HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

Entwicklung einer Mobile-App für Service-Einsätze

Bachelorarbeit

Abteilung Informatik
Hochschule für Technik Rapperswil

Frühjahrssemester 2017

Autor(en):	Noah Hendrikx & Pascal Marty
Betreuer:	Prof. Dr. Daniel P. Politze
Experte:	Ramon Schildknecht
Gegenleser:	Prof. Dr. Peter Heinzmann

Abstract

Ausgangslage

Meldet heute eine Maschine ein kritisches Problem so werden zunächst erste Daten zum Fall erhoben. Ist das Problem komplexer, müssen Servicetechniker meist zweimal vor Ort gehen, um das Maschinenproblem zu lösen. Einmal, um das Problem aufzunehmen und die erforderlichen Massnahmen zu bestimmen. Dann ein weiteres Mal, um mit den erforderlichen Ersatzteilen, die Maschine wieder in Betrieb zu nehmen. Die Vision zur Unterstützung dieses Vorgehens basiert auf einer IT-gestützten, strukturierten Analyse und Dokumentation des Problems. Dabei soll schnell festgestellt werden können, ob ein ähnliches Problem bereits woanders erfolgreich gelöst wurde und darauf basierend ein "Massnahmen-Vorschlag" (z.B. per Mobile-App) generiert werden. Damit sollen Zeit und Aufwand bei der Problemlösung vermindert werden.

Vorgehen/Technologien

Das Projekt wurde durch Verwendung von agilen Softwareentwicklungsmethoden durchgeführt. In einer ersten Phase wurde eine Analyse der relevanten Merkmale zur Beschreibung von "Service Cases" durchgeführt. Danach sind Software-Architektur und Benutzerschnittstelle entwickelt und passende Bibliotheken evaluiert worden. Auf Basis dessen erfolgte die Entwicklung eines Prototypen. Folgende Technologien wurden insbesondere eingesetzt:

- C#
- ASP.NET Core
- Microsoft Azure
- VueJS

Ergebnis

Die erarbeitete Mobile-App ermöglicht es Servicetechnikern, Service Cases strukturiert zu erfassen. Zudem unterstützt das System den Servicetechniker proaktiv bei der Erfassung von Daten und bei der Findung von bereits bestehenden Lösungen. Das zugrunde liegende System ist mit Rücksicht auf Flexibilität und Erweiterbarkeit entwickelt worden. Somit besteht die Möglichkeit externe Analyse-Systeme wie "Machine Learning" mit einzubinden.

Management Summary

Ausgangslage

Meldet heute eine Maschine ein kritisches Problem so werden zunächst erste Daten zum Fall erhoben. Ist das Problem komplexer, müssen Servicetechniker meist zweimal vor Ort gehen, um das Maschinenproblem zu lösen. Einmal, um das Problem aufzunehmen und die erforderlichen Massnahmen zu bestimmen. Dann ein weiteres Mal, um mit den erforderlichen Ersatzteilen, die Maschine wieder in Betrieb zu nehmen.

Die Vision zur Unterstützung dieses Vorgehens basiert auf einer IT-gestützten, strukturierten Analyse und Dokumentation des Problems. Dabei soll schnell festgestellt werden können, ob ein ähnliches Problem bereits woanders erfolgreich gelöst wurde und darauf basierend ein "Massnahmen-Vorschlag" (z.B. per Mobile-App) generiert werden. Damit sollen Zeit und Aufwand bei der Problemlösung vermindert werden.

Das Potenzial dieser Vision für das Unternehmen liegt im verbesserten Service-Angebot. Ausgehend von dem "Massnahmen-Vorschlag" kann gegebenenfalls der erste Service-Einsatz des Servicetechnikers entfallen und allfällige Ersatzteile können bereits früher zum Einsatzort versendet und automatisch in Rechnung gestellt werden.

Vorgehen

Das Projekt wurde durch agile Softwareentwicklungsmethoden grob in drei Phasen durchgeführt. In der ersten Phase wurde der Fokus auf die Analyse der Problemstellung und Anforderungen gelegt. Dazu gehört die Betrachtung der relevanten Merkmale zur Beschreibung eines Service Cases und deren Umsetzungsmöglichkeiten.

In einem nächsten Schritt wurden die Software-Architektur, sowie Entwürfe für die Benutzerschnittstelle entwickelt. Dabei wurden verschiedene Technologien, Bibliotheken oder Algorithmen zur Bearbeitung der Daten in Betracht gezogen und evaluiert.

Auf Basis der Entscheidungen aus den ersten beiden Phasen wurde ein Prototyp entwickelt.

Technologien

Die Anwendung wurde in einen Client und ein Backend unterteilt, welche über eine REST-Schnittstelle miteinander kommunizieren. Das Backend wurde mit ASP.NET Core in der Programmiersprache C# realisiert. Für die Persistierung und Bereitstellung der Anwendungsdaten wurden die Online-Dienste der Microsoft Azure Cloud-Computing-Plattform verwendet. Der Client wurde unter Verwendung des Webframeworks VueJS umgesetzt.

Ergebnisse

Die Anwendung unterstützt den Servicetechniker besser bei der Erfassung von Service-Einsätzen. Diese werden nun strukturiert erstellt und können an den jeweiligen Service-Einsatz angepasst werden. Zudem unterstützt das System den Servicetechniker proaktiv bei der Erfassung eines Problems, sowie bei der Lösungsfindung aus bereits bestehenden Daten.

Durch die Umsetzung einer Responsive-Webapplikation ist die Bedienung auf mobilen Geräten, sowie auf dem Desktop möglich. Dadurch ist eine Vielzahl von kompatiblen Geräten zur Benutzung der Anwendung gewährleistet.

Mit Hilfe der Reporting-Funktion erhält der Administrator nun Verbesserungsvorschläge für die Datenpflege. Diese helfen der strukturierten Erfassung von Service-Einsätzen und der optimierten Lösungsfindung zu bereits vorhandenen Problemen.

Mit der Datenhaltung auf der Microsoft Azure Cloud verfügt das System über eine einfach skalierbare Lösung mit viel Potential an zusätzlichen Cloud Services.

Ausblick

Durch die Auslegung auf ein flexibles und erweiterbares System besteht die Möglichkeit zur Verwendung von Machine Learning oder anderen Analyse-Services auf den Anwendungsdaten. Zudem können in Zukunft einfach neue Funktionen zur Anwendung hinzugefügt werden.

Der Client ist durch die Verwendung einer REST-Schnittstelle einfach auswechselbar und könnte beispielsweise durch eine native Mobile-Applikation ersetzt werden.

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

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.

Rapperswil, 13.06.2017

Pascal Marty

Noah Hendrikx

Studienprojekt: Bachelor-Arbeit WING BAW FS 2017

Entwicklung einer Mobile-App für Service-Einsätze

1. Bearbeitung durch

Herr	Hendrikx, Noah	noah.hendrikx@hsr.ch
Herr	Marty, Pascal	pascal.marty@hsr.ch

2. Termine

Ausgabe:	*Montag, 20. Februar 2017*	*Zeit*	*Ort*
Abgabe:	*Freitag, 16. Juni 2017*	*17 00 Uhr*	*Moodle*

Besprechungen: *Besprechungstermine gemäss Planung durch die Studierenden*

3. Betreuung

HSR:	Prof. Dr. Daniel P. Politze	055 222 46 05	daniel.politze@hsr.ch

4. Auftraggeber

Siehe Kap. 3

5. Ausgangslage und Aufgabe

5.1. Beschreibung der Ausgangslage

Meldet heute eine Maschine ein kritisches Problem so werden zunächst erste Daten zum Fall erhoben. Ist das Problem komplexer, müssen Service-Techniker meist zweimal vor Ort gehen, um das Maschinenproblem zu lösen. Einmal, um das Problem aufzunehmen und die erforderlichen Massnahmen zu bestimmen. Dann ein weiteres Mal, um mit den erforderlichen Ersatzteilen, die Maschine wieder in Betrieb zu nehmen.

Die Vision zur Unterstützung dieses Vorgehens basiert auf einer IT-gestützten, strukturierten Analyse und Dokumentation des Problems. Dabei soll schnell festgestellt werden können, ob ein ähnliches Problem bereits woanders erfolgreich gelöst wurde und darauf basierend ein „Massnahmen-Vorschlag" (z.B. per Mobile-App) generiert werden. Damit sollen Zeit und Aufwand bei der Problemlösung vermindert werden.

Das Potenzial dieser Vision für das Unternehmen liegt im verbesserten Service-Angebot. Ausgehend von dem „Massnahmen-Vorschlag" kann ggf. der erste Service-Einsatz des Technikers entfallen, können allfällige Ersatzteile bereits früher zum Einsatzort versendet werden und automatisch in Rechnung gestellt werden.

Zur Realisierung dieser Vision soll eine einfache Mobile-App zur strukturierten Erfassung von Service-Cases durch einen Techniker umgesetzt werden. Des Weiteren sollen über diese App Vorschläge hinsichtlich Lösungsmöglichkeit und/oder benötigte Werkzeuge/Ersatzteile abgerufen werden können. Die erfassten Cases sollen in einer Fall-Datenbank (mit Exportmöglichkeit zu externen Tools) abgelegt und verwaltet werden können. Zur einfacheren Pflege und zur Verwaltung dieser Fälle soll zudem ein einfaches Webinterface mit Editor- und Reporting-Funktion implementiert werden.

5.2. Aufgabe Bachelorarbeit - BAW FS 2017

Im Rahmen der Bachelorarbeit soll hinsichtlich der unter 5.1. beschriebenen Vision ein Prototyp für einen Showcase erstellt werden. Folgende Aufgaben und Fragestellungen sind Teil der Bachelorarbeit:

(1) Durchführen einer Problemanalyse
   a. Gezielte Aufnahme und Analyse der Anforderungen/Use-cases an den Prototypen
   b. Beschreibung der relevanten Informations-Objekte
   c. Analyse der relevanten Merkmale/Features zur Beschreibung von „Service-Cases"

(2) Entwickeln eines Lösungsentwurfs
   a. Erarbeiten einer geeigneten, flexiblen u erweiterbaren System- und Softwarearchitektur
   b. Beschreibung und Evaluation der gewählten Architektur-Komponenten
   c. Beschreibung und Evaluation geeigneter Algorithmen/Bibliotheken
   d. Entwurf von passenden Benutzerschnittstellen (für Nutzung, Pflege und Reporting)

(3) Realisierung und Test
   a. Implementierung des Pototypen auf Basis der Ergebnisse aus (1) und (2)
   b. Dokumentation und Test des Quellcodes, zudem kurze Benutzerdokumentation
   c. Nachweis der Zielerreichung durch Auswahl und Auswertung geeigneter Kennzahlen
   d. Erstellen eines Showcases zu Demonstrationszwecken

5.3. Ziele und mögliche Arbeitsergebnisse

Es sollen in dieser Bachelor-Arbeit folgende Arbeitsergebnisse erreicht werden:

- Flexible, erweiterbare Systemarchitektur für zukünftige Industrie-Anwendungen
- Nachweis der Machbarkeit durch konkrete Umsetzung einer Mobile-App
- Showcase zu Demonstrationszwecken

Die Studierenden beweisen, dass Sie ein komplexes Thema vollständig erarbeiten, dokumentieren und die Ergebnisse präsentieren können.
Es wird dabei besonderes Gewicht gelegt auf:

- Systematische und wissenschaftliche Vorgehensweise vom Verständnis der Aufgabe bis zu den abgeleiteten Schlussfolgerungen.
- Dokumentation und Bewertung von Varianten und Entscheiden
- Schlüssige und nachvollziehbare Argumentationen, sowie kritische Reflektion der eigenen Arbeit und Empfehlung für deren Verwendung / weiteres Vorgehen.

5.4. Weitere Hinweise

Parallel zu dieser Arbeit wird das Potenzial einer solchen Service-App-Lösung in einer weiteren Bachelorarbeit evaluiert. Die Ergebnisse aus dieser Arbeit stellen eine wichtige Informationsquelle für die Domänen- und Anforderungsanalyse dar.

6. Dokumentation

Ein weiteres Hauptziel ist das Erstellen und die Abgabe eines Berichtes, der für alle Adressaten verständlich sein muss.

Die Abgabe erfolgt fristgerecht über das Informatik-Archivierungstool.
Zusätzlich sind eine bestimmte **Anzahl gedruckter und gebundener Berichte** abzugeben:

- 2 Exemplare

Die Korrektur der Dokumente kann eingesehen werden, verbleibt jedoch in der Regel beim Betreuer.

7. Zur Verfügung stehende Infrastruktur

Die an der HSR verfügbare Infrastruktur kann und soll genutzt werden.

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. Ebenso alle Bestellungen (Werkstatt, HSR-3D-Lab, etc.) – Kostenstelle: 3110 1700.

8. 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.

Viel Erfolg mit vielen positiven Erkenntnissen und neuen Erfahrungen!

Inhaltsverzeichnis

Abstract **II**

Management Summary **III**

1 Einleitung **1**
1.1 Was ist ein Service Case 1
1.2 Stand der Technik 1
1.3 Vision 2

2 Problemanalyse **3**
2.1 Service Case Merkmale 3
2.1.1 Service Case 3
2.1.2 Fall 4
2.1.3 Problem 4
2.1.4 Lösungs-Vorschläge 4
2.1.5 Fall-Datenbank 4
2.1.6 Reporting 4
2.1.7 Lösung 5
2.1.8 Maschine 5
2.1.9 Algorithmus 5
2.2 Eingangsdaten 5
2.3 Ansätze Problembeschreibung 6
2.3.1 Attribut-Ähnlichkeit 6
2.3.2 Textanalyse 7
2.3.3 Entscheidungsbaum 7
2.3.4 Fazit Problembeschreibung 8
2.4 Attribut Typen 8
2.5 Form der Lösungsbeschreibung 10
2.5.1 Beschreibung 10
2.5.2 Werkzeug/Ersatzteile 10
2.6 Datenquelle für Lösungsvorschläge 10
2.6.1 Feld Daten 10
2.6.2 Musterlösung 10
2.7 Use Cases 11
2.7.1 Use Case Diagram 11
2.7.2 Aktoren & Stakeholder 12

2.7.3 Kurzbeschreibung der Use Cases . . . 12
2.8 Domainanalyse . . . 13
2.8.1 Domain Model . . . 13
2.8.2 Kurzbeschreibung der Konzepte . . . 13
2.9 Weitere Anforderungen . . . 14
2.9.1 Showcase-Szenarien . . . 15
2.10 Nichtfunktionale Anforderungen . . . 16
2.10.1 Qualitätsmerkmale . . . 16
2.10.2 Referenzsystem . . . 16
2.11 Abdeckung der Anforderungen . . . 17

3 Konzeption und Design **18**
3.1 Software-Architektur . . . 18
3.1.1 Client . . . 18
3.1.2 Backend . . . 20
3.1.3 Persistierung . . . 20
3.2 Technologien . . . 20
3.2.1 Web App Framework . . . 20
3.2.2 REST-Schnittstelle . . . 21
3.2.3 Relationale Datenbank . . . 21
3.2.4 Datei Speicher . . . 21
3.3 UI/UX-Konzept . . . 22
3.3.1 Navigationskonzept als Benutzer . . . 22
3.3.2 Navigation als Administrator . . . 23

4 Ergebnisse **24**
4.1 Deployment . . . 24
4.1.1 Deployment Diagram . . . 24
4.1.2 Azure . . . 24
4.2 Code-Schnittstellen . . . 25
4.2.1 SolutionSearch Interface . . . 26
4.2.2 SearchAlgorithmDispatcher Interface . . . 27
4.2.3 SearchAlgorithm Interface . . . 27
4.2.4 Client-Backend . . . 27
4.2.5 Backend-Azure SQL Database . . . 27
4.2.6 Backend-Azure Blob Storage . . . 28
4.3 REST-Schnittstelle . . . 29
4.4 Code Dokumentation . . . 29

4.4.1 Verwendete NuGet Bibliotheken . . . 29
4.4.2 Kommunikation: Genereller Ablauf . . . 31
4.4.3 Lösungsvorschlag Algorithmus . . . 31
4.5 Testing . . . 33
4.5.1 Verwendete NuGet Bibliotheken . . . 33
4.6 Codemetriken . . . 34
4.7 Zielerreichung . . . 34
4.7.1 Validierung Ergebnis . . . 34

5 Ausblick **36**
5.1 Attribut Typen . . . 36
5.1.1 Komma-Zahlen . . . 36
5.1.2 Diverse Datei-Typen . . . 36
5.2 Maschinen und Ersatzteile . . . 36
5.2.1 Ersatzteil Katalog . . . 36
5.2.2 Maschinentyp Katalog . . . 36
5.2.3 Maschinen Modifikation ohne Fall . . . 37
5.2.4 Vorschlag von Maschinen/Ersatzteilen . . . 37
5.3 Benutzerverwaltung . . . 37
5.3.1 Benutzerkonto . . . 37
5.3.2 Rollen . . . 37
5.3.3 Individuelle Status Ansicht . . . 38
5.4 Lösungsfindung . . . 38
5.5 Benutzer Rating . . . 38
5.5.1 Abgeschlossene Fälle . . . 38
5.6 Reporting . . . 38
5.6.1 Fortlaufende Rating-Auswertung . . . 38
5.6.2 Durchschnittliche Bearbeitungsdauer . . . 39
5.6.3 Mitarbeiter Auswertung . . . 39
5.7 Frontend Client . . . 39
5.7.1 Native Android/IOS App . . . 39
5.7.2 Offline Interaktion . . . 39
5.8 Automatische Datenerhebung . . . 39
5.9 Benutzerfeedback . . . 40
5.10 Batch Processing . . . 40

Abbildungsverzeichnis **42**

Tabellenverzeichnis **43**

A Anhang **44**
A.1 Use Cases Detail 44
A.1.1 UC1: Service Case erfassen 44
A.1.2 UC2: Service Case bearbeiten 45
A.1.3 UC3: CRUD Fall 46
A.1.4 UC4: Maschine auslesen 47
A.1.5 UC5: Reporting auslesen 48
A.1.6 UC6: CRUD Attribut 49
A.1.7 UC7: CRUD Attribut-Set 50
A.1.8 UC8: CRUD Musterlösung 51
A.2 Konzept Detail 52
A.2.1 Attribut 52
A.2.2 Attribut-Set 52
A.2.3 Service Case 53
A.2.4 Metainformationen 53
A.2.5 Fall 54
A.2.6 Problem 54
A.2.7 Lösung 55
A.2.8 Lösungsvorschlag 55
A.2.9 Ersatzteil 56
A.2.10 Maschine 56
A.3 REST-Schnittstelle 57
A.3.1 Endpunkte 57
A.4 ShowCase-Szenarien 60
A.4.1 Vorbereitung auf Szenarien 60
A.4.2 Szenario 1: Lösungshinweise erhalten 60
A.4.3 Szenario 2: Reporting und Musterlösungen 61
A.4.4 Szenario 3: Aktuelle Situation im Feld 62
A.5 Applikation Screens als Benutzer 63
A.5.1 Legende der Buttons 63
A.5.2 Algorithmus wählen 64
A.5.3 Service Case Übersicht 64
A.5.4 Service Case bearbeiten 65
A.5.5 Attribute-Set hinzufügen 66
A.5.6 Attribute-Set bearbeiten 67
A.5.7 Fall bearbeiten 68

A.5.8 Suche . . . 69
A.5.9 Lösung bearbeiten . . . 70
A.5.10 Maschinen Übersicht . . . 71
A.5.11 Maschine ansehen . . . 71
A.5.12 Maschinen Fall . . . 72
A.6 Applikation Screens als Administrator . . . 73
A.6.1 Navigation Administrator . . . 73
A.6.2 Attribute Übersicht . . . 74
A.6.3 Attribute erstellen . . . 75
A.6.4 Attribute-Set Übersicht . . . 75
A.6.5 Attribute-Set erstellen . . . 76
A.6.6 Lösungsvorschläge Übersicht . . . 77
A.6.7 Problem-Lösung Mapping . . . 77
A.6.8 Reporting . . . 79
A.7 Projektmanagement . . . 80
A.7.1 Projektplan . . . 80
A.7.2 Meilensteine . . . 80
A.7.3 Gantt-Diagramm . . . 81
A.7.4 Arbeitsaufteilung . . . 82
A.7.5 Projektverlauf . . . 83
A.7.6 Risikoanalyse . . . 84
A.7.7 Risikomanagement . . . 84
A.7.8 Risikoanalyse der Elaboration-Phase . . . 87
A.7.9 Risikoanalyse nach der Elaboration-Phase . . . 87
A.8 Persönliche Berichte . . . 88
A.8.1 Noah Hendrikx . . . 88
A.8.2 Pascal Marty . . . 89

1 Einleitung

1.1 Was ist ein Service Case

Ein Service Case stellt den Einsatz eines Servicetechnikers dar. Ein Einsatz wird häufig ausgelöst durch einen Fehlerfall oder unerwünschtes Verhalten einer Maschine, welches durch einen Kunden gemeldet wird. Zur Planung des Einsatzes und zur Behebung des Problems spielen diverse Faktoren mit.

Einerseits spielen der Status und der Zustand der einzelnen Komponenten der Maschine eine Rolle. Beispielsweise: Ist Material X genügend aufgefüllt oder ist die Aufwärmphase lange genug eingestellt? Anhand dieser Informationen kann ein erfahrener Servicetechniker bereits abschätzen, was für die Lösung des Problems hilfreich sein könnte und wie der Fehler behoben werden kann.

Andererseits ist der Kunde möglicherweise weit entfernt vom Servicetechniker und es ist sinnvoll, weitere Wartungsarbeiten mit dem Einsatz zu verknüpfen. Somit ist es wichtig zu wissen, wann welche Komponente der Maschine zuletzt ausgetauscht oder gewartet wurde und welche Ersatzteile und Werkzeuge dafür benötigt werden. Anhand dieser Informationen kann viel Zeit und Aufwand gespart werden, da bevorstehende Wartungen frühzeitig erledigt und die relevanten Ersatzteile oder Spezialwerkzeuge vor dem Einsatz mit eingeplant werden können.

1.2 Stand der Technik

Aktuell werden Service Cases häufig von Hand auf Papier erfasst, oder es werden vorgefertigte elektronische Formulare ausgefüllt. Diese Formulare sind oftmals sehr spezifisch auf gewisse Maschinentypen und somit sehr eingeschränkt nutzbar. Diese Papiere oder Formulare werden danach firmenintern weitergereicht bis zur Fakturierung und Archivierung.

Im Archiv verschwinden somit beachtlich viele Daten und Informationen zu Maschinen, Problemlösungen und Kunden. Auch wird in der Realität selten auf diese zurückgegriffen.

Firmen wie grosse Autohersteller setzen mittlerweile intern auf Softwarelösungen, durch welche sich Wartungsarbeiten und Reparaturen ihrer produzierten Autos zurückverfolgen lassen. Zudem verfügen diese auch über umfangreiche Diagnose-Tools, welche Probleme identifizieren können und Lösungsvorschläge dazu zurückgeben. Jedoch sind diese oft sehr spezifisch und ausschliesslich auf ein bestimmtes Modell anwendbar.

1.3 Vision

Die Vision ist, eine Basisplattform für die strukturierte Erfassung von Service-Einsätzen für Servicetechniker zu liefern. Diese soll es Unternehmen ermöglichen, aufgrund von Lösungsvorschlägen Zeit und Aufwand bei der Problemlösung zu vermindern. Dadurch könnte möglicherweise der erste Service-Einsatz des Servicetechnikers entfallen, da die nötigen Ersatzteile bereits aus dem Lösungsvorschlag bekannt sind. Das System kann über einen Teile-Katalog automatischen eine Abrechnung der benötigten Ersatzteile erstellen. Die Basisplattform kann mit Hilfe einer Reporting-Funktion einfach gepflegt und gewartet werden. Die Anwendungsdaten werden zusammen mit Daten über das Kundenfeedback in einem Machine Learning Algorithmus verwertet und zur optimierten Lösungssuchung verwendet.

2 Problemanalyse

2.1 Service Case Merkmale

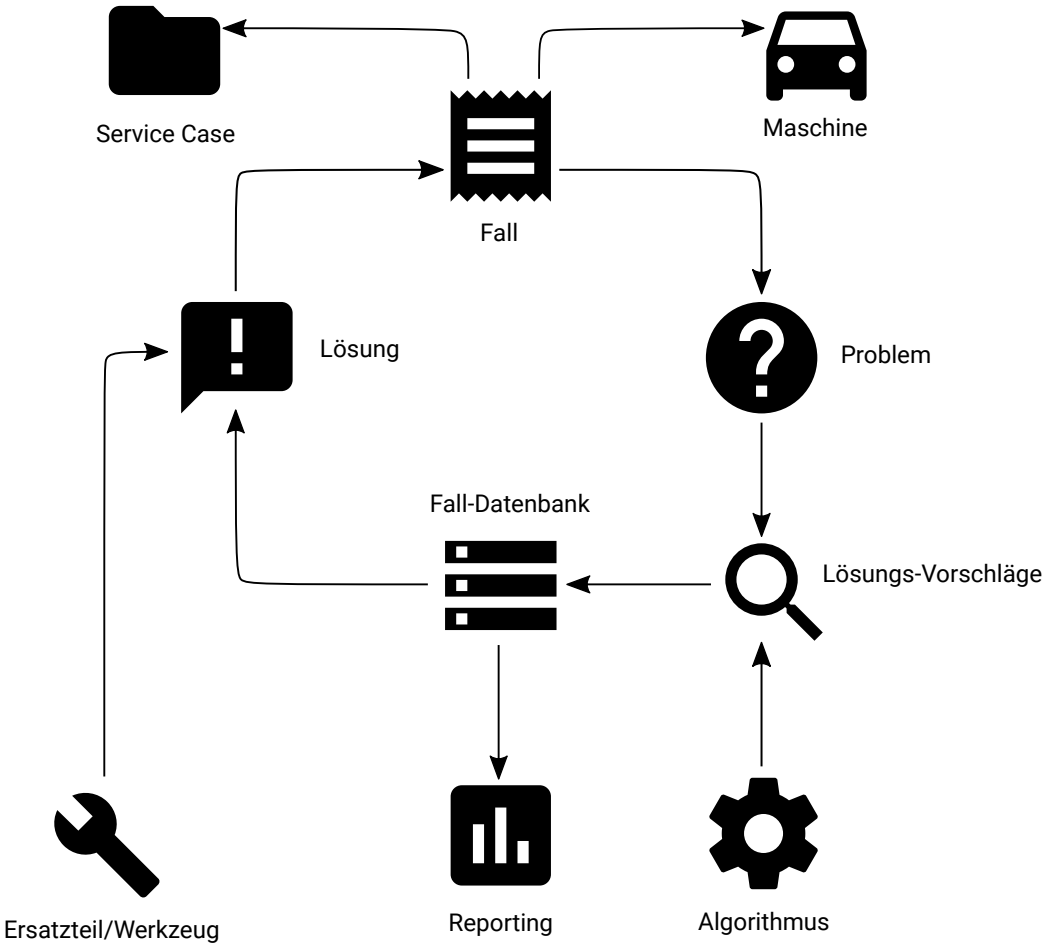


Abbildung 2.1: Übersicht der Merkmale

2.1.1 Service Case

Ein Service Case beschreibt einen Einsatz eines Servicetechnikers. Er umfasst Kundeninformationen und beinhaltet mindestens einen Fall. Zudem hat ein Service Case jeweils einen Status, beispielsweise “Offen“ oder “Geschlossen“.

2.1.2 Fall

Ein Fall umfasst alle Informationen die im Zusammenhang zu einem erkannten Defekt einer Maschine stehen. Wie der Service Case verfügt auch der Fall über einen Status.

2.1.3 Problem

Ein Problem umfasst alle Informationen, um ein Problem zu identifizieren wie beispielsweise eine Beschreibung.

2.1.4 Lösungs-Vorschläge

Nach Erfassung eines Problems können Lösungs-Vorschläge angefordert werden, welche anhand eines Algorithmus bestimmt werden. Die Vorschläge können vom Benutzer bewertet, als Lösung übernommen und gegebenenfalls angepasst werden.

2.1.5 Fall-Datenbank

Die Fall-Datenbank umfasst alle Fälle mit den damit verbundenen Informationen wie Problem, Lösung und die Service Case Informationen.

2.1.6 Reporting

Das Reporting bietet Auswertungen und Statistiken über die vorhandenen Daten wie die Anzahl der Fälle oder Service Cases.

2.1.7 Lösung

Eine Lösung besteht aus einer Lösungsbeschreibung im Freitext Format und eventuell aus Werkzeugen und Ersatzteilen, die dafür benötigt werden.

2.1.8 Maschine

Jeder Fall muss an eine Maschine gekoppelt sein, die es betrifft. Dadurch kann nachvollzogen werden, welche Änderungen an der Maschine durchgeführt wurden.

2.1.9 Algorithmus

Der Algorithmus stellt kein spezifisches Merkmal eines Service Cases an sich dar. Die Visualisierung dessen soll verdeutlichen, dass ein Algorithmus bei der Findung von Lösungs-Vorschlägen verwendet wird und das dieser austauschbar sein soll.

2.2 Eingangsdaten

Systeme welche anhand eingegebener Informationen Vorhersagen über mögliche Ausgangsdaten bestimmen müssen, sind stark abhängig von deren Eingangsdaten. Diese Daten können zudem äusserst flexibel in ihrer Repräsentation sein. Mögliche Arten von Eingangsdaten sind beispielsweise numerische Werte, Zeichensätze, Aufzählungstypen, logische Werte und Bilder, welche voneinander abhängig oder unabhängig sein können. Diese Daten werden möglicherweise von Maschinen oder Sensoren automatisch erzeugt oder durch einen Menschen mit einer gewissen Ungenauigkeit und Streuung erfasst. Diese Variabilität führt zu einer erhöhten Komplexität und Vielschichtigkeit dieser Systeme.

2.3 Ansätze Problembeschreibung

Nachfolgend werden Ansätze erläutert, die berücksichtigt wurden um ein Problem zu beschreiben und dieses gegenüber anderen auf Ähnlichkeit hin zu überprüfen. Als Inspiration dienten Verfahren, welche aus der "Case Based Reasoning"-Methodologie entwickelt wurden.

2.3.1 Attribut-Ähnlichkeit

Eine Möglichkeit um Probleme auf ihre Ähnlichkeit hin zu überprüfen besteht darin, diese mit Attributen zu versehen und deren Werte zu vergleichen [11]. Dafür wird für jedes einzelne Attribut eine Ähnlichkeits-Funktion definiert, welche zwei Werte des selben Attributs annimmt und einen Wert zwischen 0 (für überhaupt nicht ähnlich) und 1 (für genau gleich) ausgibt. Zusätzlich kann einem Attribut eine Gewichtung in Form einer Zahl mitgegeben werden, welche bestimmt, wie wichtig das Attribut im Vergleich zu anderen Attributen ist. [12]

Um zu überprüfen, ob ein neuer Fall ähnlich ist, wird nun für jeden bestehenden Fall, die Ähnlichkeits-Funktion jedes Attributs mit dem Wert des neuen Falles aufgerufen. Diese Werte werden pro Fall addiert und durch die Anzahl vergebener Gewichtungen geteilt. So ergibt sich pro Fall ein Wert zwischen 0 und 1, welcher aussagt, wie ähnlich diese Fälle sind.

Pro	**Contra**
Die Vergleiche sind eindeutig und lassen sich ganz klar ordnen.	Für jedes Attribut muss eine Funktion und Gewichtung definiert werden, was nicht immer trivial ist.
Es kann mit nicht ausgefüllten Attribut-Werten gearbeitet werden.	Bei Eingabefehlern von Werten die sich von den restlichen signifikant unterscheiden, kann die ganze Wertung durcheinander geraten.

Tabelle 2.1: Attribut-Ähnlichkeit Pro & Contra

2.3.2 Textanalyse

Denkbar ist ein Ansatz der reinen Textrepräsentation und Textanalyse [13]. Das Problem und die Lösung sind ausschliesslich Freitexte, erfasst durch die Servicetechniker. Das System soll demnach durch Textanalyse selbst herausfinden, welches die für die Lösung erforderlichen Ersatzteile sind, oder welche Werkzeuge dafür verwendet werden.
Als erste Herausforderungen stellen sich dabei unterschiedliche Schreibweisen, Schreibfehler und Sprachen heraus. Zudem wird initial eine grosse Datenmenge benötigt um das System zu trainieren. Ausserdem schränkt dieser Ansatz die Möglichkeit einer proaktiven Unterstützung durch die Applikation eher ein, da er lediglich dazu aufgefordert wird, einen Freitext einzugeben.

2.3.3 Entscheidungsbaum

Ein weiterer Ansatz ist das Verwenden eines Entscheidungsbaumes [14]. Durch Erfragen von bestimmten Attribut-Werten in der richtigen Reihenfolge, werden nicht relevante Fälle schnell aussortiert. Ausserdem kann der Benutzer einfach geführt werden und hat immer eine klare Anweisung, was als nächstes zu tun ist.

Die Schwierigkeit besteht darin herauszufinden, welche Fragen in welcher Reihenfolge gestellt werden sollen. Zudem können neue Fälle den Entscheidungsbaum stark beeinflussen, was sich bei manueller Pflege dessen als äusserst komplex herausstellt.

Pro	**Contra**
Sehr benutzerfreundlich und für eine Mobile App optimales Bedienungskonzept möglich	Generierung des Entscheidungsbaums sehr komplex

Tabelle 2.2: Entscheidungsbaum Pro & Contra

2.3.4 Fazit Problembeschreibung

Der Entscheid fiel auf den Attributs-Ähnlichkeits Ansatz, da dieser im Vergleich zur Textanalyse strukturierter erfassbar ist und mit weniger Daten bereits nützliche Lösungs-Vorschläge finden kann. Die Umsetzung eines allgemein gültigen Entscheidungsbaumes ist zudem sehr komplex, da nicht nur die Fragen, sondern auch deren Reihenfolge entscheidend sind. Zudem kann durch einen Entscheidungsbaum nur jeweils eine Frage pro Schritt beantwortet werden, während das Ausfüllen von Attribut-Werten parallel in einem Schritt gemacht werden kann. Zudem ist für den Servicetechniker schnell ersichtlich, welche Werte gewählt wurden und es ermöglicht ihm, diese im Nachhinein anzupassen.

2.4 Attribut Typen

Zur Auswahl stehen eine Reihe verschiedener Attributs Typen. Einige lassen bereits mit wenigen Werten eine Differenzierung zu, andere benötigen eine grosse Menge an Daten um als Unterscheidungsmerkmal zu dienen.

Boolesch Beispiel: Heizung AN/AUS Der Vergleich kann nur eine komplette Übereinstimmung ergeben oder gar keine.

Numerisch Beispiel: Temperatur 36

Text Beispiel: Kommentar "Das Zahnrad X klemmt"

Option Beispiel: Fehlercode E02/E05/E99

Typ	**Werte-Vergleich**	**Voraussetzung**
Boolean	Nur 1 oder 0 möglich	Bei vielen Lösungsmöglichkeiten werden auch viele Boolean Attribute nötig zur Unterscheidung
Numerisch	Ähnlichkeit & Distanz	Wenig Daten bereits nützlich
String	Häufig verwendet wird die Levenshtein Distanz: Vergleicht zwei Strings anhand der Operationen *insertion, deletion & modification*. Je weniger Änderungen nötig sind um den gleichen String zu bekommen, desto ähnlicher sind die Strings	Je mehr Varianten von Strings möglich sind, desto mehr Daten werden benötigt um eine Ähnlichkeit zu berechnen.
Option	Bei eindeutigen und nicht verschachtelten Optionen ist der Vergleich simpel. Trifft zu oder nicht. Bei verschachtelten Optionen steigt die Ähnlichkeit an, bis zum vollständigen Treffer im Blattknoten	Je mehr Varianten möglich sind, desto mehr Daten werden benötigt. Im Gegensatz zum String, sind die Optionen weniger von subjektivem Rauschen betroffen, da man sich für eine Auswahl entscheiden muss.
Freitext	Je nach Filtrierung des Freitextes bleiben relevante Wörter übrig und Schreibfehler werden bereinigt. Übereinstimmende Wörter ergeben dann den Ähnlichkeitswert	Trainierter Filter für die Problem Domäne. Komplexe Konfigurierung um den Text zu bereinigen.

Tabelle 2.3: Attribut Typen

2.5 Form der Lösungsbeschreibung

Folgende Teile sind relevant um eine Lösung zu beschreiben.

2.5.1 Beschreibung

Eine Beschreibung der Lösung in textueller Form, welche alle Schritte umfasst um das damit verknüpfte Problem zu beheben.

2.5.2 Werkzeug/Ersatzteile

Aufzählung der Werkzeuge, besonders Spezialwerkzeuge sind entscheidend für die Einsatzplanung. Ebenfalls aus den gleichen Gründen ist es wichtig, dass die benötigten Ersatzteile mitgeführt werden.

2.6 Datenquelle für Lösungsvorschläge

Um Lösungsvorschläge auf eine Problembeschreibung zurück geben zu können sind unter anderem folgende Datenquellen denkbar.

2.6.1 Feld Daten

Durch das Erfassen von Problemen und deren Lösungen, fliessen automatisch Daten in das System, welche bei ähnlichen Problemen als Lösungsvorschläge dienen können. Diese sind immer zwingend mit einem Fall verknüpft.

2.6.2 Musterlösung

Um ein neues System oder einen neuen Maschinentyp im System zu initialisieren oder eine erarbeitete Lösung in das System einzugeben, braucht es eine Möglichkeit um Lösungsvorschläge, bestehend aus Problem und Lösung, ohne einen konkreten Fall in das System eingeben zu können. Diese bezeichnen wir als Musterlösung.

2.7 Use Cases

2.7.1 Use Case Diagram

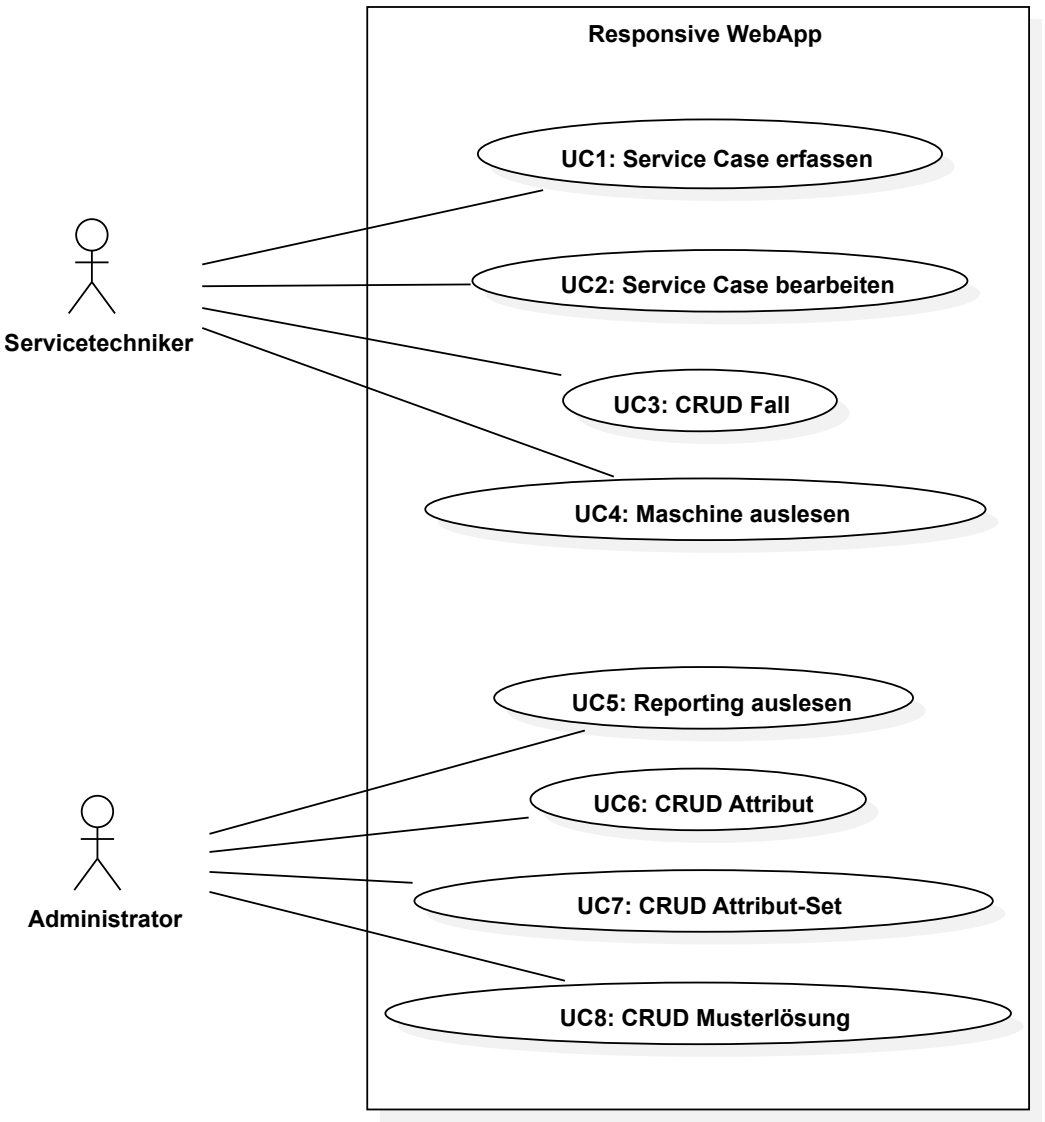


Abbildung 2.2: Use Case Diagram

2.7.2 Aktoren & Stakeholder

Servicetechniker Der Servicetechniker ist der Hauptbenutzer der Anwendung. Er verwendet die Grundfunktionalität der Applikation beim Erfassen und Bearbeiten seiner Service Cases.

Administrator Der Administrator ist für die Wartung, Pflege und Optimierung der Anwendung verantwortlich. Seine Haupttätigkeit besteht darin mithilfe der Reporting-Funktion Anpassungen an an den Attributen und Musterlösungen zu machen.

2.7.3 Kurzbeschreibung der Use Cases [1]

UC1: Service Case erfassen Ein Servicetechniker erfasst einen neuen Service Case.

UC2: Service Case bearbeiten Ein Servicetechniker bearbeitet einen erfassten Service Case.

UC3: CRUD Fall Ein Servicetechniker erstellt, bearbeitet oder löscht einen Fall.

UC4: Maschine auslesen Ein Servicetechniker sieht sich den aktuellen Zustand einer Maschine an.

UC5: Reporting auslesen Ein Administrator sieht sich einen Report über das System an.

UC6: CRUD Attribut Ein Administrator erstellt, bearbeitet oder löscht Attribute.

UC7: CRUD Attribut-Set Ein Administrator erstellt, bearbeitet oder löscht Attribut-Sets.

UC8: CRUD Musterlösung Ein Administrator erstellt, bearbeitet oder löscht Musterlösungen.

[1] Eine detailliertere Ausführung der Use Cases ist im Anhang zu finden. A.1

2.8 Domainanalyse

2.8.1 Domain Model

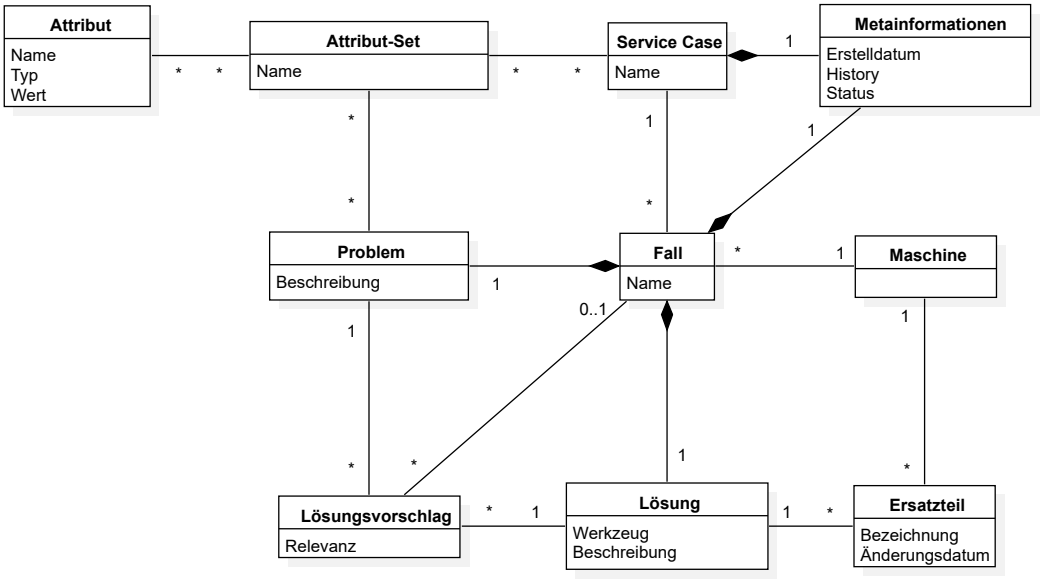


Abbildung 2.3: Domainmodel

2.8.2 Kurzbeschreibung der Konzepte [2]

Attribut Attribute enthalten Werte, welche zur Suche und Vergleich von Lösungsvorschlägen verwendet werden.

Attribut-Set Ein Attribut-Set repräsentiert eine Sammlung von Attributen. Sie können einem Service Case oder einem Problem zugeordnet werden.

Service Case Ein Service Case wird zur Erfassung von Fällen erstellt. Zusätzlich werden Metainformationen erfasst.

Metainformationen Diese enthalten zusätzliche Informationen über einen Service Case oder einen Fall.

[2] Eine detailliertere Ausführung der Konzepte ist im Anhang zu finden. A.2

Fall Ein Fall ist die Aufnahme eines Problems, für welche eine Lösung gefunden werden möchte. Ein Fall besteht aus dem Problem, möglichen Lösungsvorschlägen und der Lösung. Zudem werden Metainformationen über den Fall erstellt.

Problem Ein Problem ist ein Fehlverhalten oder Defekt, welcher behoben werden möchte.

Lösung Eine Lösung schildert die Behebung eines konkreten Problems.

Lösungsvorschlag Zu einem Fall können mehrere Lösungsvorschläge gefunden werden. Lösungen können über ein Problem als Lösungsvorschlag gefunden werden.

Ersatzteil Ein Ersatzteil ist eine Komponente einer Maschine und kann in den Lösungen enthalten sein, um die Maschine zu verändern.

Maschine Eine Maschine enthält ihren aktuellen Zustand.

2.9 Weitere Anforderungen

Die Mehrheit der funktionalen Anforderungen werden in den Use Cases behandelt. Folgend werden weitere Anforderungen aufgeführt, welche noch nicht anderswo abgedeckt wurden.

Nr.	Anforderung
R1	Die Anwendung verfügt über Szenarien für einen Showcase zu Demonstrationszwecken.
R2	Der Suchalgorithmus für die Findung von Lösungsvorschlägen kann ausgetauscht werden.

Tabelle 2.4: Weitere Anforderungen

2.9.1 Showcase-Szenarien [3]

Die Showcase-Szenarien orientieren sich an möglichen Problemen und Lösungen eines 3D-Druckers "Ultimaker 2". Für den Showcase wurden zwei Lenovo Phab 2 Pro Smartphones (kurz Phab1 und Phab2) verwendet.

Szenario 1: "Lösungshinweise erhalten"

1. Maschine hat ein Problem "Y"
2. Servicetechniker 1 erfasst das Problem "X" auf Phab1
3. Servicetechniker 1 erhält Lösungsvorschläge L1-L3 zur Behebung auf Phab1
4. Das Problem wird an der Maschine jedoch anders gelöst.
5. Servicetechniker 1 erfasst eine neue Lösung L4 zu Problem "X" auf Phab1
6. Maschine hat erneut das Problem "X"
7. Servicetechniker 2 erfasst das Problem "X" auf Phab2
8. Servicetechniker 2 erhält Lösungsvorschläge L1-L4 zur Behebung auf Phab2
9. Servicetechniker 2 wählt L4 zur Behebung des Problems

Szenario 2: "Reporting und Musterlösungen"

1. Servicetechniker 1 erfasst ein Problem "X" auf Phab1
2. Servicetechniker 1 erhält eine grosse Anzahl Lösungsvorschläge auf Phab1
3. Administrator lässt sich ein Reporting ausgeben
4. Administrator erkennt Bedarf einer Klärung aus dem Reporting
5. Administrator erstellt eine Musterlösung
6. Administrator fügt weitere Attribute zur Erhebung hinzu
7. Servicetechniker 2 erfasst nun Problem "X" auf Phab2
8. Servicetechniker 2 erhält weniger Lösungsvorschläge und die Musterlösung
9. Administrator lässt sich ein Reporting ausgeben
10. Administrator erkennt eine verbesserte Situation

[3] Eine detaillierte Beschreibung der Showcase-Szenarien findet sich im Anhang A.4

Szenario 3: "Aktuelle Situation im Feld"

1. Maschine hat ein Problem "X"
2. Servicetechniker 1 lässt sich die BOM der Maschine anzeigen
3. Servicetechniker 1 erfasst das Problem "X" auf Phab1
4. Servicetechniker 1 erhält Lösungsvorschläge L1-L3 zur Behebung auf Phab1
5. Servicetechniker 1 lässt sich die dazu notwendigen Ersatzteile anzeigen
6. Servicetechniker 1 wählt einen Vorschlag, aber ändert die Ersatzteile ab
7. Maschine erhält ein Update "U"
8. Servicetechniker 2 lässt sich die BOM der Maschine anzeigen
9. Servicetechniker 2 entdeckt die Änderungen durch Servicetechniker 1
10. Servicetechniker 2 erfasst eine Modifikation "M" auf Phab1

2.10 Nichtfunktionale Anforderungen

2.10.1 Qualitätsmerkmale

Erweiterbarkeit Die Flexibilität und Erweiterbarkeit der System- und Softwarearchitektur um neue Funktionalität ist für zukünftige Industrie-Anwendungen gewährleistet.

Kompatibilität Die Anwendung ist mit Google Chrome als Browser kompatibel.

Änderungssensitivität Die Anwendung verfügt über ein einfaches Responsive-Design, sodass die Verwendung auf Web- und Mobil-Geräten ermöglicht wird.

2.10.2 Referenzsystem

Lenovo Phab 2 Pro mit Google Chrome Version 58.0.3029.83
Lenovo ThinkPad S540 mit Google Chrome Version 58.0.3029.110

2.11 Abdeckung der Anforderungen

Zur Überprüfung der Applikationsfunktionalität dienen die Showcase-Szenarien 3. Mit folgender Tabelle wird gezeigt, welche Anforderungen und Use Cases durch welches Showcase-Szenario abgedeckt werden.

Use Cases	**Showcase**
UC1, UC3	Szenario 1
UC5, UC6, UC7, UC8	Szenario 2
UC1, UC3, UC4, UC2	Szenario 3

Tabelle 2.5: Anforderungs-Abdeckung

3 Konzeption und Design

3.1 Software-Architektur

3.1.1 Client

Für die Realisierung einer mobilen Applikation wurden folgende drei Möglichkeiten berücksichtigt:

Responsive Website Eine Webseite, welche die angeforderten HTML-Seiten und deren Inhalt serverseitig generiert und ausliefert.

Vorteile	**Nachteile**
Entlastung des mobilen Geräts, da die Seite fertig ausgeliefert wird und nur noch angezeigt werden muss	Höhere Belastung des Servers durch Generierung der angeforderten Ressourcen
	Grosses Datenvolumen, da jede Seite einzeln übertragen wird
	Hohe Kopplung des Client an das Backend

Tabelle 3.1: Responsive Website Vor- & Nachteile

Native/Cross-Platform Mobile App Für jede zu unterstützende Plattform eine eigene Applikation, oder eine Cross-Platform Applikation, welche jeweils die entsprechenden Bedienelemente der Plattform verwendet.

Vorteile	Nachteile
Verwendung von bekannten Bedienelementen vereinfacht den Umgang für Plattform Benutzer	Hoher Mehraufwand durch Implementation pro Plattform
Nutzung von Geräte-Sensoren wie Bluetooth, Gyroskop, GPS, etc.	Aufwändige Entwicklung von Bedienelementen, die nicht von der Plattform vorgesehen sind

Tabelle 3.2: Native/CrossPlatform Mobile App Vor- & Nachteile

Responsive Web App Framework Eine Web Applikation, welche im Browser ausgeführt wird.

Vorteile	Nachteile
Voraussetzung ist lediglich die relevanten Browser zu unterstützen, welche auf einer grossen Anzahl Plattformen verfügbar sind	Hohe Abhängigkeit von unterstützten Browser Features
Keine Installation notwendig	Weniger auslesbare Daten von Gerätekomponenten
Datenvolumen für die Kommunikation sehr gering	

Tabelle 3.3: Responsive Web App Framework Vor- & Nachteile

Fazit Client Eine responsive Web App mittels eines Frameworks stellt die flexibelste und am einfachsten erweiterbare Lösung dar. Dadurch wird eine klare Trennung von Backend und Client erreicht, was mit einer einfachen responsive Website mit generierten Ansichten nicht erreicht wird. Gegenüber einer nativen oder cross-platform App können Sensoren aus Mobilgeräten schlechter verwendet werden, was jedoch für diese Arbeit nicht im Fokus steht. Zudem kann eine Web App mit relativ geringem Aufwand für Desktop und Mobilgeräte nutzbar gemacht werden, was die Wiederverwendbarkeit erhöht.

3.1.2 Backend

Durch die Wahl des Clients als Responsive Web App, fiel der Entscheid eine REST-Schnittstelle [9] als Kommunikationsmittel zu verwenden. Diese ermöglicht es für einen zukünftigen Ausbau der Plattform eine Vielzahl unterschiedlicher Clients einzusetzen, solange diese das HTTP-Protokoll unterstützen.

3.1.3 Persistierung

Für die Persistierungs-Schicht wurden dokumentenbasierte und relationale Datenbanken in Betracht gezogen. Im Hinblick auf Erweiterbarkeit und Flexibilität, welche eine Grundanforderung an die Plattform stellt, fiel die Entscheidung auf eine relationale Datenbank. Zudem bietet diese umfangreichere Abfragemöglichkeiten bezüglich Objekt übergreifender Abfragen im Gegensatz zu dokumentenbasierten Datenbanken.

3.2 Technologien

Nachfolgend finden sich die Begründungen für die eingesetzten Technologien.

3.2.1 Web App Framework

Unter Berücksichtigung von Angular [1] und VueJS [15] fiel die Entscheidung zugunsten von VueJS als Framework für die Web App aus. Die Komplexität von VueJS ist geringer als die von Angular. Dies erlaubt es die Applikations-Architektur freier zu wählen, jedoch sind gewisse Grundfunktionalitäten wie eine Zustandsverwaltung clientseitig nicht direkt integriert, wie es bei Angular der Fall ist. Andererseits können benötigte Grundfunktionalitäten einfach integriert werden, was VueJS schlanker und schneller macht. Infolge dessen gestaltet sich die Lernkurve zur Anwendung von VueJS flacher und für die Realisierung eines Prototypen geeigneter.

Abbildung 3.1: VueJS Logo

3.2.2 REST-Schnittstelle

Die REST-Schnittstelle wurde mittels ASP.NET Core [3] Framework umgesetzt. Dieses wird mit C# implementiert und bietet alle Vorteile des .NET Frameworks. Zudem lässt sich das Projekt unter Windows, MacOS und Linux verwenden, was der angestrebten Flexibilität der Basisplattform entspricht. Zudem ist das Framework komplett Open Source und wird von Microsoft unterstützt und laufend erweitert.

3.2.3 Relationale Datenbank

Als Technologie für die relationale Datenbank fiel die Entscheidung auf Microsoft SQL. Dies erlaubt den Einsatz des Entity Framework Core [8] als OR-Mapper, welches die Kommunikation zur Datenbank kapselt. Das vereinfacht die Persistierung und Verwaltung der Daten, da die Generierung des Datenbank-Schemas mit einem "Code-First"-Ansatz realisiert werden kann.

3.2.4 Datei Speicher

Um Dateien wie Bilder speichern zu können wurde Azure Storage verwendet. Dies erlaubt die Verwaltung von Dateien über eine eigene REST-Schnittstelle. Microsoft bietet Bibliotheken an, welche die Kommunikation für eine Vielzahl von Programmiersprachen kapselt. Zudem kann ein Storage der Azure Cloud lokal für die Entwicklung emuliert werden mittels Azure Storage Explorer.

3.3 UI/UX-Konzept

Folgend werden die Navigationskonzepte der Anwendung aufgezeigt. Eine detailliertere Anleitung der Navigation und Funktionen innerhalb der Applikation findet man im Anhang unter A.5.

3.3.1 Navigationskonzept als Benutzer

Beim Benutzer dieser Applikation handelt es sich überwiegend um Servicetechniker. Um dem Servicetechniker eine schnelle und unkomplizierte Bedienung für den Service-Einsatz zu ermöglichen, wurde dabei auf eine möglichst flache Strukturierung geachtet. Bei den verschiedenen Seiten wurde beachtet, dass nicht zu viele Informationen auf einmal angezeigt werden, damit die Bearbeitung auf einem Mobilgerät vereinfacht wird.

Die Abbildung 3.2 zeigt eine Übersicht der wichtigsten Navigationsmöglichkeiten als Benutzer innerhalb der Applikation.

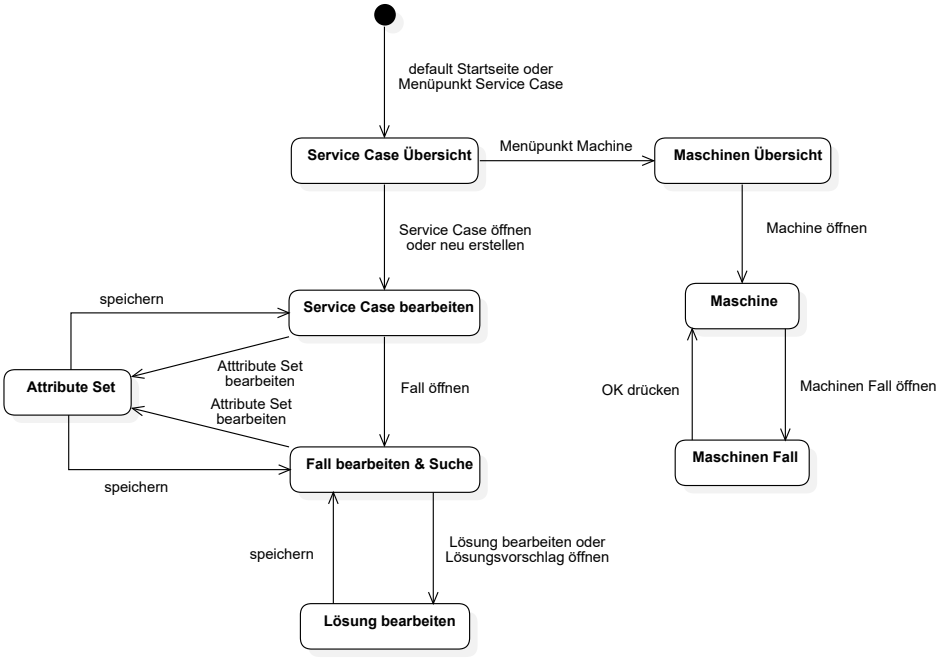


Abbildung 3.2: Navigations-Diagramm Benutzer

3.3.2 Navigation als Administrator

Der Administrator übernimmt Funktionen, welche nicht während eines Service-Einsatzes erledigt werden müssen. Daher eignet sich die Bedienung dieser Ansichten eher an einem Desktop-Computer als auf einem Mobilgerät. Bei der Navigation wurde die Bedienung an einem Mobilgerät eher vernachlässigt. Die Strukturierung ist flach gehalten und die Seiten werden auf kurzem Wege erreicht und beinhalten meistens eine Übersicht mit vielen Informationen.

Die Abbildung 3.3 zeigt eine Übersicht der wichtigsten Navigationsmöglichkeiten als Administrator innerhalb der Applikation.

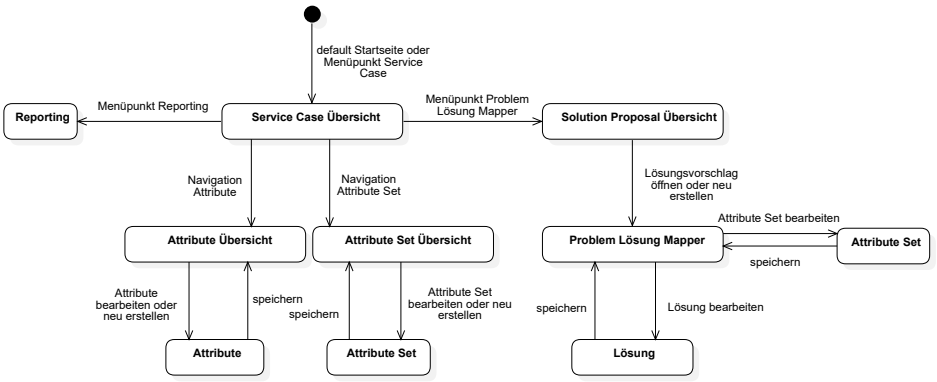


Abbildung 3.3: Navigations-Diagramm Administrator

4 Ergebnisse

4.1 Deployment

4.1.1 Deployment Diagram

Die Abbildung 4.1 zeigt die Komponenten der Applikation und mit welchen Protokollen diese untereinander kommunizieren.

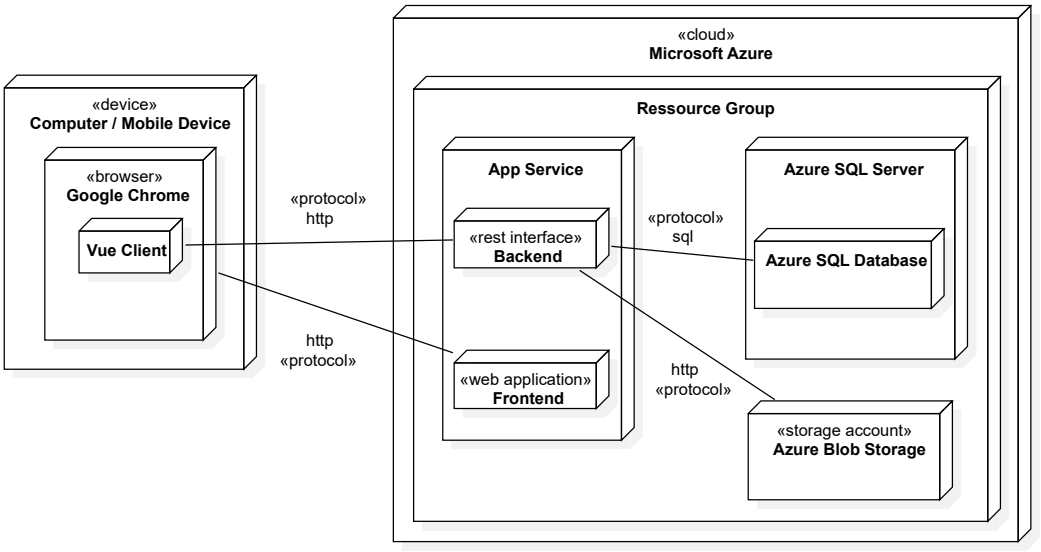


Abbildung 4.1: Deployment Diagram

4.1.2 Azure

Alle Komponenten der Applikation sind auf Azure gehosted. Dadurch wird eine hohe Skalierbarkeit und Flexibilität gewährleistet, da neue Instanzen einfach erstellt und vorhandene neu dimensioniert werden können. Die verwendeten Services von Azure umfassen einen App Service, einen Azure SQL-Server und einen Azure Blob Storage (siehe 4.1). Für eine zukünftige Weiterentwicklung kann somit der Azure Machine Learning Service einfach in die Applikationslogik integriert werden um beispielsweise den Algorithmus zur Lösungsfindung zu verbessern.

App Service Die App Service Komponente [4] erfüllt zwei Funktionen. Einerseits liefert sie den Vue Client an den Browser des Benutzers aus. Dieser beinhaltet alles was der Endbenutzer benötigt, um mit der Applikation zu kommunizieren.
Andererseits beherbergt sie die REST-Schnittstelle, über welche die Kommunikation zwischen Client und Server stattfindet. Sobald der Client an den Endbenutzer ausgeliefert wurde, werden lediglich Http-Anfragen im JSON-Format zwischen Backend und Client ausgetauscht.

Azure SQL-Server Der Azure SQL-Server [6] beherbergt die SQL Datenbank, auf welcher sich alle Applikationsdaten befinden, ausser den Dateien wie Bilder. Die Datenbank kommuniziert ausschliesslich innerhalb der Azure Cloud mit der REST-Schnittstelle. Somit ist die Verbindung zwischen Backend und der Datenbank äusserst schnell.

Azure Blob Storage Dateien und Bilder werden über die REST-Schnittstelle direkt auf den Azure Blob Storage [5] hochgeladen. Somit muss für einen Schreib-Zugriff auf den Blob Storage lediglich das Backend autorisiert sein und nicht jeder Benutzer individuell.

4.2 Code-Schnittstellen

Nachfolgend werden die wichtigsten Schnittstellen des Backends erläutert. Einerseits soll zur Laufzeit der Algorithmus, welcher die Lösungsvorschläge für ein bestimmtes Problem berechnet, ausgetauscht werden können. Andererseits soll es für die Weiterentwicklung möglich sein, weitere Implementationen des Algorithmus hinzuzufügen.

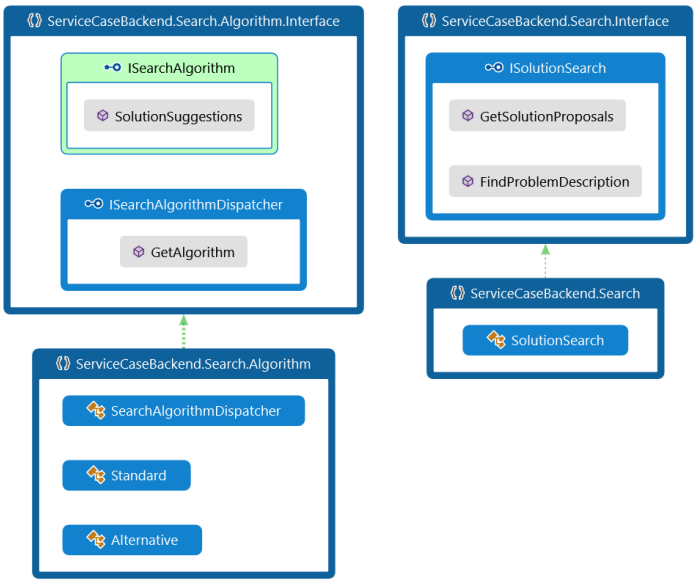


Abbildung 4.2: Search Interfaces

4.2.1 SolutionSearch Interface

Das ISolutionSearch Interface besteht aus zwei Funktionen, welche für die Vorschläge von ähnlichen Problembeschreibungen während der Problemerfassung und für die Lösungs-Vorschläge zuständig sind.

FindProblemDescription Die Implementation dieser Funktion nimmt einen Text entgegen und gibt eine Liste von ähnlichen Problembeschreibungen zurück.

GetSolutionProposals Diese Funktion nimmt einen IssueIdentifier entgegen und gibt eine Liste von Lösungsvorschlägen zurück, welche auf das angegebene Issue passen könnte. Die Idee ist, dass diese Funktion die relevanten Daten des Issues zusammenträgt und diese danach dem Algorithmus zur Lösungsfindung übergibt.

4.2.2 SearchAlgorithmDispatcher Interface

Hier wird zur Laufzeit entschieden, welcher Suchalgorithmus zur Lösungssuche angewandt wird.

GetAlgorithm Die Aufgabe dieser Funktion ist das Bestimmen, Instanzieren und Zurückgeben des korrekten Suchalgorithmus.

4.2.3 SearchAlgorithm Interface

Jede neue Implementation dieses Interfaces enthält die Logik, wie Lösungsvorschläge ermittelt werden. Wichtig ist, dass eine neue Implementation auch im SearchAlgorithmDispatcher erfasst wird, damit diese zur Laufzeit zurückgegeben werden kann.

SolutionSuggestions Diese Methode nimmt den IssueIdentifier, den Problem-Beschreibungs-Text und eine Liste von AttributSetItems entgegen und gibt die passenden Lösungsvorschläge zurück.

4.2.4 Client-Backend

Der Webclient interagiert mit dem Backend der Applikation ausschliesslich über eine REST-Schnittstelle, welche mittels HTTP-Protokoll kommuniziert. Die Informationen werden über die URL oder als JSON-Format in den HTTP-Requests versendet.

Konfiguration In der Datei "main.js" im Webclient kann die URL, über welche das Backend erreichbar ist, konfiguriert werden.

4.2.5 Backend-Azure SQL Database

Das Backend interagiert mittels Entity Framework Core mit der Azure SQL Database. Das Entity Framework selber verwendet das SQL-Protokoll, um die Transaktionen und Abfragen in SQL zu machen.

Konfiguration Azure Im App Service auf dem Azure Portal findet sich unter "Anwendungseinstellungen" der Abschnitt "Verbindungszeichenfolgen" (ConnectionStrings). Der Eintrag "DefaultConnection" definiert den Zugriff

auf die entsprechende SQL-Datenbank. Somit müssen im Sourcecode keine sensitiven Daten gespeichert werden.

Konfiguration lokale Entwicklungsumgebung In der Datei "appsettings.json" unter "ConnectionStrings" wird über den "DefaultConnection"-Eintrag die Verbindung zur SQL-Datenbank definiert.

Konfiguration Testing Für das Testing wird innerhalb der Klasse "TestFixture" eine In-Memory SQLite Datenbank erstellt. Diese wird pro Test-Fall neu erstellt.

4.2.6 Backend-Azure Blob Storage

Dateien und Bilder sendet das Backend über das HTTP-Protokoll an den Azure Blob Storage. Die Kommunikation wird im Code über die WindowsAzure.Storage Library gekapselt.

Konfiguration Azure Wie die SQL-Datenbank wird auch die Verbindung des Blob Storage mittels einer Umgebungsvariable im App Service definiert. Der Eintrag "AzureStorag" definiert die Verbindung zum entsprechenden Azure Blob Storage. Somit müssen im Sourcecode keine sensitiven Daten gespeichert werden.

Konfiguration lokale Entwicklungsumgebung In der Datei "appsettings.json" unter "ConnectionStrings" wird über den "AzureStorage"-Eintrag die Verbindung zum Blob Storage definiert. Für eine lokale Entwicklung ist es von Vorteil den Azure Storage Explorer und den dazugehörigen Azure Storage Emulator zu installieren. Dieser erlaubt es auf eine lokale, emulierte Instanz eines Azure Storage Kontos zuzugreifen. Beim Wechsel auf eine Release Umgebung, kann lediglich der Connection String angepasst werden und somit mit dem realen Azure Storage interagiert werden.

4.3 REST-Schnittstelle

Eine Auflistung der verwendeten REST-Schnittstellen Endpunkte ist im Anhang zu finden A.3.1. Zudem wird eine komplette Beschreibung der Endpunkte als interaktive HTML-Seite mittels Swagger generiert. Nachfolgend findet man eine Übersicht über die verwendeten DTOs.

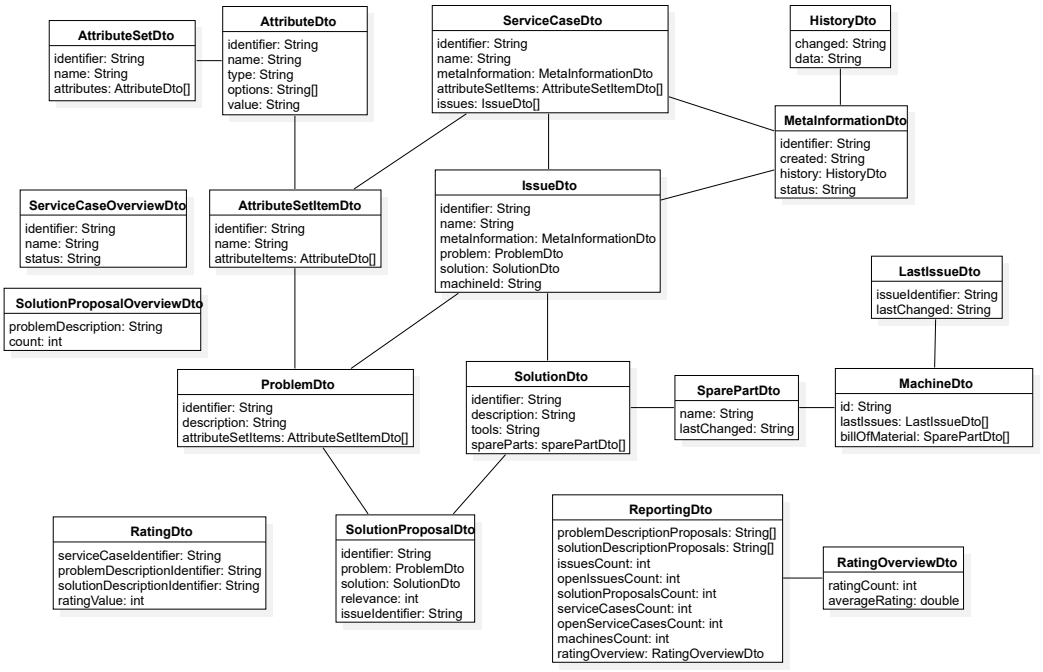


Abbildung 4.3: Dto-Diagramm

4.4 Code Dokumentation

4.4.1 Verwendete NuGet Bibliotheken

Microsoft.AspNetCore Das Framework wird verwendet um mittels Controllern die WebAPI (REST-Schnittstelle) bereit zu stellen. Es bietet die Grundlage für ein schlankes Backend und umfasst bereits nützliche Dependency Injection Mechanismen. Zusätzlich kann das Backend einfach auch für UNIX basierte Systeme kompiliert werden.

Microsoft.EntityFrameworkCore Die Verwaltung der Persistierung übernimmt das Entity Framework Core . Es kapselt die Kommunikation mit dem

SQL-Server und übernimmt wichtige Funktionen wie die Schlüsselverwaltung und die Verarbeitung von Transaktionen.

Newtonsoft.Json Die Standard-Bibliothek für die De-/Serialisierung von JSON-Objekten im .NET Framework [10]. Diese hochperformante Bibliothek erlaubt es auf einfache Weise JSON-DTOs vom Client in C#-Objekte umzuwandeln und diese mittels LINQ-Abfragen auszulesen.

Carable.Swashbuckle.AspNetCore.DocumentWithCode Ein Open Source Projekt, welches es erlaubt, Swagger Dokumente für REST-Schnittstellen aus ASP.Net Core zu generieren. Diese wird in der Dokumentation zu ASP.NET Core [2] durch Microsoft empfohlen.

WindowsAzure.Storage Die Kommunikation mit dem Azure Storage wird durch die Verwendung dieser Bibliothek gekapselt.

NinjaNye.SearchExtenstions Diese Bibliothek erweitert die Abfragemöglichkeiten vom Entity Framework um String Vergleiche. Speziell bei der Suche nach ähnlichen Problembeschreibungen unterstützt diese Möglichkeit entscheidend [7].

4.4.2 Kommunikation: Genereller Ablauf

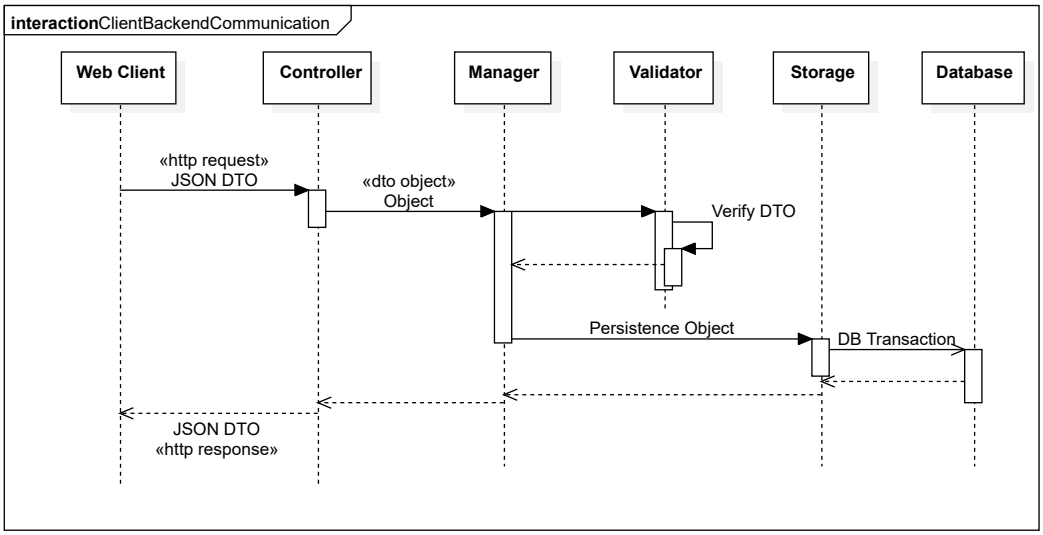


Abbildung 4.4: Client - REST Kommunikation

4.4.3 Lösungsvorschlag Algorithmus

In dieser Arbeit wurden zwei Algorithmen implementiert um sicher zu stellen, dass der Wechsel des Algorithmus zur Laufzeit funktioniert. Wie in 4.2.3 beschrieben, bestehen die Input-Daten zur Berechnung von Lösungsvorschlägen aus einer ID des Falls, welcher Lösungsvorschläge benötigt, der Problembeschreibung als Text und einer Liste von Attribut-Sets, welche alle Attribute aus dem Fall und aus dem Service Case enthält. Nachfolgend finden sich die Erläuterungen zur Berechnung der Vorschläge.

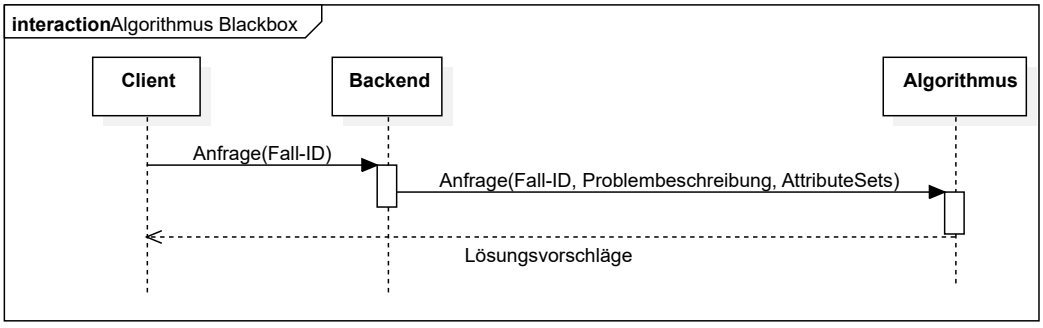


Abbildung 4.5: Algorithmus Ablauf Blackbox

Standard-Algorithmus Der Standard-Algorithmus sucht in den Musterlösungen sowie in den erfassten Fällen nach Problembeschreibungen, welche alle Wörter beinhalten, die in der Problembeschreibung des Falles oder der Musterlösung gespeichert sind. Falls die Service Cases, welche die gefundenen Fälle umfassen über Attribut-Sets verfügen, werden diese für den Vergleich ergänzt. Zudem erhalten alle Musterlösungen eine initiale Relevanz von 1000.

Anschliessend wird die Relevanz aller gefundenen Musterlösungen und der Fälle folgendermassen berechnet. Vor jedem Vergleich wird die kleinere Anzahl Attribute zwischen der Anfrage und der zu vergleichenden Lösung gespeichert. Dies stellt die maximale Anzahl Übereinstimmungen dar. Falls die Anfrage oder die zu vergleichende Lösung keine Attribute umfasst, bleibt die Relevanz der gefundenen Lösung, wie sie ist. Falls Attribute vorhanden sind, werden alle Attributs-Kombinationen miteinander verglichen. Wenn nun der Vergleich eines Paares ergibt, dass der Attributname und der Wert übereinstimmt, dann wird der Vergleichszähler um eins erhöht.

Nachdem alle Kombinationen verglichen sind, wird der Vergleichszähler durch die anfangs gespeicherte kleinere Anzahl Attribute geteilt und anschliessend mit hundert multipliziert und zur Relevanz hinzu addiert.

Somit stellt der Relevanz-Wert 100 das Maximum eines Lösungsvorschlags aus einem Fall dar und 1100 stellt das Maximum eines Lösungsvorschlags aus einer Musterlösung dar.

Stärken: Der Algorithmus ist relativ simpel und liefert nachvollziehbare Bewertungen. Zudem ist er schnell berechnet und stellt eine mögliche Umsetzung eines Algorithmus dar.

Schwächen: Der Algorithmus beschränkt die gefundenen Lösungsvorschläge nicht auf eine Auswahl, sondern es werden immer alle passenden Ergebnisse geliefert. Ausserdem ist der Vergleich von Attributen sehr einfach gehalten und bewertet keine Teilübereinstimmung des Wertes.

Alternative-Algorithmus Der Alternative-Algorithmus sucht in den Musterlösungen nach Problembeschreibungen, welche alle Wörter beinhalten, die in der Problembeschreibung des Falles gespeichert sind. Falls mindestens

eine Musterlösung gefunden wurde, wird die Lösungsbeschreibung in "ALTERNATIVE SOLUTION" geändert und zurückgeschickt.

Dieser Algorithmus ist lediglich dazu da, um zu demonstrieren, dass der Algorithmus zur Laufzeit ausgetauscht werden kann und hat keinen praktischen Nutzen.

4.5 Testing

Für die REST-Schnittstelle wurden Integrations-Tests erstellt, welche sicherstellen, dass die Interaktion des Clients mit den definierten Backend Endpunkten funktioniert.

4.5.1 Verwendete NuGet Bibliotheken

xunit Das xUnit-Testframework ist ein Open Source Test Framework für das .NET Framework und die Empfehlung der Dokumentation zu ASP.NET Core [16].

xunit.runner.visualstudio Ermöglicht das Ausführen der Tests durch den im VisualStudio integrierten Test Explorer.

Microsoft.AspNetCore.TestHost Wird benötigt um Tests für ASP.NET Core zu schreiben und auszuführen.

Microsoft.EntityFrameworkCore.Sqlite Ermöglicht die Verwendung einer relationalen In-Memory Datenbank für die einzelnen Testfälle. Somit kann für jeden einzelnen Testfall eine neue Datenbank erstellt und nach der Ausführung einfach verworfen werden.

4.6 Codemetriken

Um die Codequalität einschätzen zu können, wurde eine Analyse des Sourcecodes gemacht. Nachfolgend ist die Statistik für das Projektende aufgeführt.

Lines of Code	WebApp	3'113
Lines of Code	Backend	6'394
Total Lines of Code		**9'507**

Tabelle 4.1: Umfang

Anzahl Klassen	124
(Avg) Methoden pro Klasse	4.1
(Avg) Cyclomatic Complexity	1.41
(Max) Cyclomatic Complexity	27

Tabelle 4.2: Backend Klassen Aufteilung

4.7 Zielerreichung

Die in dieser Arbeit entwickelte Plattform entspricht dem aktuellen Stand der Technik und erlaubt es Service Cases und Fälle strukturiert zu erfassen.

4.7.1 Validierung Ergebnis

Die Showcase-Szenarien (siehe 3) sind komplett durchführbar. Diese decken alle Use Cases und die damit verbundenen funktionalen Anforderungen ab. Nachfolgend ist aufgeführt, welche nennenswerte Limitationen gelten.

- Attribute können nur vom Typ Text, Numerisch, Option, Boolean oder Bild sein.
- Bilder können nur im JPEG-Format gespeichert werden.
- Eine Modifikation einer Maschine kann nur durch einen Fall gemacht werden.

Zusammengefasst wurden folgende Punkte erreicht:

- Ein Mobile-App für Servicetechniker
- Einen Showcase zu Demonstrationszwecken
- Servicetechniker werden bei der Erfassung von Problembeschreibungen vom System unterstützt
- Das System bietet Lösungsvorschläge aus ähnlichen Problemen an
- Es können Statistiken und Auswertungen über alle Service Cases angezeigt werden
- Änderungen an Teilen von Maschinen durch Service Cases können angezeigt werden
- Die Architektur ist flexibel und erweiterbar

Somit kann für die Weiterentwicklung der Applikation eine gute Basis übergeben werden.

5 Ausblick

5.1 Attribut Typen

Die aktuelle Auswahl an Attributs-Typen kann je nach Anforderung zukünftig beliebig erweitert werden.

5.1.1 Komma-Zahlen

Zurzeit sind als numerische Werte nur Ganzzahlen möglich. Eine Überlegung wäre, rationale Zahlenwerte zu erlauben. Zu bedenken ist, dass eventuell mit angegeben werden kann, wie viele Nachkommastellen für das bestimmte Attribut relevant sind.

5.1.2 Diverse Datei-Typen

Aktuell ist es möglich für ein Attribut Bilder im JPEG-Format zu erfassen. Als Erweiterung bietet es sich an, später auch PDFs oder sogar Video-Dateien zuzulassen.

5.2 Maschinen und Ersatzteile

Die Thematik bietet wohl die interessantesten Erweiterungsmöglichkeiten.

5.2.1 Ersatzteil Katalog

Von grossem Nutzen wäre die Implementierung eines globalen Ersatzteil-Katalogs. Dort sollte es möglich sein, Ersatzteile zu definieren mit Artikelnummer etc. Zudem wäre eine Idee für gewisse Ersatzteile ein Wartungs-Intervall angeben zu können. Somit könnte das System proaktiv darauf hinweisen, falls ein Ersatzteil demnächst ausgetauscht werden sollte.

5.2.2 Maschinentyp Katalog

Ein globaler Katalog um Maschinentypen zu definieren wäre von Vorteil, um später neue Maschinen von Kunden erfassen zu können. Somit könnte man beispielsweise die Ersatzteile aus dem Ersatzteil-Katalog einem Maschinentypen zuordnen. Falls nun ein Servicetechniker auf eine noch nicht erfasste

Maschine trifft, kann er lediglich im Maschinentyp Katalog die passende Maschine auswählen und hat somit eine neue Instanz im System erfasst, auf welcher Modifikationen von Ersatzteilen getätigt werden können.

5.2.3 Maschinen Modifikation ohne Fall

Sobald einzelne Maschinen Instanzen erstellt werden können, wäre eine nützliche Funktion, dass Modifikationen von Ersatzteilen an der Maschine nachgetragen werden können, ohne dass ein spezifischer Fall vorliegt. In der Realität kann es auch vorkommen, dass eine Maschine repariert oder gewartet wird, ohne vom System erfasst worden zu sein. Optimal ist dann, wenn diese Modifikationen nachgetragen werden können.

5.2.4 Vorschlag von Maschinen/Ersatzteilen

Mit der Implementierung des Maschinen- und Ersatzteil-Katalogs kann der Client einfach erweitert werden. Während der Servicetechniker eine Ersatzteilnummer oder eine Beschreibung eintippt, könnte er bereits Vorschläge auf passende Ersatzteile erhalten. Zudem könnte anhand des Maschinentyps bereits eine Auswahl der benötigten Ersatzteile vorgeschlagen werden.

5.3 Benutzerverwaltung

Ein weiterer wichtiger Aspekt für die Verwendung der Applikation für Unternehmen ist die Benutzerverwaltung.

5.3.1 Benutzerkonto

Für die Benutzer der Applikation könnte ein Login mit Benutzernamen und Passwort vergeben werden, um die Berechtigung für den Zugriff auf die Applikation und deren Daten auf eine Gruppe einzuschränken.

5.3.2 Rollen

Eine Rollenaufteilung nach Benutzertyp bringt diverse Vorteile mit sich. Anfangs wäre eine Aufteilung auf Servicetechniker und Administrator denkbar. Die Reporting-Ansichten könnten so für Administratoren beschränkt werden und der Servicetechniker sieht nur die Ansichten, welche für ihn von Interesse sind, um einen Fall oder Service Case abschliessen zu können.

5.3.3 Individuelle Status Ansicht

Pro Benutzer könnte man sich eine individuelle Status Ansicht vorstellen. Somit wäre für den Servicetechniker beispielsweise direkt sichtbar, welche Fälle und Service Cases eine Bearbeitung benötigen, oder falls sich etwas geändert hat, was von Interesse sein könnte.

5.4 Lösungsfindung

Um die Lösungsfindung zu verbessern und erweitern gibt es mehrere Ansätze. Folgende Daten stellen einen guten Ansatz.

5.5 Benutzer Rating

Siehe 5.9. Diese Daten bieten eine gute Grundlage, um später ein "Recommender"-System beispielsweise mittels Azure ML Studio zu trainieren.

5.5.1 Abgeschlossene Fälle

Denkbar wäre ein Verarbeitungs-Prozess, welcher in einem Intervall alle Fälle und die zugehörigen Lösungen analysiert. Daraus könnte man einen Entscheidungsbaum trainieren, welcher Vorschläge zu zusätzlichen Attributen gibt, die für den vorliegenden Fall von Interesse sind.

5.6 Reporting

Ein passender Erweiterungspunkt stellt das Reporting dar. Folgende Ideen wären denkbar:

5.6.1 Fortlaufende Rating-Auswertung

Das Rating stellt die Bewertung der vorgeschlagenen Lösungen des Systems auf die gestellten Probleme dar. Interessant könnte die Beobachtung in zeitlichen Abständen sein. Der durchschnittliche Rating-Wert für jeweilige Zeitabschnitte (bspw. 30 Tage) könnte ausgewertet und in einer Grafik protokolliert werden. Somit wäre auf einer Kurve über die Zeit eine Verbesserung oder Verschlechterung des Algorithmus sichtbar.

5.6.2 Durchschnittliche Bearbeitungsdauer

Durch eine Anpassung der Historie in den Metainformationen könnte man verfolgen, wie lange ein Fall oder Service Case im Durchschnitt benötigt, um geschlossen zu werden.

5.6.3 Mitarbeiter Auswertung

Falls die Metainformationen um Benutzer/Bearbeiter Informationen ergänzt werden, könnte man sich vorstellen, eine Auswertung für die einzelnen Mitarbeiter zu machen.

5.7 Frontend Client

Durch die Verwendung einer REST-Schnittstelle zur Kommunikation mit dem Backend sind alle Client-Arten möglich, die das HTTP-Protokoll unterstützen.

5.7.1 Native Android/IOS App

Eine Variante von Client stellen native Mobile Apps für die entsprechenden Plattformen dar. Der Vorteil dieser stellt die bessere Verwendung der Sensoren im Mobilgerät dar. Somit könnte beispielsweise eine Bluetooth- oder GPS-Komponente direkt Daten erfassen und in die Applikation einfliessen lassen.

5.7.2 Offline Interaktion

In der Realität ist es denkbar, dass der Servicetechniker in gewissen Maschinenräumen keinen Internet Empfang hat. Somit wäre es von Vorteil, wenn die Applikation Daten primär lokal speichert und sich dieser Speicher bei Internetverbindung mit dem Backend synchronisiert. Mit der Vue Applikation aus dieser Arbeit wäre dies über die Verwendung von Vuex möglich.

5.8 Automatische Datenerhebung

Sobald die Erweiterung um einen Maschinen-Katalog erstellt ist, wäre es je nach Art und Typ der betroffenen Maschine denkbar, Sensoren-Werte direkt

zu überwachen und an das System zu schicken. Dies würde es einem Servicetechniker ermöglichen, beim Auftreten eines Problems direkt gewisse Daten für den vorliegenden Fall zur Verfügung zu haben. Somit könnte das System bereits eine erste Problemanalyse durchführen.

5.9 Benutzerfeedback

Zurzeit ist es für einen Servicetechniker möglich, die vorgeschlagenen Lösungen des Systems mit einer Bewertung von 1-5 Sternen zu versehen. Für jede Wertung werden die Attribute des Service Cases und des Problems, sowie die vorgeschlagene Lösung inklusive deren Attribute gespeichert. Dies ermöglicht es einem Servicetechniker dem System direkt ein Feedback zu geben. Im Optimalfall wird die durchschnittliche Wertung über die Zeit besser, was auf eine Verbesserung des Algorithmus hindeutet (siehe 5.6.1).

5.10 Batch Processing

Ein weiterer interessanter Ansatz ist die Durchführung von zeitlichen Batch Prozessen. Bei wachsender Datenmenge macht es Sinn, Auswertungen von Statistiken und vorberechnete Lösungsvorschläge in einen eigenen Dienst auszulagern. Dieser kann dann beispielsweise über Nacht die neuen Daten auslesen und die Resultate in eine für schnelle Abfragen optimierte Datenbank ablegen.

Abbildungsverzeichnis

2.1 Übersicht der Merkmale . . . 3
2.2 Use Case Diagram . . . 11
2.3 Domainmodel . . . 13
3.1 VueJS Logo . . . 20
3.2 Navigations-Diagramm Benutzer . . . 22
3.3 Navigations-Diagramm Administrator . . . 23
4.1 Deployment Diagram . . . 24
4.2 Search Interfaces . . . 26
4.3 Dto-Diagramm . . . 29
4.4 Client - REST Kommunikation . . . 31
4.5 Algorithmus Ablauf Blackbox . . . 31
A.1 Attribut-Endpunkte . . . 57
A.2 Attribut-Set-Endpunkte . . . 57
A.3 Attribut-Set Item-Endpunkte . . . 57
A.4 Configuration-Endpunkte . . . 57
A.5 Issue-Endpunkte . . . 58
A.6 Machine-Endpunkte . . . 58
A.7 Problem-Endpunkte . . . 58
A.8 Rating-Endpunkte . . . 58
A.9 Reporting-Endpunkte . . . 58
A.10 Search-Endpunkte . . . 58
A.11 Service Case-Endpunkte . . . 59
A.12 Showcase-Endpunkte . . . 59
A.13 Solution-Endpunkte . . . 59
A.14 Solution Proposal-Endpunkte . . . 59
A.15 Upload-Endpunkte . . . 59
A.16 Edit-Button . . . 63
A.17 Plus-Button . . . 63
A.18 Lösch-Button . . . 63
A.19 OK-Button . . . 63
A.20 Eye-Button . . . 63
A.21 Algorithmus wählen . . . 64
A.22 Screenshot: Service Case Übersicht . . . 64
A.23 Screenshot: Service Case bearbeiten . . . 65
A.24 Screenshot: Attribute-Set hinzufügen . . . 66
A.25 Screenshot: Attribute-Set bearbeiten . . . 67

A.26 Screenshot: Fall bearbeiten 68
A.27 Screenshot: Suche 69
A.28 Screenshot: Lösung bearbeiten 70
A.29 Screenshot: Maschinen Übersicht 71
A.30 Screenshot: Maschine ansehen 71
A.31 Screenshot: Maschinen Fall 72
A.32 Navigation Administrator 73
A.33 Attribute Übersicht 74
A.34 Attribute erstellen 75
A.35 Attribute-Set Übersicht 76
A.36 Attribute-Set erstellen 76
A.37 Attribute hinzufügen 77
A.38 Lösungsvorschläge Übersicht 77
A.39 Problem-Lösung Mapping 78
A.40 Reporting 79
A.41 Projektplan 80
A.42 Gantt-Diagramm 81
A.43 Projektverlauf 83

Tabellenverzeichnis

2.1 Attribut-Ähnlichkeit Pro & Contra 6
2.2 Entscheidungsbaum Pro & Contra 7
2.3 Attribut Typen 9
2.4 Weitere Anforderungen 14
2.5 Anforderungs-Abdeckung 17
3.1 Responsive Website Vor- & Nachteile 18
3.2 Native/CrossPlatform Mobile App Vor- & Nachteile 19
3.3 Responsive Web App Framework Vor- & Nachteile 19
4.1 Umfang 34
4.2 Backend Klassen Aufteilung 34
A.1 UC1: Service Case erfassen 44
A.2 UC2: Service Case bearbeiten 45
A.3 UC3: CRUD Fall 46
A.4 UC4: Maschine auslesen 47
A.5 UC5: Reporting auslesen 48
A.6 UC6: CRUD Attribut 49
A.7 UC7: CRUD Attribut-Set 50
A.8 UC8: CRUD Musterlösung 51
A.9 R1: Framework Unkenntnis 84
A.10 R2: Kommunikationsverlust 84
A.11 R3: Aufgabenstellung wird falsch interpretiert 85
A.12 R4: Komplexität wird unterschätzt 85
A.13 R5: Suboptimale Verwendung von Komponenten in der Toolchain 86
A.14 Risikoanalyse Elaboration 87
A.15 Risikoanalyse Construction 87

A Anhang

A.1 Use Cases Detail

A.1.1 UC1: Service Case erfassen

Description	Ein Servicetechniker erfasst einen neuen Service Case.
Primary Actor	Servicetechniker
Trigger	Ein Servicetechniker möchte einen neuen Service Case für einen Service-Einsatz eröffnen.
Stakeholder and Interests	Servicetechniker: Möchte alle Daten eines Service-Einsatzes im selben Service Case erfassen.
Preconditions	Es ist noch kein Service Case für den Service-Einsatz eröffnet worden.
Postconditions	Ein neuer Service Case wurde erstellt.
Main Success Scenario	1. Benutzer navigiert zur Service Case Übersicht. 2. Applikation zeigt vorhandene Service Cases an. 3. Benutzer erstellt einen neuen Service Case. 4. Applikation öffnet einen neuen Service Case und zeigt diesen an.
Frequency of Occurrence	Mehrmals pro Tag

Tabelle A.1: UC1: Service Case erfassen

A.1.2 UC2: Service Case bearbeiten

Description	Ein Servicetechniker bearbeitet einen erfassten Service Case.
Primary Actor	Servicetechniker
Trigger	Ein Servicetechniker möchte • einen Service Case umbenennen. • den Status eines Service Cases ändern. • sich einen Service Case ansehen. • die Service Case-Attribute neu ausfüllen.
Stakeholder and Interests	Servicetechniker: Möchte den Service Case auf den Service-Einsatz optimieren.
Preconditions	Es ist bereits ein Service Case für den Service-Einsatz eröffnet worden.
Postconditions	Die Änderungen am Service Case sind gespeichert.
Main Success Scenario	1. Benutzer wählt einen Service Case aus der Übersicht aus, um diesen zu bearbeiten. 2. Applikation öffnet den Service Case und zeigt ihn an. 3. Benutzer bearbeitet den Status, Namen oder Attribut-Sets des Service Cases und speichert die Änderungen.
Extensions	1a. Benutzer sucht einen Service Case mit Hilfe der Filterfunktion. 1b. Applikation zeigt die gefilterten Service Cases an.
Frequency of Occurrence	Mehrmals pro Tag

Tabelle A.2: UC2: Service Case bearbeiten

A.1.3 UC3: CRUD Fall

Description	Ein Servicetechniker erstellt, bearbeitet oder löscht einen Fall.
Primary Actor	Servicetechniker
Trigger	Ein Servicetechniker möchte • sich einen Fall ansehen. • einen Fall aus dem Service Case löschen. • einen Fall weiter bearbeiten. • einen neuen Fall erfassen.
Stakeholder and Interests	Servicetechniker: Möchte ein Problem strukturiert erfassen und bei der Lösungsfindung unterstützt werden.
Preconditions	Ein Service Case wurde geöffnet.
Postconditions	Die Änderungen am Fall sind gespeichert.
Main Success Scenario	1. Benutzer wählt einen Fall aus, um ihn zu bearbeiten. 2. Applikation zeigt den Fall an. 3. Benutzer bearbeitet den Status, Maschinen-ID oder Problem des Falles und sucht nach Lösungen. 4. Applikation listet mögliche Lösungsvorschläge auf. 5. Benutzer wählt passende Lösung aus und übernimmt diese.
Extensions	1a. Benutzer löscht einen Fall aus dem Service Case. 4a. Applikation findet keine Lösungsvorschläge. 4b. Benutzer bearbeitet Lösung direkt. 5a. Benutzer bearbeitet Lösungsvorschlag weiter.
Frequency of Occurrence	Mehrmals pro Tag

Tabelle A.3: UC3: CRUD Fall

A.1.4 UC4: Maschine auslesen

Description	Ein Servicetechniker sieht sich den aktuellen Zustand einer Maschine an.
Primary Actor	Servicetechniker
Trigger	Ein Servicetechniker möchte • die letzten Änderungen an der BOM der Maschine ansehen. • frühere Fälle in Verbindung mit einer Maschine ansehen.
Stakeholder and Interests	Servicetechniker: Möchte Zeit bei der Problemlösung sparen und kürzlich durchgeführte Reparaturen oder Änderungen nicht nochmals durchführen.
Preconditions	Es wurden Fälle mit dieser Maschine erfasst.
Main Success Scenario	1. Benutzer navigiert zur Übersicht der Maschinen. 2. Applikation listet alle Maschinen auf. 3. Benutzer wählt eine Maschine aus. 4. Applikation zeigt die BOM und früheren Fälle der ausgewählten Maschine an.
Frequency of Occurrence	Mehrmals pro Tag

Tabelle A.4: UC4: Maschine auslesen

A.1.5 UC5: Reporting auslesen

Description	Ein Administrator sieht sich einen Report über das System an.
Primary Actor	Administrator
Trigger	Der Administrator möchte • die aktuelle Statistik des Systems sehen. • Verbesserungsvorschläge für Musterlösungen oder Probleme erhalten.
Stakeholder and Interests	Administrator: Möchte die Lösungsfindung eines Problems verbessern.
Main Success Scenario	1. Benutzer navigiert zur Reporting Übersicht. 2. Applikation zeigt aktuelle Statistiken und Verbesserungsvorschläge an.
Frequency of Occurrence	Mehrmals pro Tag

Tabelle A.5: UC5: Reporting auslesen

A.1.6 UC6: CRUD Attribut

Description	Ein Administrator erstellt, bearbeitet oder löscht Attribute.
Primary Actor	Administrator
Trigger	Der Administrator möchte • den Namen, Typ oder Default Wert eines Attributs verändern. • sich ein bestehendes Attribut ansehen. • ein bestehendes Attribut löschen. • neue Attribute erstellen.
Stakeholder and Interests	Administrator: Möchte die strukturierte Erfassung von Service Cases und Problemen optimieren.
Postconditions	Änderungen am Attribut sind gespeichert.
Main Success Scenario	1. Der Administrator navigiert zur Attribut-Übersicht. 2. Applikation zeigt alle Attribute an. 3. Benutzer wählt ein Attribut aus, um es zu bearbeiten. 4. Applikation zeigt das Attribut im Editor an. 5. Benutzer ändert den Namen, Typen oder Defaul Wert des Attributs und bestätigt die Änderungen.
Extensions	2a. Applikation verfügt über keine Attribute. 2b. Benutzer erstellt neue Attribute. Weiter bei 4. 3a. Benutzer sucht ein Attribut mit Hilfe der Suchfunktion. 3b. Applikation zeigt die gefilterten Attribute an. 4a. Benutzer löscht ein Attribut. 4b. Applikation löscht das Attribut und zeigt die aktuelle Liste der Attribute an.
Frequency of Occurrence	Mehrmals pro Tag

Tabelle A.6: UC6: CRUD Attribut

A.1.7 UC7: CRUD Attribut-Set

Description	Ein Administrator erstellt, bearbeitet oder löscht Attribut-Sets.
Primary Actor	Administrator
Trigger	Der Administrator möchte • den Namen oder die Attribute eines Attribut-Sets bearbeiten. • sich ein bestehendes Attribut-Set ansehen. • ein bestehendes Attribut-Set löschen. • ein neues Attribut-Set erstellen.
Stakeholder and Interests	Administrator: Möchte die strukturierte Erfassung von Service Cases und Problemen optimieren.
Main Success Scenario	1. Benutzer navigiert zu der Übersicht der Attribut-Sets. 2. Applikation zeigt alle Attribut-Sets an. 3. Benutzer wählt ein Attribut-Set aus, um es zu bearbeiten. 4. Applikation zeigt das Attribut-Set im Editor an. 5. Benutzer ändert den Namen des Attribut-Sets, fügt Attribute hinzu oder löscht diese und speichert.
Extensions	2a. Applikation verfügt über keine Attribut-Sets. 2b. Benutzer erstellt ein neues Attribut-Set. Weiter bei 4. 3a. Benutzer sucht ein Attribut-Set mit der Suchfunktion. 3b. Applikation zeigt die gefilterten Attribut-Sets an. 4a. Benutzer löscht ein Attribut-Set. 4b. Applikation löscht das Attribut-Set und zeigt die aktuelle Liste der Attribut-Sets an.
Frequency of Occurrence	Mehrmals pro Tag

Tabelle A.7: UC7: CRUD Attribut-Set

A.1.8 UC8: CRUD Musterlösung

Description	Ein Administrator erstellt, bearbeitet oder löscht Musterlösungen.
Primary Actor	Administrator
Trigger	Ein Administrator möchte • Musterlösungen zu einem Problem erstellen. • Musterlösungen löschen. • Musterlösungen bearbeiten. • Musterlösungen ansehen.
Stakeholder and Interests	Administrator: Möchte die Lösungsfindung für ein Problem optimieren.
Main Success Scenario	1. Benutzer navigiert zur Übersicht der Musterlösungen. 2. Applikation zeigt erstellte Musterlösungen an. 3. Benutzer wählt Musterlösung aus, um sie zu bearbeiten. 4. Applikation öffnet Musterlösung und zeigt diese an. 5. Benutzer bearbeitet die Problembeschreibung, Lösungen oder Attribut-Sets und bestätigt die Änderungen.
Extensions	2a. Applikation verfügt über keine Musterlösungen. 2b. Benutzer erstellt eine neue Musterlösung. Weiter bei 4. 3a. Benutzer sucht eine Musterlösung mit Hilfe der Suchfunktion. 3b. Applikation zeigt die gefilterten Musterlösungen an. 4a. Benutzer löscht eine Musterlösung. 4b. Applikation löscht die Musterlösung und zeigt die aktuelle Liste der Musterlösungen an.
Frequency of Occurrence	Mehrmals pro Tag

Tabelle A.8: UC8: CRUD Musterlösung

A.2 Konzept Detail

A.2.1 Attribut

Beschreibung	Attribute enthalten Werte, welche zur Suche und Vergleich von Lösungsvorschlägen verwendet werden.
Attribute	**Name:** Ein Attribut hat einen Namen als Beschreibung. **Typ:** Der Typ definiert, um was für ein Attribut es sich handelt. **Wert:** Der Wert beinhaltet die Information des Attributs.
Beziehungen	**Attribut-Set:** Ein Attribut kann mehreren Attribut-Sets zugeordnet werden.

A.2.2 Attribut-Set

Beschreibung	Ein Attribut-Set repräsentiert eine Sammlung von Attributen. Sie können einem Service Case oder einem Problem zugeordnet werden.
Attribute	**Name:** Ein Attribute-Set hat einen Namen als Beschreibung.
Beziehungen	**Attribut:** Ein Attribut-Set kann aus mehreren Attributen bestehen. **Service Case:** Ein Attribut-Set kann einem Service Case zugeordnet werden, um ihn weiter zu beschreiben. **Problem:** Ein Attribut-Set kann einem Problem zugeordnet werden, um dieses genauer zu beschreiben.

A.2.3 Service Case

Beschreibung	Ein Service Case wird zur Erfassung von Fällen erstellt. Zusätzlich werden Metainformationen erfasst.
Attribute	**Name:** Dient zur Beschreibung eines Service Cases.
Beziehungen	**Attribut-Set:** Ein Service Case kann mehrere Attribut-Sets verwenden, um mehr und spezifischere Daten zu erfassen. **Metainformationen:** Zu einem Service Case werden zusätzlich Meta-Informationen erfasst. **Fall:** In einem Service Case können mehrere Fälle erfasst werden.

A.2.4 Metainformationen

Beschreibung	Metainformationen enthalten zusätzliche Informationen über einen Service Case oder einen Fall.
Attribute	**Erstelldatum:** Enthält das Datum an dem die Metainformation erstellt wurde. **History:** Beinhaltet die Änderungen eines Service Cases. **Status:** Gibt den Status der Metainformation an.
Beziehungen	**Service Case:** Metainformationen können sich auf einen Service Case beziehen. **Fall:** Metainformationen können sich auf einen Fall beziehen.

A.2.5 Fall

Beschreibung	Ein Fall ist die Aufnahme eines Problems, für welche eine Lösung gefunden werden möchte. Ein Fall besteht aus dem Problem, möglichen Lösungsvorschlägen und der Lösung. Zudem werden Metainformationen über den Fall erstellt.
Attribute	**Name:** Dient zur Beschreibung des Falles.
Beziehungen	**Service Case:** Ein Fall wird innerhalb eines Service Cases erfasst. **Metainformationen:** Zu einem Fall werden zusätzliche Metainformationen erstellt. **Problem und Lösung:** Ein Fall besteht unter anderem aus einem Problem und einer Lösung. **Lösungsvorschlag:** Zu einem Fall können mehrere Lösungsvorschläge gefunden werden. **Maschine:** Ein Fall bezieht sich immer auf eine Maschine.

A.2.6 Problem

Beschreibung	Ein Problem ist ein Fehlverhalten oder Defekt, welcher behoben werden möchte.
Attribute	**Beschreibung:** Beschreibt das Problem in textueller Form.
Beziehungen	**Fall:** Ein Problem ist Bestandteil eines Falles. **Lösungsvorschlag:** Zu einem Problem können mehrere Lösungsvorschläge gefunden werden. **Attribut-Set:** Ein Problem kann nebst der textuellen Beschreibung mit zusätzliche Attribut-Sets erweitert werden.

A.2.7 Lösung

Beschreibung	Eine Lösung schildert die Behebung eines konkreten Problems.
Attribute	**Beschreibung:** Textuelle Beschreibung der Lösungsanleitung. **Werkzeug:** Eine Liste von benötigten Werkzeugen für diese Lösung in textueller Form.
Beziehungen	**Fall:** Eine Lösung ist Bestandteil eines Falles. **Lösungsvorschlag:** Ein Lösungsvorschlag kann als Lösung in einem Fall verwendet werden. **Teil:** Eine Lösung kann die Bearbeitung von mehreren Ersatzteilen beinhalten.

A.2.8 Lösungsvorschlag

Beschreibung	Zu einem Fall können mehrere Lösungsvorschläge gefunden werden. Lösungen können über ein Problem als Lösungsvorschlag gefunden werden.
Attribute	**Relevanz:** Gibt an, wie treffend der gefundene Lösungsvorschlag auf das Problem passt.
Beziehungen	**Problem und Lösung:** Ein Lösungsvorschlag dient zur Zuordnung von Problemen auf ihre Lösungen. **Fall:** Durch den Algorithmus zur Zuordnung der Probleme auf ihre Lösungen können mehrere potentielle Lösungsvorschläge gefunden werden.

A.2.9 Ersatzteil

Beschreibung	Ein Ersatzteil ist eine Komponente einer Maschine und kann in den Lösungen enthalten sein, um die Maschine zu verändern.
Attribute	**Bezeichnung:** Dient zur Beschreibung des Ersatzteils.
Beziehungen	**Lösung:** Das Bearbeiten von Ersatzteilen einer Maschine kann in Lösungen enthalten sein. **Maschine:** Ein Ersatzteil kann einer Maschine zugeordnet werden.

A.2.10 Maschine

Beschreibung	Eine Maschine enthält ihren aktuellen Zustand.
Attribute	-
Beziehungen	**Ersatzteil:** Eine Maschine besteht aus einer Liste von Ersatzteilen, welche in Fällen mit dieser Maschine bearbeitet wurden. **Fall:** Eine Maschine kann von mehreren Fällen bearbeitet werden.

A.3 REST-Schnittstelle

A.3.1 Endpunkte

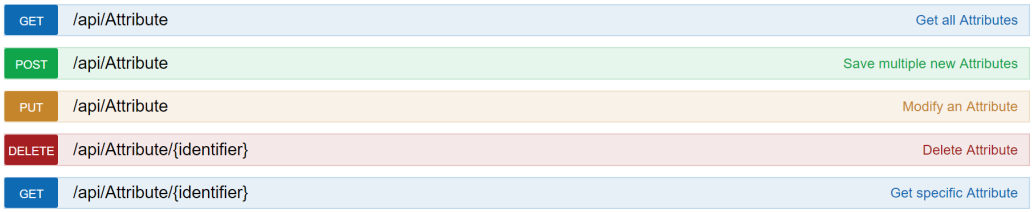


Abbildung A.1: Attribut-Endpunkte

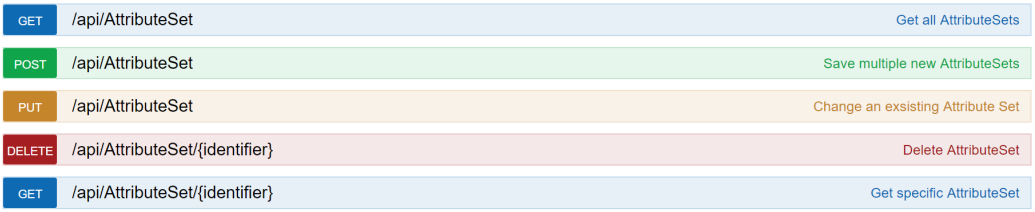


Abbildung A.2: Attribut-Set-Endpunkte

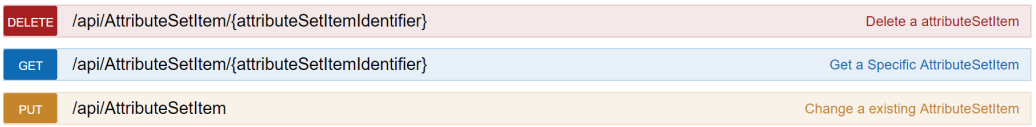


Abbildung A.3: Attribut-Set Item-Endpunkte

Abbildung A.4: Configuration-Endpunkte

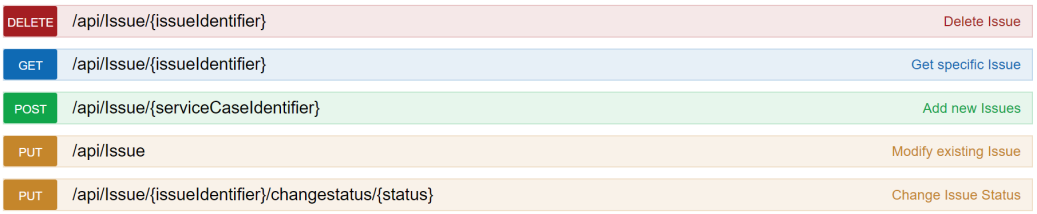


Abbildung A.5: Issue-Endpunkte

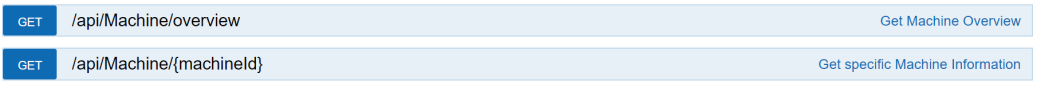


Abbildung A.6: Machine-Endpunkte

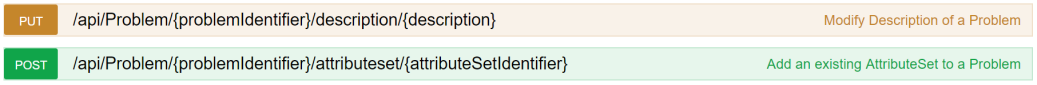


Abbildung A.7: Problem-Endpunkte

Abbildung A.8: Rating-Endpunkte

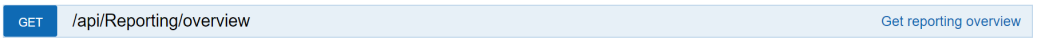


Abbildung A.9: Reporting-Endpunkte

Abbildung A.10: Search-Endpunkte

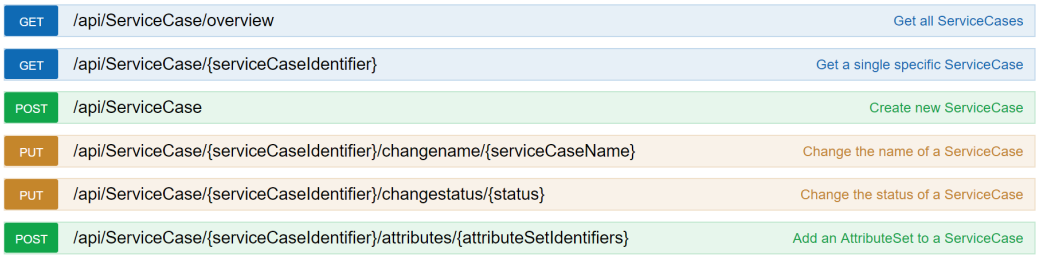


Abbildung A.11: Service Case-Endpunkte

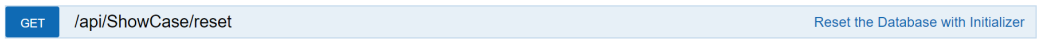


Abbildung A.12: Showcase-Endpunkte

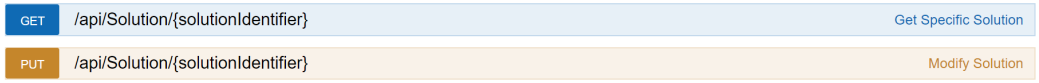


Abbildung A.13: Solution-Endpunkte

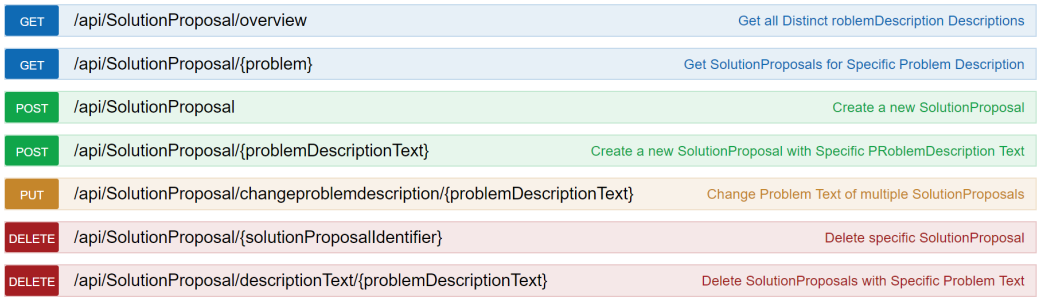


Abbildung A.14: Solution Proposal-Endpunkte

Abbildung A.15: Upload-Endpunkte

A.4 ShowCase-Szenarien

A.4.1 Vorbereitung auf Szenarien

Bevor die Szenarien durchgespielt werden können, müssen folgende Daten vorbereitet werden:

1. "Druckplattentemperatur, Numeric, 60", "Düsentemperatur, Numeric, 220" und "Klebstoff, Boolean, false" sind als Attribute (Name, Type, Default) definiert.
2. Ein Attribute-Set "Ultimaker 2 Settings" mit diesen Attributen ist erstellt.
3. Eine Musterlösung für "Druckmaterial hält nicht auf Druckplatte" wurde erstellt und enthält die Lösungen "Druckplattentemperatur erhöhen", "Düsentemperatur erhöhen" und "Klebstoff auftragen"
4. Allen Lösungen wird das Attribute-Set "Ultimaker 2 Settings" hinzugefügt
5. Die Lösung "Druckplattentemperatur erhöhen" muss mit den Attributen "Klebstoff, Ja", "Druckplattentemperatur, 50" und "Düsentemperatur, 220" erreicht werden.
6. Die Lösung "Düsentemperatur erhöhen" muss mit den Attributen "Klebstoff, Ja", "Druckplattentemperatur, 60" und "Düsentemperatur, 200" erreicht werden.
7. Die Lösung "Klebstoff auftragen" muss mit den Attributen "Klebstoff, Nein", "Druckplattentemperatur, 60" und "Düsentemperatur, 220" erreicht werden.
8. Eine Musterlösung für "Gerät schaltet sich nicht ein" ist erstellt und enthält die Lösungen "Netzteil ersetzen" mit dem Ersatzteil "Netzteil", "Stromzufuhr wiederherstellen" mit dem Ersatzteil "Sicherung" und "Stromkabel ersetzen" mit dem Ersatzteil "Stromkabel".

Alternativ kann auch die REST-API "GET /api/ShowCase/reset" mit Swagger verwendet werden, um die Daten für die Szenarien automatisch zu initieren.

A.4.2 Szenario 1: Lösungshinweise erhalten

1. Ultimaker 2 hat ein Problem "Druckmaterial hält nicht auf Druckplatte"
2. Servicetechniker 1 erfasst dieses Problem auf Phab1. Dafür erstellt er einen neuen Service Case und eröffnet einen Fall mit der Problembe-

schreibung “Druckmaterial hält nicht auf Druckplatte” und sucht nach Lösungsvorschlägen
3. Servicetechniker 1 erhält folgende Lösungsvorschläge zur Behebung auf Phab1
   - L1: Druckplattentemperatur erhöhen
   - L2: Düsentemperatur erhöhen
   - L3: Klebstoff auftragen
4. Das Problem wird an der Maschine jedoch anders gelöst
5. Servicetechniker 1 erfasst eine neue Lösung “Adhesive Plate mitdrucken” zum Problem “Druckmaterial hält nicht auf Druckplatte” auf Phab1
6. Maschine hat erneut das Problem “Druckmaterial hält nicht auf Druckplatte”
7. Servicetechniker 2 erfasst das Problem auf Phab2 und erstellt dazu einen Service Case mit einem Fall. Die Problembeschreibung lautet “Druckmaterial hält nicht auf Druckplatte”. Anschliessend sucht er nach Lösungsvorschlägen.
8. Servicetechniker 2 erhält folgende Lösungsvorschläge zur Behebung auf Phab2
   - L1: Druckplattentemperatur erhöhen (Musterlösung)
   - L2: Düsentemperatur erhöhen (Musterlösung)
   - L3: Klebstoff auftragen (Musterlösung)
   - L4: Adhesive Plate mitdrucken (keine Musterlösung)
9. Servicetechniker 2 wählt “Adhesive Plate mitdrucken” zur Behebung des Problems

A.4.3 Szenario 2: Reporting und Musterlösungen

Szenario 1 muss im Voraus absolviert worden sein, um mit Szenario 2 fortfahren zu können.

1. Servicetechniker 1 erstellt einen Service Case mit einem Fall und erfasst das Problem “Druckmaterial hält nicht auf Druckplatte” auf Phab1
2. Servicetechniker 1 erhält folgende Lösungsvorschläge auf Phab1:
   - L1: Druckplattentemperatur erhöhen (Musterlösung)
   - L2: Düsentemperatur erhöhen (Musterlösung)
   - L3: Klebstoff auftragen (Musterlösung)
   - L4: Adhesive Plate mitdrucken (keine Musterlösung)
3. Serviceadmin lässt sich ein Reporting ausgeben
4. Serviceadmin erkennt Bedarf einer Musterlösung für “L4: Adhesive Pla-

te mitdrucken" aus dem Reporting
5. Serviceadmin erstellt eine Musterlösung für "L4: Adhesive Plate mitdrucken" mit dem Attribute-Set "Ultimaker 2 Settings"
6. Serviceadmin fügt ein weiteres Attribut "Adhesive Plate, Numeric, Ja" zum Attribute-Set "Ultimaker 2 Settings" hinzu.
7. Serviceadmin bearbeitet Musterlösung, sodass die Lösung "L4: Adhesive Plate mitdrucken" mit den Attributen "Klebstoff, Ja", "Druckplattentemperatur, 60", "Düsentemperatur, 220" und "Adhesive Plate, Nein" referenziert ist.
8. Servicetechniker 2 erfasst nun einen Fall mit dem Problem "Druckmaterial hält nicht auf Druckplatte" auf Phab2
9. Servicetechniker 2 erhält besser sortierte Lösungsvorschläge und die neue Musterlösung "L4: Adhesive Plate mitdrucken"
10. Serviceadmin lässt sich ein Reporting ausgeben
11. Serviceadmin erkennt eine verbesserte Situation. Es werden keine Musterlösungen mehr vorgeschlagen.

A.4.4 Szenario 3: Aktuelle Situation im Feld

1. Ultimaker 2 hat ein Problem "Gerät schaltet sich nicht ein"
2. Servicetechniker 1 lässt sich die BOM der Maschine anzeigen
3. Servicetechniker 1 erstellt einen Service Case mit einem Fall und erfasst das Problem "Gerät schaltet sich nicht ein" auf Phab1
4. Servicetechniker 1 erhält folgende Lösungsvorschläge zur Behebung auf Phab1
   - L1: Netzteil ersetzen
   - L2: Stromzufuhr wiederherstellen
   - L3: Stromkabel ersetzen
5. Servicetechniker 1 lässt sich die dazu notwendigen Ersatzteile anzeigen
   - L1: Netzteil
   - L2: Sicherung
   - L3: Stromkabel
6. Servicetechniker 1 wählt den Vorschlag "L1: Netzteil ersetzen", aber ergänzt die Ersatzteile um ein Stromkabel
7. Maschine erhält ein Update, dass am Netzteil und Stromkabel der Maschine eine Änderung vorgenommen wurde
8. Servicetechniker 2 lässt sich die BOM der Maschine anzeigen
9. Servicetechniker 2 entdeckt die Änderungen durch Servicetechniker 1

A.5 Applikation Screens als Benutzer

A.5.1 Legende der Buttons

In diesem Kapitel werden folgende Buttons wie unten definiert referenziert:

Abbildung A.16: Edit-Button

Abbildung A.17: Plus-Button

Abbildung A.18: Lösch-Button

Abbildung A.19: OK-Button

Abbildung A.20: Eye-Button

A.5.2 Algorithmus wählen

Über das Menü am oberen Bildschirmrand kann man den Algorithmus auswählen, mit welchem die Lösungsvorschläge gesucht werden sollen.

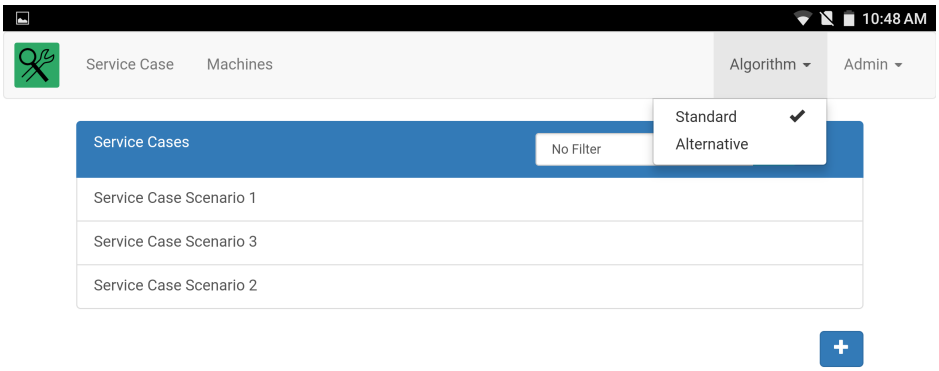


Abbildung A.21: Algorithmus wählen

A.5.3 Service Case Übersicht

Die Service Case Übersicht dient als Einstiegspunkt beim Starten der Applikation. Die Liste aller Service Cases kann zusätzlich nach deren Status gefiltert werden.

Durch Anklicken eines Service Cases kann dieser weiter bearbeitet werden. Der Plus-Button erstellt einen neuen Service Case.

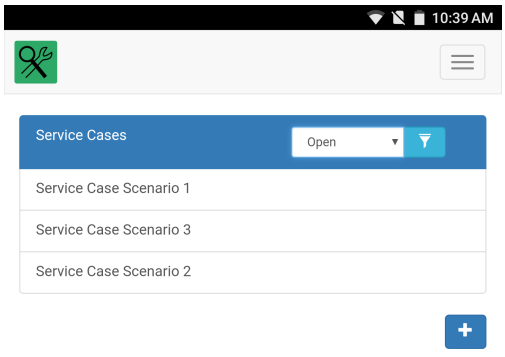


Abbildung A.22: Service Case Übersicht

A.5.4 Service Case bearbeiten

Im Service Case Editor kann ein Service Case angesehen oder bearbeitet werden. Von oben nach unten stehen folgende Funktionen zur Verfügung:

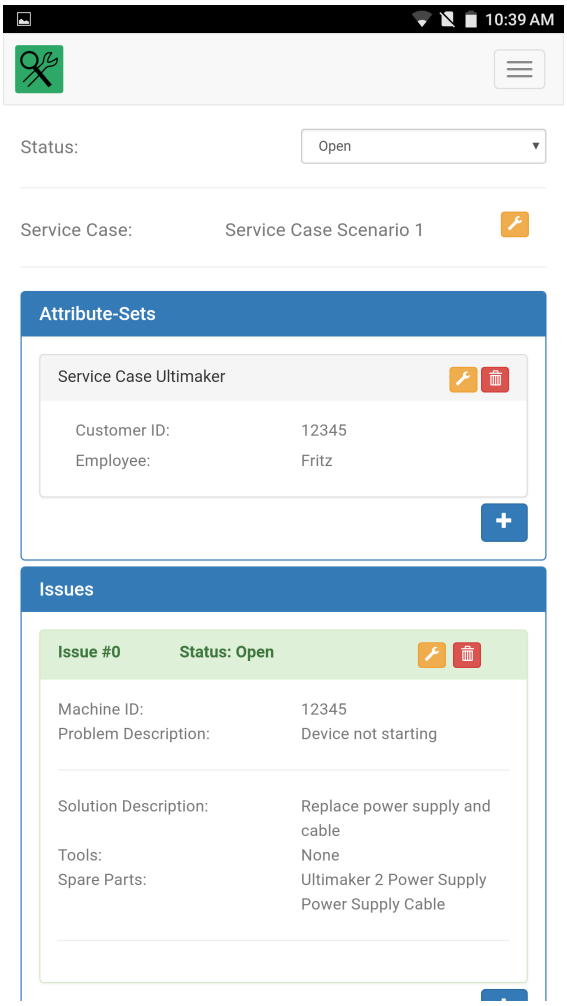


Abbildung A.23: Service Case bearbeiten

Status: Der Status des Service Cases kann ausgewählt werden.
Service Case: Durch Klicken des Edit-Buttons kann ein Name eingegeben und mit dem Save-Button bestätigt werden.
Attribute-Sets: Ausgewählte Attribute-Sets können mit dem Edit-Button bearbeitet und mit dem Lösch-Button entfernt werden. Die Liste der Attribute-Sets dieses Service Cases kann mit Hilfe des Plus-Buttons erweitert werden.
Issues: Eine Übersicht gibt Auskunft über die Fälle dieses Service Cases und deren Status. Der Edit-Button dient zur Bearbeitung, der Lösch-Button zur Löschung des Falls. Durch Anklicken des Plus-Buttons unten rechts wird ein neuer Fall an diesen Service Case angehängt.

A.5.5 Attribute-Set hinzufügen

Beim Hinzufügen eines Attribute-Sets zu einem Service Case oder einem Fall, öffnet sich ein Fenster mit den verfügbaren Attribute-Sets. Durch Klicken des Plus-Buttons wird das Attribute-Set hinzugefügt. Mit Cancel wird das Fenster geschlossen.

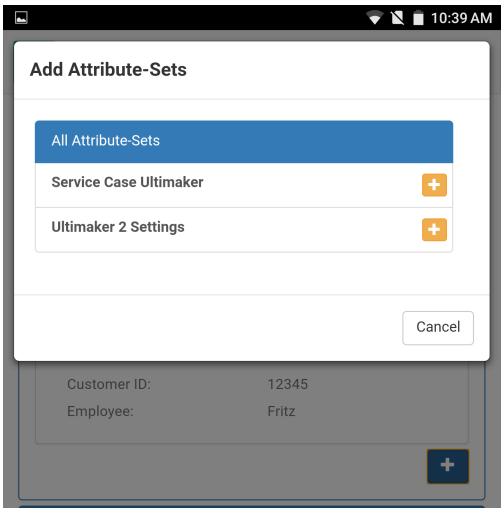
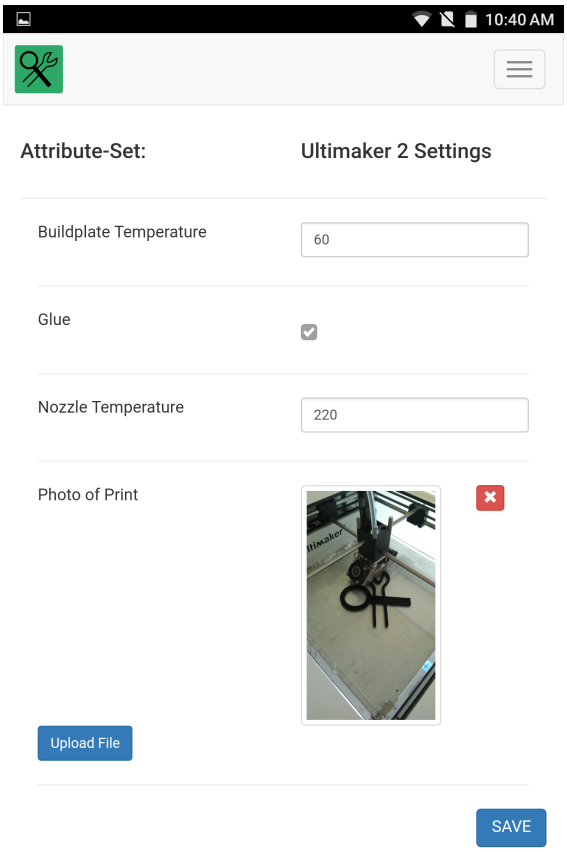


Abbildung A.24: Attribute-Set hinzufügen

A.5.6 Attribute-Set bearbeiten



Alle Attribute innerhalb dieses Attribute-Sets können hier gesetzt werden. Mit dem File-Upload können Bilder aus dem Dateiverzeichnis hochgeladen oder direkt mit der Smartphone-Kamera aufgenommen werden. Durch Klicken auf SAVE werden die Änderungen gespeichert und man wird auf die zuletzt besuchte Seite zurück geleitet.

Abbildung A.25: Attribute-Set bearbeiten

A.5.7 Fall bearbeiten

Hier kann ein Fall angesehen oder bearbeitet werden. Von oben nach unten stehen folgende Funktionen zur Verfügung:

Status: Der Status des Falles kann geändert werden.
Machine ID: Die Maschinen-ID, auf welche sich der Fall bezieht kann mit dem Edit-Button geändert und danach durch erneutes Klicken auf Save gespeichert werden.
Problem Description: Durch Klicken des Edit-Buttons kann das Problem beschrieben werden. Dabei werden Problembeschreibungen aus Musterlösungen und anderen Fällen vorgeschlagen. Mit Save werden die Änderungen gespeichert.
Attribute-Sets: Verhaltet sich gleich wie bei den Attribute-Sets im Service Case (siehe A.5.4).
Current Solution: Die aktuelle Lösung für das oben genannte Problem wird hier angezeigt. Mit dem Edit-Button unterhalb der Lösung kann diese bearbeitet werden.

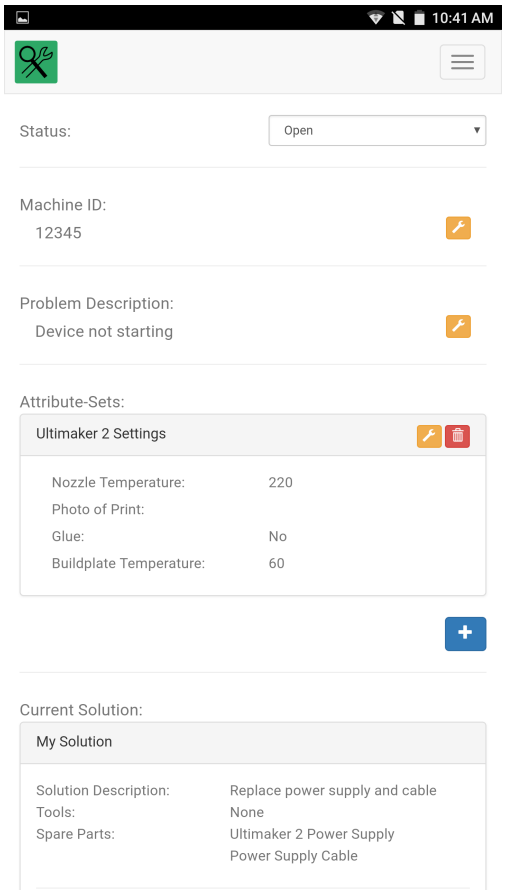


Abbildung A.26: Fall bearbeiten

A.5.8 Suche

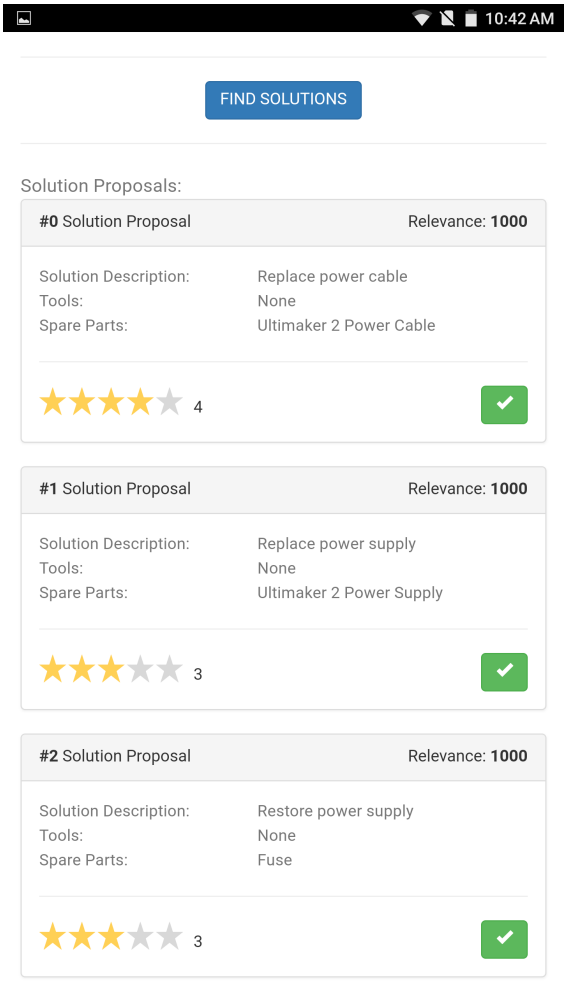


Abbildung A.27: Suche

Unterhalb des Falles aus A.5.7 befindet sich der Find-Solutions-Button. Nach dem Klicken auf diesen, werden anhand des ausgewählten Algorithmus passende Lösungen vorgeschlagen und sortiert nach deren Relevanz aufgelistet. Mit der Sternen-Bewertung kann einmalig pro Suche ein Feedback zum Algorithmus vergeben werden. Mit dem OK-Button wird der Lösungsvorschlag als Lösung übernommen und geöffnet.

A.5.9 Lösung bearbeiten

Hier können neue Lösungen erfasst oder ein ausgewählter Lösungsvorschlag bearbeitet werden.

Solution Description: Die Beschreibung zur Lösung des Problems kann hier eingegeben werden.

Tools: Die für die Lösung benötigten Werkzeuge können hier in definiert werden.

Spare Parts: Falls Ersatzteile benötigt werden, können mit dem Plus-Button Teile an diese Lösung angehängt werden. Mit dem Lösch-Button werden sie wieder von der Lösung entfernt.
Nach dem Klicken auf Save werden die Änderungen übernommen und eine Weiterleitung zu der zuletzt besuchten Seite wird vorgenommen.

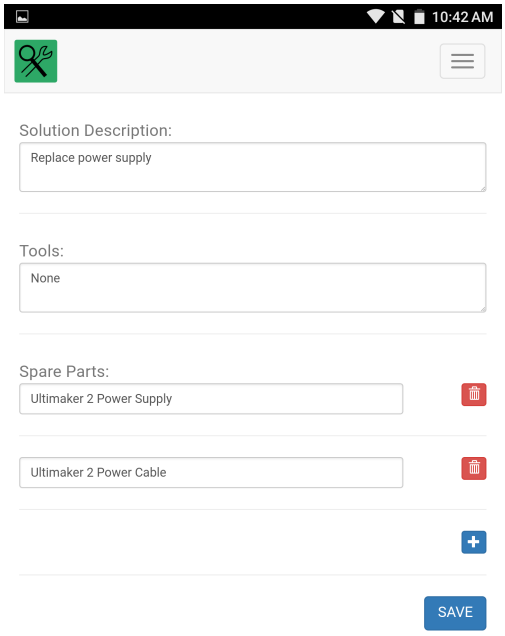
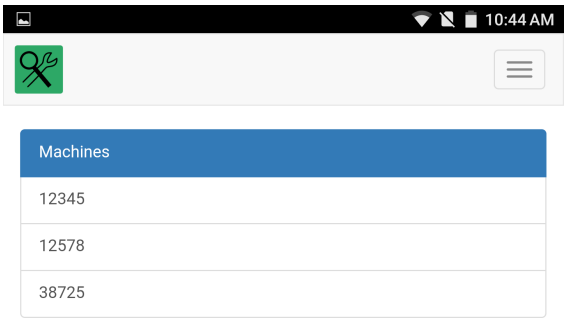


Abbildung A.28: Lösung bearbeiten

A.5.10 Maschinen Übersicht



Über den Menüpunkt Machine gelangt man zur Maschinen Übersicht. Hier sieht man alle Maschinen-ID's, welche in einem Fall referenziert wurden. Beim Anklicken einer Maschine wird die Detail-Ansicht der Maschine geöffnet.

Abbildung A.29: Maschinen Übersicht

A.5.11 Maschine ansehen

Machine ID: Die Machine ID zeigt an, über welche Maschine die folgenden Informationen sind.

Bill of Material: Hier werden alle Ersatzteile aufgelistet, welche in Lösungen für einen dieser Maschinen Fälle vorkommen. Zudem wird das Datum dieser Änderung angezeigt.

Past machine issues: Die Liste aller mit dieser Maschine verknüpften Fälle wird hier aufgelistet. Mit einem Klick auf den Eye-Button wird der Maschinen Fall angezeigt.

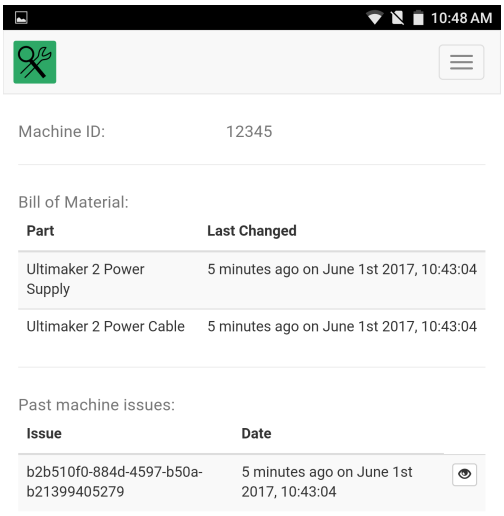


Abbildung A.30: Maschine ansehen

A.5.12 Maschinen Fall

Die Read-Only Übersicht über einen Fall aus einer Maschine wird angezeigt. Mit dem OK-Button wird man zur Maschinen Ansicht zurück geleitet.

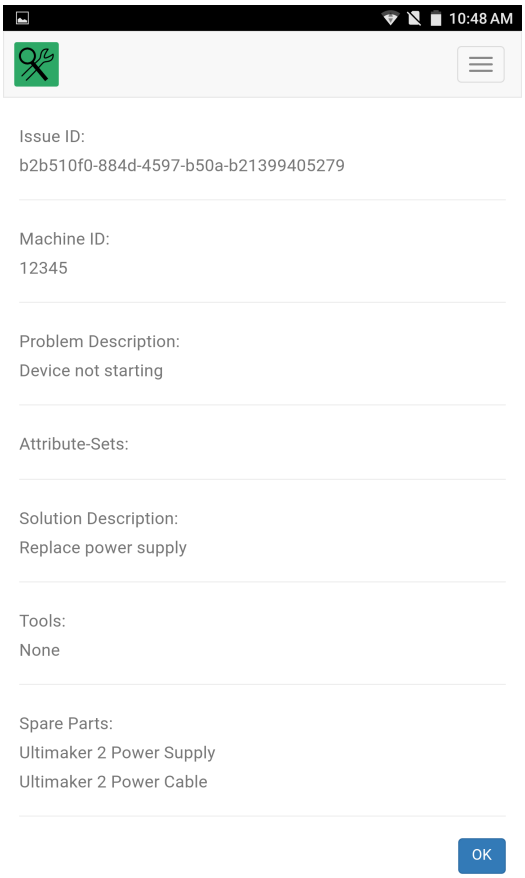


Abbildung A.31: Maschinen Fall

A.6 Applikation Screens als Administrator

A.6.1 Navigation Administrator

Über den Menüpunkt Admin am oberen Bildschirmrand gelangt man zu den Funktionen des Administrators.

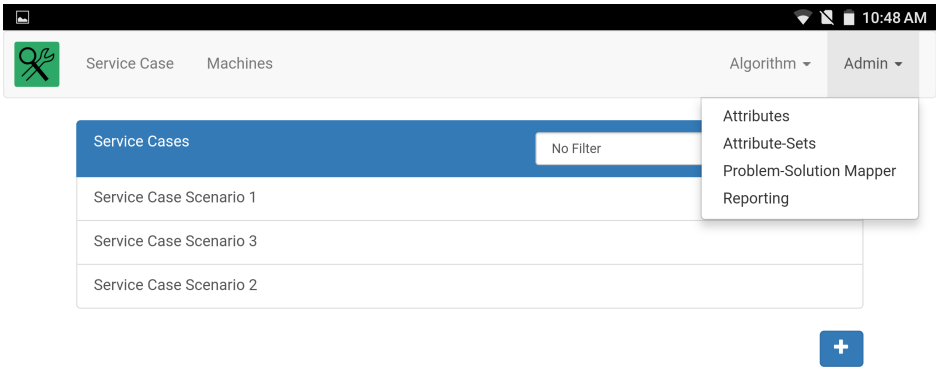


Abbildung A.32: Navigation Administrator

A.6.2 Attribute Übersicht

Über den Menüpunkt Attributes gelangt man zur Übersicht über die vorhandenen Attribute. Mit der Suchfunktion oben rechts können die Attribute nach Namen gefiltert werden. Der Edit-Button ermöglicht das Bearbeiten eines einzelnen Attributes, der Lösch-Button dient zum Löschen. Der Plus-Button unten rechts öffnet den Attribute-Editor, um neue Attribute zu erstellen.

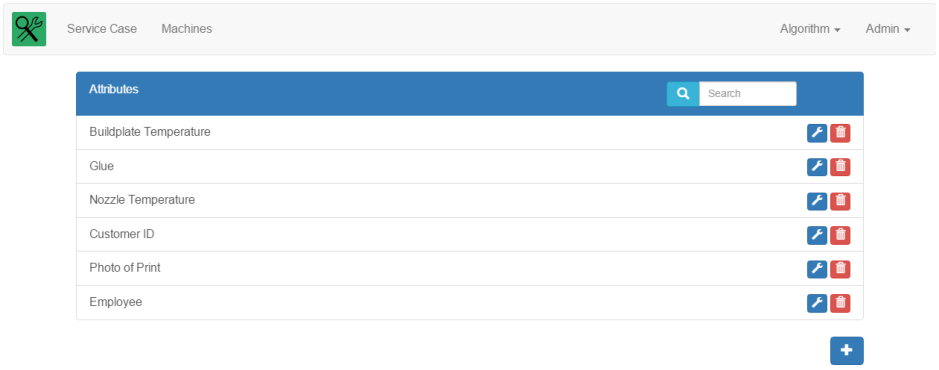


Abbildung A.33: Attribute Übersicht

A.6.3 Attribute erstellen

Der Attribute-Editor dient zum Erstellen von mehreren Attributen. Mit dem Plus-Button unten rechts können neue Attribute hinzugefügt werden. Diese können sodann oben bearbeitet werden. Mit einem Klick auf Save werden alle neu erstellten Attribute gespeichert und man gelangt zurück zur Attribute Übersicht.

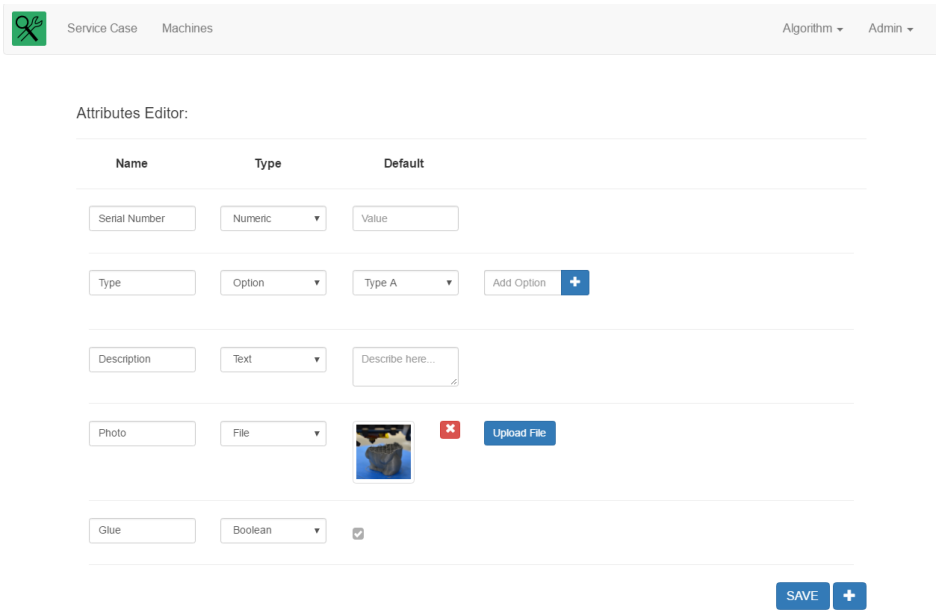


Abbildung A.34: Attribute erstellen

A.6.4 Attribute-Set Übersicht

Über den Menüpunkt Attribute-Sets gelangt man zur Übersicht über die vorhandenen Attribute-Sets. Die Funktionen für die Attribute-Sets auf dieser Seite sind dieselben wie die der Attribute Übersicht (siehe A.6.2).

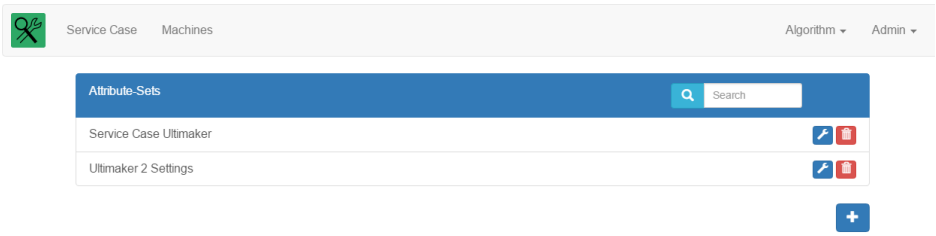


Abbildung A.35: Attribute-Set Übersicht

A.6.5 Attribute-Set erstellen

Mit Hilfe des Attribute-Set Editors kann ein Attribute-Set erstellt oder bearbeitet werden. Oben kann der Name des Sets gewählt werden. Unterhalb werden die ausgewählten Attribute angezeigt. Diese können hier nur gelesen und nicht bearbeitet werden. Über den Edit-Button unten rechts können die Attribute für dieses Set ausgewählt werden. Mit Save wird das Attribute-Set gespeichert und man gelangt zurück zur Attribute-Set Übersicht.

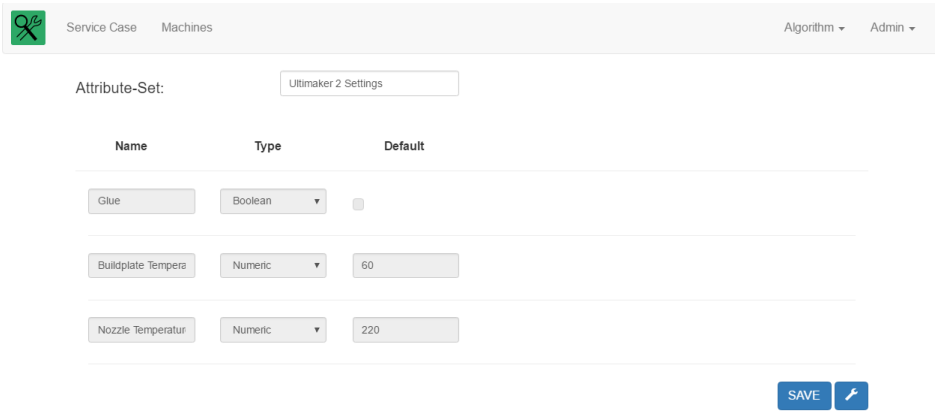


Abbildung A.36: Attribute-Set erstellen

Für das Hinzufügen oder Entfernen von Attributen in einem Set öffnet sich dieses Fenster. Mit einem Klick auf die Chevron-Buttons können Attribute von einer Seite auf die andere verschoben werden. Rechts befinden sich die für dieses Set ausgewählten Attribute. Oben rechts befindet sich ein Button,

um die verfügbaren Attribute neu zu laden, falls in der Zwischenzeit neue Attribute erstellt worden sind. Mit Save werden die Änderungen übernommen und das Fenster geschlossen.

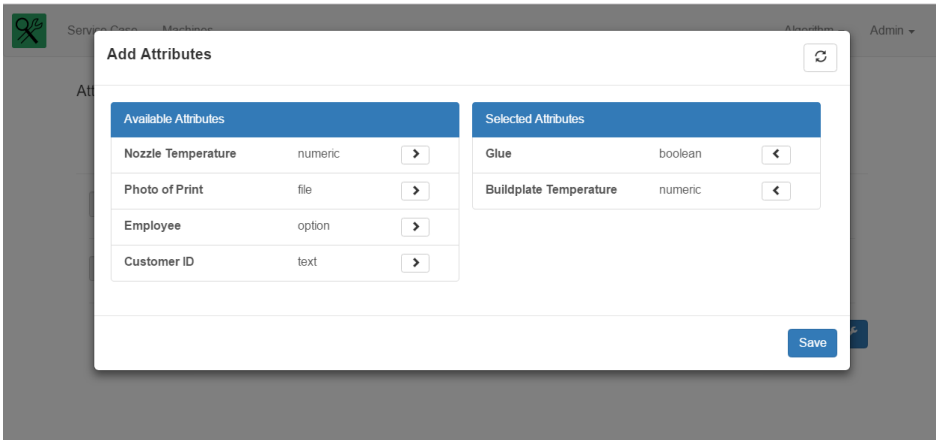


Abbildung A.37: Attribute hinzufügen

A.6.6 Lösungsvorschläge Übersicht

Über den Menüpunkt Problem Solution Mapper gelangt man zur Übersicht über die vorhandenen Lösungsvorschläge. Die Funktionen für die Lösungsvorschläge auf dieser Seite sind dieselben wie die der Attribute Übersicht (siehe A.6.2).

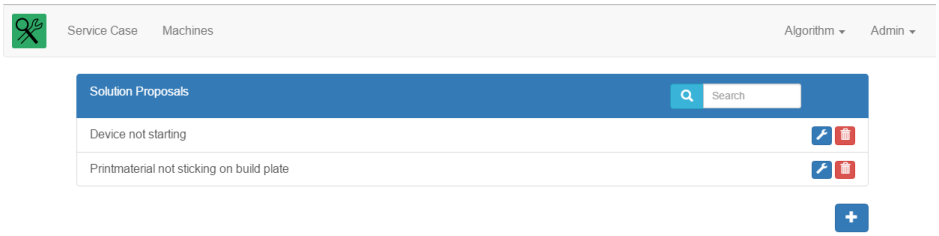


Abbildung A.38: Lösungsvorschläge Übersicht

A.6.7 Problem-Lösung Mapping

Problem Description: Mit dem Edit-Button kann die Problembeschreibung geändert werden, auf welche sich die unten aufgeführten Lösungen be-

ziehen. Mit Save werden die Änderungen gespeichert.

Solutions: Jede Lösung kann individuell mit dem Edit-Button bearbeitet werden (siehe A.5.9). Mit dem Lösch-Button wird die Lösung entfernt. Durch Klicken des Plus-Buttons unten rechts wird eine neue Lösung für dieses Problem erstellt.

Attribute-Sets: Die Attribute-Sets können mit dem Edit-Button bearbeitet (siehe A.5.6) oder mit dem Lösch-Button entfernt werden. Mit dem Plus-Button können der Problembeschreibung weitere Attribute-Sets angefügt werden (siehe A.5.5).

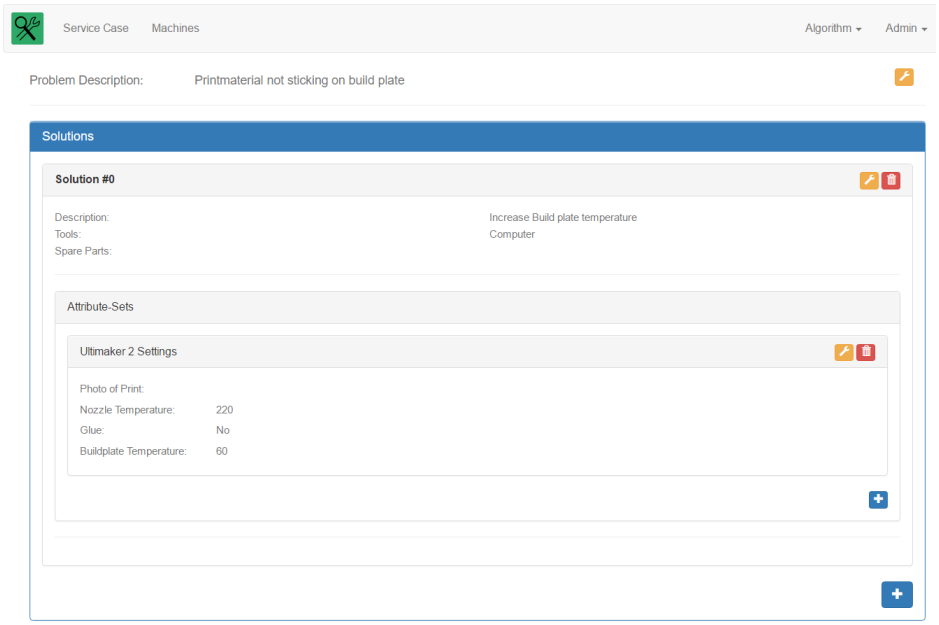


Abbildung A.39: Problem-Lösung Mapping

A.6.8 Reporting

Über den Menüpunkt Reporting gelangt man zur Report-Übersicht mit aktuellen Informationen über die Applikation.

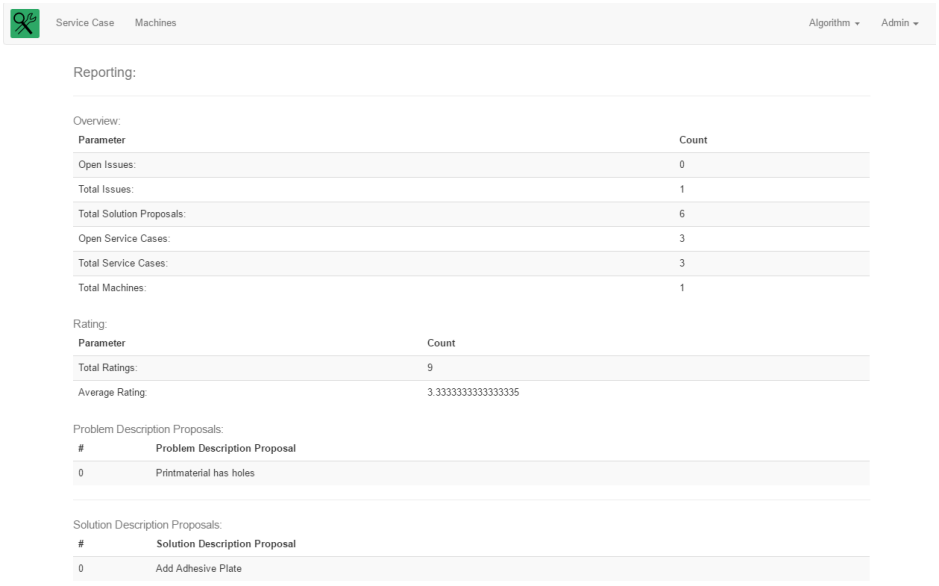


Abbildung A.40: Reporting

Literatur

[1] *Angular Website*. 10. Juni 2017. URL: https://angular.io/.

[2] *ASP.NET Core Dokumentation Swagger*. 10. Juni 2017. URL: https://docs.microsoft.com/en-us/aspnet/core/tutorials/web-api-help-pages-using-swagger.

[3] *ASP.NET Core Website*. 10. Juni 2017. URL: https://dotnet.github.io/.

[4] *Azure App Services Übersicht*. 15. Juni 2017. URL: https://docs.microsoft.com/en-us/azure/app-service/app-service-value-prop-what-is.

[5] *Azure Blob Storage Übersicht*. 2. Dez. 2016. URL: https://docs.microsoft.com/en-us/azure/storage/storage-introduction.

[6] *Azure SQL Übersicht*. 2. Dez. 2016. URL: https://docs.microsoft.com/en-us/azure/sql-database/sql-database-technical-overview.

[7] *EF SearchExtensions*. 10. Juni 2017. URL: http://ninjanye.github.io/SearchExtensions/.

[8] *Entity Framework Core Dokumentation*. 10. Juni 2017. URL: https://docs.microsoft.com/en-us/ef/core/.

[9] Roy T Fielding und Richard N Taylor. *Architectural styles and the design of network-based software architectures*. University of California, Irvine Doctoral dissertation, 2000.

[10] *Newtonsoft JSON Framework*. 10. Juni 2017. URL: http://www.newtonsoft.com/json.

[11] Michael Richter. *Case-based reasoning : a textbook*. Heidelberg: Springer, 2013. Kap. 2.5. ISBN: 978-3-642-40166-4.

[12] Michael Richter. *Case-based reasoning : a textbook*. Heidelberg: Springer, 2013. Kap. 6. ISBN: 978-3-642-40166-4.

[13] Michael Richter. *Case-based reasoning : a textbook*. Heidelberg: Springer, 2013. Kap. 4.6.1. ISBN: 978-3-642-40166-4.

[14] Michael Richter. *Case-based reasoning : a textbook*. Heidelberg: Springer, 2013. Kap. 5.2.4.5. ISBN: 978-3-642-40166-4.

[15] *VueJS Website*. 10. Juni 2017. URL: https://vuejs.org/.

[16] *xUnit Empfehlung von Microsoft*. 10. Juni 2017. URL: https://docs.microsoft.com/en-us/aspnet/core/testing/integration-testing.