Themeforest.net für \$1'400 mit erweiterter Lizenz erhältlich. Inklusive eines 6-monatigen Supports sowie lebenslang gratis Updates. Metronic wurde bereits über 60'000 Mal verkauft und hat über 5'700 positive Bewertungen.

Da Metronic nicht Open Source ist, lassen sich die Funktionen sowie der Aufbau nur durch die Online Dokumentation erahnen. Ersichtlich ist, dass nicht nur Angular native Komponenten verwendet werden sondern auch ein grösserer Anteil an jQuery. Dies wird durch schnellere Entwicklungszeit sowie mehr Möglichkeiten begründet. Das Grundgerüst der Applikation ist aber mit Angular aufgebaut [6].

Komponenten	Bewertung	Begründung
Tree	5	Unterstützt Drag & Drop, Context Menu, Checkable, Custom Icons sowie Lazy loading.
Inputs, Forms	5	Alle in der Vorarbeit verwendeten Komponenten sind vorhanden. Zudem gibt es noch weitere wie Touchspin und Timeline.
Tables	5	Diverse Beispiele mit JSON sind vorhanden. Die Tabellen unterstützen Filterung, Sortierung und Pagniation.
Charts	4	Zur Verfügung stehen amCharts, Float Charts, Google Charts sowie Morris Charts. Zum Teil muss jedoch auf den Urheber verwiesen werden.
Auth / Security	2	Ist in der Filestruktur vorhanden, aber nicht näher in der Dokumentation beschrieben.

Tabelle 5 – Auswertung Komponenten Metronic

Entscheidung

Die Bibliotheken von Nebular und PrimeNG wurden in ihrer Open Source Version genauer unter die Lupe genommen, da vom Auftraggeber eine lizenzfreie Variante bevorzugt wird. Zu diesen wurde jeweils ein Prototyp erstellt.

Bei beiden Bibliotheken konnte erfolgreich ein Prototyp erstellt werden. Dieser beinhaltet jeweils einen Tree mit auswählbaren Nodes und eine Tabelle mit Servicetechnikern, die per API-Request vom Backend geladen wird. Die Tabelle verfügt in beiden Varianten über eine funktionierende Sortierung, Filterung sowie Pagniation.

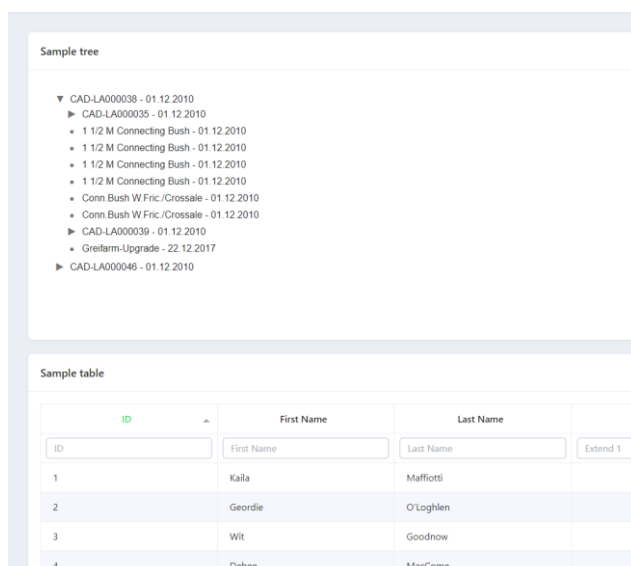


Abbildung 3 – Prototyp Nebular

Der Prototyp von Nebular kommt sehr schlank und modern daher. Vor allem die Tabelle überzeugt mit eingebauter Filterung und Sortierung. Die Einbindung der Komponenten konnte sehr gut vorgenommen werden. Dies lag auch an der ausführlichen Dokumentation und den guten Demobeispielen.

Tree example of Lego BOM

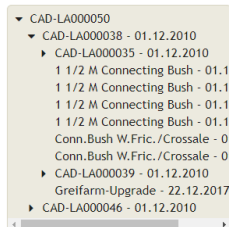


Table example with sort and filtering

Personal Number ^	First name ↕	Last name ↕	Extend 1 ↕	Extend 2 ↕	Extend 3 ↕
1	Kaila	Maffiotti			
2	Geordie	O'Lughlen			
3	Wit	Goodnow			
4	Debee	MacCome			
5	Beulah	Jamison			
6	Ivett	Heaviside			
7	Alta	Janczak			
8	Clarey	Thwalte			
9	Bryan	Connealy			
10	Karen	Marsh			

Abbildung 4 – Prototyp PrimeNG (Free)

Im Gegensatz zu Nebular sieht dieser Prototyp nicht ganz so gut aus. Dies kann natürlich mit etwas Aufwand beim Theming angepasst werden. Die Baumkomponente ist sehr ähnlich wie diese von Nebular. Bei der Tabelle steht ein globaler Filter zur Verfügung. Die Einbindung der Komponenten war wiederum relativ einfach.

Da von den Funktionalitäten kein grosser Unterschied erkennbar ist, Nebular jedoch über ein besseres Theming in der Open Source Variante verfügt und dazu ein Authentication/Security Layer vorhanden ist, wird Nebular zusammen mit Angular Bootstrap verwendet. Durch diese Entscheidung wurde Material Design etwas in den Hintergrund gerückt und Bootstrap verwendet. Bootstrap erlaubt es, das Dashboard und deren Karten genau nach Wunsch anzuordnen. Auf Metronic wurde wegen des Bezahlmodells verzichtet, obwohl die Komponenten sehr vielversprechend aussahen.

3.3 Konzeption Backend API

Die Backend API befindet sich zwischen dem Frontend und dem zur Verfügung gestellten Backbone. Die Komponenten kommunizieren über eine JSON-Schnittstelle. Die Aufgabe des Backends ist es, die Daten aus dem Backbone zusammenzufügen, zu bearbeiten und im vom Frontend benötigten Format bereitzustellen.

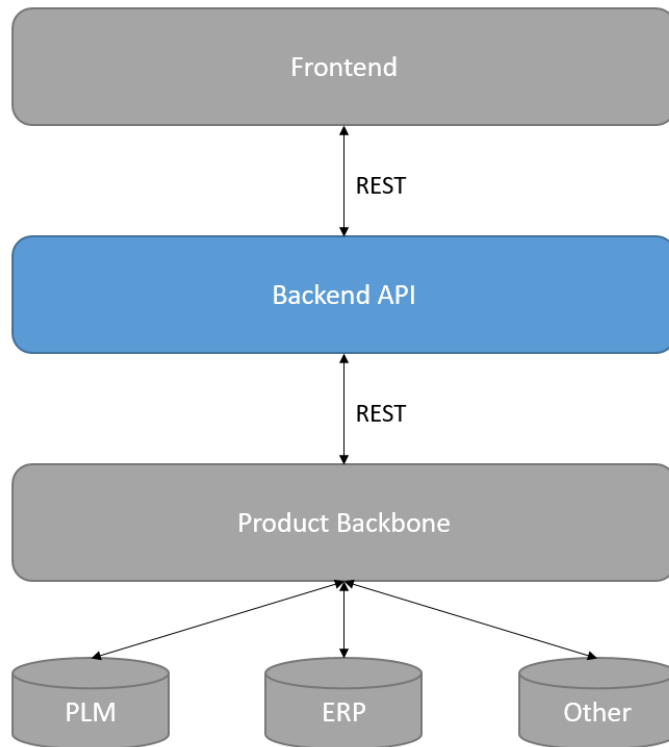


Abbildung 5 – Big Picture Backend API

Es wurde als erstes ein Prototyp erstellt, um die Architektur auf ihre Machbarkeit zu prüfen. In der Studienarbeit wurde ASP.NET Core 2.0 Web API eingesetzt. ASP.NET Core 2.1 wird voraussichtlich Mitte 2018 veröffentlicht. Es gibt jedoch schon einen Preview der neuen Version. Dieser wurde für den Prototyp eingesetzt, da ASP.NET Core bereits in der Studienarbeit erfolgreich evaluiert wurde.

3.3.1 Architektur

Die Architektur ist unterteilt in drei Schichten. Die unterste Schicht Data Access kommuniziert mit dem Product Backbone. In der Mitte befindet sich der Business Logic Layer (BLL), welcher für die Verarbeitung der Daten zuständig ist. Zuerst verwaltet die API-Schicht alle Anfragen aus dem Frontend und sendet die Daten aus dem BLL über die REST-Schnittstelle.

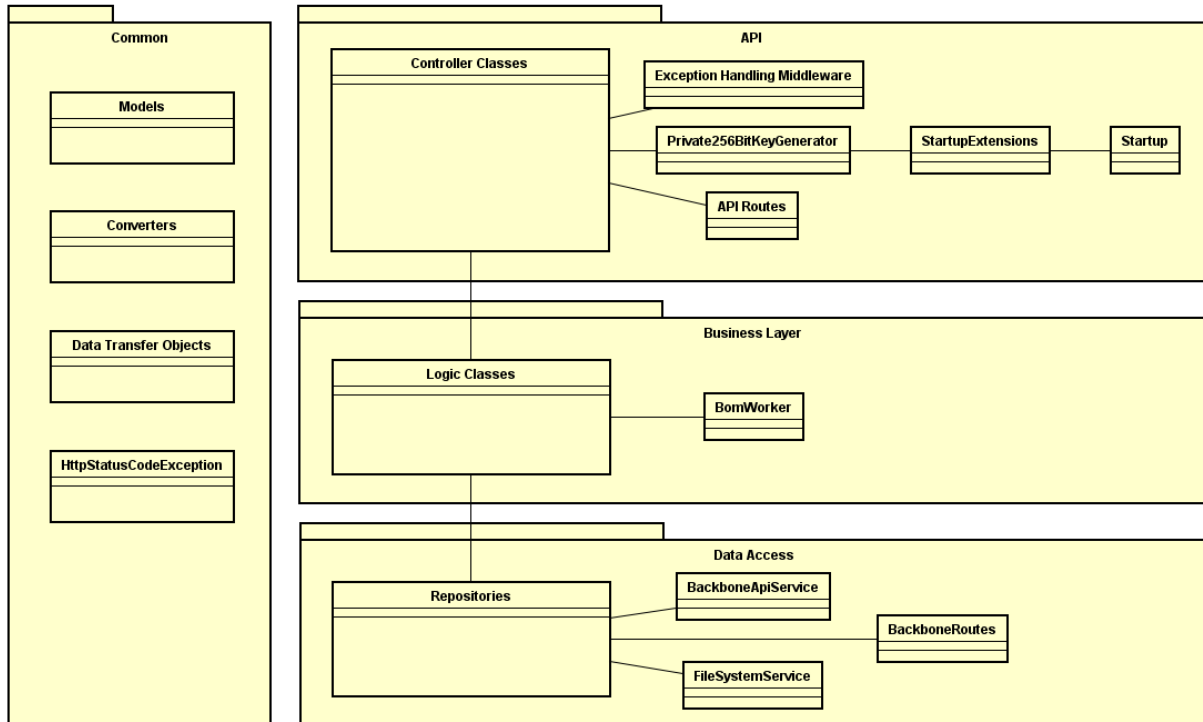


Abbildung 6 – Architektur Backend

In der folgenden Abbildung ist das von der Applikation benutzte Domänenmodell abgebildet. Ein Exemplar im Querformat ist ebenfalls im Anhang einzusehen.

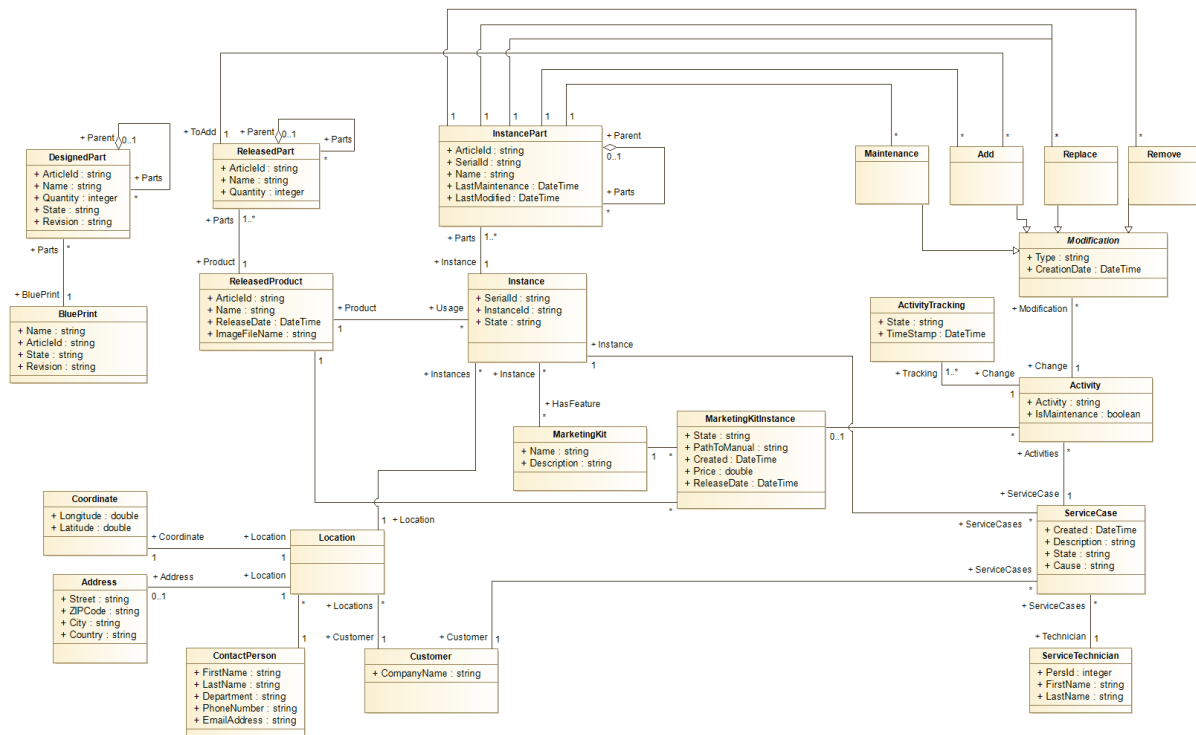


Abbildung 7 – Domänenmodell

Bei gewissen Objekten wurde ein Integer und bei anderen eine GUID für die Identifizierung genutzt. Das kommt davon, dass Integer besser von Menschen gelesen werden können. Dies ist zum Beispiel für eine Service Case Nummer oder eine Kundennummer wichtig, falls diese jemals auf ein Dokument gedruckt werden. Global eindeutige Identifikationsnummern haben den Vorteil, dass diese auf der ganzen Welt eindeutig sind. Daher muss man sich keine Gedanken betreffend doppelter Werte machen. Sie werden unter anderem aus Zeit- sowie CPU-Daten gewonnen [7].

Die Objekte, welche über die API zur Verfügung gestellt werden, haben unter Umständen andere Attribute als die Entitäten im Domänenmodell, weil versucht wurde nur die vom Frontend benötigten Eigenschaften zur Verfügung zu stellen.

Im Domänenmodell selber gibt es zwei eher spezielle Auffälligkeiten. Zum einen wurde für die Stückliste eine Composite-Struktur genutzt. Zum anderen ist die Klasse Modification mit dem Typattribut bereits eindeutig einer Modifikationsart zugewiesen. Daher werden die vier ableitenden Klassen nicht benötigt und sind nur zur Visualisierung im Domänenmodell.

3.3.2 Error Handling

Bei der Kommunikation über die Schnittstellen können Fehler auftreten. Zum Beispiel kann versucht werden, ein Objekt anzulegen, deren ID bereits vergeben ist. Damit die Applikation bei einer Exception verfügbar bleibt, wurde eine Exception Handling Middleware verwendet. Diese schaltet sich ein, wenn eine Exception bei einer Anfrage ausgelöst wird und sendet einen Fehler an den Client. Alle Ereignisse inklusive Fehler werden zwecks Nachverfolgbarkeit in ein Logfile geschrieben.

Es kann auch sein, dass vom Frontend falsche oder unvollständige Daten kommen. Diese werden mittels eines Validierungsschrittes geprüft und vorzeitig erkannt. Dies ist wichtig, damit sich Fehler nicht im System fortpflanzen können.

3.3.3 Asynchronität

Alle JSON Schnittstellen werden asynchron angesteuert. So kann gewährleistet werden, dass auch mehrere gleichzeitige Anfragen korrekt abgearbeitet werden können.

3.3.4 Testing

Durch die Aufteilung in Schichten mittels Interfaces können diese unabhängig voneinander getestet werden. Dies ist wichtig, da das Backbone nicht immer dieselben Daten liefern kann. Daher sind auch die Unit Tests auf einzelne Methoden bezogen und benötigen sehr wenige Aufbauarbeiten. Die Interfaces wurden mit NSubstitute simuliert. Dieses Framework erlaubt es für Funktionsaufrufe Rückgabewerte zu definieren ohne dass der ganze Stack aufgebaut werden muss.

3.3.5 Authentifizierung

Zur Authentifizierung wird ein JSON Web Token (JWT) eingesetzt. Dies ist ein Standard, welcher einen sicheren Weg zum Austausch von JSON Objekten bietet.

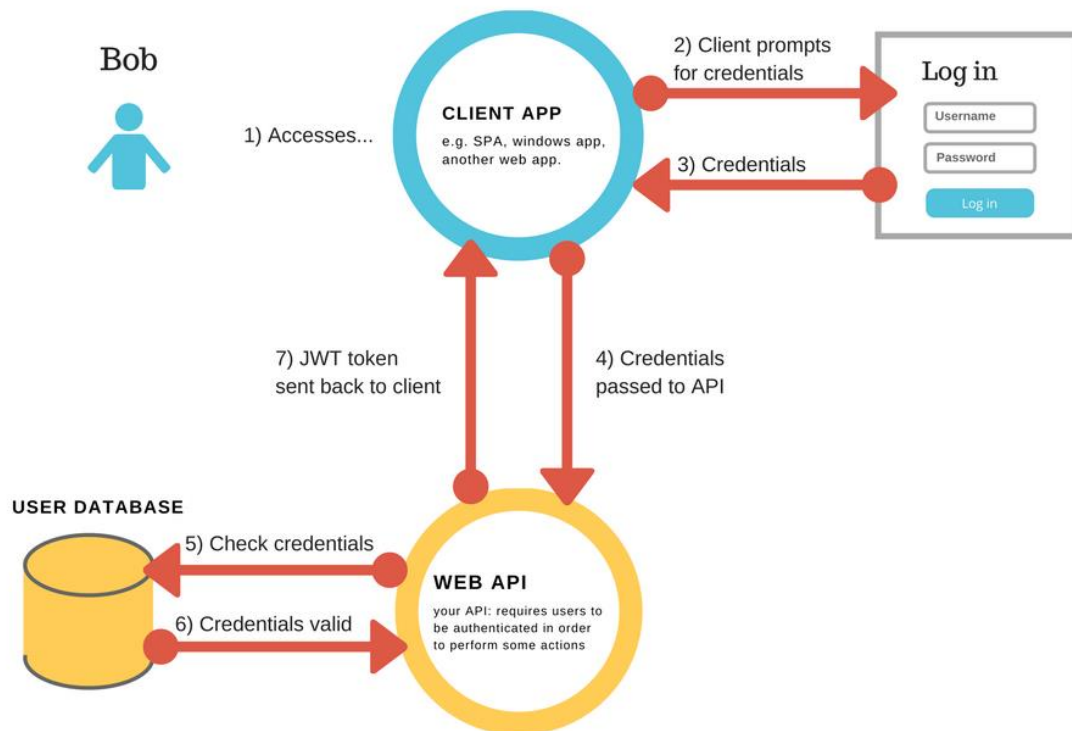


Abbildung 8 – Beispielhafter JWT Workflow [8]

Der Benutzer authentifiziert sich gegenüber der Web API über ein JSON Objekt. Das Backend generiert einen JWT. Dieses Token wird dem Benutzer gesendet sodass er dieses speichern kann. Bei jeder Anfrage muss er dieses Token mitschicken, damit die API erkennt, welche Rolle diesem Benutzer zugeordnet ist.

Da das Backend über HTTP und nicht über HTTPS kommuniziert, kann das Token mittels einer Man-in-the-Middle Attacke abgegriffen werden. Durch eine Umstellung auf HTTPS könnte dies unterbunden werden. Da es sich hier jedoch um einen Prototypen handelt, wird dies vernachlässigt.

ASP.NET Core bietet out-of-the-Box bereits die Möglichkeit, Benutzer aufgrund ihres Tokens für gewisse API Calls zu autorisieren. Dieses Feature wird im Prototyp genutzt. Die Token werden lediglich

für die Zuweisung der Rolle verwendet. Eine Identifizierung, zum Beispiel nach einem bestimmten Kunden ist nicht geplant.

3.3.6 Request Pipeline

Um Themen wie Exceptions oder Validitätschecks global ausführen zu können, werden diese als Filter in der Request Pipeline hinterlegt. Diese können mittels Dependency Injection in die Anwendung geladen werden [9].

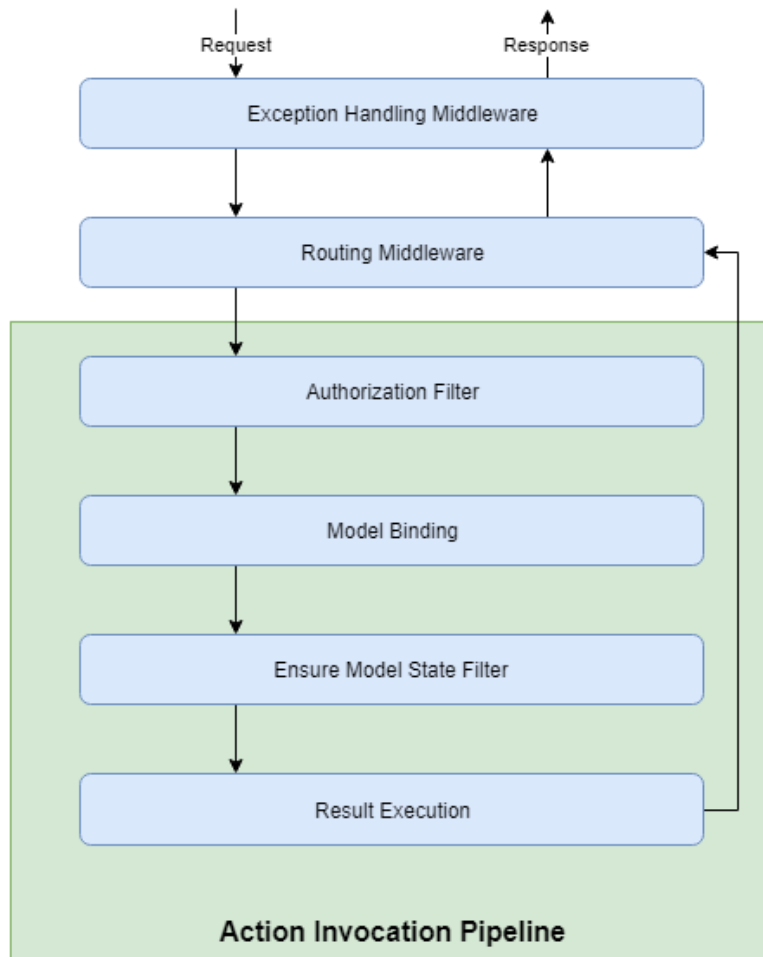


Abbildung 9 – Request Filter Pipeline

Zuoberst ist die Exception Handling Middleware angesiedelt. Sie fängt alle Exceptions ab und führt nötige Schritte zur Wiederherstellung eines konsistenten Zustands durch. Die Routing Middleware sucht die für den Request korrekte Route und den passenden Controller aus der Source und leitet den Request entsprechend an den Authorization Filter weiter. Dieser prüft ob die Berechtigungen für die gewählte Methode vorhanden sind.

Das Model Binding konvertiert eventuelle JSON-Objekte im Body der Anfrage auf tatsächliche Objekte. Danach wird im Ensure Model State Filter geprüft, ob dieses Binding korrekt funktionierte.

3.4 Product Backbone

Die Daten für das Backend sollen aus den Umsystemen wie PLM, ERP, etc. aggregiert werden. Diese Aggregation ist ausserhalb des Scopes der Bachelorarbeit. Daher werden diese Daten vorläufig selbst zur Verfügung gestellt. Zu diesem Zweck wird hinter den Controllern des Backbones ein Layer für Mockingdaten eingeführt.

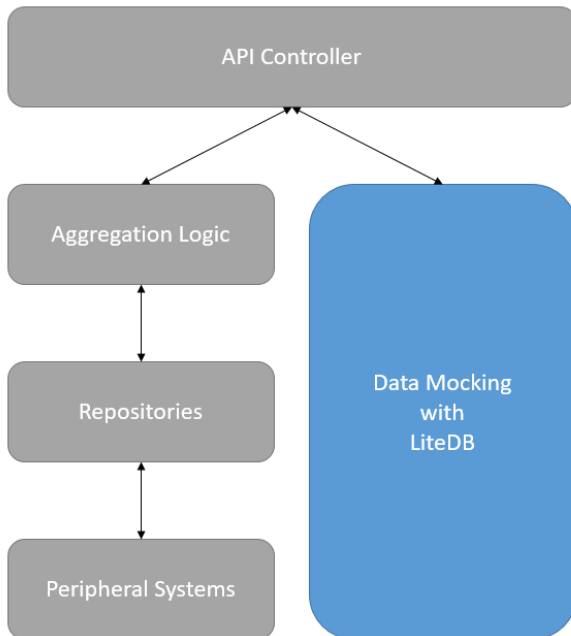


Abbildung 10 – Backbone Architektur

Auf diesem Weg kann beschrieben werden, welche Daten aus den Umsystemen erwartet werden wenn die Testdaten abgehängt werden. Für das Bereitstellen der Testdaten wurde LiteDB evaluiert. Es wurde nach einer einfachen Datenbanktechnologie gesucht, welche das Speichern und Verändern von Objekten erlaubt. Durch die Wahl einer NoSQL Technologie kann mit häufigen Änderungen am Code und dem Datenbankschema gut umgegangen werden. Auch das Arbeiten mit Objekten ist bei einer NoSQL Datenbank vereinfacht. Der grösste Anbieter von NoSQL Lösungen im Moment ist MongoDB. Dieser bietet zwar alle gewünschten Features, ist aber doch zu gross dimensioniert für dieses Projekt. Die Wahl fiel am Ende auf LiteDB da diese explizit für .NET Projekte entwickelt wurde. Die DLL ist mit ca. 350 kB sehr klein und eignet sich für einfache Anwendungsfälle wie diese. Auch bietet LiteDB Features wie ACID Transaktionen, LINQ Support und Verweise auf andere Datenkollektionen welche das Arbeiten vereinfachen [10].

Die Testdaten werden jeweils beim Start der Backbone Applikation direkt aus dem Code gelesen und in der Datenbank initialisiert. So muss bei einer Anpassung des Datenmodells kein Konsistenzcheck durchgeführt werden.

3.5 Werkzeuge und Technologien

Als Entwicklungsumgebungen werden *Visual Studio 2017 Enterprise* von Microsoft für das ASP.NET Backend sowie *WebStorm 2018* von JetBrains für das Frontend in Angular eingesetzt. Beide eignen sich sehr gut für die Entwicklung der jeweiligen Technologien und bieten viele angenehme Features wie IntelliSense, Dokumentation und statische Codeanalyse. Alle Teammitglieder verwenden dieselben Versionen der Software, um unerwarteten Fehlern vorzubeugen.

Im Backend wird *ASP.NET Core Web API* von Microsoft eingesetzt. Damit kann die API Schnittstelle zur Verfügung gestellt werden. Ausserdem gibt es grosse Vorteile beim Einsatz von Middleware, zur

Sicherstellung der Fehlertoleranz oder der Validierung der JSON Objekte. Zusätzlich hat es sich in der Studienarbeit auch schon bewährt. Es wird versucht, sich an die Richtlinien von API Best Practices zu halten, um eine wiederverwendbare und erweiterbare API anzubieten [11]. Swagger wird als API Dokumentationssoftware eingesetzt. Dies hat sich in den letzten Jahren durchgesetzt und bietet Features wie automatische API Dokumentation und das Absetzen von Anfragen zu Testzwecken während Integrationstests.

Git wurde als Versionskontrollsystem, welches direkt in WebStorm und Visual Studio integriert werden kann, evaluiert. Als CI/CD-Server wurde *Jenkins*, welches open source ist und Builds von privaten Repositories unterstützt, eingesetzt. Weiterhin werden *Astah Community* für UML-Diagramme, *Dropbox* als DMS, *Slack* als Kommunikationstool und die *Office Suite* für die Dokumentenerstellung benutzt.

3.6 Deploymentkonzept

Die Applikation wurde komplett in ASP.NET Core und Angular entwickelt und kann daher plattformunabhängig gestartet werden. Es existieren keine Abhängigkeiten zu einer Windows-Umgebung. Die Applikation ist so konfigurierbar, dass das Backbone, das Backend und das Frontend auf verschiedenen Geräten läuft. Die Skalierung ist horizontal möglich.

Werden Backbone und Backend auf demselben Server bereitgestellt, hat dies einen positiven Einfluss auf die Performance, da Netzwerkaufrufe nicht so oft geroutet werden müssen. Für die Entwicklung der Applikation wurden Staging Sites eingerichtet. So konnten Änderungen im Zusammenspiel mit den anderen Komponenten getestet werden.

Das Deployment der produktiven Instanz wurde mit Jenkins realisiert. Über einen Webhook wird Jenkins über neue Versionen im Master Branch des VCS benachrichtigt. Jenkins holt sich dann automatisch den neusten Code, testet diesen und liefert ihn aus. Die Testumgebungen und das Backbone wurden mit Skriptdateien nach demselben Verfahren ausgeliefert. Dabei war es wichtig, persönlich die Kontrolle über die Versionen des Stagings zu haben. Deshalb werden die Skripte manuell auf dem Stagingserver ausgeführt.

3.7 Schnittstellenkonzept

Die API des Backends ist die Schnittstelle für das Angular Frontend und stellt die benötigten Daten zur Verfügung. Folgende Endpunkte sind im Backend definiert.

Endpunkt	Verbs	Optionale GET-Parameter
Authentication	POST	
ContactPersons	GET	
ContactPersons/{id}	GET	
Customers	GET	
Customers/{id}	GET	
Locations	GET	CustomerId
Locations/{id}	GET	
MarketingKits	GET, POST	ReleasedProductId MarketingKitInstanceId
MarketingKits/{id}	GET	
MarketingKits/{id}/Marketingkitinstances	POST	
MarketingKits/{id}/Marketingkitinstances/{id}	PUT	
MarketingKits/{id}/Marketingkitinstances/{id}/Manual	GET, POST	
ProductInstances	GET	CustomerId
ProductInstances/{id}	GET	
ProductInstances/{id}/Bom	GET	

ReleasedProducts	GET	
ReleasedProducts/{id}	GET	
ReleasedProducts/{id}/Part/{id}	GET	
ReleasedProducts/{id}/Bom	GET	
ServiceCases	GET, POST	ProductInstanceId CustomerId ReleasedProductId
ServiceCases/{id}	GET, PUT	
ServiceCases/{id}/Activities	GET, POST	
ServiceCases/{id}/Activities/{id}	GET, PUT	
ServiceCases/{id}/Activities/{id}/ActivityTrackings	POST	
ServiceCases/{id}/Activities/{id}/Modifications	GET, POST	
ServiceTechnicians	GET, POST	
ServiceTechnicians/{id}	GET	
Statistics/TopMachineTypesByServiceCase	GET	
Statistics/TopMarketingKitsToSell	GET	

Tabelle 6 – API Endpoints

Eine vollständige API-Dokumentation wird jeweils immer durch Swagger generiert. Diese ist unter dem Endpunkt [http://\[BACKENDADRESSE\]/swagger](http://[BACKENDADRESSE]/swagger) zu finden. Ein Export der Dokumentation zu Projektende ist als HTML in der digitalen Abgabe enthalten.

3.8 Berechtigungskonzept

Innerhalb der Applikation werden vier verschiedene Rollen unterschieden. Diese sind Verkäufer, Produktmanager, Servicetechniker und Kunde. Der Kunde ist dabei speziell, da dieser auf einen tatsächlichen Kunden verweist und er auch der Einzige ist, welcher ausserhalb der Unternehmung Zugriff auf das System hat. Die anderen Rollen haben für jeden angemeldeten Benutzer derselben Rolle dasselbe Verhalten.

Hat sich ein Benutzer angemeldet, bekommt er seine Rolle zugewiesen und die Applikation liefert alle Ansichten, welche für die Arbeit benötigt werden. So kann zum Beispiel ein Kunde kein neues Marketing Kit erfassen oder Änderungen an Service Cases vornehmen.

Die Berechtigungen der Rollen werden an zwei verschiedenen Orten durchgesetzt. Einerseits werden im Frontend nur die Daten dargestellt, auf welche die Rolle Zugriff hat. Andererseits kann bei den Backend API Aufrufen anhand des oben beschriebenen JSON Web Token die Rollenzugehörigkeit geprüft werden.

Die einzelnen Berechtigungen wurden auf Stufe Use Case umgesetzt. Die Zuteilung der Rollen auf die einzelnen Use Cases sind im Use Case Diagramm ersichtlich. Dieses befindet sich aufgrund des Umfangs im Anhang.

Die Berechtigungen im Frontend können zentral angepasst werden. Dazu müssen in der Datei *role.config.ts* die entsprechenden Ressourcen in der Variable *SECURITY_CONFIG* den Wünschen entsprechend verändert werden.

Um die Berechtigungen im Backend zu konfigurieren, werden Attribute des Packages *Microsoft.AspNetCore.Authorization* verwendet. Diese können über einen ganzen Controller oder auch nur für einzelne Routen verwendet werden [12].

3.9 Funktionsmanagement

Ein wichtiger Teil der Applikation ist die Verwaltung von Funktionen der Produkte. Damit könnte zum Beispiel eine Mobilitätsfunktion oder eine zusätzliche Verarbeitungsfunktion gemeint sein. Die Grundidee ist es, Produkte nicht nur anhand von physisch miteinander interagierenden Teilen und Baugruppen zusammenzustellen und zu entwickeln, sondern mittels Funktionen eine andere Sicht auf das Produkt zu erhalten. So kann man zum Beispiel ein Mobiltelefon wie folgt in Funktionen unterteilen.



Abbildung 11 – Funktionsbeispiel Smartphone

Wie in der Abbildung zu sehen ist, werden der Entität Mobiltelefon gewisse Basisfunktionen als gegeben zugewiesen. Funktionen wie die Frontkamera sind zusätzliche Funktionen. Diese bestehen nicht einfach als Baugruppe, welche in ein Teil des Mobiltelefons eingebaut wird. Sondern sie besteht unter Umständen aus Hardwareteilen an verschiedenen Orten im Telefon und auch aus Software, welche die Funktion unterstützen muss.

Diese Art der Produktspezifikation ist vor allem wichtig für den Produktmanager und den Kunden. Sie können sich Fragen stellen wie „Wann kann meine Maschine auch Holz verarbeiten?“ oder „Ist es möglich, zwei Produkte gleichzeitig zu produzieren?“. So ist es möglich durch ein gutes Management von solchen Funktionen die Kommunikation aus der Entwicklung zum Kunden zu verbessern und auch effizient nach Funktionen zu entwickeln.

Die Frage ist nun, wie ein solches Management auszusehen hat. Ein zur Verfügung gestellter Vorschlag sieht wie folgt aus.

```

Function{
  name      String      1
  type      FunctionType 1
  var       String      0..1
  cond      String      0..*
  sub       Function    0..*
  in        Descriptor  1..*
  out       Descriptor  1..*
  props    Property     0..*
}

Descriptor{
  type      DescriptorType 1
  flow      Flow           1
  var       String         0..1
  cond      String         0..*
  act       ActivityType   0..1
  before    Descriptor     0..1
  after     Descriptor     0..1
}

Flow{
  name      String      1
  type      FlowType    1
}
    
```

Abbildung 12 – Konzeptionsvorschlag Betreuer

Die Klassen sind in nachfolgender Tabelle genauer beschrieben.

Eigenschaft	Bedeutung
Name	Ein treffender Name zur Beschreibung der Funktion.
Type	Eine Klassifikation der Funktion je nach Bedürfnis.
Var	Bezeichnet verschiedene Varianten der Funktion falls verschiedene Ausführungen der Funktion verfügbar sind.
Cond	Beschreibt die zu erfüllenden Vorbedingungen, welche nötig sind um die Funktion zu verbauen.
Sub	Eine Menge an Unterfunktionen, welche die betreffende Funktion beinhalten.
In	Beschreibt eine Menge von Deskriptoren, welche den Input spezifizieren. Dieses Modell setzt mindestens einen Input voraus.
Out	Beschreibt eine Menge von Deskriptoren, welche den Output spezifizieren. Dieses Modell setzt mindestens einen Output voraus.
Props	Hier werden alle Eigenschaften der Funktion als Name-Value Paare hinterlegt.

Tabelle 7 – Beschreibung Klasse Function

In dieser Spezifikation fehlen jedoch jegliche Managementinformationen. Zum Beispiel ist nicht ersichtlich, wann die Funktion marktreif sein könnte oder welchen Produkttypen sie zugewiesen werden kann. Auch ein Bedürfnis für den Verkäufer und den Kunden ist der schlussendliche Kaufpreis der Funktionalität. Für den Servicetechniker ist ausserdem wichtig zu wissen, wie die Funktion beim Kunden verbaut werden soll. Unter Berücksichtigung dieser Punkte wurde folgendes Konzept erarbeitet.

```

MarketingKit {
    Name                String                1
    Description          String                1
    ProductInstances    ProductInstance      0..*
    MarketingKitInstances MarketingKitInstance 0..*
}

MarketingKitInstance {
    State                DevelopmentState    1
    PathToManual        String                0..1
    Price                Double               0..1
    Created              Date                 1
    ReleaseDate          Date                 1
    ReleasedProduct     ReleasedProduct      1
}

```

Abbildung 13 – Konzept Featuremanagement

Der Name ist analog zum Vorschlag des Betreuers möglichst sprechend zu vergeben. Bei der Beschreibung ist es möglich, mehr Informationen zu hinterlegen. In den Produkt Instanzen wird festgehalten in welchen Maschinen diese Funktion bereits verbaut ist.

Die Marketing Kit Instanzen sind dann einzelne Umsetzungen der Funktion welche jeweils auf einen bestimmten Maschinentyp zugeschnitten sind. Darin sind spezifische Informationen, wie das (geplante) Release Datum, der Preis oder der Pfad zur Einbauanleitung für den Servicetechniker gespeichert. Dieses Modell erlaubt allen Rollen, die benötigten Informationen an einem Ort abzufragen.

Erweiterungen des Modells wie Subfunktionen, Varianten, Vorbedingungen oder komplementäre Funktionen wurden aussen vor gelassen, da diese vor allem darauf abzielen ob ein gewisses Feature

überhaupt verbaut werden kann, beziehungsweise das gleichzeitige Verbauen von zwei Varianten zugelassen ist.

Durch die Subfunktion wird eine logische Hierarchie erstellt. Dies spricht gegen das reine Anforderungskonzept. Dabei sollen Features eigenständig sein und nur Funktionen einer Maschine abbilden und keine Struktur haben.

Auch die Input- und Outputdeskriptoren wurden gestrichen da es nur wenige Use Cases gibt bei welchen eine solche Beschreibung verwendet werden kann. Falls dies nötig ist können diese in der Beschreibung des Marketing Kits textuell erfasst werden.

3.10 Testkonzept

Während der Entwicklungsphase wird kontinuierlich getestet. Dabei sollten während der Entwicklung eines Arbeitspakets jeweils Unit Tests für die einzelnen Methoden geschrieben werden. Nach dem Abschluss eines Arbeitspakets werden Integrationstests durchgeführt, um zu prüfen ob die Funktionalität durch alle Layer des Frontends beziehungsweise Backends korrekt ist. Gegen Ende der Construction Phase werden Systemtests durchgeführt, um die Erfüllung der spezifizierten Anforderungen zu überprüfen. Zusätzlich wird während der Entwicklung ein Usability-Test erstellt, um die Benutzerfreundlichkeit zu verbessern.

Die einzelnen Module der Software werden mittels Unit Tests auf Korrektheit geprüft. Unit Tests laufen automatisiert ab und sind daher nach jeder Änderung an der Software schnell und einfach durchzuführen. Sie testen das Verhalten einzelner Methoden und damit den feinsten Detaillierungsgrad. In der Applikation wurde das Backend sehr gründlich mittels Unit Tests getestet, um die korrekte Funktionsweise der Controller und der API sicherzustellen. Dies ist wichtig, damit sich Fehler im System nicht fortpflanzen. Dazu wurde das Microsoft Framework MSTest eingesetzt. Alternativen mit mehr Funktionalität oder einer anderen Philosophie wie NUnit oder xUnit wurden kurz evaluiert. Jedoch sind für die geplanten Tests keine speziellen Anforderungen nötig. MSTest verwendet im Hintergrund jedoch auch xUnit und bietet mittlerweile eine signifikant bessere Performance [13].

Ausgewählte Frontend Elemente wurden mittels Jasmine ebenfalls mit Unit Tests geprüft. Für die Benutzeroberfläche machen Unit Tests jedoch keinen Sinn und stehen nicht in einem guten Aufwand- und Ertragsverhältnis. Das User Interface wurde jedoch vor allem mittels Integrationstests kontrolliert. Jasmine wurde gewählt, da Angular CLI dieses direkt in das Projekt integriert und somit eine vollständige Unterstützung bietet. Ausserdem ist es das präferierte Framework an den WED3 Vorlesungen der HSR.

Das Zusammenspiel der Module wird mit Hilfe von Integrationstests getestet. Backend sowie Frontend wurden während der Entwicklung bei jedem Review und vor jedem Pullrequest mittels Integrationstests überprüft. So wurden zum Beispiel HTTP-Anfragen direkt an die API gesendet, um zu prüfen ob die Antwort dem erwarteten Wert entspricht. Die Integrationstests wurden jeweils nicht dokumentiert da keine konkreten Testszenarien definiert wurden. Dies bietet den Vorteil, dass alle Entwickler eigene Szenarien ausdenken und somit die Testabdeckung besser ist als mit vordefinierten Szenarien. Fehlerfälle wurden im Review erfasst, behoben und erneut getestet.

Der Systemtest prüft im Gegensatz zum Integrationstest zusätzlich zu den funktionalen auch nichtfunktionale Anforderungen. Es handelt sich um einen Black-Box Test. Somit ist kein Wissen über das System nötig. Systemtests prüfen konkrete Anwendungsfälle der Applikation und decken Mängel in der Benutzerfreundlichkeit sowie in der Validation der Benutzereingabe auf. Die Szenarien sind im Kapitel 4 zu finden. Die Protokolle der Systemtests sind im Anhang abgelegt.

Ein Usability-Test wird eingesetzt, um die Benutzerfreundlichkeit des Systems zu testen. Im Gegensatz zum Systemtest wird der Test mit Benutzern erstellt, welche das System nicht kennen. Dadurch kann herausgefunden werden, wie diese auf das System ansprechen. Die Testfälle und deren Auswertung sind im Kapitel 4 einzusehen. Die Protokolle der Tests sind im Anhang zu finden.

4 Umsetzung und Testing

4.1 Systemübersicht

Ähnlich der Vorarbeit ist die Bereitstellung auf einem einzigen Server möglich. Jedoch gibt es von der Seite des Projekts aus keine Plattformeinschränkung mehr da MSSQL wegfällt. Es ist aber möglich, dass das Backbone noch MSSQL benötigt. Dieses könnte jedoch auch auf einem dedizierten Server laufen. Backend und Backbone sind in einer eigenen, unabhängigen Instanz laufend. Dies, da in IIS ein eigener Application Pool definiert wurde [14]. Das Backend greift über eine REST-Schnittstelle auf das Backbone zu und stellt seinerseits eine API zur Verfügung. Diese ist auch von ausserhalb über einen Reverse Proxy verfügbar. Das Frontend konsumiert diese Schnittstelle und stellt die Benutzeroberfläche zur Verfügung.

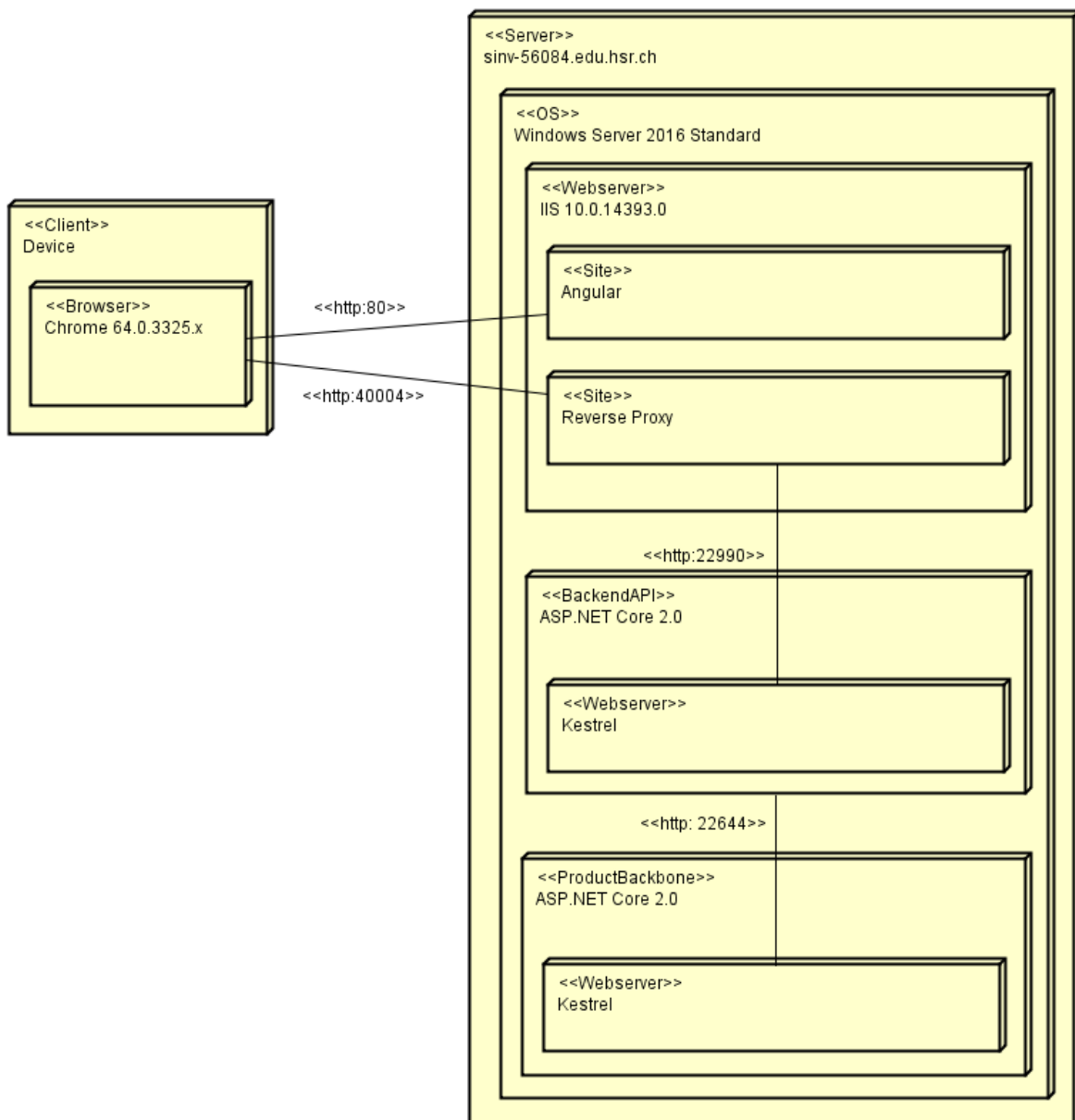


Abbildung 14 – Systemübersicht

4.2 Usability-Test

Bei der Durchführung eines Usability-Tests werden verschiedene Szenarien mit einer projektfremden Person durchgespielt. Die Schwierigkeiten und Unklarheiten werden dabei von der Begleitperson aufgenommen und festgehalten. Diese Ergebnisse geben Aufschluss über die Bedienbarkeit und Benutzerfreundlichkeit der Applikation.

Die Testperson ist dazu angehalten, während des Lösen der Aufgabe laut zu denken, sodass die Begleitperson erfährt wonach gerade gesucht wird.

4.2.1 Aufgaben

Für alle durchzuführenden Szenarios werden die Testdaten neu in das Backbone geladen. So wird sichergestellt, dass alle Tests gleich durchgeführt werden können. In den nicht-funktionalen Anforderungen ist die Rede von einer einstündigen Einführung. Diese wurde bewusst nicht durchgeführt.

Test Case 1: Neue Erweiterungen

Als Produktmanager möchtest Du ein neues Marketing Kit für die Maschinentypen *LegoRoboter* und *TechMachine* erfassen. Das Marketing Kit soll ein neues Feature *Waterproof* zur Verfügung stellen, welches die betreffenden Maschinen wasserdicht macht. Die von Dir geplanten Releasezeitpunkte sind der 16. Juni 2019 (*LegoRoboter*) und 20. August 2019 (*TechMachine*). Da mit der Entwicklung der Erweiterungen noch nicht begonnen wurde, kannst Du auch nicht genau sagen, wie viel die neuen Features am Ende kosten werden oder wie diese verbaut werden sollen.

Etwas Zeit ist nun vergangen. Wir schreiben den 5. Mai 2019. Die Erweiterung für den *LegoRoboter* ist von der Entwicklung fertiggestellt worden. Die Produkttests wurden auch erfolgreich absolviert. Du möchtest nun die Erweiterung freigeben und passt diese entsprechend an. Der Preis wird CHF 5'000 betragen. Erfasse dies im System und speichere ab.

Test Case 2: Kundenanfrage

Du hast im Moment einen Kunden am Telefon. Er hat Fragen zu einer Maschine. Leider weisst du nicht von welcher Maschine er redet. Finde mittels Rückfragen (Dein Betreuer spielt den Kunden) heraus, um welche Maschine von welchem Kunden es sich handelt und lies folgende Daten aus.

- Wie viele offene Service Cases beziehen sich auf diese Maschine?
- Wo steht die Maschine?
- Welche Teile sind auf dem *Logic Board* verbaut?

Test Case 3: Featureanfrage

Der Kunde MyTech GmbH möchte wissen, ob seine Maschine mit der Seriennummer *SN5* bereits die Kamerafunktion unterstützt, welche er bei einer anderen Maschine gesehen hat. Finde heraus, ob dieses Feature bereits verfügbar ist. Wenn nein, teile dem Kunden das Releasedatum mit.

Test Case 4: Featureinstallation

Du hast dem Kunden MyTech GmbH nun das Feature *Outdoor* für die Maschine mit der Seriennummer *SN6* verkauft. Erfasse nun einen Service Case mit entsprechender Aktivität. Der Service Case soll vom Servicetechniker *Ralf Hek* ausgeführt werden.

Logge Dich nun aus und als Servicetechniker wieder ein. Um das Feature erfolgreich zu verbauen, soll in der Baugruppe mit der Artikel-ID *Ald24* ein Mikrocontroller *Ald24212* hinzugefügt werden. Erfasse dies im System und schliesse die Aktivität sowie den Service Case ab.

4.2.2 Ergebnisse

Es wurden zwei Usability-Tests mit anwendungsfremden Personen durchgeführt. Das zu testende System war ein Build, in welchem bereits alle Basisfunktionalitäten umgesetzt wurden. In der nachfolgenden Tabelle sind die erkannten Probleme aufgeführt und beschrieben welche Massnahmen zur Behebung getroffen worden sind.

Test Case	Beobachtung	Massnahme
TC 1	Menüpunkte in Hauptnavigation unklar	Die Menüpunkte werden in der einstündigen Einführung (siehe NFA) behandelt
TC 1	„No data found“ war verwirrend und deutet auf einen Fehler hin, obwohl noch keine Daten angezeigt werden können	Umbenennung in „No data available“
TC 1	Verwirrung der Begriffe Marketing Kit und Marketing Kit Instance	Einfügen eines neuen Titels bei der Tabelle „Marketing kit instances“ zur Klarstellung des Begriffs
TC 1	Beim Freischalten einer Instanz ist nicht klar, dass das Releasedatum angepasst werden soll	Beim Setzen auf Released wird das Datum automatisch auf den heutigen Tag angepasst
TC 1	Beim Klick auf das Input Feld des Date Pickers wurde bereits der Dialog zur Auswahl eines Datums erwartet	Wird umgesetzt
TC 1	Bei Tabellen wurde mittels Doppelklick eine Auswahl erwartet	Doppelklick wird in allen Tabellen umgesetzt
TC 2	Bei der Karte sollte ein Scrollen nicht ohne weiteres möglich sein	Scrollen mit Mausrad deaktiviert, Zoom ist noch über Doppelklick und Zoom Panel möglich
TC 2	Maschinentyp sollte in der Maschinentabelle ersichtlich sein	Wird zusätzlich hinzugefügt
TC 2	Bezeichnung „BOM“ unklar	Umbenennung in „Bill of materials“
TC 3	Erwartete klickbare Badges für Marketing Kits oder informativere Tooltips	Tooltip wird mit mehr Informationen erweitert
TC 3	Scrollen für das Anzeigen der Marketing Kits in der Maschinentypenübersicht ist nicht ideal	Aufteilung der Ansicht auf 50/50 der Breite mit Bootstrap Breakpoints
TC 4	Bei der Auswahl der Servicetechniker sollte die Liste nach Nachname und nicht nach Vorname sortiert sein	Anzeigen und Sortieren der Liste als [Nachname], [Vorname]
TC 4	Description Feld der Aktivität ist zwingend hat aber keinen Stern (*)	Stern hinzufügen bei dieser Form
TC 4	Im Stepper sollte klar angezeigt werden, wo man sich im Moment befindet und was ausgewählt wurde	Auswahl in einzelnen Schritten anzeigen
TC 4	Bei der Spezifikation eines Marketing Kits sollte die Vorauswahl komplett markiert werden wenn das Inputfeld angewählt wird	Wird umgesetzt
Allgemein	Weiterleitungszeit wurde als zu lange empfunden	Die Weiterleitungszeit wird halbiert
Allgemein	Bei Statusänderungen könnte es besser sein, dies über einen Haken oder einen Button zu lösen als über eine Auswahlliste	Da nur bei zwei Status möglich, wird dies nicht umgesetzt

Tabelle 8 – Auswertung Usability-Test

Die Testpersonen fanden sich schnell zurecht und wurden in der kurzen Testzeit immer effizienter im Finden der gesuchten Informationen. Dies war laut ihnen auf das konsistente Design zurückzuführen. Sie experimentierten mit der Zeit auch mit den Filtern, um die korrekten Tabelleneinträge zu finden.

Weiterhin wurden während dem Usability-Test einige Unschönheiten und kleine Inkonsistenzen aufgedeckt. Diese wurden separat in einem GitHub Issue eingetragen und behoben.

4.3 Systemtest

Um die ganze Applikation im Sinne eines Blackbox-Tests zu kontrollieren, wurden die in der nachfolgenden Tabelle aufgeführten Testszenarien entworfen. Diese dienen dazu, die verschiedenen Features der Software zu testen und auch Fehlerfälle und Falscheingaben zu berücksichtigen. Der Systemtest wird am Ende der Entwicklung durchgeführt, um eventuelles Fehlverhalten aufzudecken. Die Protokolle sind im Anhang zu finden.

Die Tabellen sind nach den verschiedenen Rollen aufgeteilt, damit auch die Umsetzung der Autorisierung geprüft werden kann.

Testcase	Erwartetes Resultat
TC01: Kunde in Konfiguration auswählen	<ul style="list-style-type: none"> - Karte mit Standorten wird angezeigt - Maschinen sind gefiltert - Service Cases sind gefiltert
TC02: Kunde in Konfiguration auswählen, Zoom in Auswahlkarte ändern, Navigation auf Maschinentypen, Navigation auf Dashboard	<ul style="list-style-type: none"> - Karte mit Standorten wird angezeigt - Maschinen sind gefiltert - Service Cases sind gefiltert - Zoom Level wurde zurückgesetzt
TC03: Auswahl Maschine SN5 nach der Filterung nach Kunde <i>MyGmbH</i>	<ul style="list-style-type: none"> - Maschinen Metadaten werden angezeigt - Karte mit Standort wird angezeigt - Bei Hover über das Kamera Feature wird das Release Datum angezeigt - Service Cases der Maschine werden angezeigt - Stückliste wird angezeigt
TC04: Nach TC03 wird ein neuer Service Case hinzugefügt	<ul style="list-style-type: none"> - Submit Button ist nicht aktiv solange die Felder nicht ausgefüllt sind und weniger als drei Zeichen enthalten - Nach Submit wird in die Service Case Detailansicht gewechselt - Es ist nicht möglich eine Aktivität zu erfassen
TC05: Nach TC04 wird der Service Case mit dem entsprechenden Button editiert	<ul style="list-style-type: none"> - Die Felder für den Servicetechniker und das Erstellungsdatum bleiben deaktiviert - Der Save Button ist deaktiviert solange nichts geändert wurde oder die Felder nicht wenigstens drei Zeichen enthalten - Nach Save wird auf das Dashboard weitergeleitet
TC06: Wechsel auf Maschinentypenansicht	<ul style="list-style-type: none"> - Es werden alle Maschinentypen und die Grafik mit den Top 3 Maschinen angezeigt
TC07: Nach TC06 wird ein Maschinentyp ausgewählt	<ul style="list-style-type: none"> - Die Marketing Kits werden dargestellt
TC08: Nach TC07 wird ein Marketing Kit ausgewählt	<ul style="list-style-type: none"> - Das Marketing Kit mit allen Instanzen wird dargestellt
TC09: Nach TC08 wird eine Instanz ausgewählt	<ul style="list-style-type: none"> - Die Detailansicht wird dargestellt - Es können keine Änderungen durchgeführt werden

TC010: Ohne Sortierung eine Maschine auswählen Neuer Service Case erstellen	- Fehler: Es muss ein Kunde ausgewählt werden
TC11: Klick auf Logout	- Der Benutzer ist ausgeloggt - Das Token ist aus dem lokalen Speicher entfernt

Tabelle 9 – Testsznarien Verkäufer

Testcase	Erwartetes Resultat
TC12: Kunde in Konfiguration auswählen	- Karte mit Standorten wird angezeigt - Maschinen sind gefiltert - Service Cases sind gefiltert
TC13: Kunde in Konfiguration auswählen, Zoom in Auswahlkarte ändern, Navigation Maschinentypen, danach Dashboard	- Karte mit Standorten wird angezeigt - Maschinen sind gefiltert - Service Cases sind gefiltert - Zoom Level wurde zurückgesetzt
TC14: Auswahl Maschine SN5 nach Filterung nach Kunde „MyGmbH“	- Maschinen Metadaten werden angezeigt - Karte mit Standort wird angezeigt - Bei Hover über das Kamera Feature wird das Release Datum angezeigt - Service Cases der Maschine werden angezeigt - Stückliste wird angezeigt - Es ist nicht möglich einen Service Case zu erfassen
TC15: Wechsel auf Marketing Kit Ansicht	- Alle Marketing Kits werden angezeigt - Es ist möglich ein Marketing Kit zu erfassen
TC16: Erstellen eines neuen Marketing Kits	- Create Button ist deaktiviert solange nicht alle Felder mindestens drei Zeichen beinhalten - Nach Create wird auf die Detailansicht gewechselt
TC17: In der Marketing Kit Ansicht wird das Kamera Feature gewählt	- Die Details werden angezeigt - Es ist möglich eine neue Instanz zu erstellen
TC18: Nach TC17 wird eine neue Instanz erstellt, dabei wird die Legomaschine ausgewählt	- Create ist deaktiviert solange nicht alle benötigten Felder ausgefüllt sind - Nach Create wird ein Fehler angezeigt
TC19: Wie TC18 aber diesmal mit BigMachine	- Create ist deaktiviert solange nicht alle benötigten Felder ausgefüllt sind - Weiterleitung auf Übersicht
TC20: Nach TC19 Instanz auswählen	- Alle Details werden angezeigt
TC21: nach TC20 wird die Instanz editiert	- Save ist deaktiviert solange nicht alle benötigten Felder ausgefüllt sind - Es können nur positive Preis ausgewählt werden - Es können nur PDF hinterlegt werden - Nach Save wird weitergeleitet auf Übersicht
TC22: Wechsel auf Maschinentypenansicht	- Es werden alle Maschinentypen und die Grafik mit den Top 3 Maschinen angezeigt
TC22: Nach TC21 wird ein Maschinentyp ausgewählt	- Die Marketing Ktis werden dargestellt
TC23: Nach TC22 wird ein Marketing Kit ausgewählt	- Das Marketing Kit mit allen Instanzen wird dargestellt
TC24: Nach TC23 wird eine Instanz ausgewählt	- Die Detailansicht wird dargestellt
TC25: Klick auf Logout	- Der Benutzer ist ausgeloggt - Das Token ist aus dem lokalen Speicher entfernt

Tabelle 10 – Testsznarien Produktmanager

Testcase	Erwartetes Resultat
TC26: Kunde in Konfiguration auswählen	<ul style="list-style-type: none"> - Karte mit Standorten wird angezeigt - Maschinen sind gefiltert - Service Cases sind gefiltert
TC27: Kunde in Konfiguration auswählen, Zoom in Auswahlkarte ändern, Navigation auf Maschinentypen, Navigation auf Dashboard	<ul style="list-style-type: none"> - Karte mit Standorten wird angezeigt - Maschinen sind gefiltert - Service Cases sind gefiltert - Zoom Level wurde zurückgesetzt
TC28: Auswahl Maschine SN5 nach Filterung nach Kunde „MyGmbH“	<ul style="list-style-type: none"> - Maschinen Metadaten werden angezeigt - Karte mit Standort wird angezeigt - Bei Hover über das Kamera Feature wird das Release Datum angezeigt - Service Cases der Maschine werden angezeigt - Stückliste wird angezeigt
TC29: Nach TC28 wird ein neuer Service Case hinzugefügt	<ul style="list-style-type: none"> - Submit Button ist nicht aktiv solange die Felder nicht ausgefüllt sind und weniger als 3 Zeichen enthalten - Nach Submit wird in die Service Case Detailansicht gewechselt - Es ist nicht möglich eine Aktivität zu erfassen
TC30: Nach TC29 wird der Service Case mit dem entsprechenden Button editiert	<ul style="list-style-type: none"> - Die Felder für den Servicetechniker und das Erstellungsdatum bleiben deaktiviert - Der Save Button ist deaktiviert solange nichts geändert wurde oder die Felder nicht wenigstens drei Zeichen enthalten - Nach Save wird auf das Dashboard weitergeleitet
TC31: Auswahl eines geschlossenen Service Cases	<ul style="list-style-type: none"> - Es können keine Aktivitäten hinzugefügt werden
TC32: Auswahl eines offenen Service Cases	<ul style="list-style-type: none"> - Es können Aktivitäten hinzugefügt werden
TC33: Im Service Case „Desc0027“ wird eine Aktivität erstellt, Marketing Kit „TRob+“	<ul style="list-style-type: none"> - Marketing Kits „TRob+“ und „Outdoor“ sind verfügbar - Create ist deaktiviert solange nicht alle Felder mindestens drei Zeichen enthalten - Nach dem Create wird in die Service Case Übersicht weitergeleitet
TC34: Nach TC33 wird der Service Case geschlossen	<ul style="list-style-type: none"> - Fehler
TC35: Nach TC33 wird die Aktivität bearbeitet und geschlossen	<ul style="list-style-type: none"> - Nur der Status und die Beschreibung können angepasst werden - Nach der Änderung Weiterleitung auf Service Case Übersicht
TC36: Nach TC35 wird der Service Case geschlossen	<ul style="list-style-type: none"> - Nach der Schliessung wird das Marketing Kit „TRob+“ in den Maschinendetails als verbaut geführt
TC37: Versuch, geschlossene Service Cases und Aktivitäten zu bearbeiten	<ul style="list-style-type: none"> - Nicht möglich
TC38: Hinzufügen von Ald221 in Ald42 bei Service Case „Desc1577“	<ul style="list-style-type: none"> - Auswahl wird in Titel des Steppers angezeigt - Summary wird angezeigt - Nach Save wurde die BOM angepasst

TC39: Entfernen des hinzugefügten Teils	<ul style="list-style-type: none"> - Auswahl wird in Titel des Steppers angezeigt - Summary wird angezeigt - Nach Save wurde die BOM angepasst
TC40: Ersatz des Teils Ald421	<ul style="list-style-type: none"> - Auswahl wird in Titel des Steppers angezeigt - Summary wird angezeigt - Nach Save wurde die BOM angepasst
TC41: Wartung des Teils Ald421	<ul style="list-style-type: none"> - Auswahl wird in Titel des Steppers angezeigt - Summary wird angezeigt - Nach Save wurde die BOM angepasst
TC42: Ohne Sortierung eine Maschine auswählen Neuer Service Case erstellen	<ul style="list-style-type: none"> - Fehler: Es muss ein Kunde ausgewählt werden
TC43: Klick auf Logout	<ul style="list-style-type: none"> - Der Benutzer ist ausgeloggt - Das Token ist aus dem lokalen Speicher entfernt

Tabella 11 – Testszzenarien Servicetechniker

Testcase	Erwartetes Resultat
TC44: Login mit „TEST“	<ul style="list-style-type: none"> - Fehler
TC45: Login mit „MyTech GmbH“	<ul style="list-style-type: none"> - Maschinen und Service Cases des Kunden werden angezeigt
TC46: Auswahl Maschine SN5	<ul style="list-style-type: none"> - Maschinen Metadaten werden angezeigt - Karte mit Standort wird angezeigt - Bei Hover über das Kamera Feature wird das Release Datum angezeigt - Service Cases der Maschine werden angezeigt - Stückliste wird angezeigt
TC47: Nach TC46 wird ein neuer Service Case hinzugefügt	<ul style="list-style-type: none"> - Submit Button ist nicht aktiv solange die Felder nicht ausgefüllt sind und weniger als 3 Zeichen enthalten - Customer, Support ist vorausgewählt - Nach Submit wird in die Service Case Detailansicht gewechselt - Es ist nicht möglich eine Aktivität zu erfassen - Es ist nicht möglich den Service Case zu bearbeiten
TC48: Klick auf Logout	<ul style="list-style-type: none"> - Der Benutzer ist ausgeloggt - Das Token ist aus dem lokalen Speicher entfernt

Tabella 12 – Testszzenarien Kunde

4.4 Erweiterung der Applikation

Im Backbone werden Testdaten mittels API Controller zur Verfügung gestellt. Um Erweiterungen im Backbone vorzunehmen, können die Models im Assembly *DomainModel.API* bearbeitet oder neu erstellt werden. Im Assembly *Data.Mocking* werden die Testdaten erstellt und die Datenbank initialisiert. Das Datenbankschema wird in der Klasse *DataInitializer* definiert. Danach können die entsprechenden Controller in den *API.** Assemblies angepasst werden.

Im Backend sind die zu verwendenden Models im gleichnamigen Assembly definiert. Es muss darauf geachtet werden, dass jeweils ein Data Transfer Object und ein Entity Objekt erstellt, beziehungsweise angepasst wird. Die Konvertierung zwischen den Objekten ist in der Erweiterungsklasse *ConverterExtensions* realisiert. Im *DataAccess* Assembly ist die Verbindung zum Backbone entwickelt worden. Dort können Routen zum Backbone angepasst oder hinzugefügt werden. Ebenfalls ist dort die Verwendung des Dateisystems geregelt. Über dem Datenzugriffslayer ist der Business Logic Layer angesiedelt. Dort werden die Daten aus dem Backbone für das Backend aufbereitet. Für jeden

Controller existiert mindestens eine Logikklasse. Zuerst befinden sich die Controller, in welchen Endpunkte bearbeitet oder neu hinzugefügt werden können.

Beim Frontend können Komponenten, Services oder Models analog bestehender Implementationen erstellt werden. Beim Erstellen oder Ändern ist jeweils darauf zu achten, dass die zu exportierenden Elemente in den *index.ts* Dateien hinterlegt wurden. Die benötigten Importe sind jeweils in der Klasse *[NAME].module.ts* zu spezifizieren. Allgemein sollten Komponenten so generisch wie möglich erstellt werden um die Wiederverwendbarkeit zu gewährleisten.

Sollten Textbausteine im Frontend angepasst werden müssen, können diese in der Datei *string-handler.ts* an einer zentralen Stelle verändert werden.

4.5 Code Metriken

4.5.1 Backend

Statische Code Metriken machen es Entwicklern möglich, problematische Stellen im Code zu erkennen. Dies kann vor allem bei einer grossen Codebasis sinnvoll sein. Zur Erstellung der Code Metriken im Backend wurde eine Testversion von NDepend verwendet. Da die Analyse der Code Metriken erst am Ende des Projekts wichtig ist, spielt es keine Rolle, dass nur eine zeitlich begrenzte Testversion verwendet wurde. Ähnliche Tools unterstützen noch immer kein ASP.NET Core.

Metrik	Wert
Physische Codezeilen	9'627 LoC
Effektive Codezeilen	5'718 LoC
Logische Codezeilen	2'786 LoC
Maximale zyklomatische Komplexität	14 (ActivitiesLogic.cs)
Durchschnittliche zyklomatische Komplexität	1.35
Anzahl Unit Tests	198
Testabdeckung	68 %

Tabella 13 – Code Metriken Backend

Die physischen Codezeilen sagen aus, wie viele Zeilen physikalisch im Backendcode vorhanden sind. Sie leiten sich aus der Anzahl Zeilenumbrüchen in den Source Code Dateien ab. Die effektiven Codezeilen rechnen davon noch alle Kommentarzeilen, leeren Zeilen und solche mit nur einer öffnenden oder schliessenden Klammer weg. Die logischen Codezeilen wurden mit NDepend ausgewertet. Diese werden anhand der PDB Dateien von Visual Studio berechnet. Die Anzahl logischer Codezeilen einer Methode ist gleich der Sequenzpunkte für diese Methode in den PDB Dateien. Ein Sequenzpunkt markiert einen Punkt im IL Code, welcher auf eine spezifische Lokation im Source Code verweist. Diese werden für das Debugging verwendet. Interfaces, abstrakte Klassen, abstrakte Methoden und Enumerationen haben daher keine logischen Codezeilen. Auch Namespaces, Typen, Fields und Methodendeklarationen werden nicht gewertet, da sie keinen korrespondierenden Sequenzpunkt haben. Diese Art der Berechnung hat einige Vorteile. Der Coding Style hat somit keinen Einfluss auf die Berechnung der logischen Codezeilen. Auch ist diese Metrik unabhängig von der benutzten Sprache. Werte aus Assemblies in einer anderen Programmiersprache können verglichen oder summiert werden [15].

Die zyklomatische Komplexität sagt aus, wie viele mögliche Verzweigungen und Pfade eine Methode hat. Eine hohe Zahl gibt einen Hinweis darauf, dass die Methode sehr schwer zu verstehen und schlecht wartbar ist [16]. Die Methode *CreateModificationByActivity* sticht dabei besonders negativ heraus. Mit dem Wert 14 liegt sie jedoch noch knapp im wartbaren Bereich. Der nächsthöchste Wert beträgt 10. Der Grund für so eine hohe Komplexität liegt an der Prüfung vieler Vorbedingungen und vor allem an dem Switch-Case. Dieser erzeugt trotz seiner guten Lesbarkeit schnell eine hohe zyklomatische

Komplexität. Der Durchschnitt von 1.35 ist wiederum sehr gut und weist auf eine gut verständliche und wartbare Codebasis hin.

Die Unit Tests wurden vor allem für die Controller und die Validierung geschrieben. Auch ausgewählte Logikmethoden wurden getestet. Die Testabdeckung ist jedoch über das ganze Projekt berechnet worden. So kann der tiefe Wert erklärt werden.

4.5.2 Frontend

Wie bereits in der Studienarbeit wurde für die Codemetriken in der Bachelorarbeit das Plugin *Statistics* für WebStorm verwendet. Jedoch ist es noch immer nicht möglich, für ein Angular Projekt die zyklomatische Komplexität zu ermitteln. Aus diesem Grund wurden die Anzahl Codezeilen und Dateien pro Dateitypen ermittelt.

Dateityp	Codezeilen	Anzahl Dateien
Typescript	4'851	196
HTML	982	52
SCSS	299	15

Tabella 14 – Code Metriken Frontend

Bei den Codezeilen handelt es sich um die effektiven Codezeilen ohne Kommentare und Zeilenumbrüche. Die Anzahl Typescript Dateien ist im Vergleich zu den anderen Dateitypen hoch. Dies ist mit den Barrels zu begründen [17]. Diese Dateien erlauben es, einen Export über mehrere Module zu machen und vereinfachen dadurch den Import in andere Module.

Im Gegensatz zur Studienarbeit funktioniert diese Applikation mit Lazy-Loading. Das heisst, dass bei einer ersten Anfrage das Mainmodul sowie Polyfills geladen werden. Die anderen Module werden erst in Form von Chunks nachgeladen, wenn diese benötigt werden. Bei einem produktiven Build mit AOT Compilation werden folgende Dateigrößen erreicht [18]. Die Grösse des Main Modules lässt sich damit erklären, dass das ngx-charts Framework von Swimlane nur bei einem Import der kompletten Bibliothek funktioniert. Dieser Import trägt rund 1.6 MB zur Modulgrösse bei, dadurch muss bei Performance-Optimierungen Rücksprache mit den Entwicklern gehalten oder ein alternatives Framework gesucht werden.

Modul	Grösse
Main	2.7 MB
Polyfills	60 kB
Styles	350 kB
Runtime	2 kB
Chunks	Max. 20 kB

Tabella 15 – Dateigrößen Frontend

Angular verwendet standardmässig keine Komprimierung mehr beim Build, da dies durch die Server und Browser automatisch erledigt wird, falls sie dies unterstützen. Dadurch reduzieren sich die Dateigrößen nochmals enorm.





 styles.c30fdb225d6c605fe7bd.css	200 OK	stylesheet	login Parser	49.0 KB 343 KB
 runtime.502a8161465641319e4f.js	200 OK	script	login Parser	2.2 KB 2.0 KB
 polyfills.67bd8e3fb59dc15e9dae.js	200 OK	script	login Parser	19.5 KB 58.0 KB
 main.1968bd06e97f7d991fb6.js	200 OK	script	login Parser	590 KB 2.6 MB

Abbildung 15 – Komprimiertes Laden

4.6 Showcases

Die folgenden Showcases sollen die Möglichkeiten des Prototyps anhand spezifischer Anwendungsfälle aufzeigen. Es wird nicht der gesamte Funktionsumfang demonstriert sondern lediglich die wichtigsten Funktionen.

4.6.1 Showcase „Häufig auftretendes Problem“

Der Produktmanager bemerkt, dass bei den Maschinen des Typs *LegoRoboter* häufig die Vorderachse kaputt geht. Die Reparatur dieses Maschinenteils ist mit einem sehr hohen Aufwand verbunden und kostet die Firma viel Geld. Daher beschliesst er die Entwicklung damit zu beauftragen, ein Upgrade dieser Achse zu entwerfen, welche stabiler ist und den enormen Belastungen Stand hält. Da der Maschinentyp *TechMachine* eine sehr ähnliche Achse enthält, soll dafür ebenfalls ein gleiches Upgrade entworfen werden. Aufgrund der Dringlichkeit soll das Upgrade für den *LegoRoboter* priorisiert werden. Die Entwicklungszeit wird ca. 4 Monate respektive 8 Monate für die *TechMachine* betragen.

Nach der erfolgreichen Entwicklung für den *LegoRoboter*, wird diese Instanz für den Markt freigegeben. Der Preis wird festgelegt und die Installationsanleitung hochgeladen.

4.6.2 Showcase „Upgrade“

Der Verkäufer der Firma bemerkt beim Kunden *RoboTech AG* ein grosses Potenzial für den Verkauf von stabileren Achsen aus einem besseren Kunststoff. Er ruft den Kunden an und führt ein Verkaufsgespräch.

Der Kunde ist nach dem Gespräch interessiert. Jedoch will er sich die betroffenen Maschinen und deren Vergangenheit nochmals im Kundenportal ansehen. Er stellt fest, dass das betreffende Teil schon mehrmals ausgewechselt werden musste. Um Ausfallzeiten in Zukunft zu minimieren, bestellt er das Upgrade.

Der Verkäufer erstellt einen entsprechenden Service Case pro Maschine.

Der Servicetechniker, welcher mit dem Upgrade der Maschinen beauftragt wurde sieht sich die Service Cases an und plant seine Tätigkeiten. Dabei wirft er auch einen Blick in die Einbauanleitung des neuen Marketing Kits. Vor Ort baut er die Teile gemäss der Anleitung ein und schliesst die Aktivitäten und Service Cases ab.

Der Kunde sieht nun die abgeschlossenen Service Cases und dass die Marketing Kits in den Maschinen vorhanden sind.

4.6.3 Showcase „Einsatz des Servicetechnikers“

Die Firma *ValSys* hat Probleme mit einer ihrer Maschinen. Die Maschine habe sich in den vergangenen Tagen immer mal wieder von selbst ausgeschaltet. Nun laufe sie gar nicht mehr an. Der Servicetechniker bereitet sich auf den Einsatz vor. Zuerst schaut er die Vergangenheit der Maschine an, findet dabei aber nichts Auffälliges. Die Problembeschreibung lässt einen Fehler in der Steuerung vermuten. Er nimmt daher die entsprechenden Ersatzteile zum Kunden mit.

Durch einen ausgefallenen Ventilator am Gehäuse der Steuerung ist die Maschine überhitzt. Als es einmal zu heiss wurde ging auch noch eine Batterie kaputt. Dies wäre mit automatischer Übertragung von Sensordaten nicht passiert. Der Servicetechniker tauscht die defekten Teile aus und schliesst den Service Case ab.

Er bemerkt weiterhin, dass ein Wireless-Modul an der Maschine installiert wurde. Diese Anpassung ist nicht in der Stückliste ersichtlich. Daher erfasst er die bemerkte Abweichung in einem separaten Service Case.

5 Ergebnis und Ausblick

5.1 Zielerreichung

Das Erreichen des Projektziels kann an den funktionalen und nicht-funktionalen Anforderungen gemessen werden. Die als Bronze und Silber eingestufteten Use Cases konnten alle umgesetzt werden. Danach wurde der Fokus aufgrund des Interesses des Betreuers auf die Marketing Kits gelegt. Dadurch konnten in diesem Bereich alle Gold Use Cases umgesetzt werden.

Use Case	Ausbaustufe	Umgesetzt
UC 100 – Create marketing kit	Bronze	Ja
Extension UC 100a – Set price independent per machine type	Gold	Ja
UC 110 – Show marketing kit per machine type	Bronze	Ja
UC 120 – Show marketing kit per machine	Silber	Ja
UC 130 – Update marketing kit instance	Gold	Ja
Extension UC 130a – Update state independent per machine type	Gold	Ja
UC 140 – Show customer feedback and suggestions	Gold	Nein
Extension UC 140a – Filter feedback and suggestions	Gold	Nein
UC 150 – Update customer feedback and suggestions for salesman	Gold	Nein
UC 200 – Show machines by customer	Bronze	Ja
UC 210 – Show machine properties	Bronze	Ja
Extension UC 210a – Show service cases of machine	Bronze	Ja
Extension UC 210b – Show history of activities	Silber	Ja
Extension UC 210c – Show maintenance plan	Gold	Nein
Extension UC 210d – Show bill of materials	Silber	Ja
UC 300 – Show maintenance plan by customer	Gold	Nein
UC 310 – Show marketing kits by customer	Gold	Ja
UC 320 – Show customer locations	Bronze	Ja
UC 330 – Show customer contact by location	Bronze	Ja
UC 340 – Show service cases by customer	Bronze	Ja
UC 400 – Create service case	Bronze	Ja
UC 410 – Show service case with all activities	Bronze	Ja
UC 420 – Update service case	Silber	Ja
UC 430 – Create activity from service case	Bronze	Ja
Extension UC 430a – Add part	Bronze	Ja
Extension UC 430b – Remove part	Bronze	Ja
Extension UC 430c – Replace part	Bronze	Ja
Extension UC 430d – Maintenance	Silber	Ja
UC 440 – Read activity from service case view	Bronze	Ja
UC 450 – Update activity from service case view	Bronze	Ja
UC 500 – Login per role	Silber	Ja
UC 510 – Logout	Silber	Ja

Tabelle 16 – Erfüllung der funktionalen Anforderungen

Die User Stories, aus welchen sich die Use Cases ableiteten, konnten bis auf die User Story 07 mit den Wartungsintervallen sinngemäss umgesetzt werden. Dies lag daran, dass der Fokus sich während der Arbeit auf die Marketing Kits verschob und daher die Anforderungen für die Wartung in den Hintergrund rückten.

Die definierten Showcases können erfolgreich durchgespielt werden. Anhand dieser wird die Funktionalität der Applikation aufgezeigt.

Ein wichtiger Punkt ist auch die Erfüllung der nicht-funktionalen Anforderungen. Zu diesem Zeitpunkt werden nur die Kriterien des Prototyps bewertet, da kein produktiver Einsatz geplant ist.

NFA	Erfüllt	Bemerkung
Kompatibilität	Ja	
Erweiterbarkeit	Ja	Nicht messbar, jedoch subjektiv erfüllt
Änderungssensivität	Ja	
Datenintegrität	k/A	Durch die fehlende Anbindung der Umsysteme im Backbone ist diese Anforderung nicht testbar
Bedienbarkeit	Ja	Konnte im Usability-Test beobachtet werden
Sicherheit	Ja	

Tabelle 17 – Erfüllung der nicht-funktionalen Anforderungen

Die Punkte der Erweiterbarkeit und der Bedienbarkeit sind nicht objektiv zu testen. Bei der Erweiterbarkeit ist jedoch zu sagen, dass neue Module während der Entwicklung laufend hinzukamen. Dies ist ein Indikator, dass das Hinzufügen neuer Module auch nach dem Abschluss der Arbeit möglich ist. Die Bedienbarkeit ist aufgrund fehlender Testpersonen, welche diese Software am Ende wirklich einsetzen nicht möglich. Jedoch hat der Usability-Test gezeigt, dass sich Benutzer auch ganz ohne Einführung gut in der Applikation zurechtfinden können.

Die Anforderung der Datenintegrität wurde in der Annahme erstellt, dass die Umsysteme bis zum Ende der Arbeit angebunden werden würden. Durch das Fehlen von wirklichen Datenquellen im Hintergrund, konnte keine Aussage getroffen werden.

5.2 Übergang zu produktivem System

Um die Applikation in ein produktives Umfeld zu verschieben, müssen einige Schritte durchgeführt werden. Das wichtigste ist die Anbindung der produktiven Daten im Backbone. Dazu wird das *Assembly Data.Mocking* vollständig aus dem Backbone entfernt. Ausserdem müssen die entsprechenden Repositories und Adapter für die Umsysteme implementiert werden. Diese Daten können danach aufbereitet an die entsprechenden Endpunkte weitergeleitet werden.

Dabei kann es sein, dass gewisse Funktionen wie die Änderungen an einer Stückliste in einem Umsystem geschehen müssen. In diesem Fall kann die entsprechende Logik aus dem Backend ersetzt werden. Diese wurden dort zwecks Präsentation des Prototyps platziert.

Im Frontend und Backend ist es ratsam, eine Authentifizierung mittels Benutzername und Passwort einzuführen, um die Zuteilung der Rollen zu implementieren. Dazu ist eine Einführung einer entsprechenden Datenhaltung notwendig.

Weiterhin müssen die nicht-funktionalen Anforderungen aus dem zweiten Kapitel überprüft und gegebenenfalls angepasst werden. Dabei sind auch die Anforderungen für das produktive Umfeld zu beachten.

5.3 Ausblick und Erweiterungen

In den folgenden Abschnitten wird dargelegt, welche Möglichkeiten zur Weiterentwicklung und Verbesserung der Applikation bestehen. Diese werden als grobe Konzepte vorgestellt, wobei die Details genauer zu analysieren und evaluieren sind.

Azure als Cloud-Lösung wurde etwas genauer unter die Lupe genommen, da sich durch eine Applikation in der Cloud viele weitere Möglichkeiten für Erweiterungen eröffnen.

5.3.1 Azure Cloud

Über eine Cloud-Lösung ist es möglich, die Applikation hochverfügbar anzubieten. Doch nicht nur dies ist ein Vorteil. Man könnte sich vorstellen, dass in Zukunft Sensordaten direkt an eine Queue in der Cloud gesendet werden. Auch muss man sich nicht mehr um die ganze Serverinfrastruktur kümmern da der Cloud-Anbieter diese zur Verfügung stellt. Ausserdem bietet die Cloud eine ganze Palette an Lösungen für typische IIoT³ Anwendungsfälle. Dazu gehört auch eine Einführung von Scannern, mit welchen die einzelnen Teile gescannt und identifiziert werden können. Dies spart Zeit und verhindert Eingabefehler.

Durch eine Verschiebung in die Cloud können zum Beispiel unter Verwendung von Azure Notification Hub, Pushnachrichten an alle möglichen Plattformen gesendet werden. Dies ist zum Beispiel denkbar wenn ein Kunde einen Service Case erfasst und der zuständige Verkäufer das sofort merken soll.

Durch Azure Time Series Insights können Sensordaten sehr einfach visualisiert und ausgewertet werden. Fehler und Unstimmigkeiten können mit dem korrekten Einsatz schnell ausgemacht und behoben werden [19]. Im Folgenden ist ein Screenshot der Demo angehängt, um in etwa die Darstellung der Daten zu zeigen.

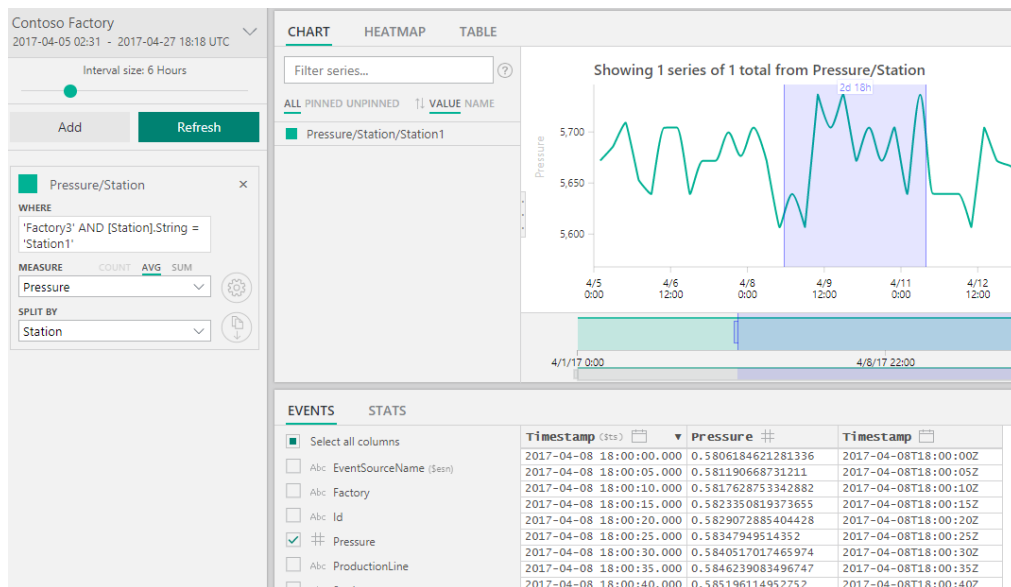


Abbildung 16 – Screenshot Time Series Demo

Es wurde Azure als Cloud-Anbieter evaluiert, da dieser über eine Testversion mit vollem Funktionsumfang verfügt. Um die Applikation auf ihre Cloud-Fähigkeit zu testen wurde sie in einem App Service auf Azure hochgeladen und gestartet. Dies hat ohne weitere Anpassungen funktioniert. Die Applikation ist also bereit für die Cloud. Eine Anleitung für das Deployment sowie die Adresse zu der Cloud-Applikation finden sich im Anhang.

³ Industrial Internet of Things

Ausserdem wurde eine ungefähre Kostenschätzung der Applikation inklusive Erweiterungen für einen einzelnen kleinen Betrieb erstellt.

Microsoft Azure Estimate				
Ihre Schätzung				
Service type	Custom name	Region	Description	Estimated Cost
App Service		West Europe	Standard Tarif, 1 S2 (2 Kern(e), 3.5 GB RAM, 50 GB Speicher) x 31 Days; Windows Betriebssystem	CHF134.38
Time Series Insights		West Europe	S1-Tarif, 2 Einheit(en)	CHF270.93
Notification Hubs		West Europe	Basic-Tarif, 2 Million(en) zusätzliche Pushvorgänge	CHF9.03
Service Bus		West Europe	Standard-Tarif, 20, 1.000 Brokerverbindung(en), 0 Hybrid Connect-Listener + 0 Überschreitung(en) pro GB, 0 Relaystunde(n), 0 Relaynachricht(en)	CHF13.92
Monthly Total				CHF428.26
Annual Total				CHF5'139.13
Disclaimer				
All prices shown are in Swiss Franc. (CHF). This is a summary estimate, not a quote. For up to date pricing information please visit this estimate was created at 6/9/2018 5:01:29 PM UTC.				

Abbildung 17 – Kostenrechnung Azure

Mittels einem kleinen App Service für die Applikation und Time Series Insights, mit bis zu zwei Millionen Nachrichten pro Tag, sowie der Möglichkeit Benachrichtigungen und den Service Bus zu nutzen, bezahlt man monatlich unter CHF 500.00. Aussen vor gelassen wurden natürlich die Entwicklungskosten und die Supportkosten seitens Microsoft.

5.3.2 Sicherheitserweiterungen

Im Thema Sicherheit können auch einige Verbesserungen und Erweiterungen vorgenommen werden. Zum Beispiel das Einführen von Benutzerkonten für die eindeutige Identifizierung von einzelnen Mitarbeitern birgt grosses Potenzial. So könnte zum Beispiel dem Servicetechniker auf dem Dashboard seine tägliche Route angezeigt werden. Auch für die Nachverfolgbarkeit und Benutzerfreundlichkeit ist es wichtig, dass das System genau weiss, wer die Applikation bedient. Dafür ist jedoch eine eigene Datenhaltung zu entwerfen, um Benutzernamen und Passwörter zu speichern. Alternativ kann zum Beispiel auch das Active Directory von der jeweiligen Firma benutzt werden. Technisch ist dies mit Auth0 möglich [20]. Dies hat den Vorteil, dass keine eigenen Accounts für die Nutzung der Applikation angelegt werden müssen.

Weiterhin muss zu einem Zeitpunkt auch das Backbone abgesichert werden. Dieses ist im Moment noch frei zugänglich. Sobald die Umsysteme an das Backbone angeschlossen wurden darf die API auch nicht mehr ungeschützt im Netz stehen. Dabei könnte auch dieselbe Methode wie im Backend verwendet werden. Aufgrund der Daten der Umsysteme ist jedoch die Rechtevergabe eher komplex.

5.3.3 Offline Verfügbarkeit

Eine grosse Herausforderung wird die Verfügbarkeit der Services, wenn der Servicetechniker periodisch Netzwerkunterbrüche erlebt. Dies kann bei der Arbeit in Produktionshallen öfters der Fall sein.

Ein Teil des Problems wird gelöst, wenn die wichtigsten Ansichten als Cache bereits im Voraus zu einem Fall, eventuell anhand von GPS Informationen, heruntergeladen werden können. So ist sichergestellt, dass der Servicetechniker die benötigten Informationen abrufen kann. Jedoch erfasst dieser meist auch noch Aktivitäten oder Modifikationen. Das Erstellen von diesen erfordert direkten Kontakt mit dem Backend.

Bei einer solchen Art von Problem bietet sich eine Cachelösung an. Diese muss bei der Wiederherstellung der Verbindung die Daten synchronisieren und bei einer Abweichung in der Datenbank entscheiden, welche Informationen behalten werden sollen.

6 Literaturverzeichnis

- [1] Akveo, «Nebular,» 2017. [Online]. Available: <https://akveo.github.io/nebular>. [Zugriff am 24 April 2018].
- [2] Akveo, «Admin template based on Angular 5+, Bootstrap 4 and Nebular,» 23 April 2018. [Online]. Available: <https://github.com/akveo/ngx-admin>. [Zugriff am 24 April 2018].
- [3] PrimeTek, «The Most Complete User Interface Suite,» [Online]. Available: <https://primefaces.org/primeng>. [Zugriff am 24 April 2018].
- [4] PrimeTek, «Layout Licenses,» 2018. [Online]. Available: <https://www.primefaces.org/layouts/licenses/>. [Zugriff am 24 April 2018].
- [5] keentthemes, «Metronic - Responsive Admin Dashboard Template,» 2 April 2018. [Online]. Available: https://themeforest.net/item/metronic-responsive-admin-dashboard-template/4021469?s_rank=1. [Zugriff am 24 April 2018].
- [6] keentthemes, «Metronic Documentation,» 1 April 2018. [Online]. Available: <https://keentthemes.com/metronic/documentation.html>. [Zugriff am 24 April 2018].
- [7] P. Leach, M. Mealling und R. Salz, «A Universally Unique IDentifier (UUID) URN Namespace,» Juli 2005. [Online]. Available: <https://www.rfc-editor.org/rfc/rfc4122.txt>. [Zugriff am 24 April 2018].
- [8] J. Hilton, «Secure your ASP.NET Core 2.0 API,» 11 Oktober 2017. [Online]. Available: <https://jonhilton.net/2017/10/11/secure-your-asp.net-core-2.0-api-part-1---issuing-a-jwt/>. [Zugriff am 9 April 2018].
- [9] R. Anderson, T. Dykstra und S. Smith, «Filters in ASP.NET Core,» 10 April 2018. [Online]. Available: <https://docs.microsoft.com/en-us/aspnet/core/mvc/controllers/filters?view=aspnetcore-2.1>. [Zugriff am 24 April 2018].
- [10] M. David, «LiteDB - A .NET NoSQL Document Store,» 9 September 2017. [Online]. Available: <https://github.com/mbdavid/LiteDB/wiki>. [Zugriff am 28 Mai 2018].
- [11] C. Bennage, «API Design,» Microsoft, 1 Dezember 2018. [Online]. Available: <https://docs.microsoft.com/en-us/azure/architecture/best-practices/api-design>. [Zugriff am 21 März 2018].
- [12] R. Anderson, «Policy-based authorization in ASP.NET Core,» 21 November 2017. [Online]. Available: <https://docs.microsoft.com/en-us/aspnet/core/security/authorization/policies?view=aspnetcore-2.0>. [Zugriff am 29 Mai 2018].
- [13] V. S. B. Team, «Test Experience Improvements,» 16 November 2017. [Online]. Available: <https://blogs.msdn.microsoft.com/visualstudio/2017/11/16/test-experience-improvements/>. [Zugriff am 15 Mai 2018].

- [14] R. Rohith, «Quick Reference : IIS Application Pool,» Microsoft, 8 Oktober 2017. [Online]. Available: <https://blogs.msdn.microsoft.com/rohithrajan/2017/10/08/quick-reference-iis-application-pool/>. [Zugriff am 24 April 2018].
- [15] «Code Metrics Definitions,» 2018. [Online]. Available: <https://www.ndepend.com/docs/code-metrics#NbLinesOfCode>. [Zugriff am 12 Juni 2018].
- [16] T. J. McGabe, «A Complexity Measure,» *IEEE Transactions on Software Engineering*, pp. 308-320, Dezember 1976.
- [17] A. Fâciu, «Barrel files: to use or not to use ?,» 4 März 2018. [Online]. Available: <https://medium.com/@adrianfaciu/barrel-files-to-use-or-not-to-use-75521cd18e65>. [Zugriff am 12 Juni 2018].
- [18] Google, «Angular Glossary,» 2018. [Online]. Available: <https://angular.io/guide/glossary#ahead-of-time-aot-compilation>. [Zugriff am 12 Juni 2018].
- [19] Microsoft, «Time Series Insights,» 2018. [Online]. Available: <https://azure.microsoft.com/en-us/services/time-series-insights/>. [Zugriff am 9 Juni 2018].
- [20] Auth0 Inc., «Authenticate Angular 2+ with Active Directory using Auth0,» 2018. [Online]. Available: <https://auth0.com/authenticate/angular2/active-directory/>. [Zugriff am 9 Juni 2018].
- [21] Microsoft, «Hey students—start building the future with Azure,» 2018. [Online]. Available: <https://azure.microsoft.com/en-us/free/students/>. [Zugriff am 9 Juni 2018].
- [22] C. Spielmann, «Virtual Infrastructure Guide,» Rapperswil, 2017.

7 Abbildungsverzeichnis

Abbildung 1 – Umfeld und Projektabgrenzung (Bildherkunft: HSR/Microsoft).....	2
Abbildung 2 – Backbone Architektur.....	3
Abbildung 3 – Prototyp Nebular.....	12
Abbildung 4 – Prototyp PrimeNG (Free)	13
Abbildung 5 – Big Picture Backend API	14
Abbildung 6 – Architektur Backend.....	15
Abbildung 7 – Domänenmodell.....	16
Abbildung 8 – Beispielhafter JWT Workflow [8]	17
Abbildung 9 – Request Filter Pipeline	18
Abbildung 10 – Backbone Architektur.....	19
Abbildung 11 – Funktionsbeispiel Smartphone	22
Abbildung 12 – Konzeptionsvorschlag Betreuer	22
Abbildung 13 – Konzept Featuremanagement	23
Abbildung 14 – Systemübersicht.....	26
Abbildung 15 – Komprimiertes Laden.....	34
Abbildung 16 – Screenshot Time Series Demo	38
Abbildung 17 – Kostenrechnung Azure	39
Abbildung 18 – Projektplanung	45
Abbildung 19 – Zeitauswertung nach Aktivitäten	48
Abbildung 20 – Backbone Architektur.....	49
Abbildung 21 – Domänenmodell Querformat	50
Abbildung 22 – Use Case Diagramm Produktmanagement	51
Abbildung 23 – Use Case Diagramm Maschinendetails	51
Abbildung 24 – Use Case Diagramm Kundenbeziehungen	52
Abbildung 25 – Use Case Diagramm Service Case	52
Abbildung 26 – Use Case Diagramm Sessionmanagement.....	53
Abbildung 27 – IIS Tabelle	61
Abbildung 28 – Frontend Application Pool	62
Abbildung 29 – ASP.NET Core Application Pool	62
Abbildung 30 – IIS Sites	62
Abbildung 31 – Frontend Ordner Share	63
Abbildung 32 – Frontend (environment.prod.ts).....	63
Abbildung 33 – Backend (BackboneRoutes.cs)	63
Abbildung 34 – Frontend Build.....	64
Abbildung 35 – Backend lokal	65
Abbildung 36 – Frontend lokal	65
Abbildung 37 – Projekt veröffentlichen	66
Abbildung 38 – App Service Ziel auswählen.....	66
Abbildung 39 – App Service erstellen.....	67
Abbildung 40 – App ausliefern	67

8 Tabellenverzeichnis

Tabelle 1 – Übersicht der funktionalen Anforderungen und deren Ausbaustufe.....	6
Tabelle 2 – Betriebsumgebungen.....	8
Tabelle 3 – Auswertung Komponenten Nebular	11
Tabelle 4 – Auswertung Komponenten PrimeNG	11
Tabelle 5 – Auswertung Komponenten Metronic	12
Tabelle 6 – API Endpoints.....	21
Tabelle 7 – Beschreibung Klasse Function.....	23
Tabelle 8 – Auswertung Usability-Test.....	28
Tabelle 9 – Testsznarien Verkäufer	30
Tabelle 10 – Testsznarien Produktmanager	30
Tabelle 11 – Testsznarien Servicetechniker	32
Tabelle 12 – Testsznarien Kunde.....	32
Tabelle 13 – Code Metriken Backend.....	33
Tabelle 14 – Code Metriken Frontend	34
Tabelle 15 – Dateigrößen Frontend	34
Tabelle 16 – Erfüllung der funktionalen Anforderungen	36
Tabelle 17 – Erfüllung der nicht-funktionalen Anforderungen	37
Tabelle 18 – Beschreibung der Meilensteine	44
Tabelle 19 – Risikomatrix; Inception Phase.....	46
Tabelle 20 – Bewertung Risiken; Inception Phase.....	46
Tabelle 21 – Risikomatrix; Ende der Elaboration	47
Tabelle 22 – Bewertung Risiken; Ende der Elaboration	47
Tabelle 23 – Zeitauswertung nach Projektmitglied.....	48
Tabelle 24 – Usability-Test Protokoll Lukas Koller, Test Case 1	54
Tabelle 25 – Usability-Test Protokoll Lukas Koller, Test Case 2	55
Tabelle 26 – Usability-Test Protokoll Lukas Koller, Test Case 3	55
Tabelle 27 – Usability-Test Protokoll Lukas Koller, Test Case 4	56
Tabelle 28 – Usability-Test Protokoll Raphael Jöhl, Test Case 1	57
Tabelle 29 – Usability-Test Protokoll Raphael Jöhl, Test Case 2	58
Tabelle 30 – Usability-Test Protokoll Raphael Jöhl, Test Case 3	58
Tabelle 31 – Usability-Test Protokoll Raphael Jöhl, Test Case 4	58
Tabelle 32 – Usability-Test Protokoll Raphael Jöhl, Allgemeine Bemerkungen.....	59
Tabelle 33 – Systemtest Protokoll.....	60
Tabelle 34 – Wichtigste Serveradressen bei Abgabzeitpunkt	72

Anhang

Anhang A – Projektorganisation

Meilensteine

Meilenstein	Beschreibung	Termin
MS1: Projektplan	<ul style="list-style-type: none"> - Risiken sind analysiert und bewertet - Die wichtigsten Funktionen der Applikation sind bekannt - Die Infrastruktur ist bekannt - Es existiert eine Planung der Iterationen - Showcases wurden definiert 	25.02.2018
MS2: GUI Design, UCs, Domain Model	<ul style="list-style-type: none"> - Für das GUI ist ein grobes Wireframe erstellt - Die Use Cases sind briefly definiert. Falls nötig casually - Die Datenstruktur ist bekannt und dokumentiert inklusive Domänenmodell - Nicht-funktionale Anforderungen wurden definiert und vom Betreuer abgenommen - Die zur Verfügung gestellte Schnittstelle wurde analysiert 	11.03.2018
MS3: End of Elaboration	<ul style="list-style-type: none"> - Die Infrastruktur ist getestet und für die Anwendung geeignet - Cockpitlösungen wurden evaluiert und das UI Design dementsprechend angepasst - CI-Lösung, Git, Webstorm und Visual Studio sind für alle Mitglieder konfiguriert - Die Projektplanung wurde nach neusten Erkenntnissen überarbeitet - Es existiert ein Architektur Diagramm - Die Arbeitspakete für die Construction Phase sind aufgeschlüsselt - Eingesetzte Technologien wurden reevaluiert - Die API Schnittstelle ist definiert 	25.03.2018
MS4: Basisfunktionen	<ul style="list-style-type: none"> - Alle Bronze Use Cases wurden gemäss Definition abgeschlossen - Es wurde ein Usability-Test durchgeführt 	22.04.2018
MS5: Feature Freeze	<ul style="list-style-type: none"> - Es werden ab diesem Zeitpunkt keine neuen Features mehr hinzugefügt - Es wurden alle Tests gemäss Testspezifikation durchgeführt 	20.05.2018
MS6: End of Construction	<ul style="list-style-type: none"> - Die Software ist gemäss den Anforderungen fertiggestellt - Alle Tests laufen erfolgreich - Code Freeze - Das UI ist ansprechend gestaltet 	03.06.2018
MS7: Abgabe	<ul style="list-style-type: none"> - Alle abzugebenden Dokumente sind vorhanden - Showcases sind erfolgreich durchführbar 	15.06.2018

Tabelle 18 – Beschreibung der Meilensteine

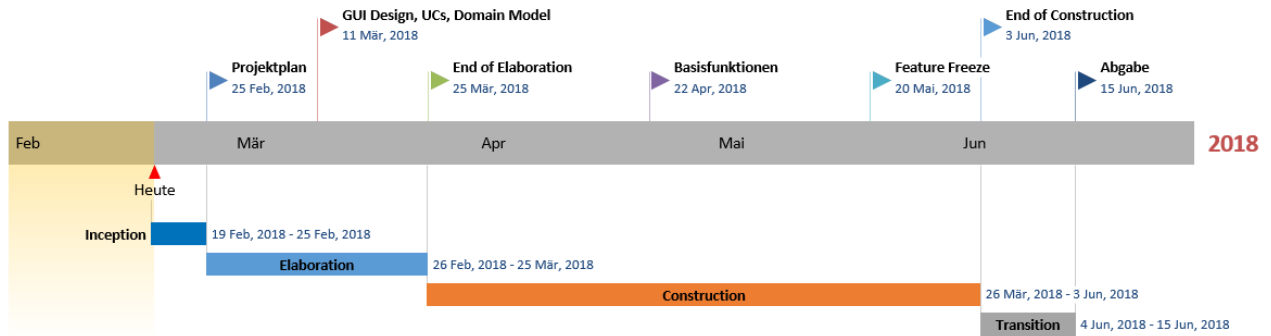


Abbildung 18 – Projektplanung

Risikoanalyse

Nr.	Titel	Beschreibung	Vorbeugung	Verhalten beim Eintreten
R1	Einrichtung des CI Server gestaltet sich schwierig	Konfiguration von Jenkins als CI Server für das ProductBackbone auf dem HSR Git ⁴ gestaltet sich schwierig.	Bei HSR Developer nach den Erfahrungen mit HSR Git und Jenkins erkunden und diesen Lösungsansatz verwenden.	Auf CI Server verzichten und Tests lokal ausführen. Via Skript das lauffähige, getestete ProductBackbone auf dem HSR Server bereitstellen.
R2	Einrichtung des Webservers	Konfiguration des IIS Servers für das Ausliefern mehrerer Webseiten (Staging/Production) sowie SSL-Zertifikat gestaltet sich schwierig.	Proof of Concept erstellen für Zertifikat und mehrere Deployments	Projektstruktur anpassen oder alternativer Service installieren. Auf SSL-Zertifikat verzichten.
R3	Entwicklung des Product Backbones	Entwicklung des Product Backbones durch die HSR dauert länger als eingeplant und ist während der BA nicht verfügbar.	Kontinuierliches Statusupdate und Kommunikation mit den HSR Developer.	Daten auf dem Product Backbone mocken und auf die Anbindung an Drittsysteme vorerst verzichten.
R4	Authentifizierung und Autorisierung im Backend	Implementierung der Authentifizierung und Autorisierung für das Rollenkonzept gestaltet sich komplexer als gedacht im Backend.	Während der Elaboration mehrere Frameworks und Methoden evaluieren.	Rücksprache mit erfahrenen Developers halten. Nur Basisautorisierung implementieren.
R5	Authentifizierung und Autorisierung im Frontend	Implementierung der Authentifizierung und Autorisierung für das Rollenkonzept gestaltet sich komplexer als gedacht im Frontend.	Während der Elaboration mehrere Frameworks und Methoden evaluieren.	Analog der Studienarbeit, das Rollenkonzept durch die Navigation ändern für den Prototyp.

⁴ <https://git.hsr.ch/>

R6	Unterschiedliche Definitionen von Artefakten und Prozessen	Verschiedene Ansichten der zu verwendeten Artefakten sowie des Prozesses, welcher modelliert werden soll zwischen dem Betreuer und dem BA Team.	Showcases möglichst früh definieren. Definitionen und das Datenmodell vom Betreuer absegnen lassen.	Meeting zur Aufklärung der Grundlagen einberufen.
-----------	--	---	---	---

Tabelle 19 – Risikomatrix; Inception Phase

Die Bewertung der Risiken nach dem Meilenstein Projektplan sieht wie folgt aus.

Nr.	Max. Schaden [h]	Eintrittswahrscheinlichkeit	Gewichteter Schaden [h]
R1	6	80%	4.8
R2	14	50%	7.0
R3	10	50%	5.0
R4	16	60%	9.6
R5	16	60%	9.6
R6	10	30%	3.0

Tabelle 20 – Bewertung Risiken; Inception Phase

Während der Elaboration Phase wurden einige der Risiken adressiert und konnten teilweise eliminiert werden.

- R1** Das Deployment mit Jenkins funktionierte wie gewünscht und konnte planmässig konfiguriert werden.
- R2** Grundsätzliche Möglichkeit für SSL ist vorhanden. Jedoch ist noch nicht klar, ob dies im Scope berücksichtigt wird. Die Auslieferung mehrerer Seiten wurde eingerichtet und funktioniert einwandfrei.
- R3** Es wurde ein Backbone zur Verfügung gestellt, welches alle Schichten bereits implementiert. Die Daten können gemockt werden. Aus diesem Grund ist der Schaden geringer bei Eintritt des Risikos. Jedoch wird vermutet, dass die Umsysteme am Ende des Projekts nicht komplett eingebunden sein werden.
- R4** Auf der Backendseite konnte ein JWT gelöst und verschickt werden. Die Autorisierung funktionierte auch. Ein Restrisiko ist die Identifizierung von Kunden in der Applikation.
- R5** Im Frontend wurde das Authentifizierungs- und Sicherheitspaket von Nebular verwendet, welche bereits die Verarbeitung des JWT übernimmt und vereinfacht. Wie im Backend bleibt das Identifizieren der Kunden sowie das evaluieren des Rollenkonzepts ein Restrisiko.
- R6** Während der Evaluationsphase wurde in verschiedenen Meetings die Anforderungen abgeklärt. Bisher ist das Risiko noch nicht eingetreten.

Folgende Risiken wurden am Ende der Elaboration neu hinzugefügt.

Nr.	Titel	Beschreibung	Vorbeugung	Verhalten beim Eintreten
R7	Marketing Kits	Die Abbildung und Erstellung von Marketing Kits bereitet Schwierigkeiten	Ausführliche Konzeption, frühes Implementieren um allfällige Probleme schnell zu erkennen.	Anpassung des Scopes in Rücksprache mit dem Betreuer.

Tabelle 21 – Risikomatrix; Ende der Elaboration

Die Bewertung der Risiken nach dem Ende der Elaboration sieht wie folgt aus.

Nr.	Max. Schaden [h]	Eintrittswahrscheinlichkeit	Gewichteter Schaden [h]
R3	3	80%	2.4
R4	6	20%	1.2
R5	6	20%	1.2
R6	7	30%	2.1
R7	15	30%	4.5

Tabelle 22 – Bewertung Risiken; Ende der Elaboration

Das Backbone wurde wie erwartet nicht an die Umsysteme angeschlossen. Dies hat jedoch auch Vorteile für die Entwicklung, da die Testszenarien und Showcases ohnehin mit Testdaten durchgeführt werden müssen. Die Authentifizierung und Autorisierung in Backend und Frontend funktionierten wie geplant und konnten auch so eingesetzt werden. Durch die kontinuierliche Präsentation des Fortschritts konnten unterschiedliche Visionen und Vorstellungen schnell miteinander besprochen werden. Das Feature der Marketing Kits kann wie gewünscht umgesetzt werden. Dieses Risiko ist somit nicht aufgetreten.

Zeitauswertung

Alle auszuführenden Arbeiten wurden als Arbeitspakete im Projektmanagementtool Redmine erfasst. Die aufgewendete Arbeitszeit ist dort ebenfalls eingetragen worden. Diese Aufwendungen wurden exportiert und damit konnten folgende Auswertungen erstellt werden.

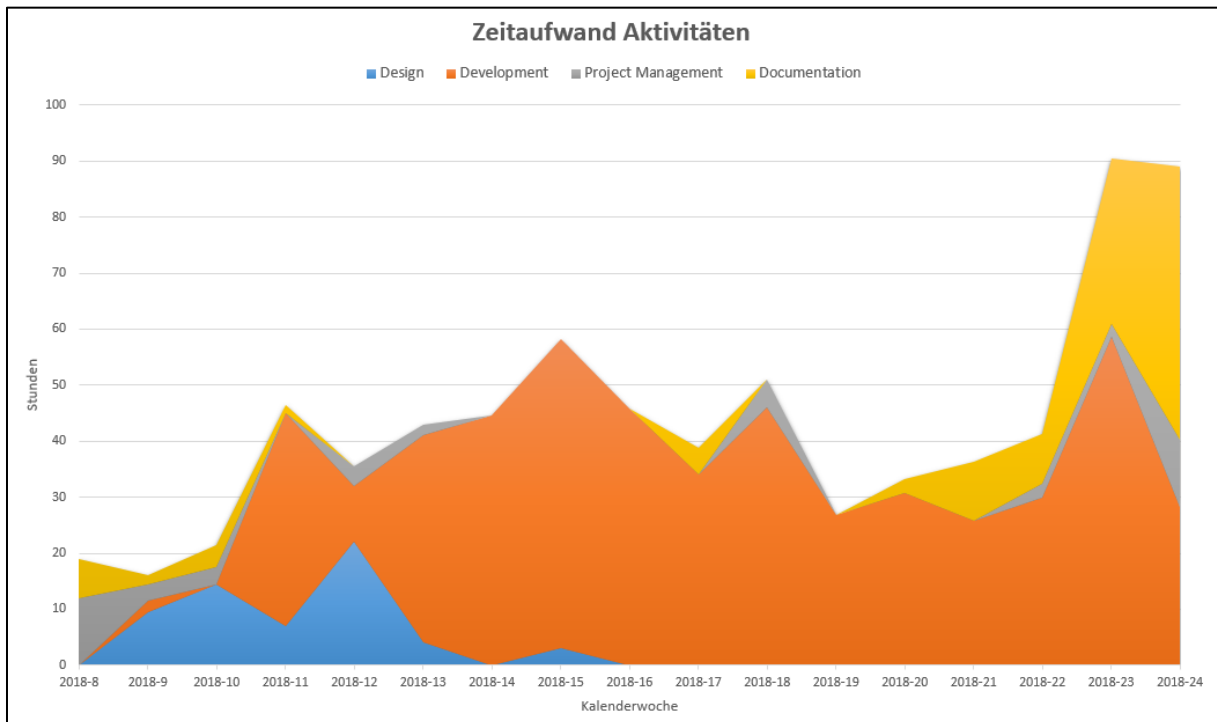


Abbildung 19 – Zeitauswertung nach Aktivitäten

In diesem gestapelten Flächendiagramm sind die Zeitaufwände pro Aktivität sowie der gesamte Zeitaufwand pro Woche ersichtlich. Gut zu sehen ist ein eher schleppender Start. Dieser war auch schon bei der Studienarbeit zu erkennen. Auch die Belastung durch mehrere Abgaben gegen Ende des regulären Semesters ist durch den Einbruch in der Kalenderwoche 19 ersichtlich. In den letzten beiden Wochen konnte dieser jedoch wieder kompensiert werden. Zu sehen ist auch, dass schon sehr früh mit der Entwicklung angefangen wurde. So wurde sichergestellt, dass das Design und die Architektur praktikabel sind.

User	Februar	März	April	Mai	Juni	Total
Luca Salzani	15 h	71.5 h	100.5 h	85.25 h	97.75 h	370 h
Fabian Gübeli	13 h	82 h	98.25 h	86.25 h	87.25 h	366.75 h
Total	28 h	153.5 h	198.75 h	171.5 h	185 h	736.75 h

Tabelle 23 – Zeitauswertung nach Projektmitglied

Es wird deutlich, dass jedes Projektmitglied etwa gleich viel zu dem Projekt beigetragen hat. Auch konnte die Zeitvorgabe von 360 h pro Person erreicht werden.

Anhang B – Diagramme

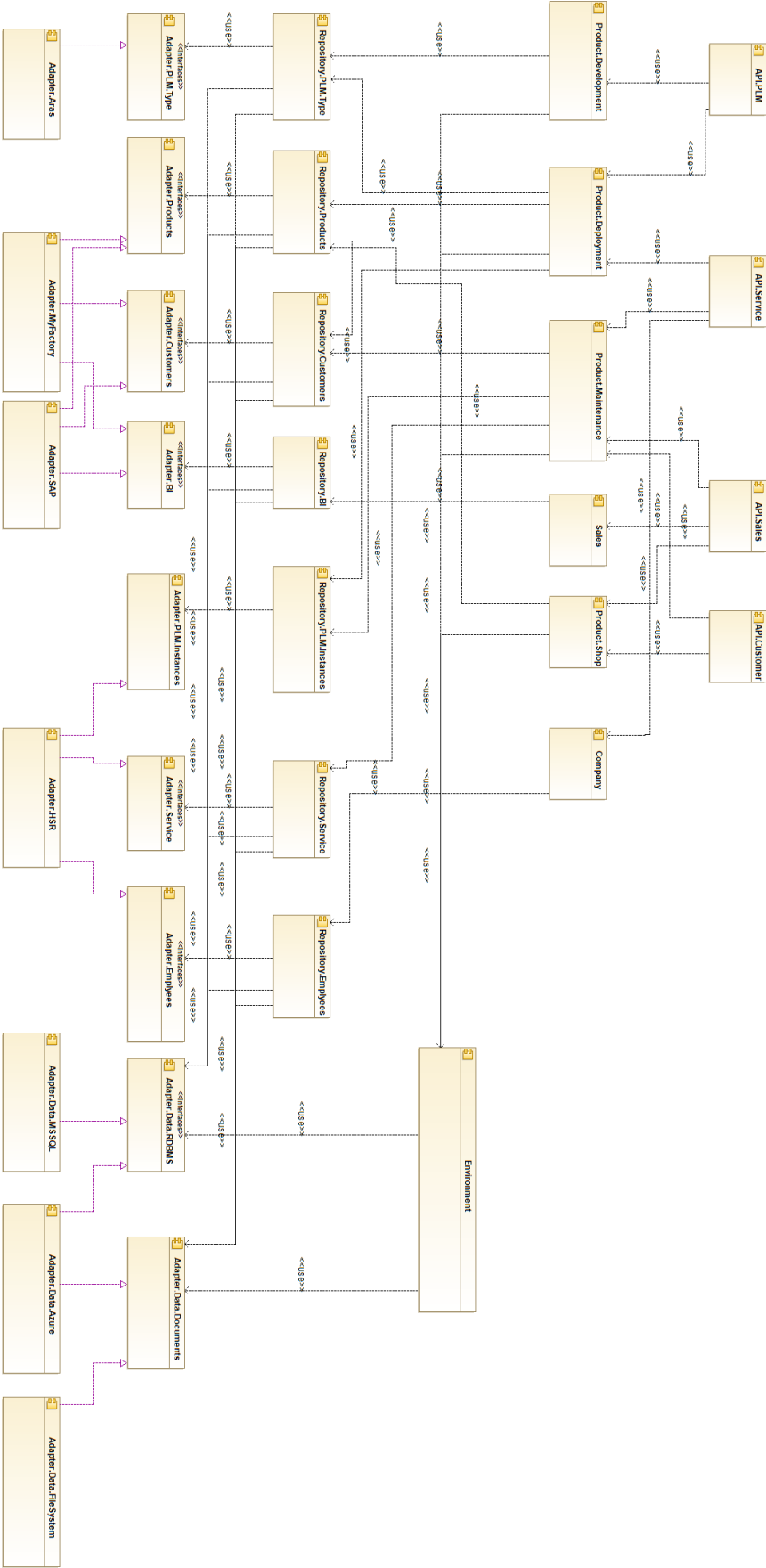


Abbildung 20 – Backbone Architektur

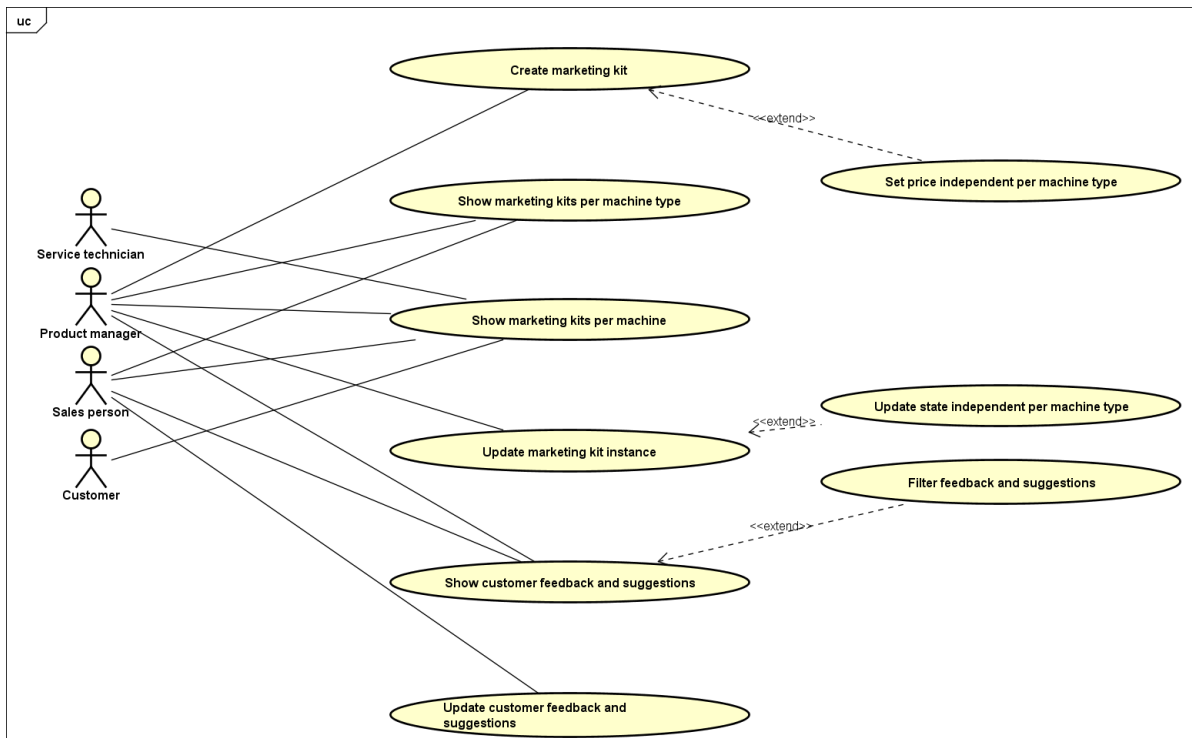


Abbildung 22 – Use Case Diagram Produktmanagement

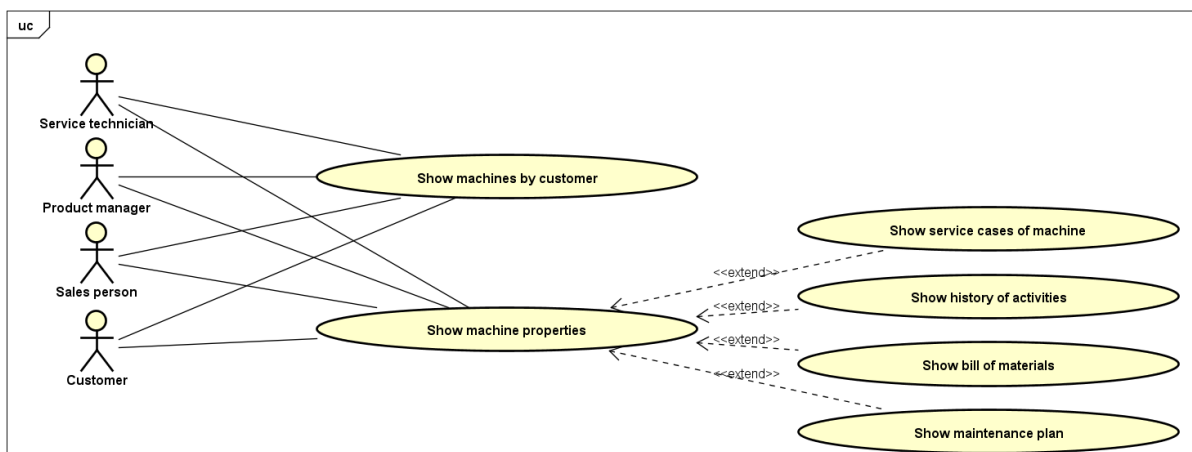


Abbildung 23 – Use Case Diagram Maschinendetails

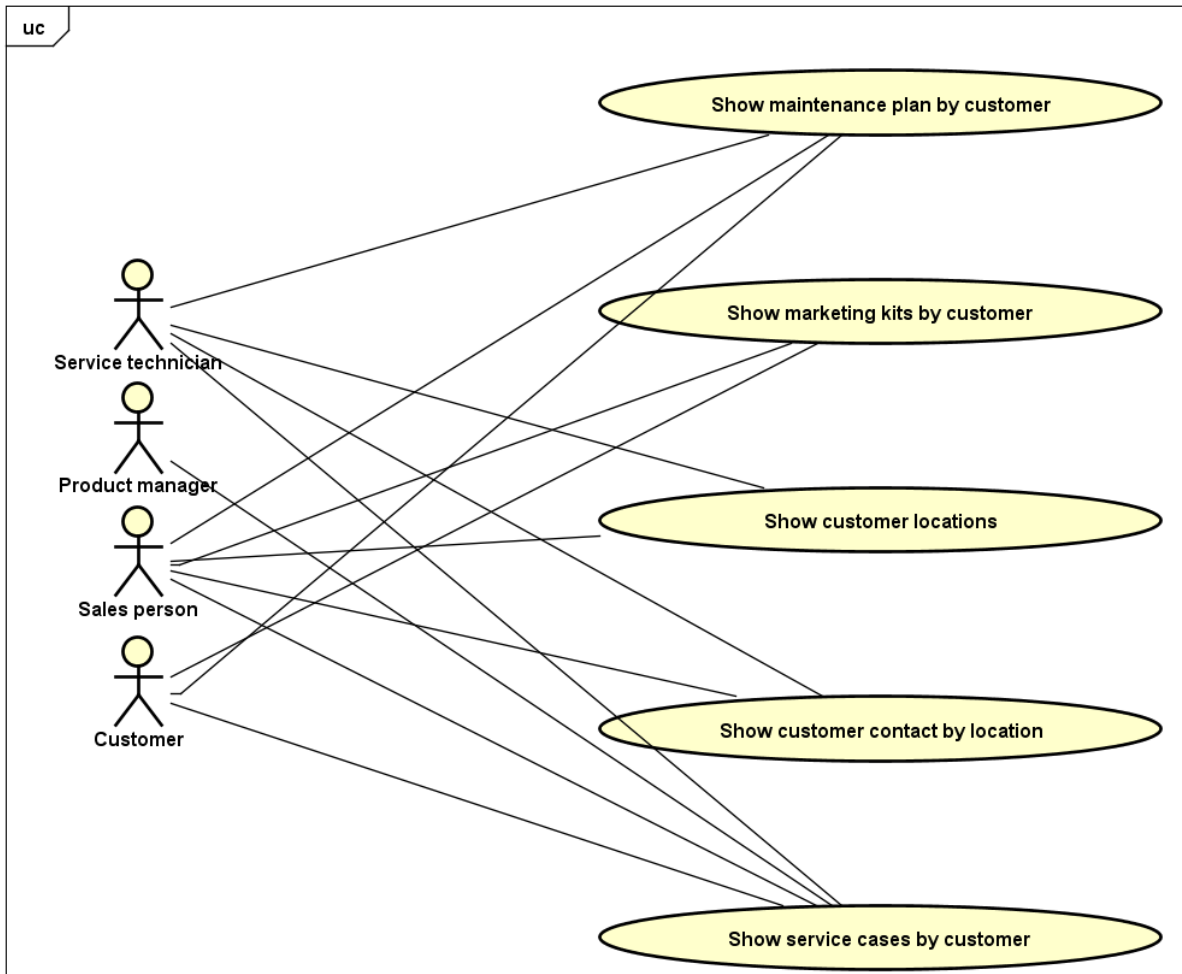


Abbildung 24 – Use Case Diagramm Kundenbeziehungen

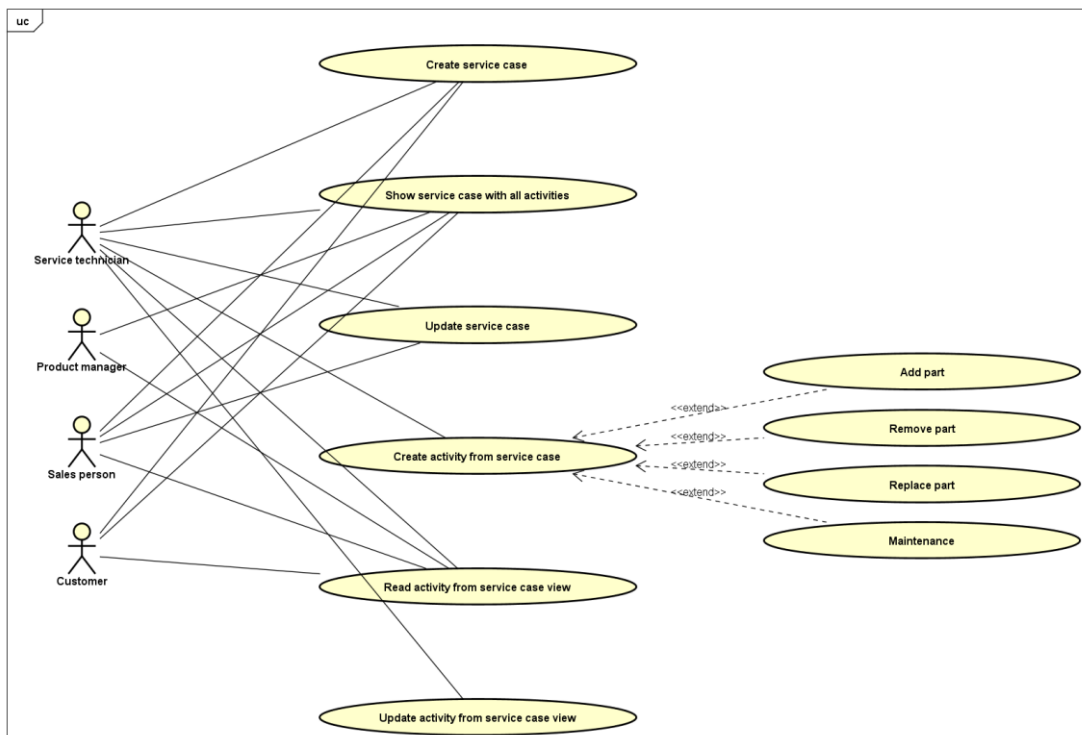


Abbildung 25 – Use Case Diagramm Service Case

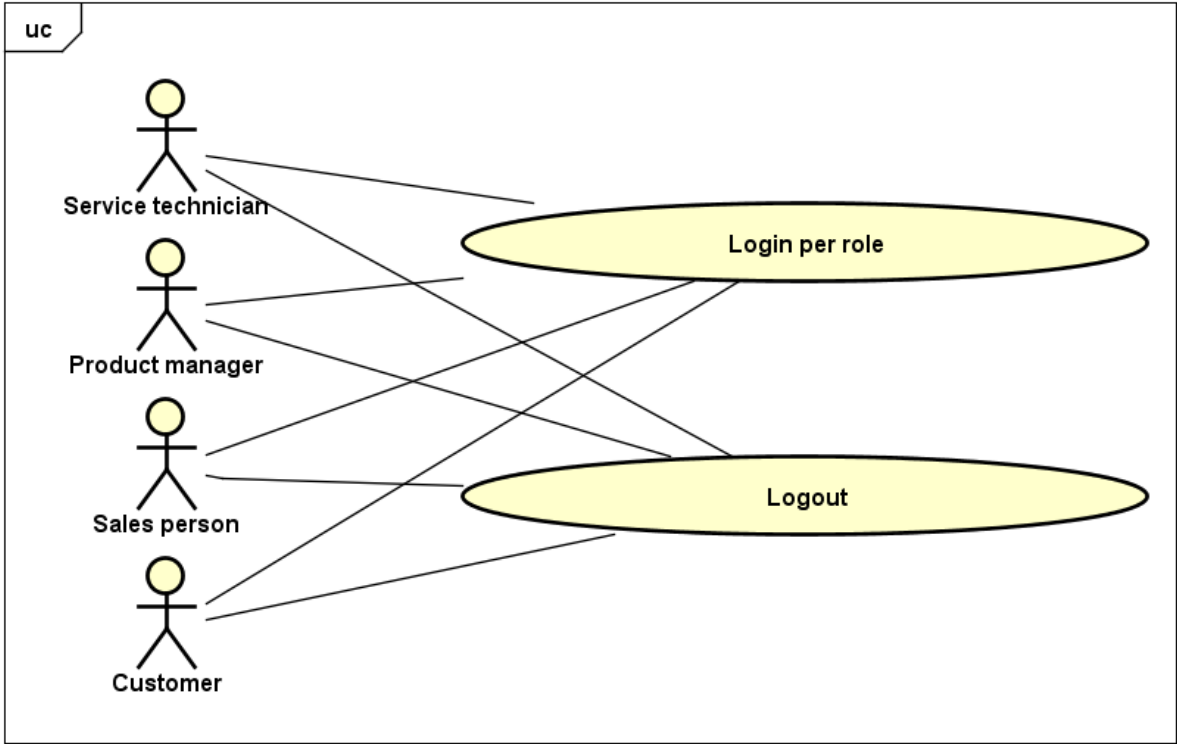


Abbildung 26 – Use Case Diagramm Sessionmanagement

Anhang C – Test Protokolle

Usability-Test 22.05.2018

Angaben zur Durchführung

Software Version: Build #70
 User: Lukas Koller / Student Informatik
 Betreuer: Luca Salzani
 Datum: 22. Mai 2018
 Methodik: Lautes Denken

Einführung

Herzlich Willkommen zu unserem Usability-Test. Wir danken Dir schon im Voraus für die aufgebrauchte Zeit. Durch diesen Test wollen wir auf keinen Fall Dein Können oder sogar Dich persönlich testen - Du kannst absolut nichts falsch machen.

Wir wollen mit diesem Test unsere Software unter die Lupe nehmen und allfällige Schwierigkeiten in der Benutzerführung, beziehungsweise dem Design aufdecken.

Testablauf

Der Test wird wie folgt funktionieren:

- Der Betreuer gibt bekannt, welcher Test Case durchgeführt wird.
- In jedem Test Case findest Du eine Aufgabe, die Du **selbstständig** in unserer Software lösen sollst. Du wirst vom Betreuer – falls nicht anders spezifiziert – keine Hilfe erhalten.
- Während dem Lösen der Aufgabe bitten wir Dich, immer alle Gedankengänge laut zu denken, damit der Betreuer Notizen machen kann.

Durchführung

Aufgabe

Aufgabenstellung

Als Produktmanager möchtest Du ein neues Marketing Kit für die Maschinentypen „LegoRoboter“ und „TechMachine“ erfassen. Das Marketing Kit soll ein neues Feature „Waterproof“ zur Verfügung stellen, welches die betreffenden Maschinen wasserdicht macht. Die von dir geplanten Releasezeitpunkte sind der 16. Juni 2019 (LegoRoboter) und 20. August 2019 (TechMachine). Da mit der Entwicklung der Erweiterungen noch nicht begonnen wurde, kannst Du auch nicht genau sagen, wie viel die neuen Features am Ende kosten werden oder wie diese verbaut werden sollen.

Etwas Zeit ist nun vergangen. Wir schreiben den 5. Mai 2019. Die Erweiterung für den „LegoRoboter“ ist von der Entwicklung fertiggestellt worden. Die Produkttests wurden auch erfolgreich absolviert. Du möchtest nun die Erweiterung freigeben und passt dies entsprechend an. Der Preis wird CHF 5'000 betragen. Erfasse dies im System und speichere ab.

Notizen Betreuer

- Nach längerem Suchen erst den Menüpunkt zu den Marketing Kits gefunden -> Unklare Symbole
- Button für das Hinzufügen nicht auf Anhieb entdeckt
- „No Data found“ nicht ganz korrekt, da noch gar keine gefunden werden können
- Konzept der Marketing Kit Instances unklar (Link zwischen Maschinentypen und Marketing Kit Instanzen)
- Gute Eingaben für Marketing Kit Instanz
- Unklares Datumsformat
- Erwartete beim Klick auf das Inputfeld Datum einen Picker
- Testperson versuchte per Doppelklick ein Element aus der Liste auszuwählen

Tabelle 24 – Usability-Test Protokoll Lukas Koller, Test Case 1

Aufgabe

Aufgabenstellung

Du hast im Moment einen Kunden am Telefon. Er hat Fragen zu einer Maschine. Leider weisst du nicht von welcher Maschine er redet. Finde mittels Rückfragen (Dein Betreuer spielt den Kunden) heraus um welche Maschine von welchem Kunden es sich handelt und lies folgende Daten aus.

- Wie viele offene Service Cases beziehen sich auf diese Maschine?
- Wo steht die Maschine?
- Welche Teile sind auf dem Logic Board verbaut?

Notizen Betreuer

SN6, BigMachine, MyTech GmbH, 3 SCs, Standort HSR, 2 Batterien/Microcontroller/BombTimer

- Testperson klickte sich zuerst durch das Menü und stellte dann die Frage um welchen Kunden es geht -> MyTech GmbH
- Filterung nach Kunde sofort geklappt
- Erwartete klickbare Marker auf der Map (2 Maschinen wurden als 1 Marker angezeigt)
- Fragte nach der Seriennummer -> SN6 -> Maschine gefunden, Details angezeigt (wieder über Doppelklick)
- Adresse verschoben
- „BOM“ nicht klar -> sollte ausgeschrieben sein
- Alle Fragen korrekt beantwortet
- Map sollte nicht ohne weiteres scrollbar sein

Tabella 25 – Usability-Test Protokoll Lukas Koller, Test Case 2

Aufgabe

Aufgabenstellung

Der Kunde MyTech GmbH möchte wissen, ob seine Maschine mit der Seriennummer „SN5“ bereits die Kamerafunktion unterstützt welche er bei einer anderen Maschine gesehen hat. Finde heraus, ob dieses Feature bereits verfügbar ist. Wenn nein, teile dem Kunden das Releasedatum mit.

Notizen Betreuer

- Kunde und Maschine per Filter ausgewählt -> Farbe bei Filter (Violet) sollte grün sein
- Testbenutzer war sich nicht sicher, ob seine Antwort (nein) korrekt ist, wusste nicht wo er das Releasedatum nachschauen kann (Suchte im Tooltip)
- Danach erinnerte er sich an Test Case 1 und versuchte, über die Marketing Kit Ansicht die Frage zu beantworten. Jedoch wusste er nicht, von welchem Typ die Maschine SN5 war. Also wieder zurück zu Maschinendetail. Danach konnte die Frage beantwortet werden.

Tabella 26 – Usability-Test Protokoll Lukas Koller, Test Case 3

Aufgabe

Aufgabenstellung

Du hast dem Kunden MyTech GmbH nun das Feature „Outdoor“ für die Maschine mit der Seriennummer „SN6“ verkauft. Erfasse nun einen Service Case mit entsprechender Aktivität. Der Service Case soll vom Servicetechniker Ralf Hek ausgeführt werden.

Logge dich nun aus und als Servicetechniker wieder ein. Um das Feature erfolgreich zu verbauen, soll in der Baugruppe mit der Artikel-ID „Ald24“ ein Mikrocontroller (Ald24212) hinzugefügt werden. Erfasse dies im System und schliesse die Aktivität sowie den Service Case ab.

Notizen Betreuer

- Tester suchte im Menü das Marketing Kit Icon welches er nicht fand da er als Sales eingeloggt war. Sein Weg führte zuerst über diese Ansicht
- Danach suchte er das „+“ Icon in der Service Case Ansicht. Der letzte Weg führte über die Maschine. Dort konnte der Service Case hinzugefügt werden.
- „Cause“ konnte nicht klar bestimmt werden -> Gleiche Eingabe wie bei Description
- Anordnung Vorname, Nachname bei den Servicetechnikern ist nicht geläufig
- Bei Klick auf Input von Marketing Kit sollte das vorausgewählte markiert werden.
- Description wirkte redundant
- Wenn man keine Wahl hat (Status: open), sollte das Feld nicht vorhanden oder deaktiviert sein.
- SC über Filter als Servicetechniker gefunden
- Modification Type nicht ganz klar
- Der Rest funktionierte gut. Vor allem durch die Weiterleitung nach Save konnte schnell abgeschlossen werden.

Tabelle 27 – Usability-Test Protokoll Lukas Koller, Test Case 4

Usability-Test 23.05.2018**Angaben zur Durchführung**

Software Version: Build #70
 User: Raphael Jöhl / Student Informatik
 Betreuer: Luca Salzani
 Datum: 23. Mai 2018
 Methodik: Lautes Denken

Einführung

Herzlich Willkommen zu unserem Usability-Test. Wir danken Dir schon im Voraus für die aufgebrauchte Zeit. Durch diesen Test wollen wir auf keinen Fall Dein Können oder sogar Dich persönlich testen - Du kannst absolut nichts falsch machen.

Wir wollen mit diesem Test unsere Software unter die Lupe nehmen und allfällige Schwierigkeiten in der Benutzerführung, beziehungsweise dem Design aufdecken.

Testablauf

Der Test wird wie folgt funktionieren:

- Der Betreuer gibt bekannt, welcher Test Case durchgeführt wird.
- In jedem Test Case findest Du eine Aufgabe, die Du **selbstständig** in unserer Software lösen sollst. Du wirst vom Betreuer – falls nicht anders spezifiziert – keine Hilfe erhalten.
- Während dem Lösen der Aufgabe bitten wir Dich, immer alle Gedankengänge laut zu denken, damit der Betreuer Notizen machen kann.

Durchführung**Aufgabe****Aufgabenstellung**

Als Produktmanager möchtest Du ein neues Marketing Kit für die Maschinentypen „LegoRoboter“ und „TechMachine“ erfassen. Das Marketing Kit soll ein neues Feature „Waterproof“ zur Verfügung stellen, welches die betreffenden Maschinen wasserdicht macht. Die von dir geplanten Releasezeitpunkte sind der 16. Juni 2019 (LegoRoboter) und 20. August 2019 (TechMachine). Da mit der Entwicklung der Erweiterungen noch nicht begonnen wurde, kannst Du auch nicht genau sagen, wie viel die neuen Features am Ende kosten werden oder wie diese verbaut werden sollen.

Etwas Zeit ist nun vergangen. Wir schreiben den 5. Mai 2019. Die Erweiterung für den „LegoRoboter“ ist von der Entwicklung fertiggestellt worden. Die Produkttests wurden auch erfolgreich absolviert. Du möchtest nun die Erweiterung freigeben und passt dies entsprechend an. Der Preis wird CHF 5'000 betragen. Erfasse dies im System und speichere ab.

Notizen Betreuer

- MKits schnell gefunden, + auch, Unklar ob das Feature im mkit „waterproof“ heisst
- Mkit instance schnell gefunden und hinzugefügt, gut ausgefüllt.
- Doppelklick versucht
- Show details gut gefunden, edit kurz gesucht, gute Eingaben
- Release Date nicht angepasst -> Unklar

Tabelle 28 – Usability-Test Protokoll Raphael Jöhl, Test Case 1

Aufgabe
<p>Aufgabenstellung</p> <p>Du hast im Moment einen Kunden am Telefon. Er hat Fragen zu einer Maschine. Leider weisst du nicht von welcher Maschine er redet. Finde mittels Rückfragen (Dein Betreuer spielt den Kunden) heraus um welche Maschine von welchem Kunden es sich handelt und lies folgende Daten aus.</p> <ul style="list-style-type: none"> - Wie viele offene Service Cases beziehen sich auf diese Maschine? - Wo steht die Maschine? - Welche Teile sind auf dem Logic Board verbaut?
<p>Notizen Betreuer</p> <p>SN6, BigMachine, MyTech GmbH, 3 SCs, Standort HSR, 2 Batterien/Microcontroller/BombTimer</p> <ul style="list-style-type: none"> - Zuerst nach Maschinentyp gesucht, keine Instanzen gefunden, danach über Kunde - Map unerwartet - Scroll auf map nicht ok - Maschinentyp in Tabelle nicht sichtbar - Fragen gut beantwortet, logic board nicht direkt gefunden - „BOM“ unklar

Tabelle 29 – Usability-Test Protokoll Raphael Jöhl, Test Case 2

Aufgabe
<p>Aufgabenstellung</p> <p>Der Kunde MyTech GmbH möchte wissen, ob seine Maschine mit der Seriennummer „SN5“ bereits die Kamerafunktion unterstützt welche er bei einer anderen Maschine gesehen hat. Finde heraus, ob dieses Feature bereits verfügbar ist. Wenn nein, teile dem Kunden das Releasedatum mit.</p>
<p>Notizen Betreuer</p> <ul style="list-style-type: none"> - Maschine gut gefunden - Erwartete klickbare Badges für mkits - Kurz gesucht, danach gefunden über Maschinentypen - Scrollen für das Anzeigen von Marketing Kits nicht ideal, Tabelle der mkits nur md6

Tabelle 30 – Usability-Test Protokoll Raphael Jöhl, Test Case 3

Aufgabe
<p>Aufgabenstellung</p> <p>Du hast dem Kunden MyTech GmbH nun das Feature „Outdoor“ für die Maschine mit der Seriennummer „SN6“ verkauft. Erfasse nun einen Service Case mit entsprechender Aktivität. Der Service Case soll vom Servicetechniker Ralf Hek ausgeführt werden.</p> <p>Logge dich nun aus und als Servicetechniker wieder ein. Um das Feature erfolgreich zu verbauen, soll in der Baugruppe mit der Artikel-ID „Ald24“ ein Mikrocontroller (Ald24212) hinzugefügt werden. Erfasse dies im System und schliesse die Aktivität sowie den Service Case ab.</p>
<p>Notizen Betreuer</p> <ul style="list-style-type: none"> - Hinzufügen von Service Case zuerst über SC Tabelle dann über Maschinentyp gesucht - Danach gut gefunden - Vorname Name nicht korrekt aufgelistet - Cause, Description das gleiche eingetragen - Mkits sind eingerückt - Description Feld hat kein * ist aber required - Erwartete das Erfassen einer Activity bei Erfassen des SC - Unklar, dass dies im SC passieren soll. Brauchte einen hint um modification zu finden - Sofort add gewählt - Verwechselte parent part mit target part, selber gemerkt und korrekt angepasst - SC und Activity gut geschlossen

Tabelle 31 – Usability-Test Protokoll Raphael Jöhl, Test Case 4

Allgemeines**Notizen Betreuer**

- Logout nicht direkt gefunden
- Filterung nicht aktiv bei dashboard (nicht reproduzierbar)
- Abschliessen lieber über Button als edit
- Dashboard nur für SC soll so einfach wie möglich sein
- Aktivität ist editable obwohl geschlossen, anders als bei SC
- Im Stepper klar zeigen, was ausgewählt wurde bei jedem Schritt
- Weiterleitungszeit ist zu lang

Tabelle 32 – Usability-Test Protokoll Raphael Jöhl, Allgemeine Bemerkungen

Systemtest 08.06.2018

Build: #80
 Testumgebung: Windows 10, Testdaten neu initialisiert
 Tester: Luca Salzani

Testcase	Erfüllt	Testcase	Erfüllt
TC01	Ja	TC25	Ja
TC02	Ja	TC26	Ja
TC03	Ja	TC27	Ja
TC04	Ja	TC28	Ja
TC05	Ja	TC29	Ja
TC06	Ja	TC30	Ja
TC07	Ja	TC31	Ja
TC08	Ja	TC32	Ja
TC09	Ja	TC33	Ja
TC10	Ja	TC34	Ja
TC11	Ja	TC35	Ja
TC12	Ja	TC36	Ja
TC13	Ja	TC37	Ja
TC14	Ja	TC38	Ja
TC15	Ja	TC39	Ja
TC16	Ja	TC40	Ja
TC17	Ja	TC41	Nein
TC18	Ja	TC42	Ja
TC19	Ja	TC43	Ja
TC20	Ja	TC44	Ja
TC21	Ja	TC45	Ja
TC22	Ja	TC46	Ja
TC23	Ja	TC47	Ja
TC24	Ja	TC48	Ja

Tabelle 33 – Systemtest Protokoll

Bemerkungen

Test TC41 schlug fehl, da die zu testende Funktionalität noch nicht verfügbar ist. Ein dementsprechender Issue wurde eröffnet und zugeteilt.

Anhang D – Installationsanleitung

In diesem Abschnitt wird erklärt, wie die Applikation eingerichtet werden kann. Die Anleitung richtet sich vor allem an Entwickler und nicht an Benutzer der Applikation.

Vorbedingung Client

- Visual Studio inkl. folgender Workloads
 - .NET Desktopentwicklung
 - ASP.NET und Webentwicklung
 - .NET Framework 4.6.1 SDK
 - Plattformübergreifende .NET Core-Entwicklung
- Angular IDE nach Wahl – Entwickelt wurde mit Webstorm 2018
- Google Chrome ab Version 64
- Dotnet Version 2.1+

Vorbedingungen Server

- Internet Information Services (IIS) Version 10+

Installation

1. IIS Pakete

Über den Server Manager sind folgende IIS Pakete hinzuzufügen.

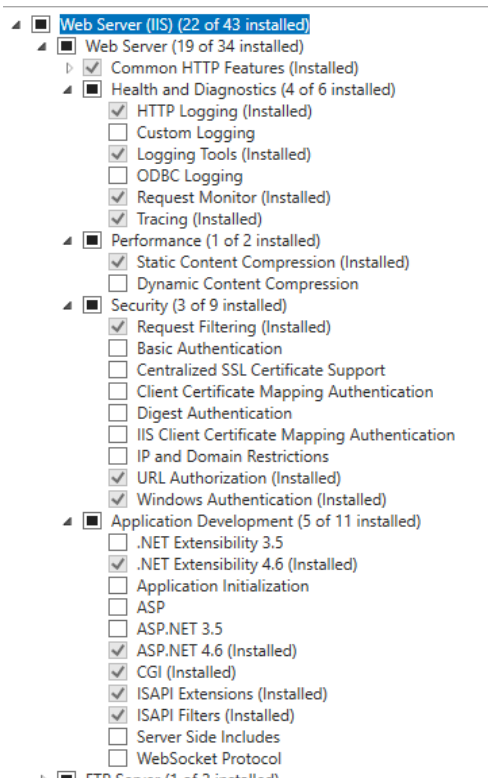


Abbildung 27 – IIS Tabelle

2. Application Pools

In der IIS Konsole sind folgende Application Pools zu erstellen.

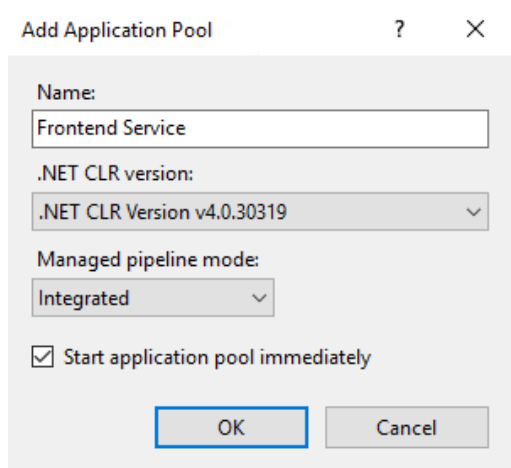


Abbildung 28 – Frontend Application Pool

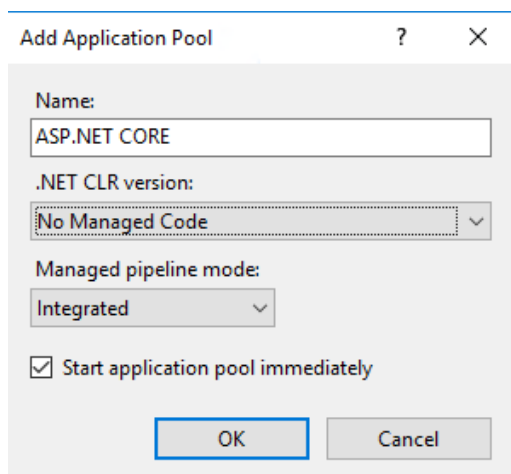


Abbildung 29 – ASP.NET Core Application Pool

3. IIS Webseiten

Danach werden die Webseiten in der IIS Konsole erstellt. Dabei sind die Pfade im System für das Deployment auszuwählen. Für das Backbone und das Backend ist der ASP.NET Core Application Pool zu verwenden. Im Frontend entsprechend der Frontend Service Pool.

Name	Status	Binding	Path
ProductBackbone	Started (http)	*:40003 (http)	C:\Users\Administrator\Desktop\BackboneDeployment
BackendApi	Started (http)	*:40004 (http)	C:\Users\Administrator\Desktop\Backend_HSR-Bachelorarbeit
FrontendService	Started (http)	*:80 (http)	C:\Users\Administrator\Desktop\FrontendDeployment

Abbildung 30 – IIS Sites

4. Shares für Deploymentordner

Der Ordner welcher das Deployment des Frontends beinhaltet unter Windows mit Lesezugriff für „Everyone“ freigeben.

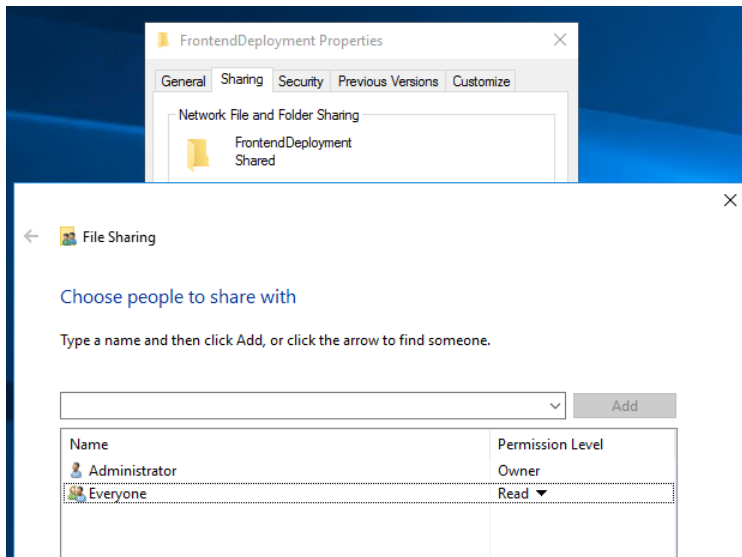


Abbildung 31 – Frontend Ordner Share

5. Anpassung der Serveradressen

Die Verbindungsadressen des Frontends und des Backends auf die entsprechenden Adressen des neuen Backends, beziehungsweise Backbones ändern. Für das Frontend ist die Datei *environment.prod.ts* und für das Backend die Datei *BackboneRoutes.cs* im Data Access Assembly anzupassen.

```
export const environment = {
  production: true,
  urls: {
    baseEndpoint: 'http://152.96.56.84:40004/api',
    staticImageFileLocation: 'http://152.96.56.84:40004/images/',
    authentication: '/authentication',
  }
}
```

Abbildung 32 – Frontend (*environment.prod.ts*)

```
42 references | Luca Salzani, 53 days ago | 1 author, 7 changes | 7 work items
public static class BackboneRoutes
{
  private const string BaseUrl = "http://152.96.56.84:40003/api";
}
```

Abbildung 33 – Backend (*BackboneRoutes.cs*)

6. Backend Build

Mittels der Kommandozeile im Ordner der Solution kann das Backend erstellt werden. Nach der Erstellung sollte das Ergebnis gegebenenfalls noch auf den Deploymentordner des Servers kopiert werden.

```
dotnet publish BackendAPI -o <GEWÜNSCHTER DEPLOYMENT PFAD>
```

7. Frontend Build

Im Terminal von Webstorm sollten vor dem Build alle Abhängigkeiten installiert werden. Danach kann der Build durchgeführt werden. War dieser erfolgreich, sind die Inhalte des Ordners *dist* im Frontendverzeichnis in den Deploymentordner zu kopieren.

```
npm install -g @angular/cli@latest
npm install
ng build --prod --aot
```

```
D:\HSR\_MODULE\Bachelorarbeit\BackendApi\Frontend\ProductFrontend>ng build --prod --aot

Date: 2018-06-08T13:39:23.672Z
Hash: 31fb9f1faa333a925e14
Time: 137828ms
chunk [0] 0.f24d2b25c8747303b61c.js () 5.11 kB [rendered]
chunk [1] 1.f8de6b15b567240461ff.js () 5.62 kB [rendered]
chunk [2] 2.4aace90a5ff63803b9e1.js () 4.57 kB [rendered]
chunk [3] 3.092229ca0cf0b0cdc87d.js () 16.6 kB [rendered]
chunk [4] 4.7a984c151d96ad4adc1e.js () 3.98 kB [rendered]
chunk [5] 5.efcfbad56482979c6f86.js () 4.55 kB [rendered]
chunk [6] 6.ccca80e18a30a76d7ab5.js () 5.48 kB [rendered]
chunk [7] 7.aade5eaed1cb8a6e446a.js () 4.49 kB [rendered]
chunk [8] 8.08684e98a86a29692849.js () 2.92 kB [rendered]
chunk [9] runtime.0371d03ebf01bab4a6a3.js (runtime) 2.04 kB [entry] [rendered]
chunk [10] styles.a90e0ce9a1522afe92c2.css (styles) 351 kB [initial] [rendered]
chunk [11] polyfills.67bd8e3fb59dcl5e9dae.js (polyfills) 59.4 kB [initial] [rendered]
chunk [12] main.bda3bf958f0a5fd4bla9.js (main) 2.69 MB [initial] [rendered]
```

Abbildung 34 – Frontend Build

Die Applikation ist nun auf Port 80 des Servers erreichbar.

Lokales Ausführen

Backend sowie Frontend können auch lokal ausgeführt werden.

Backend

Die Solution sollte in Visual Studio geöffnet sein. Ausserdem muss auf einem Server oder dem Client eine Instanz des Backbone Servers gestartet sein. Die Adresse des Backbone Servers ist in der Datei *BackboneRoutes.cs* analog zur vorherigen Anleitung einzutragen. Danach kann die Applikation über die Runkonfiguration *BackendAPI* gestartet werden. Die Swaggerseite sollte nun ersichtlich sein. Der Output ist in einem separaten Fenster ersichtlich.

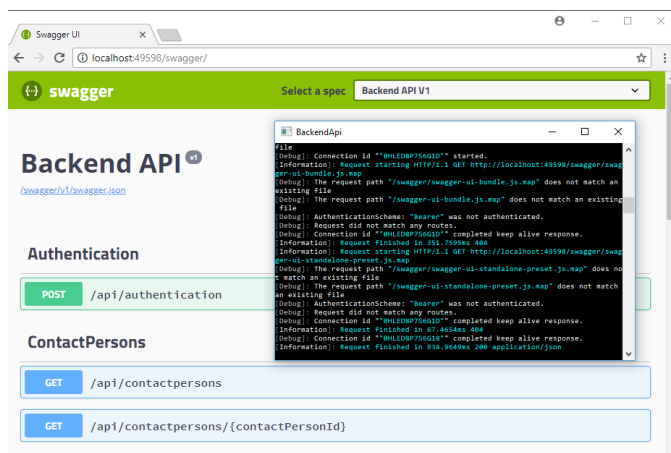


Abbildung 35 – Backend lokal

Frontend

Die Adresse einer laufenden Backendinstanz sollte in der Datei *environment.ts* (ACHTUNG: Nicht die Produktionsumgebung) eingetragen sein. Mittels folgender Befehle kann die Applikation dann gestartet werden. Die Applikation läuft nun unter der eingblendeten Adresse (hier: <http://localhost:4200>).

```
npm install -g @angular/cli@latest
npm install
npm start
```

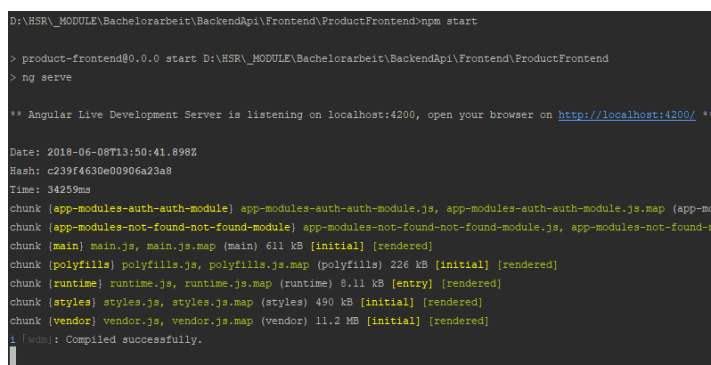


Abbildung 36 – Frontend lokal

Deployment auf Microsoft Azure

Im Zuge der Erweiterungen wurde die Applikation auch auf Microsoft Azure gestartet. Dazu ist eine Subscription nötig. Mit einer @hsr.ch Email-Adresse bekommt man \$100 als Guthaben gratis [21].

1. Projekt veröffentlichen

Mit einem Rechtsklick auf das Webprojekt – Publish das Projekt veröffentlichen.

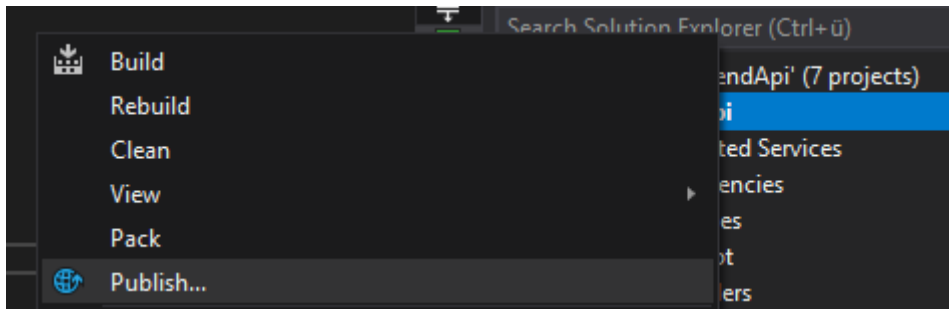


Abbildung 37 – Projekt veröffentlichen

2. Ziel spezifizieren

Es muss App Service ausgewählt werden. Dabei sollte „Create New“ ausgewählt werden, um in der Subscription einen neuen App Service zu erstellen.

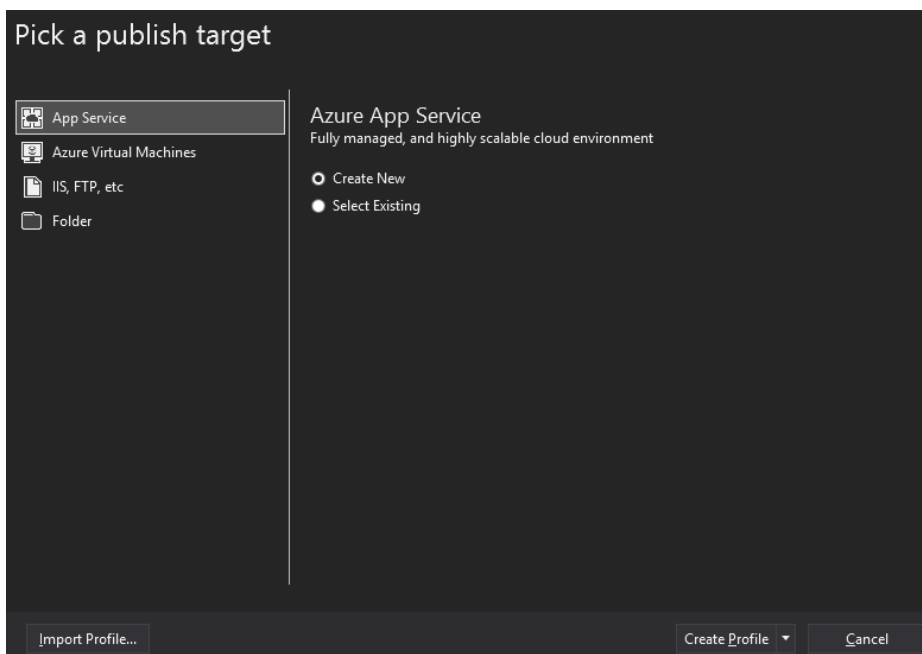


Abbildung 38 – App Service Ziel auswählen

3. App Service erstellen

Als erstes sollte man sich mit dem entsprechenden Account der Subscription anmelden. Der App Name kann frei gewählt werden. Die Ressourcengruppe wird automatisch neu erstellt. Beim Hosting Plan kann man zwischen den verschiedenen Grössen wählen. Hier wurde B1 gewählt. F1 (gratis) funktioniert als Test auch.

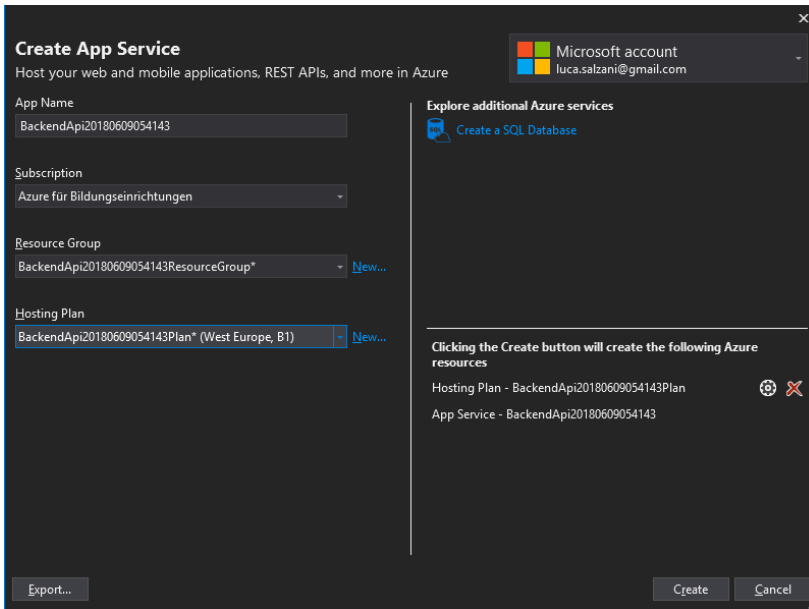


Abbildung 39 – App Service erstellen

4. App ausliefern

Danach kann die Applikation über den Button „Publish“ ausgeliefert werden. Über den angezeigten Link wird man auf die Applikation weitergeleitet.

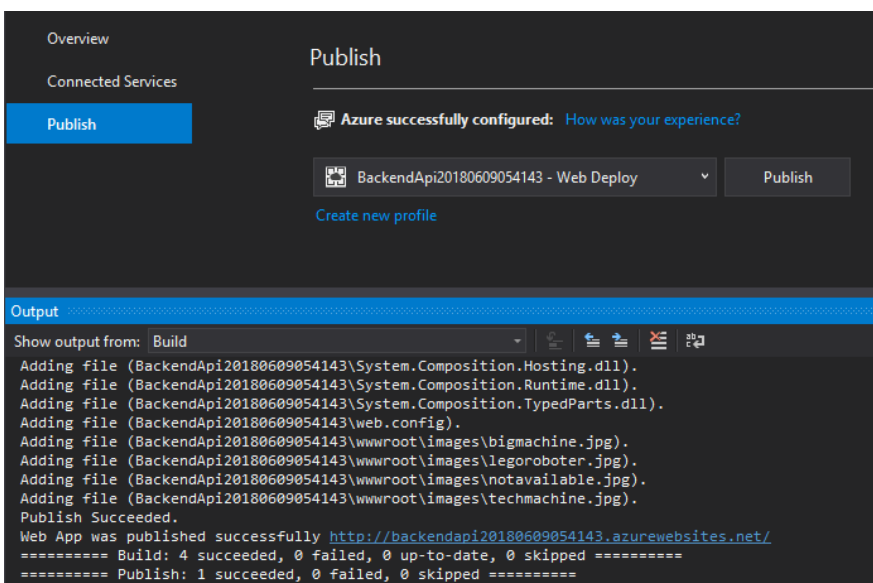


Abbildung 40 – App ausliefern

Anhang E – Sitzungsprotokolle

Sitzungsprotokoll: Schnittstellen Umsysteme

Projekt: SPA für Management von Maschinenlebenszyklen
Woche: 2
Datum / Zeit: 26.02.18 09:00
Protokollführung: Fabian Gübeli

Sitzungsteilnehmer

Prof. Dr. Daniel P. Politze
Lukas Kretschmar, MSc FHO in Engineering (ICT)

Salzani Luca Isalzani
Gübeli Fabian fguebeli

Traktanden

- Domainmodell
- Schnittstellen Umsysteme
- Aufgabenstellung
- Show Cases
- Feedback Studienarbeit

Beschlüsse / Ergebnisse

- Feedback Studienarbeit
 - Saubere Aufnahme der NFA
 - Code wurde nicht sehr genau inspiziert
 - Saubere UI Umsetzung und Trennung was umgesetzt wurde und was nicht
- Product Back Bone
 - Aufbau: Service Layer – Business Layer – Repository Layer – Interface für Adapter – Adapter für Drittsysteme
 - PLM Software ist ARAS
 - As Build BOM kommt aus dem ERP
 - As Maintained BOM als NoSQL speichern
 - Versionierung bei internen Interfaces: Bleibt bei der BA aussen vor, ist erst später ein Thema
 - API-Schnittstelle, kann durch uns bearbeitet werden und so können wir mit Dummy Daten Mocken. Der Rest für das Anbinden der 3t Systeme wird durch Lukas und Berni erledigt.
 - **Suggestion für mögliche Ersatzteile werden in dieser Arbeit nicht berücksichtigt.**
 - Eigener Business Layer designen mit z.B. Modelio
 - HSR Git Server wird verwendet
- Security
 - Rollenkonzept einführen für das Dashboard -> eventuell nur mit Dropdown
 - API-KEY wie zum Beispiel JWT
- Lukas sendet Domainmodel und Zugangsdaten für Git

Sitzungsprotokoll: Definition NFA/FA/Domainmodel

Projekt: SPA für Management von Maschinenlebenszyklen
 Woche: 3
 Datum / Zeit: 05.03.18
 Protokollführung: Fabian Gübeli

Sitzungsteilnehmer

Prof. Dr. Daniel P. Politze

Salzani Luca Isalzani
 Gübeli Fabian fguebeli

Traktanden

- FA
- NFA
- Prototype
- Domainmodel – PLM Anbindung notwendig?
- Showcases – Upgrade Kit, was müssen wir dabei abdecken?
- Upgrade Paket – Wie erkennen wir das? Was kann verkauft werden?
- BOM Anpassung – Was kann durch was ersetzt werden?

Beschlüsse / Ergebnisse

- Dashboard Evaluierung: Kosten von 35\$ wären kein Problem, jedoch müssen Lizenzen, Support und Funktionalitäten beachtet werden.
- FA
 - Anwender definieren als Rolle
 - Upgrade Kit ↔ Marketing Kit ↔ Feature Kit
 - Produktemanager als Rolle definieren
 - Auswertungen über defekte Teile an einem Maschinentyp
 - Probleme im Feld z.B. auch nach Regionen
 - Produkt kommt aus der Entwicklung, jetzt müssen Pakete erstellt werden, die dann dem Verkäufer angezeigt werden
 - Kunde sieht wie ein Online Shop welche Features für seine Maschinen noch verfügbar wären und welche zum Beispiel verbaut wurden
 - (Kunden können Wünsche erfassen, damit der Produktemanager aus Service Cases und Kundenwünsche neue Features evaluieren kann)
 - Bei Problemen werden Verkäufer oder direkt der Service via Hotline kontaktiert
- NFA
 - Das gleiche wie bei der SA – Prototypentwicklung
- Domainmodel
 - Wann ist das Produkt im PLM und wann im ERP? Haben wir etwas mit der Freigabe zu tun? -> Definition erstellen
 - Ersatzteil und Verschleissteil sind im PLM bereits definiert. Ersatzteil-Kit wird im PLM definiert, aber Preise oder Kompatibilität im ERP oder Excel.
 - Upgrade Kit wird zum Teil auch in einer Testphase produktiv in einer Maschine eingesetzt, jedoch wird dies heute noch nicht sauber definiert, wo diese eingesetzt werden.
- BOM Anpassung
 - Feature Kit definiert wo es eingebaut werden kann und welche Teile dabei ersetzt werden. Ansonsten werden Teile nur 1:1 ersetzt.

Sitzungsprotokoll: Domainmodel

Projekt: SPA für Management von Maschinenlebenszyklen
 Woche: 5
 Datum / Zeit: 19.03.2018, 10:00
 Protokollführung: Isalzani

Sitzungsteilnehmer

Prof. Dr. Daniel P. Politze
 Bernhard Fradl

Salzani Luca Isalzani
 Gübeli Fabian fguebeli

Traktanden

- Wann ist das Produkt im PLM und wann im ERP? Haben wir etwas mit der Freigabe zu tun?
 - Workflow neue Maschine? Müssen diese im GUI erfasst werden?
- Ersatzteil und Verschleissteil sind im PLM bereits definiert. Ersatzteil-Kit wird im PLM definiert, aber Preise oder Kompatibilität im ERP oder Excel? Zu welcher Maschine gehören die Teile?
- Wo werden Marketingpakete definiert und abgelegt inkl. Preis und Status (Im Verkauf, Sold out..).. wie oft kann dieses verbaut werden?
- Kunden, Mitarbeiter über Umsysteme oder selber verwaltet?
- Welche Informationen sind auf den Produkten in PLM und ERP vorhanden?
 - Ersatzteil/Verschleissteil/Upgrade
 - Kompatibilität auf Maschinentypen (evtl. mehrere)
 - Preis
 - Freigabe anhand ERP/PLM oder durch Flag
 - Unterscheidung Bauplan/Instanz
 - Baugruppe oder Teil
 - Metainformationen wie ArtNr, SNr, Releasedatum, Baudatum, Aktiv/SoldOut/Konzeption, Bezeichnung
 - Einzelne Objekte für Quantität oder als Property
- Statusupdate
 - Durchstich Backend
 - Prototypen Frontend -> Entscheid Technologie

Beschlüsse / Ergebnisse

- PLM (Aras) = Kunden unabhängig, BluePrint / Im PLM sind im Produkt mehr Möglichkeit als in der realen Maschine möglich (z.B. zwei Motoren)
- ERP = As Ordered entspricht as Build und wird dort gespeichert
- Marketing Paket: Wird im PLM erstellt, Status anhand vom Status der Teile
- PLM: Boolean-Flag Verschleissteil/Ersatzteil
- PLM: Flag «State» - In Arbeit, In Review, Freigegeben
- PLM: Neues Attribut: «Soll Entwicklungsdatum»

Sitzungsprotokoll: Marketing Kits

Projekt: SPA für Management von Maschinenlebenszyklen
Woche: 15
Datum / Zeit: 28.05.2018, 13:00
Protokollführung: Isalzani

Sitzungsteilnehmer

Prof. Dr. Daniel P. Politze

Salzani Luca Isalzani
Gübeli Fabian fguebeli

Traktanden

- Abklären ob Marketing Kits korrekt umgesetzt wurden
- Allgemeine Vorführung des Projektstandes
- Administratives
 - Formulare
 - Abgabezeitpunkt
 - Abgabeort
 - Wie viele gedruckte Exemplare

Beschlüsse / Ergebnisse

- Administratives
 - Je eine gedruckte Version ohne Anhang für Mirko Stocker und Daniel Politze
 - Digitale Abgabe komplett an alle
 - Abgabe von Daniel Politze an Fr. Breitenmoser (3.120)
- Mehrere Marker auf Karte im Dashboard gewünscht
- Maschinendetailansicht effizienter gestalten
- Backbone auch etwas dokumentieren
- Marketing Kit Badges sollten klickbar sein
- Service Case Tabelle ändern
 - Datum, Beschreibung, Status
- In der Stückliste die Artikelnummer ergänzen
- Präsentation
 - Ideen
 - Vorgehensweise
 - Methodik
 - Ergebnis
 - Ca 1h inklusive Fragen -> 20-30 Minuten Präsentationszeit

Anhang F – Zugangsdaten

Gmail

Email:

Passwort:

Google Maps API-Key:

RedmineAdresse: <http://sinv-56084.edu.hsr.ch:40000/redmine>

Benutzername:

Passwort:

JenkinsAdresse: <http://sinv-56084.edu.hsr.ch:40001/jenkins>

Benutzername:

Passwort:

HSR-Server

IP: 152.96.56.84

Benutzername:

Passwort:

Adressen der bestehenden Installation

Bei Abgabezeitpunkt sind folgende Adressen und Zugangspunkte konfiguriert.

Adresse	Beschreibung
http://sinv-56084.edu.hsr.ch	Produktives Frontend Deployment mit Jenkins
http://sinv-56084.edu.hsr.ch:40000/redmine	Redmine-Repository des Projekts für Iterationsplanung und Zeiterfassung
http://sinv-56084.edu.hsr.ch:40001/jenkins	Jenkins-Installation für CI/CD
http://sinv-56084.edu.hsr.ch:40003	Produktives Backbone Deployment per Skript
http://sinv-56084.edu.hsr.ch:40004	Produktives Backend Deployment mit Jenkins
http://sinv-56084.edu.hsr.ch:40006	Frontend Staging Deployment per Skript
http://sinv-56084.edu.hsr.ch:40007	Backend Staging Deployment per Skript
http://backendapi20180609054143.azurewebsites.net	Test-Backend auf Azure Manuelles Deployment

Tabelle 34 – Wichtigste Serveradressen bei Abgabezeitpunkt

Anhang G – Persönliche Reflexion

Luca Salzani

Aufgrund der als Studienarbeit durchgeführten Vorarbeit gelang es mir schnell, mich im Projektumfeld zurechtzufinden. Dabei half auch, dass der Projektpartner sowie der Betreuer gleich blieben. So konnte schon gegen Ende des letzten Semesters ein Workshop durchgeführt werden, in welchem die Anforderungen für die Bachelorarbeit festgelegt wurden. Durch die positiven Erfahrungen aus der Studienarbeit konnten wir viele Vorgehensweisen direkt übernehmen. Wir verwendeten wieder die bewährte Review Prozedur und die Aufgabenteilung in Backend und Testing, welche ich durchführte, und das Frontend und Design, welches Fabian bearbeitete.

Während der Entwicklungsphase haben wir zu wichtigen Zeitpunkten den aktuellen Stand der Applikation mit dem Betreuer besprochen. Dies half dabei, die Bedürfnisse der Industrie nicht aus den Augen zu verlieren. Es wurden gute Ansätze für Erweiterungen der Software geliefert. Während der Entwicklung haben wir für das Frontend und für das Backend je ein Code Review durchgeführt. Beim Frontend konnte Silvan Gehrig einige Inputs für die Struktur und die Lösung verschiedener Probleme zur Verfügung stellen. Das Backend wurde mit Lukas Kretschmar zusammen angesehen. Dabei kamen einige Punkte zu Tage an welche wir nicht gedacht haben. Diese Code Reviews waren sehr hilfreich für die Entwicklung der Applikation. Durch die grössere Erfahrung der beiden Reviewer konnten wir auch für zukünftige Projekte profitieren.

Mit der Qualität der Applikation bin ich zufrieden. Es ist eine klare Steigerung im Vergleich zu der Studienarbeit zu erkennen. Sowohl im Angular Projekt wie auch im Backend. Vor allem die saubere Strukturierung und das starke Trennen der einzelnen Layer waren sehr hilfreich beim Testen der Applikation. Auch bei einer eventuellen Weiterentwicklung wird dies die Arbeit einfacher machen.

Dank der eingeplanten Pufferzeit von zwei Wochen kamen wir dieses Mal auch nicht in einen Zeitdruck. Ich bin sehr froh, dass wir diese eingeplant haben. So konnten wir bis zum Ende die Codequalität auf demselben Niveau halten. Das Mitführen der Dokumentation während der Planung und der Entwicklung erwies sich wieder als sehr nützlich. So konnte eine gesunde Balance zwischen Entwicklung und Dokumentation gefunden werden. Schade fand ich jedoch, dass ähnlich der Studienarbeit der letzte Drucktermin für die Dokumentation bereits zwei Tage vor der Abgabe ist. Dieser könnte meiner Meinung nach auch auf den letzten Abgabetag gelegt werden. So steht allen die Möglichkeit offen, auf Kosten des Studiengangs zu drucken.

Die Struktur der Bachelorarbeit als Teamarbeit finde ich persönlich sehr sinnvoll. So lassen sich grössere Projekte umsetzen und es wird auch die Arbeit in einem Entwicklerteam simuliert. Da diese einen grossen Teil der Arbeit in der Wirtschaft ausmacht begrüsse ich dies sehr. Die Teamarbeit lief sehr gut. Da wir nun schon einige Projekte zusammen umgesetzt haben, konnten wir unsere Fähigkeiten gut ergänzen. Auch die Zusammenarbeit mit den anderen Projektmitgliedern der HSR klappte gut. Treffen mussten dann jedoch aufgrund des hohen Beschäftigungsgrads sehr früh geplant werden. Dies konnte unseren Zeitplan aber nicht durcheinanderbringen, da jederzeit an einem anderen Arbeitspaket weitergearbeitet werden konnte.

Ich bin der Meinung, dass wir in diesem Semester gute Arbeit geleistet haben und uns im Vergleich zu der Studienarbeit nochmals steigern konnten. Aufgrund der hohen Belastung durch andere Module in diesem Semester konnten wir uns nicht ganz so viel Zeit nehmen wie wir es vorgehabt haben.

Fabian Gübeli

Die Bachelorarbeit wurde als Aufbau auf die Studienarbeit ausgelegt. Daher war mir das Themengebiet bereits bekannt. Sehr positiv fand ich, dass bereits vor dem Semesterstart ein Workshop organisiert werden konnte, in dem alle Beteiligten der HSR teilnahmen um möglichst realitätsnahe Anforderungen zu definieren. Die Einarbeitung in ein Vorprojekt ist deshalb entfallen und die dadurch gesparte Zeit konnte anderweitig investiert werden. Ausserdem verwendeten wir in der Bachelorarbeit dieselben Technologien, die wir bereits in der Studienarbeit erfolgreich eingesetzt haben. Insbesondere dort hat sich gezeigt, dass Luca und ich bereits von den vorherigen Erfahrungen profitieren konnten und sich die Effektivität gesteigert hat.

Parallel zur Bachelorarbeit besuchte ich das Modul Web Engineering + Design 3 (WED3), welches unter anderem den Schwerpunkt Single Page Applikation mit Angular hatte. Da ich in der Bachelorarbeit für das Frontend verantwortlich war, konnte ich vieles aus dem Modul WED3 mitnehmen und direkt in der Praxis einsetzen. In der Gliederung des Projektes und der Wiederverwendbarkeit sind grosse Fortschritte gegenüber der Studienarbeit zu erkennen. So wurde auch Wert auf die Sicherheit und die Performance gelegt, welches neue interessante Herausforderungen mit sich brachte. Es hat sich wieder einmal mehr gezeigt, dass sich eine saubere Evaluierung am Projektanfang längerfristig lohnt. So konnten bereits viele neue Anforderungen durch die richtige Evaluation des Cockpits abgedeckt werden.

Während der Entwicklung wurden regelmässig Termine mit dem Betreuer festgelegt, damit wir die funktionalen und optischen Erwartungen an das Frontend abgleichen konnten und die Applikation auf dem richtigen Weg ist. Der Code im Frontend wurde durch Silvan Gehrig, welcher WED3 mit Angular unterrichtete, frühzeitig einem Review unterzogen. Dieses wurde extra früh angesetzt, da der gesamte Projektaufbau mit Modulen bei solch einer Projektgrösse nicht mehr ganz so trivial ist und eine saubere Projektorganisation von Anfang an sehr wichtig ist. Beim Review kamen noch einige Punkte zum Vorschein, diese konnten erfolgreich behoben werden und das Lazy-Loading funktioniert wie gewünscht. Das Backend, welches vor allem Luca Salzani bearbeitete, wurde durch Lukas Kretschmar einem Review unterzogen. Beim Review entstanden spannende Diskussionen und sehr gute Inputs, die bestimmt auch in der Arbeit nach dem Studium noch angewendet werden können.

Die Zusammenarbeit im Team mit Luca Salzani funktionierte perfekt, da wir schon einige Projekte zusammen erfolgreich abgeschlossen haben. Jedes Teammitglied war sich seiner Stärken bewusst und konnte diese einsetzen, um das beste Resultat zu erzielen. Da in den Umsystemen der Bachelorarbeit auch noch Mitarbeiter der HSR involviert waren und diese sehr ausgebucht sind, mussten Termine frühzeitig gesetzt werden. Die Überbrückung bis zum nächsten Termin konnten wir jedoch ohne Probleme durch eine dynamische Entwicklung des Front- bzw. Backend bewerkstelligen.

Unter dem Semester waren wir stark ausgelastet durch weitere Module, die Abgaben innerhalb des Semesters verlangten. Trotzdem haben wir nie den Anschluss an die Bachelorarbeit verpasst und sind mit voller Motivation am Ball geblieben. Mit dem Endresultat bin ich sehr zufrieden und freue mich dieses zu präsentieren.

Anhang H – Administrative Anhänge

Recherche-Arbeit / Bachelor-Arbeit**Stücklistenmanagement für Maschinen****1. Bearbeitung durch**

Herr	Gübeli, Fabian	+41 76 543 03 93	fquebeli@hsr.ch
Herr	Salzani, Luca	+41 76 332 93 51	lsalzani@hsr.ch

2. Termine

Ausgabe:	Mittwoch, 21. Februar 2018	17 30 Uhr	3.109
Abgabe RA:	Freitag, 01. Juni 2018	23 59 Uhr	Moodle
Abgabe BA:	Freitag, 15. Juni 2018	23 59 Uhr	Moodle

Das Druckexemplar darf am Montag der Folgewoche nachgereicht werden.

Kick-off mit Kunde: gemäss Absprache

Besprechungen: Weitere Termine mit dem Kunden gemäss Planung durch den Studenten
Mit dem Dozenten: ca. alle 2-3 Wochen: Fortschrittsbericht

3. Betreuung

HSR:	Prof. Dr. Daniel P. Politze	055 222 4605	daniel.politze@hsr.ch
------	-----------------------------	--------------	--

4. Auftraggeber

Kein externer Auftraggeber

5. Aufgabenstellung**5.1. Ausgangslage**

Nach dem Vertrieb einer Maschine fallen bei dem Produktionsbetrieb weitere Arbeiten bezüglich des Lebenszyklus der Maschine an. Einzelne Systeme wie PLM, ERP oder Projektmanagement arbeiten isoliert. Ein System welches die Daten dieser Systeme in Verbindung setzt und die Abbildung eines Workflows erlaubt fehlt. Die Servicetechniker und Verkäufer können ihre gewünschten Informationen nicht auf einen Blick einsehen.

5.2. Zielsetzung

Aufbauend auf der Studienarbeit HS 2017 soll ein Rollenkonzept und Workflows erarbeitet und implementiert werden um folgende Features zu unterstützen.

- Dashboard mit Übersicht über benötigte Informationen pro Rolle (Kunde, Servicetechniker und Verkäufer)
- (Preventive Maintenance)
- Erfassung und Tracking von Service Cases
- Unterstützung bei Maschinenupgrades
- Umfassende Reportingmöglichkeiten

Eigene Computer-Programme und Software-Pakete dürfen nur eingesetzt werden, wenn:

- legale Lizenzen vorhanden sind,
- die Daten auch an der Schule gelesen und weiter bearbeitet werden können.

Weitere Ressourcen (z.B. kundenspezifische) müssen mit dem Betreuer abgesprochen werden.

6.3. Vertraulichkeit

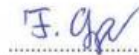
Interne Informationen, Daten und Resultate sowie abgegebenen Dokumente und Zeichnungen müssen vertraulich behandelt werden und dürfen nicht an Dritte weitergegeben werden.

Die Ergebnisse der Arbeit gehören der Schule. Der Auftraggeber erhält das Recht, die Ergebnisse für seine eigenen Zwecke zu verwenden.

Zur Verfügung gestellte Ressourcen müssen sauber, unbeschriftet und in gutem Zustand mit der Arbeit zurückzugeben werden.

6.4. Unterschriften

Rapperswil, den 5.3.18



Fabian Gübeli

Rapperswil, den 5.3.18



Luca Salzani

Rapperswil, den 5.3.18



Dr. Prof. Daniel P. Politze



Eigenständigkeitserklärung

Erklärung

Ich erkläre hiermit,

- dass ich die vorliegende Arbeit selber und ohne fremde Hilfe durchgeführt habe, ausser derjenigen, welche explizit in der Aufgabenstellung erwähnt ist oder mit dem Betreuer schriftlich vereinbart wurde,
- dass ich sämtliche verwendeten Quellen erwähnt und gemäss gängigen wissenschaftlichen Zitierregeln korrekt angegeben habe.
- dass ich keine durch Copyright geschützten Materialien (z.B. Bilder) in dieser Arbeit in unerlaubter Weise genutzt habe.

Ort, Datum: Rapperswil, 15.05.2018

Name, Unterschrift:

Fabian Trübeli
F. Trübeli

Luca Salzani
Luca Salzani

Einverständniserklärung Publikation auf eprints.hsr.ch

 SA BATitel der Arbeit: Stücklistenmanagement für MaschinenTeam: Luca Salzani, Fabian GubeliBetreuer: Prof. Dr. Daniel P. Poltze

Wir sind mit der Publikation unserer Arbeit auf eprints.hsr.ch einverstanden, sofern für diese Arbeit keine Geheimhaltungsvereinbarung unterzeichnet wurde.

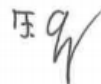
Nach Bekanntgabe der Note haben wir die Möglichkeit innert 14 Tagen Einsprache zu erheben und das Einverständnis zur Publikation der Arbeit auf eprints.hsr.ch zurückzuziehen. In diesem Falle wird nur der Abstract publiziert.

Rapperswil, 05.03.2018

Name(n)

Unterschrift(en)

Fabian Gubeli



Luca Salzani



Daniel Poltze





Vereinbarung

1. Gegenstand der Vereinbarung

Mit dieser Vereinbarung werden die Rechte über die Verwendung und die Weiterentwicklung der Ergebnisse der Bachelorarbeit „Stücklistenmanagement für Maschinen“ von Fabian Gübeli und Luca Salzani unter der Betreuung von Dr. Prof. Daniel P. Politze geregelt.

2. Urheberrecht

Die Urheberrechte stehen der Studentin / dem Student zu.

3. Verwendung

Die Ergebnisse der Arbeit dürfen sowohl von der Studentin / dem Student wie von der HSR nach Abschluss der Arbeit verwendet und weiter entwickelt werden

Rapperswil, den 5.3.18
.....
Fabian Gübeli

Rapperswil, den 5.3.18
.....
Luca Salzani

Rapperswil, den 5.3.18
.....
Dr. Prof. Daniel P. Politze