HOCHSCHULE FÜR TECHNIK RAPPERSWIL

STUDIENARBEIT

Aufbereitung und Visualisierung von Maschinendaten einer Industrie 4.0-Maschine

---

Autoren:
Simon MÜLLER
Lukas KOLLER

Betreuer:
Prof. Dr. Daniel P. POLITZE

21. Dezember 2018

HOCHSCHULE FÜR TECHNIK RAPPERSWIL

Abteilung Informatik
Internet-Technologien und -Anwendungen

Abstract

Aufbereitung und Visualisierung von Maschinendaten einer Industrie 4.0-Maschine

von Simon MÜLLER
Lukas KOLLER

Die vorliegende Studienarbeit beschreibt die Erweiterung einer bestehenden Single Page Application um die Aufbereitung und Visualisierung von Maschinendaten einer Industrie 4.0-Maschine.

Aktuell sind weltweit mehrere Crimpmaschinen für Fahrzeugkabelbäume der Firma Komax AG im Einsatz. Diese Maschinen generieren fortlaufend interessante Daten, sowohl für die Komax AG wie auch deren Kunden. Stand heute sind diese Daten jedoch nicht, oder nur in geringer Weise abgreifbar. Dies erschwert für die Komax AG die effiziente Unterstützung ihrer Kunden bei der Fehlersuche.

In Zusammenarbeit mit der Abteilung Maschinentechnik wurde eine Crimpmaschine mit Sensoren ausgestattet und an das Internet angeschlossen. Daraufhin wurde ein Kommunikationskanal zu einem Websocket-Server implementiert, welcher die gesammelten Maschinendaten entgegennimmt und in kurzen, regelmässigen Zeitintervallen dauerhaft speichert.

Die aus einer Vorarbeit entstandene SPA mit zugehöriger Web API wurde in einem späteren Schritt um eine Echtzeitanalyse und Visualisierung der persistierten Daten erweitert.

Durch alle Layer hindruch wurde hierbei ein möglichst generischer Ansatz verfolgt um weitere Maschinentypen ohne Mehraufwand an das System anbinden und im Frontend auswerten zu können.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

AG	**A**ktien**G**esellschaft
API	**A**pplication **P**rogramming **I**nterface
AT	**AT**tention
CRM	**C**ustomer **R**elationship **M**anagement
ERP	**E**nterprise **R**essource **P**lanning
GSM	**G**lobal **S**ystem for **M**obile and Communications
GUID	**G**lobally **U**nique **ID**entifier
HSR	**H**och**S**chule für Technik **R**apperswil
HTTP	**H**yper **T**ransfer **T**ext **P**rotocol
HTTPS	**H**yper **T**ransfer **T**ext **P**rotocol **S**ecure
ID	**ID**entifier
IDE	**I**ntegrated **D**evelopment **E**nvironment
ISO	**I**nternational **S**tandard **O**rganisation
JSON	**J**ava**S**cript **O**bject **N**otation
IPEK	**I**nstitut für **P**oduktdesign **E**ntwicklung und **K**onstruktion
KB	**K**ilo**B**yte
M2M	**M**achine to **M**achine
OPC/UA	**O**pen **P**latform **C**ommunications / **U**nified **A**rchitecture
REST	**Re**presentational **S**tate **T**ransfer
SPA	**S**ingle **P**age **A**pplication
SSL	**S**ecure **S**ockets **L**ayer
UI	**U**ser **I**nterface
UC	**U**se **C**ase
UX	**U**ser **E**xperience

Kapitel 1

Management Summary

1.1 Ausgangslage

Die industrielle Revolution 4.0 gewinnt zunehmend an Bedeutung. Mit dem heutigen technologischen Fortschritt ist es möglich geworden nicht nur Computer, sondern auch handelsübliche Geräte, wie Uhren und Kühlschränke, aber auch Industriemaschinen über das Internet kommunizieren zu lassen.

Komax AG ist eine Firma mit Standort in der Schweiz. Sie gehört zur Komax Group, eine global tätige Technologiegruppe, die auf Automatisierungslösungen für ausgewählte Prozesse spezialisiert ist. Dabei unterstützt sie Kabelverarbeitung und wirtschaftliche Fertigungsabläufe, insbesondere bei Automobilzulieferern. Komax AG konstruiert und vertreibt unter anderem Industriemaschinen für das Zuschneiden von Kabelbäumen für Fahrzeuge der Automobilbranche.

Stand heute sind diese Maschinen nicht fähig über das Internet zu kommunizieren. Dadurch gehen der Komax AG interessante Informationen verloren, welche unter anderem für die Fernunterstützung ihrer Kunden, das Sammeln von Erfahrungswerten, wie auch das Tracking von Maschinenstandorten von Bedeutung sind.

1.2 Ziel der Arbeit

Zukünftig sollten Produkte der Komax AG mit Sensoren ausgestattet werden. Diese Sensoren dienen dazu, Maschinendaten zu sammeln, um diese danach via Netzwerkkommunikation an einen Server zu senden und zu persistieren. Ausserdem sollten die gesammelten Daten grafisch ausgewertet und dargestellt werden.

Das Ziel dieser Arbeit ist das Bereitstellen eines Kommunikationskanals für die Maschinen der Komax AG. Dieser sollte die gesendeten Daten über einen Webserver in eine Datenbank speichern. Schliesslich werden die gesammelten Informationen in einer SPA ausgewertet und in grafischer Form dargestellt.

1.3 Vorgehen

Die Komax AG hat für die Realisierung dieser Arbeit eines ihrer Produkte, eine Crimpmaschine vom Typ „Alpha 355 S“, zur Verfügung gestellt. In Zusammenarbeit mit einem Maschinentechniker konnte die Maschine mit Sensoren und einem

simplen E/A-Board mit Mikrocontroller ausgestattet werden. Danach ist ein Kommunikationskanal über einen Websocket eröffnet worden, welcher die Maschinendaten entgegennimmt und in einer Persistenzschicht speichert. Eine SPA wurde daraufhin so erweitert, dass diese über eine Web API die Maschinendaten abgreift und visualisiert.

1.4 Technologien

Als Kommunikationsprotokoll zwischen Maschine und Webserver wurde Websocket verwendet. Die Persistenzschicht ist eine ASP.Net Core 2.1 Web API mit einer LiteDB als Datenbank. Das Frontend ist eine in Angular geschriebene SPA, welche auf einer ASP.Net Core 2.1 Web API aufbaut. Diese Web API dient als Abstraktion der Persistenzschicht und vereinfacht den Zugriff auf deren Daten.

1.5 Ergebnisse

Dank sehr generischem Ansatz konnte eine flexible, robuste und leicht erweiterbare Lösung implementiert werden. Die bestehende SPA wurde um fünf Views erweitert, welche die gesammelten Maschinendaten auswerten und in passenden Diagrammen darstellen. Das Frontend reagiert komplett dynamisch auf die abgegriffenen Daten, was eine Konfiguration der Metadaten erlaubt. Auf Grund der Verwendung eines Websocket Protokolls können die Maschinendaten sehr performant und in kleinen Datenpaketen zwischen Maschine und Webserver transportiert werden. Zudem bildet das Websocket Protokoll die perfekte Grundlage für eine bidirektionalen Kommunikation.

1.6 Ausblick

Die Arbeit wird zusammen mit der Abteilung Maschinentechnik am 11. Januar 2018 präsentiert. Eine Folgearbeit soll zudem die SPA um zusätzliche Funktionalitäten ausbauen. Denkbar wäre zum Beispiel eine Visualisierung der Sensoren an der Maschine vor Ort über Augmented Reality. Ebenfalls könnte eine Verfeinerung und Weiterführung der bisherigen Lösung ein Thema für nachfolgende Arbeiten sein. Ziel ist es, dass die Firma Komax AG ein Endprodukt erhält, welches als Prototyp für weitere Arbeiten dient.

Kapitel 2

Einleitung und Aufgabenstellung

Das nachfolgende Kapitel dient dazu, einen Überblick über die Ausgangslage der Arbeit zu vermitteln. Es wird zuerst erklärt, in welchem Kontext dieses Projekt entstanden ist. Später folgt die ausgearbeitete Aufgabenstellung, die zusammen mit dem Betreuer und dem Kunden erarbeitet wurde. Da die Aufgabenstellung separat zu diesem Dokument entstanden ist, lässt sich in diesem Kapitel lediglich eine Kopie davon finden.

2.1 Einleitung

Die vorliegende Arbeit wurde im Auftrag eines Industriepartners, namentlich Komax AG, erarbeitet. Die Komax AG konnte von Herr Prof. Dr. Daniel Patrick Politze, einem Institutpartner am IPEK der HSR, akquiriert werden, welcher auch die Rolle als Betreuer für diese Arbeit übernahm.

Komax AG ist eine Firma mit Standort in der Schweiz. Sie gehört zur Komax Group, eine global tätige Technologiegruppe, die auf Automatisierungslösungen für ausgewählte Prozesse spezialisiert ist. Dabei unterstützt sie Kabelverarbeitung und wirtschaftliche Fertigungsabläufe, insbesondere bei Automobilzulieferern.

Ziel ist es, eine von der Komax AG zur Verfügung gestellten Industriemaschine mit Sensorik auszustatten und die generierten Daten für alle Stakeholder in einer Weboberfläche anzuzeigen. Es handelt sich bei der Maschine um eine Crimpmaschine vom Typ „Alpha 355 S“, die für das Zuschneiden von Kabelbäumen für Fahrzeuge der Automobilbranche eingesetzt wird.

Die Maschine wird mit einer Software ausgeliefert, über die die Maschine bedient werden kann. Über diese können verschiedene Einstellungen vorgenommen werden, wie z.B. in welcher Länge die Kabel geschnitten werden oder die Anzahl der zu schneidenden Kabel. Nebst diesen Einstellungen sammelt die Software auch noch Daten zur verarbeiteten Stückzahl oder einzelnen Ereignissen in Form eines Logs. Die Software läuft auf dem Betriebssystem Windows 7 und bietet keine Kommunikation über ein Netzwerk.

Da die Maschine noch über keine Sensorik verfügte, musste diese zuerst von einem Experten angebracht werden. Auch der Zugang zum Internet musste mittels Erweiterung um ein GSM-Modul gewährleistet werden. Diese Aufgaben sind Teil einer Bachelorarbeit der Abteilung Maschinentechnik der HSR und werden daher nicht in diesem Dokument behandelt. Das vorliegende Dokument behandelt in erster Linie die Übermittlung der Maschinendaten, deren Persistenz in einer Datenbank und

schliesslich die Auswertung in einem Frontend. Da die Zusammenarbeit mit der Abteilung Maschinentechnik aber essentiell für die Kommunikation zwischen Maschine und Webserver ist, werden im Verlaufe der Arbeit immer wieder Referenzen auf die genannte Bachelorarbeit gemacht.

Neben der Crimpmaschine dient eine bereits vorhandene SPA mit zugehörigem Webservice als Grundlage dieser Arbeiten. Diese wurde ebenfalls an der HSR in mehreren Studien- und Bachelorarbeiten erarbeitet.

2.2 Aufgabenstellung

Semesterarbeit HS 18/19 Abteilungen Informatik

Aufgabenstellung "Aufbearbeitung und Visualisierung von Sensordaten einer Industrie 4.0-Maschine"

Dieses Dokument beinhaltet die Aufgabenstellung für die im Rahmen einer Studienarbeit geplanten Erweiterung einer SPA um die Aufbearbeitung und Visualisierung von Sensordaten einer Industrie 4.0-Maschine. Die Arbeit wird vom Studiengang Informatik an der HSR realisiert.

Bearbeitung durch

Lukas Koller (Informatik)	Tel. +41 79 559 25 35	Email: lukas.koller@hsr.ch
Simon Müller (Informatik)	Tel. +41 76 347 47 52	Email: simon.mueller1@hsr.ch

Betreuung

Prof. Dr. Daniel P. Politze	Tel. +41 55 222 46 05	Email: daniel.politze@hsr.ch

Auftraggeber

Komax Group, Pius Anderhub Email: pius.anderhub@komaxgroup.com

Kick-off

19. September 2018, 15:00 Uhr

Abgabetermin

21. Dezember 2018, 12.00 Uhr

Ausgangslage

Aufbauend auf der im Frühjahrssemester 2018 durchgeführten Bachelorarbeit "SPA für das Management von Produktlebenszyklen" sollte ein bestehendes Product Cockpit um eine Visualisierung von Sensordaten erweitert werden. Die Sensordaten werden dabei von einer mit Sensoren ausgestatteten Industrie 4.0-Maschine geliefert und in einem Backbone zwischengespeichert. Das Cockpit wiederum bezieht die Visualisierungsdaten vom Backbone. Das Backbone wird uns von dem IPEK[1] zur Verfügung gestellt und repräsentiert eine Abstraktion von third-party Applikationen. Alle benötigten API-Informationen sind dokumentiert und für uns ersichtlich.

Zielsetzungen

Die bestehende SPA soll um eine Sensordatenvisualisierung erweitert werden und in regelmässigen Zeitintervallen aktualisiert werden. Als Grundlage hierfür dienen uns Sensordaten, die live von einer zur Verfügung gestellten Industrie 4.0-Maschine geliefert wird.

- Schnittstellenentwicklung zwischen der Industrie 4.0-Maschine und dem Backbone
- Persistierung der Sensordaten
- Cockpit um regelmässig Auswertung der Sensordaten erweitern
- Cockpit auf den Securitystandard des Backbones heben

[1] Zentrum für Produktentwicklung, PDM/PLM, CAx, Ein Institut der Hochschule für Technik Rapperswil

Rapperswil, 26. September 2018 Simon Müller, Lukas Koller

Semesterarbeit HS 18/19 Abteilungen Informatik

Vorgehen

- Project Specification and Setup
  - In die Problemstellung und das Cockpit einarbeiten
  - Entwicklungsumgebung aufsetzen
  - Grundverständnis für bestehende APIs entwickeln
  - Aufgabenstellungen definieren
  - Projektplan erstellen
- Requirements Analysis
  - Szenarios, User Stories und Use Cases definieren
  - Domänenanalyse durchführen und Domain Model erweitern
  - Funktionale und nicht-funktionale Anforderungen beschreiben
  - Visualisierungskonzept ausarbeiten (Schnittstelleninformationen)
- Prototypes and Interface-Definitions
  - Clickable-Prototyp entwickeln
  - "Durchstich"-Prototyp, der alle Layers umfasst, realisieren
  - Architekturprototyp entwerfen
- Sensor Persistence Service
  - Neuer Service zum Abgreifen und Persistieren der gelieferten Sensordaten entwickeln
- Sensor Cockpit Service
  - Neuer Service zum Laden der persistierten Sensordaten entwickeln
- Product Cockpit UI
  - Erweiterung des bestehenden Cockpit um Visualisierung der Sensordaten
- Completion and Presentation
  - Mockingdaten der bestehenden Applikation mit sinnvollen Testdaten anreichern
  - Cockpit auf den Securitystandard des Backbones heben

Weitere Hinweise

Die allgemeine Aufgabenstellung der beiden Bereichen Maschinentechnik und Informatik sind in einem separaten Dokument beschrieben.

Vertraulichkeit

Interne Informationen, Daten und Resultate sowie abgegebene Dokumente und Zeichnungen müssen vertraulich behandelt werden und dürfen nicht an Dritte weitergegeben werden. Die Ergebnisse der Arbeit gehören der Schule.

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

Unterschrift Betreuer: _____

Datum: 28.9.18

Unterschrift Teammitglied 1: _____

Datum: 26.10.18

Unterschrift Teammitglied 2: _____

Datum: 26.10.18

Rapperswil, 26. September 2018 Simon Müller, Lukas Koller

Kapitel 3

Situationsanalyse

In diesem Kapitel werden die Ausgangslage und die Ziele dieser Arbeit beschrieben. Zudem wird gezeigt, mit welchem Vorgehen die Anforderungen erfüllt werden konnten. Das Vorgehen wurde bereits in einer frühen Projektphase erarbeitet und als Projektplan festgehalten. Die Ziele leiten sich aus der Aufgabenstellung ab, welche vom Betreuer gegengelesen und bestätigt wurde.

3.1 Ausgangslage

Nach dem Vertrieb einer Maschine hat die Komax AG keine Übersicht über die Daten der Maschinen wie z.B. wo diese steht oder in welchem Zustand sie sich befindet. So entgehen der Komax AG wichtige Informationen, die sie für die Verbesserung ihrer Services einsetzen könnten.

Im Rahmen einer Studien- und Bachelorarbeit wurde bereits der erste Schritt gemacht. Darin wurde ein Cockpit erstellt, das Informationen aus verschiedenen Umsystemen zusammenträgt und an einem zentralen Punkt visualisiert. Die Umsysteme bestehen aus verschiedenen externen Systemen, wie z.B. einem ERP, einem CRM oder einer Verwaltung von Maschineninstanzen. Diese haben jeweils eine unabhängige Datenerhaltung, welche in eine Persistenzschicht, dem Product Backbone, synchronisiert wird. Somit bildet das Product Backbone das Rückgrat für alle in dieser Arbeit implementierten Services.

Eines der wichtigsten Merkmalen der bestehenden Arbeit war das Rollenkonzept für die verschiedenen Stakeholders von Komax AG. Dank diesem Konzept ist es möglich, nur die relevanten Informationen für bestimmte User anzuzeigen, um so den Arbeitsprozess so simpel wie möglich zu gestalten.

Mit diesen Massnahmen wurde die Grundlage für weitere Arbeiten auf dem Gebiet der Maschinenlebenszyklen geschaffen. Eine dieser Arbeiten ist die Datenbeschaffung der vertriebenen und im einsatzstehenden Maschinen.

3.2 Ziele

Ziel ist es, eine automatisierte Lösung zu schaffen, welche sowohl Metainformationen wie auch Livedaten zu einer Industrie 4.0-Maschine sammelt und im Product Cockpit darstellt. Das Anbringen der Sensorik an der Maschinen, die zu Testzwecken von der Komax AG zur Verfügung gestellt wurde, ist Teil einer Bachelorarbeit in der Fachrichtung Maschinentechnik.

Da eine enge Zusammenarbeit mit der Abteilung Maschinentechnik Voraussetzung ist, wird nachfolgend beschrieben, welche Ziele in dieser Arbeit gesteckt wurden, um eine klare Abgrenzung zu erhalten. Der Wortlaut ist derselbe wie der in der Aufgabenstellung.

- Schnittstellenentwicklung zwischen der Industrie 4.0-Maschine und dem Product Backbone
- Persistierung der Maschinendaten
- Cockpit um regelmässige Auswertung der Maschinendaten erweitern
- Cockpit auf den Sicherheitsstandard des Backbons heben

3.3 Vorgehen

Bevor mit der Arbeit gestartet werden konnte, wurde zuerst ein Projektplan erstellt. Dieser wurde insgesamt in sieben Phasen unterteilt, die zu einem erfolgreichen Projektverlauf führen sollen. Die Phasen wurden in einer iterativen Arbeitsweise durchgearbeitet. Zu jeder Phase sind Tickets definiert worden, welche in einer Projektmanagementsoftware verwaltet wurden. Im nachfolgenden Projektplan sind die verschiedenen Phasen in einem Zeitstrahl über die gesamte Projektdauer visualisiert.

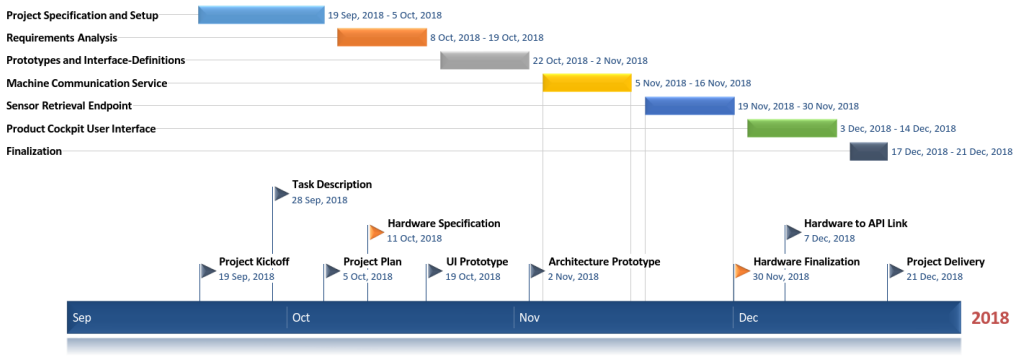


ABBILDUNG 3.1: Projektplan

Orange gefärbte Meilensteine sind solche, die von der Abteilung Maschinentechnik gesetzt wurden. Sie beinhalten Ziele, wie die Spezifikation der Hardware und deren Beschaffung. Eine grössere Version des Projektplans kann dem Anhang D entnommen werden.

Kapitel 4

Anforderungsanalyse

Im nachfolgenden Kapitel sind die funktionalen sowie die nicht-funktionalen Anforderungen, die mit der Komax AG erarbeitet wurden, beschrieben. Diese dienen als Grundlage für die Konzeption und sind iterativ erarbeitet worden. Daraus resultierten zum Schluss die Use Cases, welche in drei Prioritätsstufen gegliedert sind.

4.1 Funktionale Anforderungen

Die Anforderungen an Maschinen-, Sensor- und Metadaten wurden von der Komax AG definiert und in einem Dokument festgehalten. Die erarbeiteten User Szenarien und User Stories sind von den Anforderungen im erwähnten Dokument abgeleitet. Das Dokument ist im Anhang G zu finden.

	Statusparameter	Sichtbar für Komax	Sichtbar für Kunde	Prio
1	Maschinentyp / Seriennummer	ja	ja	1
2	Softwareversion Maschinensoftware	ja	ja	1
3	Softwareversion Betriebssystem	ja	ja	1
4	Laufleistung (Stückzahl / Betriebsst.)	ja	ja	1
5	Standortkoordinaten	ja	nein	3
6	Signalmaststatus: Rot = Störung / Grün = Produktion / Grün blinkend = Warten Bestätigung / Orange = Quality / Blau = Maintenance	ja	ja	1
7	Maschinenstatus: Auswertung Verfügbarkeit in % (Production, Setup, Interruption)	ja	ja	4
8	Temperatur Umgebung	ja	nein	2
9	Temperatur Schaltschrank (vorn / hinten)	ja	nein	2
10	Luftfeuchtigkeit Umgebung	ja	nein	2
11	Luftfeuchtigkeit Schaltschrank (vorn / hinten)	ja	nein	2
12	Temperatur Servomotoren (Bandantrieb, Schwenker, Abzug, Schneideinheit)	ja	nein	2
13	Luftdruck Maschine	ja	ja	2
14	Aktuelle oder letzte Fehlermeldung bei Maschinenstopp	ja	nein	3

ABBILDUNG 4.1: Von der Komax AG ausgearbeitete Anforderungen

4.1.1 User Szenarien

In diesem Kapitel werden möglichst reale Szenarien beschrieben, wie mit dem Endprodukt in Zukunft gearbeitet wird.

Kunde

Der Kunde bemerkt, dass seine Maschine nicht die gleiche Leistung hat wie sonst, deshalb interessiert er sich für die Laufleistung in den letzten zwei Wochen. Er meldet sich beim Product Cockpit an, woraufhin er mit der Rolle „Customer“ auf das Dashboard weitergeleitet wird. Im Dashboard wählt der Kunde seine Maschine aus und klickt auf „Show Details“. In den Details sieht der Kunde die Metainformationen seiner Maschine, unter anderem auch den Maschinenstatus. Nun klickt er auf den Button „Show Sensors“ worauf er zur grafischen Darstellung der verfügbaren Sensoren weitergeleitet wird. In der Übersicht sieht er neben dem Luftdruck der Maschine ebenfalls die Auswertung der Verfügbarkeit und die Laufleistung der letzten drei Wochen. Der Kunde bemerkt, dass die Laufleistung merklich abgenommen hat. Deshalb schaut er sich die Auswertung der Verfügbarkeit genauer an. Die Maschine befand sich in den letzten zwei Wochen zu 20 % im Status „Interruption“. Daraufhin überprüft der Kunde den Luftdruck in dieser Zeit und sieht, dass dieser jeden Tag zu einer gewissen Uhrzeit sinkt. Um die Laufleistung seiner Maschine wieder zu normalisieren, informiert er den Servicetechniker. Der Servicetechniker repariert ein defektes Ventil und bringt die Laufleistung wieder in den normalen Bereich.

Produktmanager

Der Produktmanager hat von seinem Chef den Auftrag erhalten, die Servomotoren, die im Feld sind, zu analysieren und ihre Lebensdauer um 10 % zu erhöhen. Er meldet sich beim Product Cockpit an, woraufhin er mit der Rolle „Product manager“ auf das Dashboard weitergeleitet wird. Im Dashboard wählt er eine Maschine mit vielen Servomotoren aus und klickt auf „Show Details“. Nun klickt er auf den Button „Show Sensors“ wodurch er zur grafischen Darstellung der angehängten Sensoren weitergeleitet wird. In der Übersicht hat er mehrere Grafiken zur Verfügung, die jeweils die Temperaturen der Servomotoren anzeigen. Der Produktmanager analysiert die Daten über das letzte Jahr und merkt, dass die Temperatur über dem spezifizierten Limit ist. Daher unterbreitet er seinem Chef ein Konzept um die Temperatur in den Servomotoren zu senken.

Servicetechniker

Ein Kunde meldet sich beim Servicetechniker. Es gibt ein Problem bei seiner Maschine. Er selbst kann den Fehler nicht ausfindig machen und bittet darum um Hilfe. Der Servicetechniker meldet sich beim Product Cockpit an, woraufhin er mit der Rolle „Service technician“ auf das Dashboard weitergeleitet wird. Im Dashboard wählt der Servicetechniker die Maschine des Kunden aus und klickt auf „Show Details“. In den Maschinendetails sieht der Servicetechniker, dass der Signalmaststatus auf grün ist. Deshalb klickt er auf den Button „Show Sensors“. Die Maschine verfügt über mehrere Sensoren, dazu gehören auch Temperatur- und Feuchtigkeitssensoren. Als erstes überprüft der Servicetechniker das Log mit den Fehlermeldungen, wird dabei jedoch nicht fündig. Deshalb überprüft er die Temperatur und die Luftfeuchtigkeit in den letzten zwei Tagen. Dabei sieht er, dass es eine enorme Temperaturänderung von 20 Grad Celsius gegeben hat. Der Servicetechniker meldet dem Kunden die mögliche Fehlerquelle mit einem Vorschlag um die Temperatur konstant halten zu können.

Verkäufer

Der neue Verkäufer von Komax AG hat den Auftrag bekommen eine neue Maschinensoftware zu vertreiben. Sein Chef setzt ihm das Ziel, dass 80 % aller Kunden mit der neuen Software arbeiten. Er meldet sich beim Product Cockpit an, woraufhin er mit der Rolle „Sales person“ auf das Dashboard weitergeleitet wird. Im Dashboard wählt der Verkäufer einen Kunden aus, klickt auf die erste Maschine und dann auf „Show Details„. In den Details sieht der Verkäufer in den Metainformationen der Maschine die Softwareversion, die auf der Maschine installiert ist. Da der Kunde mit einer veralteten Version arbeitet, ruft der Verkäufer den Kunden an und informiert ihn über die neue Software und welche Vorteile diese hat. Der Kunde entscheidet sich die Software zu kaufen.

4.1.2 User Stories

Die User Stories definieren Personen und deren Interesse an den Informationen. Zusätzlich beschreiben sie den Nutzen, der durch die Umsetzung der Anforderung erreicht wird. Sie leiten sich aus den User Szenarien ab und dienen als Grundlage für die Use Cases.

US01 Als Komax Mitarbeiter oder Kunde möchte ich Maschinendaten, wie Maschinentyp, Seriennummer, Softwareversion der Maschinensoftware und Softwareversion des Betriebsystems, sehen, um fehlende Updates der Maschinen ausfindig zu machen.

US02 Als Komax Mitarbeiter oder Kunde möchte ich die Laufleistung (Stückzahl / Betriebsstück) der verbundenen Maschine sehen, um zu analysieren, ob die Maschinen auf voller Leistung laufen und wann die Maschinen diese Leistung nicht erreicht haben.

US03 Als Komax Mitarbeiter möchte ich sehen wo sich die Maschine befindet, um Anfahrtswege für mögliche Supportfälle planen zu können.

US04 Als Komax Mitarbeiter oder Kunde möchte ich sehen, wenn sich etwas am Signalmaststatus ändert, um ggf. bei einem Fehler zu wissen, in welchem Status sich die Maschine befunden hat.

US05 Als Komax Mitarbeiter oder Kunde möchte ich den aktuellen Maschinenstatus sehen, um ggf. Massnahmen einzuleiten, damit die Maschine wieder mit voller Leistung produzieren kann.

US06 Als Komax Mitarbeiter möchte ich die Temperatur der Umgebung und des Schaltschranks einsehen können, um aussergewöhnliche Schwankungen und deren Einfluss auf die Maschine zu analysieren.

US07 Als Komax Mitarbeiter möchte ich die Luftfeuchtigkeit der Umgebung und des Schaltschranks einsehen können, um aussergewöhnliche Schwankungen und deren Einfluss auf die Maschine zu analysieren.

US08	Als Komax Mitarbeiter möchte ich die Temperatur der Servomotoren (Bandantrieb, Schwenker, Abzug, Schneideinheit, ect.) einsehen können, um aussergewöhnliche Schwankungen und deren Einfluss auf die Maschine zu analysieren.
US09	Als Komax Mitarbeiter oder Kunde möchte ich den Luftdruck der Maschine einsehen können, um mögliche Fehlerquellen ausschliessen zu können.
US10	Als Komax Mitarbeiter oder Kunde möchte ich zu jeder Instanz und jedem Sensor eine Historie der gesammelten Daten einsehen können, um Erfahrungswerte aus den Daten ziehen zu können.
US11	Als Komax Mitarbeiter möchte ich in einem Log alle Fehlermeldungen sehen, die von der Maschine produziert wurden, um kleinere Fehler remote lösen zu können.
US12	Als Komax Mitarbeiter oder Kunde möchte ich verschiedene Maschinendaten im gleichen Diagramm miteinander vergleichen können.

TABELLE 4.2: Ausgearbeitete User Stories

4.1.3 Use Cases

Aus den User Stories wurden die Use Cases abgeleitet, welche in diesem Abschnitt genauer erläutert werden. Im Gegensatz zu den User Stories sind diese detaillierter und können besser als Feature formuliert werden. Um die Use Cases zu priorisieren, wurden diese nach den Ausbaustufen Bronze, Silber und Gold gegliedert. Gold bedeutet in diesem Fall „vergoldet“ und wurde mit der kleinsten Priorität behandelt. Das Formality-Level wurde „brief“ gehalten und auf Use Case Models wurde gänzlich verzichtet.

Da die Use Cases auch zur Erstellung der Tickets im Ticketingsystem verwendet wurden, werden in diesem Abschnitt die Begrifflichkeiten verwendet, welche auch im Entwicklungsprozess Anwendung fanden. Maschinen werden als Instanzen bezeichnet und Sensoren als Parts.

INSTANZÜBERSICHT

UC01	Bronze	Instanzname unter Instanzübersicht in Textform anzeigen lassen.
UC02	Bronze	Instanztyp unter Instanzübersicht in Textform anzeigen lassen.
UC03	Bronze	Seriennummer unter Instanzübersicht in Textform anzeigen lassen.

UC04	Bronze	Instanzsoftware-Version unter Instanzübersicht in Textform anzeigen lassen.
UC05	Bronze	Betriebssystem unter Instanzübersicht in Textform anzeigen lassen.
UC06	Bronze	Betriebssystem-Version unter Instanzübersicht in Textform anzeigen lassen.
UC07	Bronze	Standort unter Instanzübersicht auf Google-Maps-Karte anzeigen lassen.
UC08	Bronze	Signalmaststatus unter Instanzübersicht graphisch anzeigen lassen.
UC09	Bronze	Instanzstatus unter Instanzübersicht in Textform anzeigen lassen.
UC10	Silber	Instanzfehlermeldung unter Instanzlog in Textform anzeigen lassen.
UC11	Silber	Meldungen zu fehlenden / fehlerhaften Instanzinformationen unter Instanzübersicht in Textform anzeigen lassen.
UC12	Bronze	Letzter Aktualisierungszeitpunkt der Instanzinformationen unter Instanzübersicht in Textform anzeigen lassen.
PARTÜBERSICHT		
UC13	Bronze	Laufleistung unter Partübersicht und Detailansicht graphisch im Verlauf anzeigen lassen.
UC14	Bronze	Signalmaststatus unter Partübersicht und Detailansicht graphisch im Verlauf anzeigen lassen.
UC15	Bronze	Instanzstatus unter Partübersicht und Detailansicht graphisch im Verlauf anzeigen lassen.
UC16	Bronze	Umgebungstemperatur unter Partübersicht und Detailansicht graphisch im Verlauf anzeigen lassen.
UC17	Bronze	Schaltschranktemperatur unter Partübersicht und Detailansicht graphisch im Verlauf anzeigen lassen.
UC18	Bronze	Umgebungsluftfeuchtigkeit unter Partübersicht und Detailansicht graphisch im Verlauf anzeigen lassen.
UC19	Bronze	Schaltschrankluftfeuchtigkeit unter Partübersicht und Detailansicht graphisch im Verlauf anzeigen lassen.

UC20	Bronze	Servomotortemperatur unter Partübersicht und Detailansicht graphisch im Verlauf anzeigen lassen.
UC21	Bronze	Instanzluftdruck unter Partübersicht und Detailansicht graphisch im Verlauf anzeigen lassen.
UC22	Silber	Meldungen zu fehlenden / fehlerhafte Partinformationen unter Partübersicht und Detailansicht in Textform anzeigen lassen.
UC23	Bronze	Letzter Aktualisierungszeitpunkt der Partinformationen unter Partübersicht und Detailansicht in Textform anzeigen lassen.
UC24	Bronze	Zu jedem Part eine Detailansicht mit Informationen, wie Datentyp, Interval, Einheit unter Partübersicht und Detailansicht in Textform anzeigen lassen.
UC25	Silber	Für jede graphische Darstellung eine Einschränkung für einen bestimmen Zeitraum auswählen.
UC26	Silber	Für jede graphische Darstellung einen Normalwertebereich anzeigen lassen.
UC27	Silber	Für jeden Part einen Log mit Errorinformationen anzeigen lassen.
UC28	Gold	Partinformationen in der Partübersicht nach Name sortieren und anzeigen lassen.
UC29	Gold	Partinformationen in der Partübersicht nach Name suchen und anzeigen lassen.
KONFIGURATION		
UC30	Gold	Konfiguration, welche graphischen Daten miteinander im selben Diagramm angezeigt werden, über ein simples Textfile vornehmen.
UC31	Gold	Darstellungskonfiguration über ein UI vornehmen.

TABELLE 4.4: Abgeleitete Use Cases

4.2 Nicht-funktionale Anforderungen

Die nachfolgenden nicht-funktionalen Anforderungen richten sich nach der Checkliste der ISO 9126 [1].

4.2.1 Funktionalität

Genauigkeit

Die Genauigkeit der Sensoren ist grundsätzlich von den Sensoren abhängig. Dabei unterstützt die Applikation die Genauigkeiten der primitiven Datentypen von C#. Die Daten werden bereits auf dem Arduino Client sinnvoll aufbereitet und gerundet. Es sind keine Abweichungen der Werte zwischen dem versendeten Wert des Sensors und dem im Product Cockpit angezeigten Wert zu erwarten.

Interoperabilität

Damit die Interoperabilität gewährleistet werden kann, wurde eine generische Schnittstellendefinition entwickelt, sodass diese Daten durch alle Applikationen gereicht werden können. Die Daten, die von den Sensoren gesendet werden, werden in derselben Struktur und im selben Format im Backbone gespeichert. Das Cockpit bezieht die Daten dann in darstellbarer Form über den Retrieval Service.

Sicherheit

Der Anmeldebereich wurde bereits in einer Vorarbeit umgesetzt und wird nicht um zusätzliche Sicherheit erweitert. Ebenfalls Teil davon ist das bereits existierende Rollenkonzept. Dies wurde auch in der neuen Version des Cockpits verwendet, damit Maschinendaten nur von Benutzern betrachtet werden können, die auch die Berechtigung dafür haben. Die Kommunikation von und zum Backbone, das vom IPEK der HSR zur Verfügung gestellt wurde, ist durch ein SSL-Zertifikat verschlüsselt und nur über HTTPS erreichbar. Alle anderen Kommunikationskanäle sind nicht verschlüsselt. Die Verschlüsselung der Kommunikation dieser Schnittstellen wird für einen späteren Schritt empfohlen und gilt momentan noch als optional.

4.2.2 Zuverlässigkeit

Fehlertoleranz

Da es sich in dieser Arbeit um einen Prototypen handelt, sind Sensor- oder Netzwerkausfälle nicht abgedeckt. Sollten Daten oder Schnittstellen nicht erreichbar sein, wird dies im User Interface entsprechend mit einer Meldung angezeigt. Das User Interface ist auch bei fehlenden Maschinendaten bedienbar.

Wiederherstellbarkeit

Die persistierten Daten liegen im Backbone. Eine Wiederherstellbarkeit der Daten muss deshalb durch die HSR gewährleistet werden.

4.2.3 Benutzbarkeit

Verständlichkeit

Das Cockpit soll ohne Einarbeitungsaufwand bedienbar und selbsterklärend sein. Klare Beschriftungen, optimierte Darstellungsformen und Responsive Design unterstützen die Verständlichkeit.

Erlernbarkeit

Das Product Cockpit beinhaltet keine komplexen Funktionen und ist damit einfach und schnell erlernbar.

Bedienbarkeit

Eine einheitliche Menüführung mit übereinstimmendem Farbkonzept unterstützt den User bei der Navigation durch die Applikation. Die neue Version soll wie die Vorgängerversion über das Mobile bedienbar sein. Die Detailansicht ist auf dem Mobile nur eingeschränkt verfügbar, da Diagramme in gewissen Fällen nicht sinnvoll angezeigt werden können. Die Anwendung ist mit Google Chrome ab Version 64.0.3325.x verwendbar. Auf Optimierung der Accessibility wurde verzichtet.

4.2.4 Effizienz

Zeitverhalten

Die Maschinendaten sollen unter Onlinebedingungen spätestens eine Minute nach Erfassen bei der Maschine im Cockpit angezeigt werden.

4.2.5 Übertragbarkeit

Anpassbarkeit

Die Anpassbarkeit der Applikation ist gut bis sehr gut. Alle Komponenten arbeiten losgelöst voneinander, sind austauschbar und konfigurierbar. Die Koppelung zwischen den Komponenten ist so lose wie möglich. Die Schnittstelle zwischen Maschine und Backbone nimmt generische Daten entgegen, sodass neue Maschinen- oder Sensorkombinationen ohne weiteren Aufwand an das System angeschlossen werden können

Installierbarkeit

Die Installation kann auf einem Windowsserver vorgenommen werden. Dies ist jedoch nicht trivial, da mehrere Applikationen miteinander kommunizieren. Deshalb kann dies nur durch eine geschulte Person erledigt werden.

Kapitel 5

Lösungskonzeption

In diesem Kapitel wird die Konzeptionsphase und deren Resultate beschrieben. Dabei werden grundlegende Konzeptionsentscheide erläutert und begründet. Der Fokus liegt vor allem auf den UI- und Schnittstellenentscheiden. Auf Entscheide, die in der Vorgängerversion bereits gemacht wurden, wird nicht genauer eingegangen.

5.1 Übermittlung und Persistenz der Maschinendaten

Die Persistenzschicht für das Speichern der Maschinendaten wurde stark in Zusammenarbeit mit der Abteilung Maschinentechnik konzipiert. Entscheide, über welches Endgerät und mittels welcher Transportschicht gearbeitet wird, mussten in beiden Arbeiten berücksichtigt werden. Deshalb war eine genaue Abstimmung zwischen den Abteilungen unabdingbar. Nur so konnte gewährleistet werden, dass die nötigen Endpoints zur Verfügung gestellt sind und eine simple Persistenz der Maschinendaten möglich ist.

5.1.1 Netzwerkprotokoll

Vor der Implementierung der Server-Client-Kommunikation wurde zuerst evaluiert, welches Netzwerkprotokoll für die Übermittlung der Maschinendaten am besten geeignet ist. Einander gegenübergestellt wurden die zwei häufigsten im Web zu findenden Netzprotokolle HTTP und Websocket.

HTTP mit REST-Paradigma

HTTP ist das meistverbreiteste Netzwerkprotokoll im World Wide Web [2]. Es wird hauptsächlich zur Übertragung von Webseiten zu Webbrowsern verwendet. Da das Protokoll zustandlos ist, bietet es den Vorteil, dass beliebig idempotente Zugriffe von verschiedensten Punkten im Netz gemacht werden können.

Gleichzeitig ist dies aber auch der grösste Nachteil in Anbetracht des Use Cases dieser Arbeit. Denn die Zustandslosigkeit führt dazu, dass für jede Datenübertragung ein Kommunikationsoverhead von mehreren Bytes anfällt.

Da die Maschinendaten in Abständen von wenigen Sekunden geschickt werden, führt dies längerfristig zu einer erhöhten Netzwerklast und somit zu längeren Verarbeitungszyklen der gesamten Kommunikation.

Desweiteren ist keine bidirektionale Kommunikation zwischen Client und Server möglich, da der Server beim REST-Paradigma als zustandsloser Anfragepunkt

agiert, welcher nur begrenzt Informationen über den Anfragesteller hat. Zukünftigen Szenarien, bei denen der Server der Maschine Anfragen schicken muss, wären durch HTTP mit REST nicht umsetzbar.

Websocket-Protokoll

Das Websocket-Protokoll baut auf der TCP-Schicht auf und ermöglicht eine bidirektionale Kommunikation über ein Netzwerk [3]. Sowohl der Client wie auch der Server erstellen bei der Initialisierung einen Socket, über welchen Daten gesendet und empfangen werden können. Eine Verbindung besteht dann, wenn ein erfolgreicher Websocket-Handshake durchgeführt wurde, bei welchem sich beide Parteien auf ein Kommunikationsupgrade von HTTP auf Websocket einigen.

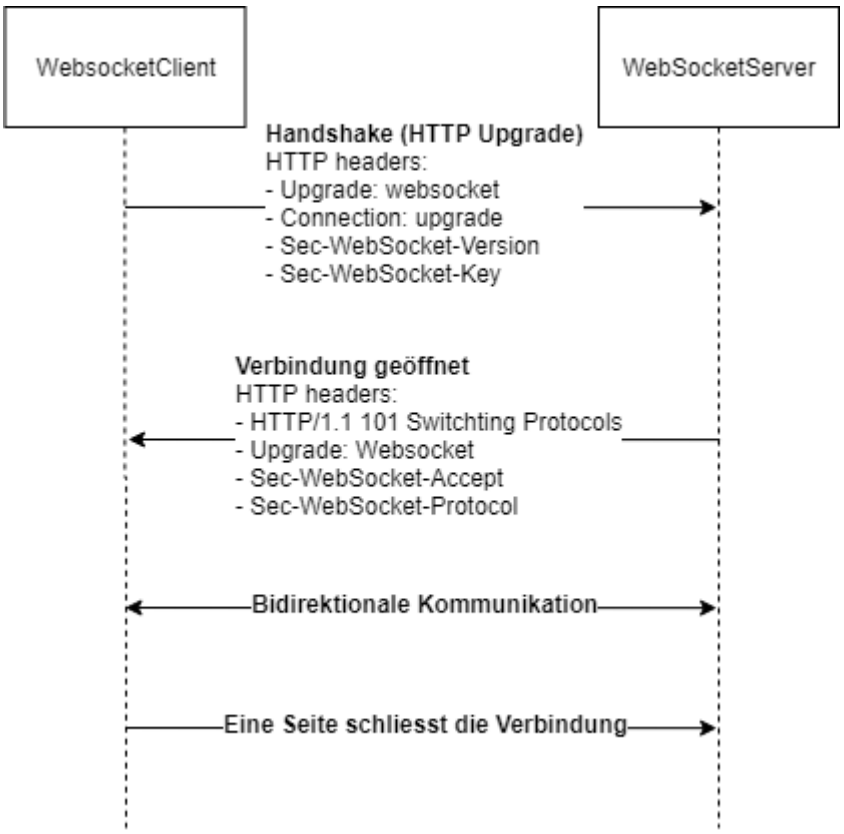


ABBILDUNG 5.1: Ablauf der Kommunikation über das Websocket-Protokoll

Websockets bieten den Vorteil, dass nach einem initialen Kommunikationsaufbau die Verbindung bestehen bleibt, bis eine der beiden Parteien die Verbindung schliesst oder ein unvorhergesehenes Ereignis eintritt - z.B. ein Verbindungsunterbruch durch einen Proxy oder Ähnliches. Der Vorteil dabei ist, dass neben der bidirektionalen Kommunikation eine sehr schnelle Datenübertragung möglich wird, da der Kommunikationsoverhead nur einmal anfällt.

Entscheidung

Für den Use Case eines schnellen und beständigen Datenstreams erfüllt das Websocket-Protokoll alle Anforderungen. Es ist performant, leicht zu implementieren und bietet eine bidirektionale Kommunikation. Für eine spätere Einführung des OPC/UA Standards bildet es zudem die perfekte Grundlage, da dieser ebenfalls auf Sockets beruht [4].

5.1.2 Arduino Websocket Client

Die Maschinendaten werden über die Plattform Arduino gesammelt. Arduino bietet ein komplettes Paket aus Soft- und Hardware um Physical Computing möglichst einfach zu realisieren. Ein weiterer Vorteil ist, dass beide Komponenten open-source sind und von einer breiten Community unterstützt werden.

Das Softwaremodul besteht aus einer IDE und einer an C und C++ angelehnten Programmiersprache. Mittels sogenannten AT-Commands können Websocket-Requests über das GSM-Modul an den Machine Communication Service gesendet werden. Der Websocket Client kann standortunabhängig die Daten an die API senden, sofern die gesendeten Informationen einer registrierten Maschinen- und Sensorinstanz zugeordnet werden können, welche im Vorfeld im Backbone eingetragen werden müssen.

Die detaillierte Konzeption und Realisierung des Arduino Websocket Clients kann aus der entsprechenden Dokumentation zur Arbeit der Abteilung Maschinentechnik entnommen werden.

5.1.3 Machine Communication Service

Der Machine Communication Service wurde als Websocket Server aufgebaut. Er bietet eine Schnittstelle zwischen dem Arduino Websocket Client und dem Product Backbone, welches den Persistenzlayer zur Verfügung stellt. Wie bereits bei der Cockpit Backend API ist die Entscheidung auf eine ASP.NET Core 2.1 Implementation gefallen, mit dem Unterschied, dass über das Websocket-Protokoll kommuniziert wird. Dieser Entscheid dient der Einheitlichkeit und birgt den Vorteil, dass die Schnittstelle plattformunabhängig einsetzbar ist.

Im Vorfeld zur Realisierungsphase des Machine Communication Service wurde ein Architekturprototyp entworfen, um die Machbarkeit zu evaluieren. Hierbei wurde ein Endpoint definiert, welcher ein JSON-Objekt mit den Maschinendaten akzeptiert. Mittels eines simplen Websocket Client Plugins [5], welches im Chrome Store erhältlich ist, wurden JSON-Objekte an den Prototyp gesendet, welcher die empfangenen Daten anschliessend loggte.

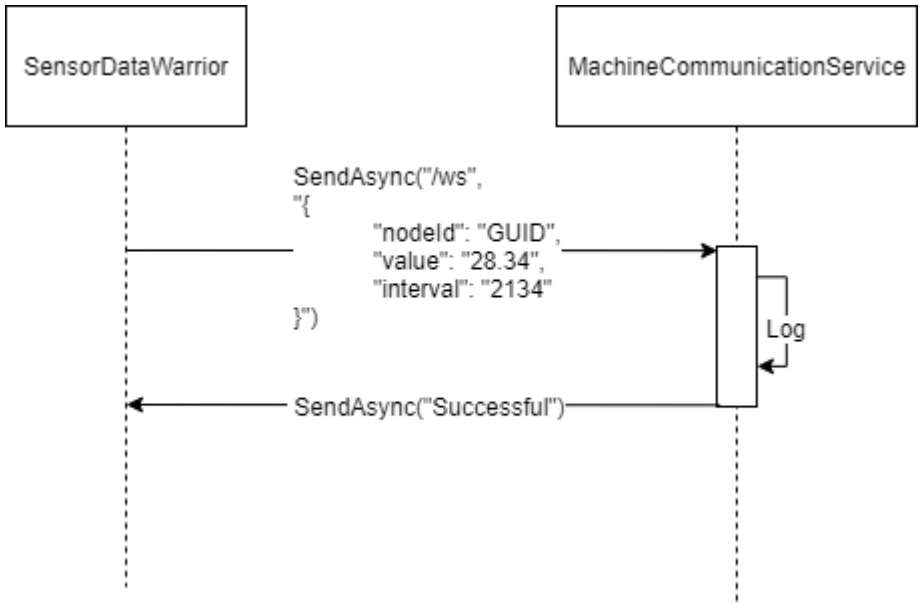


ABBILDUNG 5.2: Machine Communication Service Prototyp

So konnte schon früh erkannt werden, ob der richtige Lösungsansatz gewählt wurde, wie sich grössere Lasten auf das System auswirken und in welcher Form die Daten gesendet werden müssen. Die Ergebnisse haben gezeigt, dass mit dieser Lösung ein richtiger Weg eingeschlagen wird, jedoch noch Bedarf an der Umsetzung weiterer Endpoints im Product Backbone besteht, da sich der Payload als zu gross erwies. Dies wird mit folgender Abbildung verdeutlicht.

VARIANTE 1

Temperature Send Data:


```
{
        "nodeId": "3e3e5966-7027-4b45-8fb9-5d5218ed0a97",
        "nodeName": "Temperatur Umgebung",
        "instanceId": "2f3bb3d4-4615-42d3-842b-65673cc338f0",
        "value": "28.34",
        "dataType": "float",
        "interval": "2134"
        "informationType": "temperature"
}
```


VARIANTE 2

Temperature Metadata:


```
{
        "nodeId": "3e3e5966-7027-4b45-8fb9-5d5218ed0a97",
        "nodeName": "Temperatur Umgebung",
        "instanceId": "2f3bb3d4-4615-42d3-842b-65673cc338f0",
        "dataType": "float",
        "informationType": "temperature"
}
```


Temperature Send Data:


```
{
        "nodeId": "3e3e5966-7027-4b45-8fb9-5d5218ed0a97",
        "value": "28.34"
}
```


ABBILDUNG 5.3: Beispiel von Sensormetadaten- und Sensordaten-Objekte

In der Variante 1 ist die ursprüngliche Konzeption ersichtlich. Diese wurde in der Variante 2 so angepasst, dass statische Daten (Metadaten) nicht bei jedem Request mitgesendet werden, sondern separat im Product Backbone abgelegt und referenziert werden. Daraus ergaben sich zwei neue Endpoints für das Ablegen der Instanz- und Part-Metadaten, welche im Architekturbeschrieb dokumentiert sind.

Der Vorteil an dieser Aufteilung ist, dass sich die für jeden Sensor gesendete Datenmenge spürbar verkleinert. Zudem wird so eine höhere Flexibilität erreicht, da Metadaten geändert werden können, ohne dass Änderungen am Arduino Websocket Client vorgenommen werden müssen.

5.2 Sensor Retrieval Endpoint

Das Cockpit Backend wurde um einen Endpoint erweitert, welcher Sensordaten aus dem Product Backbone zur Verfügung stellt. Er dient als Schnittstelle für das Product Cockpit und ist ebenfalls als ASP.NET Core 2.1 Web API implementiert. Auf Grund der klaren Strukturierung und der Erweiterbarkeit wurde als Architekturstil REST verwendet Der erstellte Prototyp wurde direkt im Cockpit Backend integriert und stellte ein einfacher Endpoint zur Verfügung, um alle Sensordaten zu erhalten.

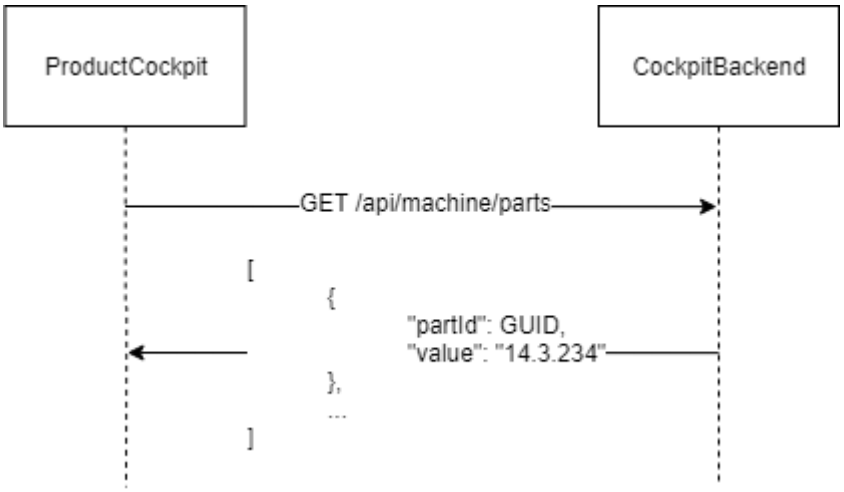


ABBILDUNG 5.4: Sensor Retrieval Endpoint Prototype

5.3 Product Cockpit Erweiterung

Das Product Cockpit wurde um einen Bereich erweitert, welcher die Maschinendaten in geeigneter Form visualisiert. Als geeignete Form ist hier vor allem die textuelle wie auch die graphische Auswertung gemeint. Der Technologie-Stack wurde von der bestehenden Version adaptiert und wird hier nicht weiter erläutert.

5.3.1 Clickable Prototype

Da der Kunde noch keine klare Vorstellung von passenden Darstellungen hatte, wurde in einer ersten Phase ein Clickable Prototype erstellt. Dieser wurde dem Kunden in einer kurzen Präsentation gezeigt und half bei der Entscheidungsfindung.

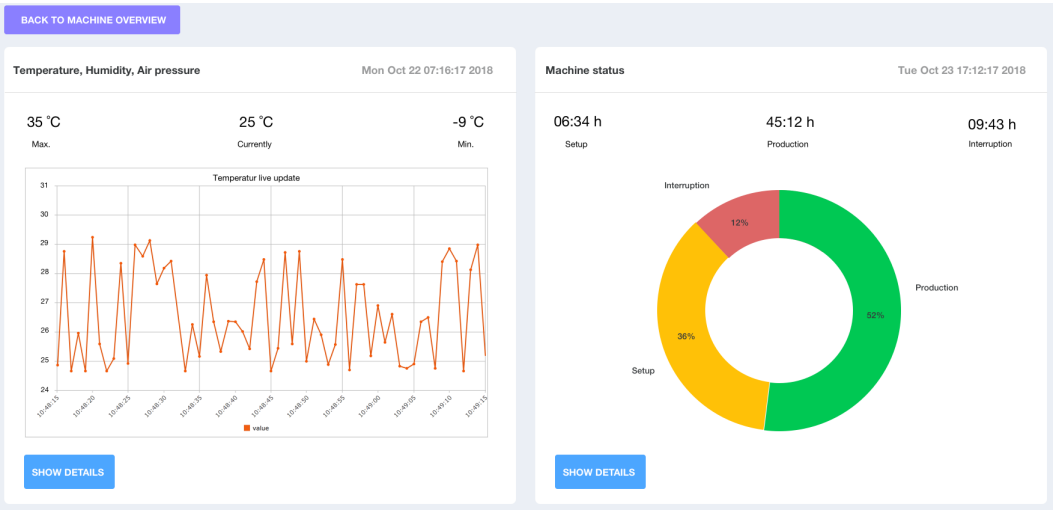


ABBILDUNG 5.5: Beispielpage des Clickable Prototypes

Der komplette Prototyp ist im Anhang E zu finden.

Ergebnisse

Aus dem Prototyp konnten sich verschiedene Anforderungen ableiten lassen, welche in der Anforderungsanalyse (Kapitel 4) genauer beschrieben sind. Eines der wichtigsten Ergebnisse war die Anforderung an die hohe Flexibilität des Cockpits. Es sollte möglich sein, ohne grossen Implementationsaufwand die Darstellungsinformationen der einzelnen Sensoren zu ändern und dynamisch erstellen zu lassen. Ebenfalls war es ein Wunsch, dass Sensorwerte in den Diagrammen miteinander verglichen werden können.

5.3.2 Evaluation der User Experience

Da die Arbeit die Erweiterung eines bestehenden UI vorsieht, wurde dieses genauer untersucht, um Diskrepanzen in der User Experience vorzubeugen.

Das bestehende Product Cockpit enthält den Darstellungsansatz von nebeneinander liegenden Kacheln, welche sich in maximal zwei Spalten aufteilen. Grössere Datenmengen werden in filterbaren Listviews innerhalb der Kacheln dargestellt und navigieren mit einem Doppelklick oder per Buttonklick zu einer Detailview. User Aktionen können über Buttons getriggert werden, die sich innerhalb der Kacheln befinden. Eine Navigationsbar im oberen Bereich der Page ist ausserdem über alle Sichten verfügbar.

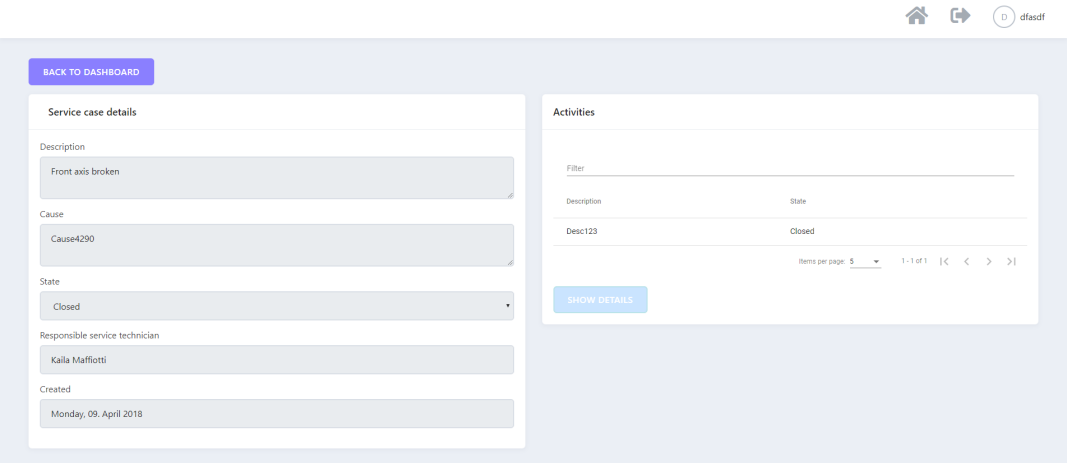


ABBILDUNG 5.6: Bestehendes User Interface

Innerhalb einer Page kann über einen Navigationsbutton, der sich oben links befindet, zurück zur Dashboardübersicht navigiert werden. Nur in einer spezifischen Ausnahme navigiert der Button den Benutzer vom Erstellen einer Activity zurück zur Service Case Übersicht. Aus UX-Sicht ist dies schlecht, da sich der Benutzer gewohnt ist, dass dieser Button ihn zurück zur Dashboardübersicht bringt.

Für die Erweiterung der Visualisierung der Maschinendaten wurde entschieden, dass dieser Button von der Detail- zurück zur Masterview navigiert. Anpassungen an der bestehenden Lösung wurden keine gemacht, jedoch würde es Sinn machen, dieses Verhalten durchgehend zu implementieren.

5.3.3 Vergleich der Grafik-Bibliotheken

Es gibt sehr viele und unterschiedliche Bibliotheken für die Visualisierung von Grafiken in Angular. Nachfolgende werden die evaluierten kurz vorgestellt und anschliessend die Entscheidung erläutert.

Ngx-charts [6] ist ein Grafik Framework das von dem MIT entwickelt wurde. Das Framework verwendet Angular für das Rendering der Grafiken.

JqxChart [7] ist ein Framework, das von über 300'000 Entwickler und von bekannten Unternehmen wie HP, nVidia, Boeing, Xerox, Ericsson, American Express und NASA verwendet wird. JqxChart bietet verschiedene Rendering-Technologien an, darunter auch HTML5 Canvas, SVG und VML.

Angular Chart [8] baut auf AngularJS und Chart.js auf und bietet für Angular Applikationen eine erweiterte Auswahl an Diagrammelementen.

CanvasJs Angular Charts [9] ist ebenfalls ein sehr bekannte Bibliothek und wird regelmässig erweitert. Die Charts werden mit der Technologie HTML5 Canvas gerendert.

Entscheidung

Alle Bibliotheken können für nicht-kommerzielle Zwecke gratis verwendet werden. Ebenfalls bieten alle Bibliotheken ähnliche Grundfunktionalitäten wie Line, Area und Column-Charts an. JqxChart und CanvasJs Angular Charts sind im Markt stark vertreten und werden regelmässig weiterentwickelt. Dementsprechend ist auch die Community und die Dokumentation entsprechend gut.

Aufgrund dessen wurden JqxChart und CanvasJS genauer überprüft, insbesondere weil für die Erweiterung des User Interface komplexe und zeitgemässe Grafiken gefordert wurden. CanvasJS unterstützt die Rendering Technologie HTML5 Canvas, bei JqxChart sind es alle verfügbaren, also HTML5 Canvas, SVG und VML [10]. Die Entscheidung fiel schlussendlich auf die Bibliothek JqxChart, da sie eine sehr spezifische Konfiguration der Charts im Code erlaubt. Die aktuellen Use Cases erfordern kein 3D Rendering von Grafiken [11]. Ebenfalls ein positiver Aspekt von JqxChart ist, dass die Library relativ lightweight ist.

5.4 Weitere Entscheide

5.4.1 Flexibility

Über die gesamte Konzeption musste berücksichtigt werden, wie eine möglichst hohe Flexibilität gewährleistet werden kann. In zukünftigen Lösungen muss es zum Beispiel möglich sein, neue Maschinen- / Sensorkombinationen an den Service anschliessen und im Frontend auswerten zu können.

Im Persistence Layer wurde dies so gelöst, dass ein sehr generisches JSON-Objekt von der API akzeptiert und im Backbone abgespeichert wird. Grundanforderungen an das JSON-Objekt sind lediglich eine passende Instanz- und Part-ID, sowie ein entsprechender Sensorwert, welcher einem Property zugeordnet werden kann.

Wie bereits erwähnt, werden übrige Informationen (Metadaten), wie der Sensorname oder die Seriennummer, separat im Backbone abgelegt, da sich diese Daten selten bis nie ändern. Instanz-IDs, Part-IDs und Metadaten werden in einer ersten Version händisch im Product Backbone gepflegt. Eine automatische Registrierung ist für Folgearbeiten angedacht.

Ziel dieser Lösung ist es, dass möglichst wenig Implementationsaufwand anfällt, sobald eine neue Maschinen- / Sensorkombination an den Service angeschlossen wird. Einzig im Frontend kann es zu Anpassungen kommen, falls zum Beispiel ein neues Diagramm für einen bestimmten Sensor erstellt werden muss. Doch auch hier sollte ein möglichst generischer Ansatz implementiert werden.

5.4.2 OPC Unified Architecture

Als grosser M2M-Kommunikationsstandard gilt heute OPC Unified Architecture (OPC/UA) [12]. Dieser definiert ein eigenes Protokoll, über welches Maschinen mit ihrem Umsystem kommunizieren können. Desweiteren werden Security, Permission und Data Access Konzeptionen vorgegeben. Es gibt diverse Implementationen dieses Standards, welche als Frameworks zur Verfügung stehen. Eine Implementation die auf ASP.NET Core aufbaut ist als Open-Source-Framework über GitHub verfügbar.

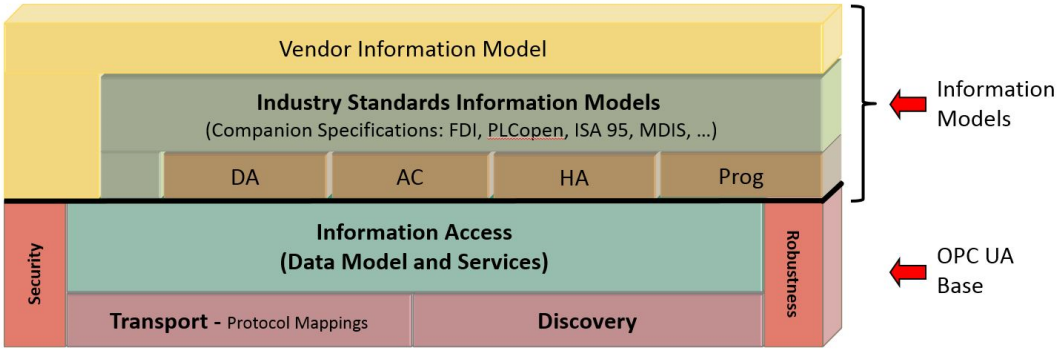


ABBILDUNG 5.7: OPC/UA Standardarchitektur

Trotz Eignung dieses Frameworks wurde in dieser Arbeit bewusst darauf verzichtet. Gründe hierfür waren unter anderem, dass eine Umsetzung viel Einarbeitungsaufwand mit sich bringt, da im Team keine Erfahrungswerte vorhanden sind und die Onlinedokumentation dazu relativ dürftig ist. Zudem hätte ein Framework unnötigen Boilerplate-Code verursacht, was die Wartbarkeit verschlechtert hätte.

Trotzdem wurde darauf geachtet, dass nahe an dem Standard gearbeitet wird, damit eine spätere Portierung auf diesen möglichst ohne grossen Aufwand möglich ist. So wurden Architekturentscheide aus OPC/UA in die Arbeit miteinbezogen, wie z.B. dass die Kommunikation auf Websockets beruht.

Falls die Notwendigkeit einer Portierung besteht, müsste in einem ersten Schritt der Kommunikationsaufbau umgeschrieben werden. Die Kommunikation würde neu über das Binärprotokoll opc.tcp erfolgen und müsste verschiedene Sicherheitsaspekte berücksichtigen. Desweiteren würden die Daten nicht mehr als Custom-Objekte gesendet werden, sondern als sogenannte Nodes. Diese werden von OPC/UA vorgegeben und repräsentieren das Address Space Model.

5.4.3 Error Handling

Ein ausgereiftes Error Handling ist ausserhalb des Scopes dieser Arbeit. Zwar soll im Frontend ersichtlich sein, wenn ein Sensor keine aktuellen Daten mehr liefert, jedoch werden jegliche kommunikationsspezifische Probleme jeweils im Layer abgefangen, in welchem diese auftreten, sprich sie werden nicht an das Frontend propagiert.

5.4.4 Security, Availability, Scalability und Performance

Für die erste Version der Schnittstellen wurden die Aspekte Security, Availability, Scalability und Performance aus zeitlichen Gründen bewusst vernachlässigt. Damit das Backbone dennoch vor Angriffen geschützt ist, können Daten nur abgelegt werden, wenn Kenntnisse über die abzulegende Datenstruktur vorhanden sind.

Die Kommunikation über das Internet ist momentan noch unverschlüsselt und kann von Dritten theoretisch abgehört werden. Von einem Upgrade auf HTTPS ist daher in einer späteren Version zu raten.

Die Availability der Services ist nicht gewährleistet. Manipulative Angriffe wie z.B. DDoS sind theoretisch denkbar. Um sich von solchen Angriffen schützen zu können, müsste die Serverinfrastruktur mit entsprechenden Gegenmassnahmen ausgestattet werden.

Eine horizontale Skalierung der Services ist bereits heute denkbar, aber nicht umgesetzt. Hierfür müsste man mehrere Instanzen der Services laufen lassen und diese via Loadbalancer oder Ähnlichem gleichverteilt betreiben.

Die Performance ist unter Berücksichtigung der nicht-funktionalen Anforderungen (Kapitel 4) eingehalten. Ob der Service aber bei sehr hoher Last - hunderten von Instanzen - performant bleibt, wurde nicht weiter untersucht.

5.5 Werkzeuge und Tools

Es wurden verschieden IDEs für die Backend- und Frontend-Entwicklung analysiert. Nachfolgend ist beschrieben, wie die verschiedenen IDEs evaluiert wurden und aufgrund welcher Eigenschaften die Entscheidung getroffen wurde.

5.5.1 Frontend

Für die Frontend-Entwicklung wurde Visual Studio Code und WebStorm evaluiert. Ein essentielles Entscheidungskriterium war, dass die IDE lightweight ist und sie nach eigenen Anforderungen erweitert werden kann. Visual Studio Code konnte vor allem in dessen Simplizität überzeugen. Es kann mit spezifischen Extensions erweitert werden, womit schliesslich alle Anforderungen erfüllt werden konnten.

Die verwendeten Extensions sind nachfolgend aufgeführt.

- Angular 7 Snippets [13]
- Angular Files [14]
- Angular Follow Selector [15]
- Auto Import [16]
- Debugger for Chrome [17]
- GitLens - Git supercharged [18]
- Path Intellisense [19]
- TSLint [20]

5.5.2 Backend

Bei der Evaluation der Backend IDE wurde anhand der Entscheidung, dieselbe Technologie wie in der Vorarbeit zu verwenden (ASP.NET Core), klar, dass das Produkt Visual Studio 2017 von Microsoft am besten geeignet ist. Visual Studio ist für ASP.NET Core ausgelegt und bietet eine volle Integration des Frameworks. Deshalb wurden keine weiteren IDEs evaluiert. Als Erweiterung wurde ReSharper [21] verwendet. Dies half unter anderem durch eine erweiterte Autocompletion, Programmier- und Formatierungs- Standards und das Verhindern von Code Smells.

Kapitel 6

Architektur

Wie in der Lösungskonzeption bereits erwähnt, wurde auf eine lose Kopplung der einzelnen Applikationen geachtet. Die folgende Systemübersicht zeigt dies.

Der Communication Service dient als Schnittstelle zwischen dem Backbone und dem Arduino Websocket. Die Schnittstelle sorgt dafür, dass die Maschinendaten, die vom Arduino Websocket Client gesendet werden, an das Backbone gelangen. Im Backbone, welches vom IPEK der HSR entwickelt wurde, werden die Maschinendaten im spezifiziertem Format persistiert, welches in diesem Kapitel genauer beschrieben wird.

Damit die Maschinendaten im Frontend visualisiert werden können, wurde das bestehende Backend so erweitert, dass die neuen Endpoints im Backbone angesprochen werden können. Durch diese Erweiterung stellt das Backend diverse Endpoints für das Frontend zur Verfügung, um die Maschinendaten, die zuvor über den Communication Service gespeichert wurden, auszulesen.

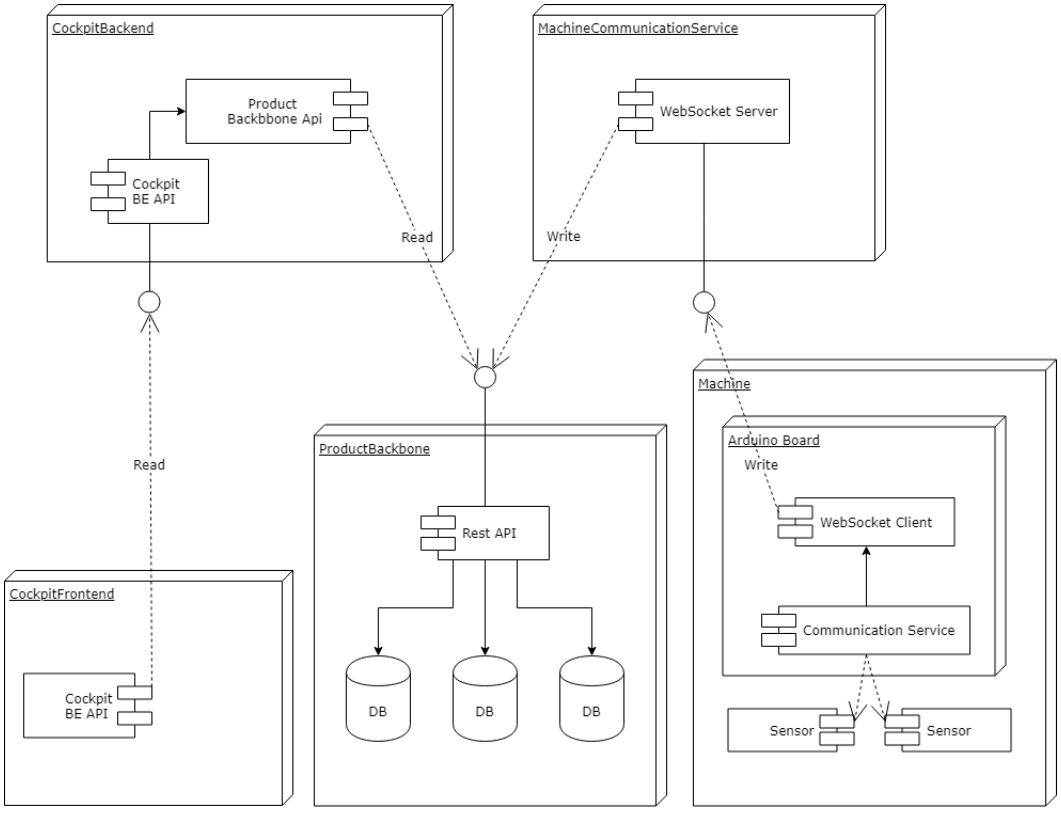


ABBILDUNG 6.1: Übersicht des Gesamtsystems

6.1 Persistenz der Maschinendaten

Im Kapitel 5 wurde erläutert, wie das Websocket Protokoll für die Kommunikation zwischen der Maschine und dem Machine Communication Service eingesetzt werden kann. In diesem Abschnitt wird genauer darauf eingegangen und die übertragenen Datenpakete beschrieben.

6.1.1 Datenübermittlung zum Machine Communication Service

Der Machine Communication Service besteht aus einem Socket und einem Listener, der auf eingehende Datenpakete auf diesem Socket hört und fungiert als Websocket Server. Sobald ein Websocket Client, in diesem Falle die Maschine selbst, eine Verbindung auf den Endpoint „/ws“ macht und dabei im HTTP Header vermerkt, dass er einen Websocket Upgrade machen will, antwortet der Server mit der Nachricht „101 Switching Protocol“.

Eine aktive und permanente Verbindung ist nun aufgebaut und der Server startet einen neuen Thread für die Kommunikation mit diesem Client. Der Thread durchläuft eine while-Schlaufe, in welcher synchron auf ankommende Datenpakete gewartet wird. Im Main-Thread wird weiterhin auf einkommende Verbindungen gewartet, was bedeutet, dass sich mehrere Clients gleichzeitig auf einen Socket verbinden können.

Sobald ein Datenpaket eintrifft, wird dieses von der Binärenform in einen String konvertiert. In den Settings ist definiert, dass ein ankommendes Datenpaket nicht grösser als 4 KB sein darf, was in diesem Anwendungsfall bei weitem reicht. Eine Erhöhung der Message-Size ist theoretisch möglich. Sie sollte aber aus Sicherheitsgründen so klein wie möglich gehalten werden.

Die empfangene Message wird gegen ein strikt definiertes JSON-Schema geprüft. Dies dient primär der Sicherheit, damit keine unpassenden oder böswilligen Daten in das Product Backbone geschrieben werden können. Sollte die Message nicht dem definierten Schema entsprechen, antwortet der Server mit der Nachricht „Invalid machine data“. Bei einer passenden Nachricht lautet die Antwort „Successfully inserted new entry of type [Instance oder Part]“.

Das erhaltene Datenpaket wird nun in ein Model-Objekt konvertiert, über die Business Logik passend aufbearbeitet und in das Product Backbone gespeichert. Je nach Typ (Instanz oder Part) wird ein anderer Product Backbone Endpoint angesprochen.

6.1.2 Erläuterung der gesendeten Daten-Pakete

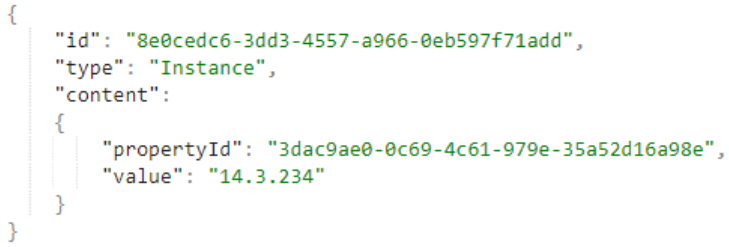


ABBILDUNG 6.2: Übermittelte Maschinendaten

id repräsentiert je nach Typ eine Instanz- oder Part-Id und muss eine valide GUID sein.

type definiert, ob es sich um Maschinen-, oder Sensordaten handelt, damit daraus der richtige Endpoint im Product Backbone angesprochen werden kann.

content enthält die eigentlichen Maschinendaten, die in das Product Backbone gespeichert und im Cockpit Frontend angezeigt werden.

propertyId definiert zu welchem Property der gesendete Wert gehört. Auch diese Id muss eine valide GUID repräsentieren. Die Bedeutung der Properties wird im nächsten Abschnitt erklärt.

value speichert den gesendeten Wert. Er ist immer vom Typ String und wird gemäss den Metadaten in einen passenden Datentyp konvertiert.

TABELLE 6.2: Beschreibung der Maschinendaten

6.1.3 Product Backbone

Das Product Backbone bildet im System die Datenhoheit und speichert alle Informationen der Umsysteme. Es wurde vom IPEK der HSR bereitgestellt. In diesem Abschnitt werden die Endpoints zu den Maschinendaten genauer beschrieben. Eine umfangreiche Dokumentation zum Product Backbone findet sich in einem separaten Dokument.

Für die Verarbeitung der Maschinendaten stellt das Product Backbone folgende Endpoints zur Verfügung.

GET	*https://[host]/api/machine/instances/instanceid/data*
POST	*https://[host]/api/machine/instances/instanceid/data*
GET	*https://[host]/api/machine/instances/instanceid/metadata*

POST	*https://[host]/api/machine/instances/instanceid/metadata*
GET	*https://[host]/api/machine/instances/instanceid/parts/partid/data*
POST	*https://[host]/api/machine/parts/partid/data*
GET	*https://[host]/api/machine/instances/instanceid/parts/partid/metadata*
POST	*https://[host]/api/machine/parts/partid/metadata*

TABELLE 6.4: Relevante Product Backbone Endpoints

Sowohl für Maschineninstanzen (Instances), wie auch für Sensorinstanzen (Parts) werden jeweils Endpoints für das Abrufen bzw. Speichern der Maschinendaten bereitgestellt. Alle gespeicherten Daten werden mit einem Zeitstempel versehen, der den Zeitpunkt der Speicherung festhält.

Es wird zwischen *Data* und *Metadata* unterschieden, wobei *Data* alle dynamischen und *Metadata* alle statischen Maschinendaten umfassen. Über eine GUID können die einzelnen Instanzen bzw. Sensoren identifiziert werden.

Die zurückgegebene Datenmenge kann mittels Zeitbereich eingeschränkt werden. Hierfür müssen bei den *GET* Requests lediglich ein Datum über die URL-Parameter *from* und/oder *to* mitgegeben werden.

Die Daten, welche der Machine Communication Service empfängt, werden direkt in die jeweiligen */data* Endpoints gespeichert. Metadaten werden in einer ersten Version händisch über die */metadata* Endpoints erfasst. Später ist angedacht, dass diese Daten über umliegende Systeme erfasst und in das Product Backbone gespeichert werden.

Persistierte Daten können grundsätzlich nicht verändert werden. Stattdessen muss ein neuer Eintrag erstellt werden, um den alten Eintrag als obsolet zu kennzeichnen. Über den Zeitstempel kann dann ausfindig gemacht werden, welcher Eintrag der aktuellere ist. Der Vorteil von diesem Ansatz ist, dass die Daten automatisch historisiert werden.

6.1.4 Erläuterung der persistierten Metadaten-Pakete

ABBILDUNG 6.3: Persistierte Maschinenmetadaten

name beschreibt die Maschine und dient als Anzeigename im Frontend.

type definiert einen Maschinentyp und wird im Frontend angezeigt.

serialId stellt die Seriennummer der Maschine dar.

properties enthält eine Liste von Property-Objekten. Maschinen- und Sensorinstanzen können mehrere Properties enthalten, welche verschiedene Informationstypen definieren. Bei einem Sensor wären das bspw. ein Value-Property für den gesendeten Wert und ein Error-Property für jegliche Error. Es können beliebig viele Properties definiert werden.

propertyId enthält eine GUID um das Property referenzieren zu können.

dataType wird gebraucht, um den gesendeten String in den passenden Datentyp konvertieren zu können.

displayChart ist ein Boolean und gibt an, ob ein Property als Chart, oder als Text gerendert wird. Diese Lösung ist ein Workaround und müsste zukünftig durch eine UI-Konfiguration abgelöst werden, in welcher für ein Property definiert werden kann, wie dieses im Frontend angezeigt wird.

label definiert das im UI angezeigte Label. Auch dieser Wert könnte zukünftig durch eine UI-Konfiguration abgelöst werden.

interval enthält die Information, wie oft ein Sensordaten sendet. Diese Angabe ist in Millisekunden und gibt dem Frontend an, wie oft dieses nach neuen Werten fragen soll

TABELLE 6.6: Beschreibung der persistierten Maschinendaten

```
{
    "name": "MotorTempPart1",
    "type": "sensor",
    "serialId": "S01234567",
    "informationType": "temperature",
    "minValue": -20,
    "maxValue": 60,
    "loadingTimeSpan": 100000,
    "properties": [
        {
            "propertyId": "5c107bf8-6e1d-4d2c-82d4-5f69b2eadfbc",
            "dataType": "float",
            "displayChart": false,
            "label": "Temperatur",
            "interval": "5000"
        }
    ]
}
```


ABBILDUNG 6.4: Persistierte Sensormetadaten

name beschreibt den Sensor und dient als Anzeigename im Frontend.

type definiert einen Parttyp. Für die momentane Implementation sind alle Parts vom Typ „Sensor“. In zukünftigen Szenarien könnten aber auch zu anderen Parttypen Daten persistiert werden.

serialId stellt die Seriennummer des Sensor dar.

informationType beschreibt, was für einen Informationstyp der jeweilige Sensor sendet. Informationstypen können zum Beispiel sein, „temperature“, „location“ oder „humidity“

minValue wird benötigt, um die untere Grenze des möglichen Wertebereichs des Sensors zu definieren. Im Cockpit Frontend kann dank diesem Wert ein passendes Chart gerendert werden.

maxValue wird benötigt, um die obere Grenze des möglichen Wertebereichs des Sensors zu definieren. Im Cockpit Frontend kann dank diesem Wert ein passendes Chart gerendert werden.

loadingTimeSpan gibt für das Cockpit Frontend an, wie weit zurück in die Vergangenheit Sensordaten geladen werden je Chart, damit nicht eine zu hohe Datenmenge dargestellt wird. Dieser Wert sollte zukünftig über eine UI-Konfiguration gesetzt werden und dient momentan als Workaround.

properties enthält eine Liste von Property-Objekten, welche gleich geliedert sind, wie oben bereits beschrieben.

TABELLE 6.8: Beschreibung der persistierten Sensordaten

6.2 Cockpit Backend

Das Cockpit Backend wurde bereits in der Vorgängerarbeit erstellt. Es bildet die Grundlage für das Cockpit Frontend und besteht aus einer Web API, welche auf das Product Backbone aufbaut.

6.2.1 Relevante Endpoints

Um die zusätzlichen Anforderungen an das Frontend unterstützen zu können, musste das Backend um folgende Endpoints erweitert werden.

1) GET	*https://[host]/api/machine/instances/instanceid*
2) GET	*https://[host]/api/machine/instances/instanceid/properties/propertyid*
3) GET	*https://[host]/api/machine/instances/instanceid/parts*
4) GET	*https://[host]/api/machine/instances/instanceid/parts/partid*

TABELLE 6.10: Neu implementierte Cockpit Backend Endpoints

Der Aufbau der Endpoints wurde stark an die Definition des Product Backbones angelehnt. Die Endpoints orientieren sich an den benötigten Daten im Frontend, um ein möglichst performanter Datentransfer zu ermöglichen.

Über Endpoint **1)** können die *Metadata* und *Data* zusammengeführt in einem JSON-Objekt für eine bestimmte Instanz angefragt werden. Dabei werden ausschliesslich die aktuellsten Daten zurückgegeben. Dieser Call wird beim initialen Laden der Maschinenoverview getätigt.

Der Endpoint **2)** dient zum Abfragen der *Data* eines bestimmten Properties. Es werden alle *Data* dieses Properties zurückgegeben, es sei denn, es wird mittels *from* Parameter eine zeitliche Einschränkung gemacht. Mittels *from* können anhand eines Timestamps immer die aktuellsten Propertydaten angefordert werden. Dieser Call wird in kurzen Intervallen sowohl auf der Maschinenübersicht, auf der Sensorenübersicht, wie auch auf der Sensordetailansicht ausgeführt.

Mittels Endpoint **3)** können alle Sensoren zu einer bestimmten Maschine zurückgegeben werden. Es werden jeweils die aktuellsten *Metadata* und die *Data* der Sensoren zurückgegeben. Da das Laden aller *Data* sehr teuer ist, werden hier nur diejenigen geladen, die sich in einem bestimmten Zeitbereich befinden. Der Zeitbereich wird über das Property *loadingTimeSpan*, welches weiter oben beschrieben wurde, definiert. Dieser Call wird auf der Sensorenübersicht getätigt.

Der Endpoint **4)** stellt die Daten eines bestimmten Sensors zur Verfügung. Es werden immer die aktuellsten *Metadata* und alle *Data* des Sensors zurückgegeben. Dieser Call wird in der Sensordetailansicht getätigt.

6.2.2 Beschreibung der zusammengeführten Daten

Das folgende Bild zeigt einen JSON-Dump des **1)** Endpoints. Da die anderen Endpoints ähnliche Daten zurückgeben, wurde nicht weiter auf diese eingegangen.


```
{
    "metadata": {
        "name": "Crimp Machine",
        "type": "n1-standard-96",
        "serialId": "M01234567",
        "properties": [
            {
                "propertyId": "e9b7a094-d0cc-4411-a1df-f8451f0677df",
                "dataType": "float",
                "label": "Latitude",
                "displayChart": false,
                "unit": "°",
                "interval": 604800
            },
            {
                "propertyId": "3d2834c2-23e5-4a61-8f95-5bab7599990b",
                "dataType": "float",
                "label": "Longitude",
                "displayChart": false,
                "unit": "°",
                "interval": 604800
            }
        ]
    },
    "content": [
        {
            "propertyId": "7bb00381-b88b-450d-8c96-4f8a80010052",
            "timestamp": "2018-12-18T22:16:37.745+01:00",
            "value": "4.535.2"
        },
        {
            "propertyId": "0ab0c6dd-5753-49ba-b0da-5c1978efb179",
            "timestamp": "2018-12-18T22:16:31.9+01:00",
            "value": "Windows"
        }
    ]
}
```


ABBILDUNG 6.5: Beispiel von zusammengeführten Daten

metadata beschreibt die *Metadata* einer Instanz. In diesem Beispiel werden die aktuellsten Maschinenmetadaten zurückgegeben.

content definiert die *Data* einer Instanz. In diesem Beispiel werden die aktuellsten Maschinendaten zurückgegeben.

TABELLE 6.12: Beschreibung der zusammengeführten Daten

6.3 Cockpit Frontend

Auch das Cockpit Frontend wurde bereits in der Vorgängerarbeit erstellt. Das Frontend wurde um die jeweiligen Ansichten erweitert, um die Maschinendaten visualisieren zu können. Die Maschinendaten werden über die neuen Endpoints des Cockpit Backends geladen.

Für die Visualisierung der Daten wurden folgende drei Grafiktypen erstellt und für die jeweiligen Typen der Daten wiederverwendet.

1)	*Detail Line Chart*
2)	*Live Updates Chart*
3)	*Stacked Column Chart*

TABELLE 6.14: Grafiktypen

In den nachfolgenden Unterkapitel werden die Ansichten erläutert und gezeigt, wo diese Grafiktypen verwendet wurden.

6.3.1 Instanzdaten

Maschinendetails

Die Metadaten werden, wie bereits in der Vorgängerversion, auf der Instanzebene angezeigt

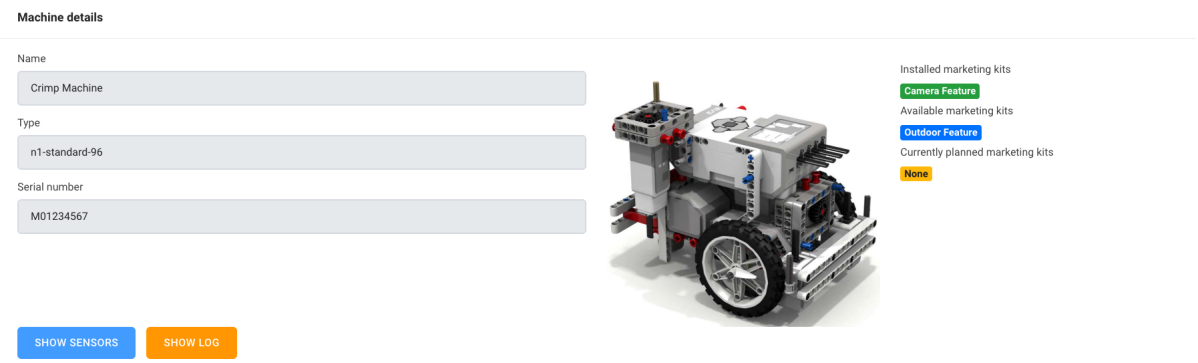


ABBILDUNG 6.6: Maschinendetails

Instanz *Properties*

Die *Properties* einer Instanz werden in einer Tabelle, in einem separaten Card angezeigt. Alle Werte in der Tabelle und in der nachfolgenden Ansicht werden generische geladen und angezeigt. Dies bedeutet, dass für ein neues *Property* einer Instanz nur die Metadaten einer Instanz erweitert werden müssen. Im Frontend sind keine Änderungen nötig.

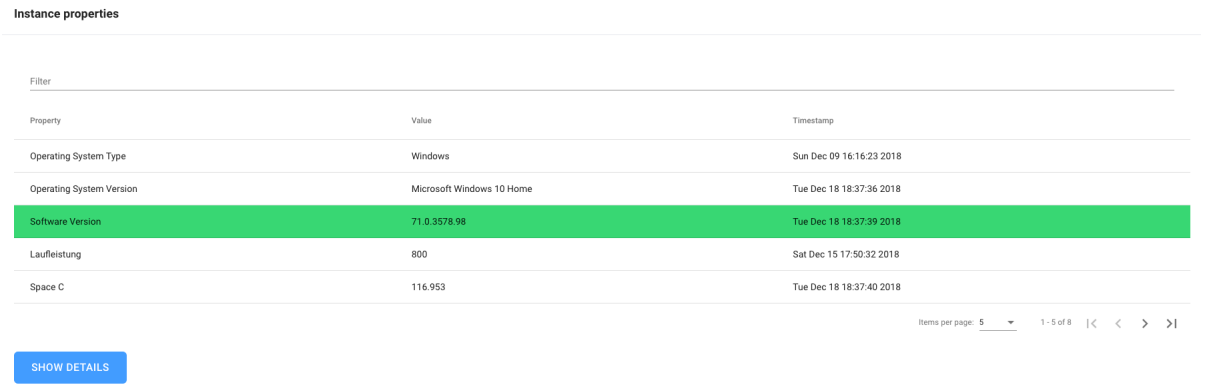


ABBILDUNG 6.7: Instanz *Properties*

Instanz *Property* Details

In der Detailansicht der Instanz *Properties* wird anhand des dataTypes des *Properties* entschieden, mit welchem Grafiktyp die Daten gerendert werden. Ist das *Property* vom Typ *string* werden die Daten in der Grafik **Stacked Column Chart** gerendert.

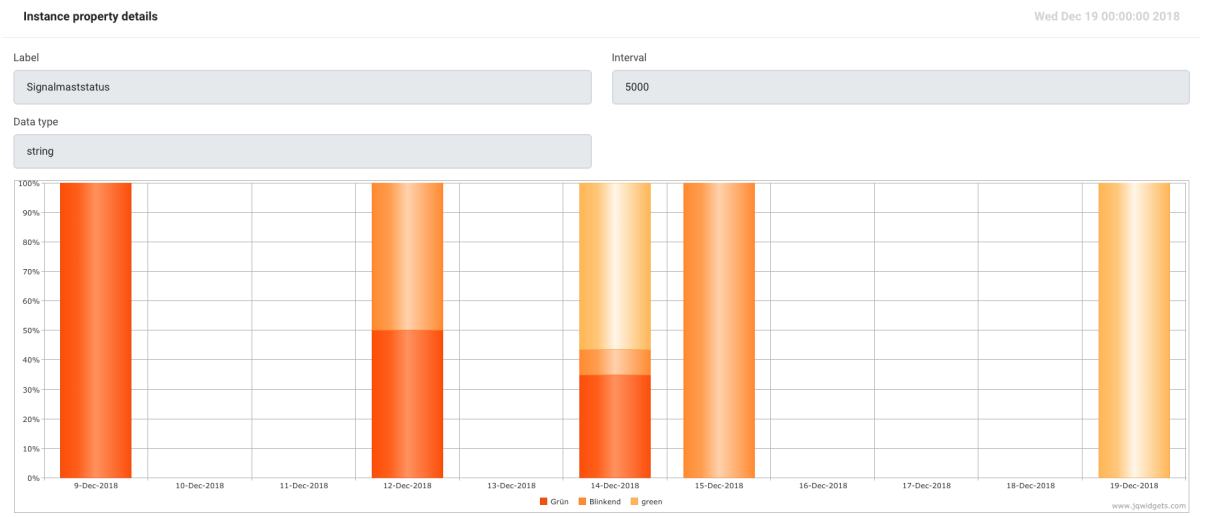


ABBILDUNG 6.8: Instanz *Property* Stacked Column Chart

Für alle anderen Typen, insbesondere nummerische Typen, wird der Grafiktyp **Detail Line Chart** verwendet.

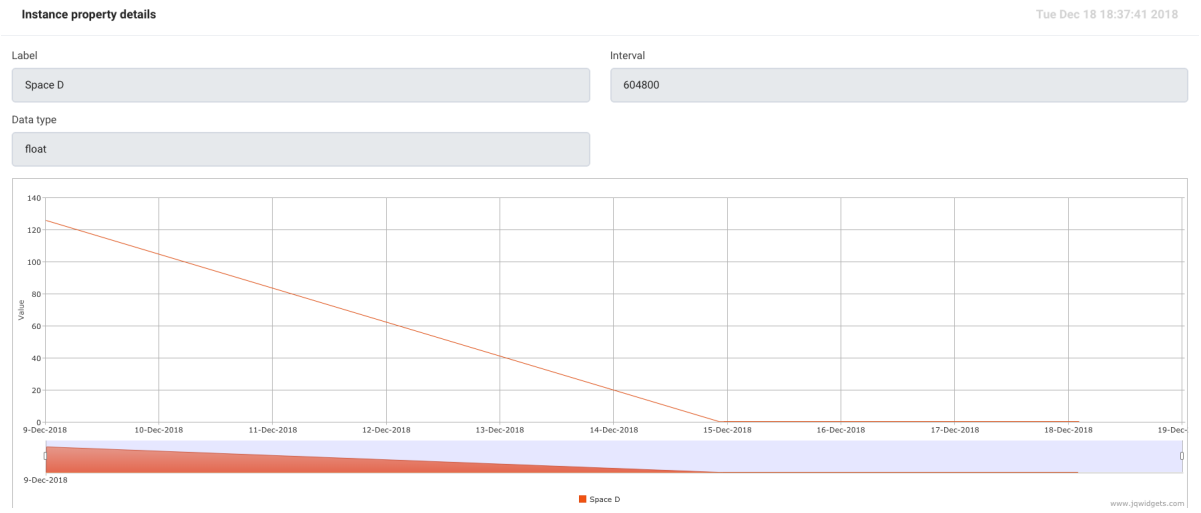


ABBILDUNG 6.9: Instanz *Property* Line Chart

Instance Error Log

Ebenfalls Teil der Instanz *Properties* sind Error-Meldungen der Maschine. Diese werden in einer separaten Ansicht visualisiert. Anhand des Intervalls lässt sich konfigurieren in welchem Abstand die Daten aktualisiert werden sollen. Sind die Daten nicht mehr aktuell, wird der Benutzer über eine rote Box benachrichtigt.

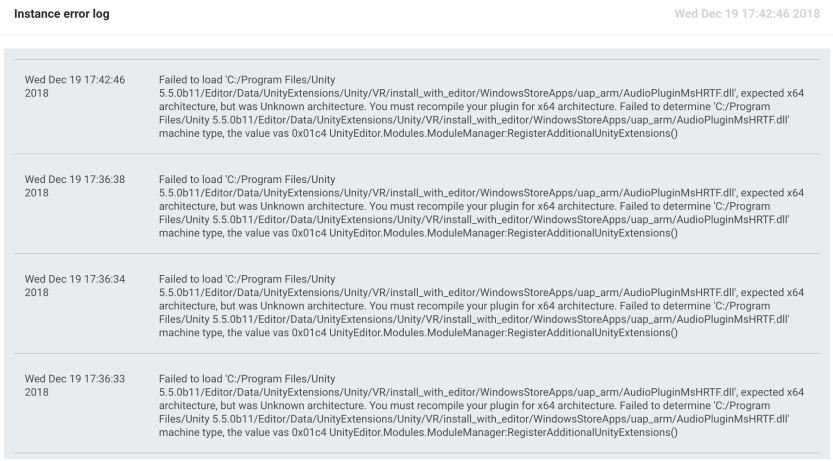


ABBILDUNG 6.10: Instanz Error Log

6.3.2 Partdaten

Part Übersicht

Für jeden Part oder Sensor gibt es in der Übersicht einen eigenen Card mit den Livedaten, die mit dem Grafiktyp **Live Updates Chart** gerendert werden. In diesem Grafiktyp fliessen die Daten von rechts nach links. Das heisst der aktuellste Werte

ist immer auf der rechten Seite zu finden. Sind die Daten nicht mehr aktuell, wird der Benutzer über eine rote Box benachrichtigt.

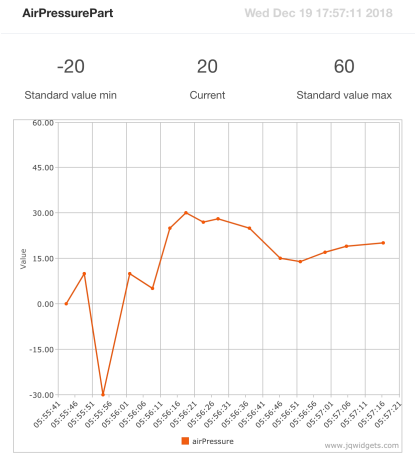


ABBILDUNG 6.11: Part Übersicht - Live Updates

Part Details

Neben den Part Metadaten dient diese Ansicht für einen Überblick über alle existierenden Daten. Durch den *RangeSelector* können die Daten auf einen gewissen Zeitraum eingeschränkt werden. Für die Visualisierung der Daten wurde der Grafiktyp **Detail Line Chart** verwendet.

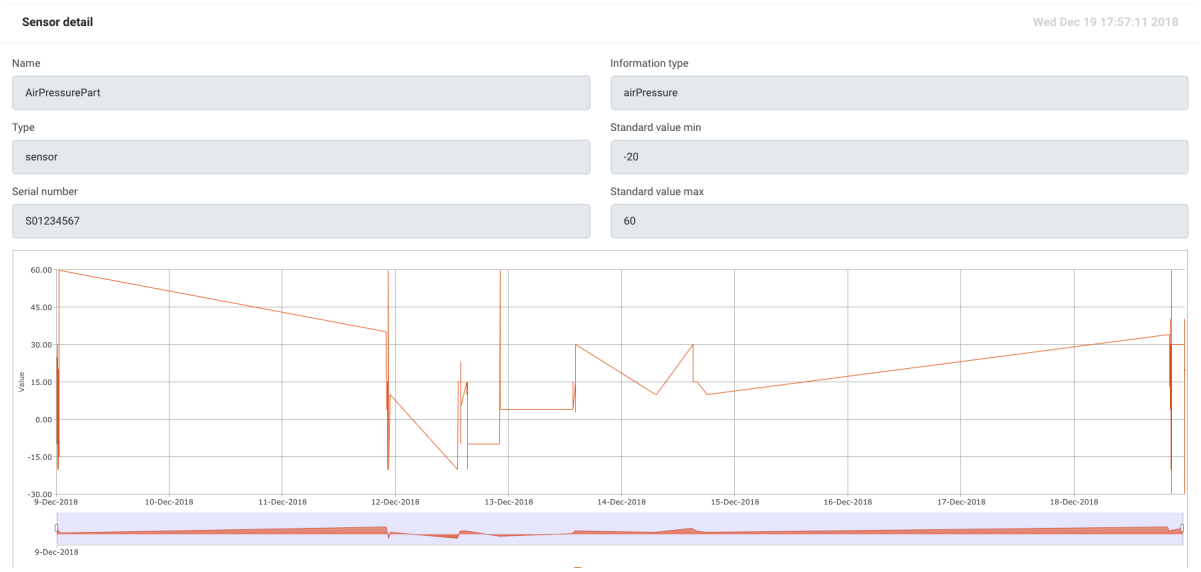


ABBILDUNG 6.12: Part Details

Kapitel 7

Qualitätssicherung

7.1 Code Review und Refactoring

Für jedes Ticket wurde ein neuer Branch auf Basis des Masters erstellt. Sobald alle Anforderungen eines Tickets umgesetzt waren, wurde ein Pull-Request für diesen Branch erstellt. Dies hatte zur Folge, dass jede Änderung durch einen zweiten, unabhängigen Entwickler angeschaut werden musste. Das Feedback zum Pull-Request wurde, falls nötig, bilateral besprochen.

Refactorings wurden unabhängig von der Entwicklung der Anforderungen durchgeführt. Dabei wurde unter anderem auf eine lose Kopplung zwischen Klassen und das Single Responsibility Principle [22] geachtet.

7.2 Metriken

Für die IDE Visual Studio Code gibt es kein Plugin um Codemetriken über das gesammte Projekt zu generieren. Daher wurde das Plugin Statistics [23] von der IDE WebStorm [24] verwendet.

Nachfolgende eine Auflistung der erstellten Files und Anzahl Codezeilen. In der Auflistung ist zu beachten, dass die angegebene Zahl jeweils die differenz des Gesamten Projekts zur Vorarbeit repräsentiert. Folglich kann es sein, dass bestehende Files angepasst oder gelöscht wurden.

Extension	Codezeilen	Anzahl Datein
Typescript	3181	92
HTML	361	18
SCSS	95	3

TABELLE 7.2: Frontend Codemetriken

Um die zyklomatische Komplexität zu ermitteln wurde das Plugin CodeMetrics verwendet. Über das Plugin wird der Entwickler über die Komplexität der aktuellen Methode informiert, dies hilft klaren und verständlichen Code zu schreiben.

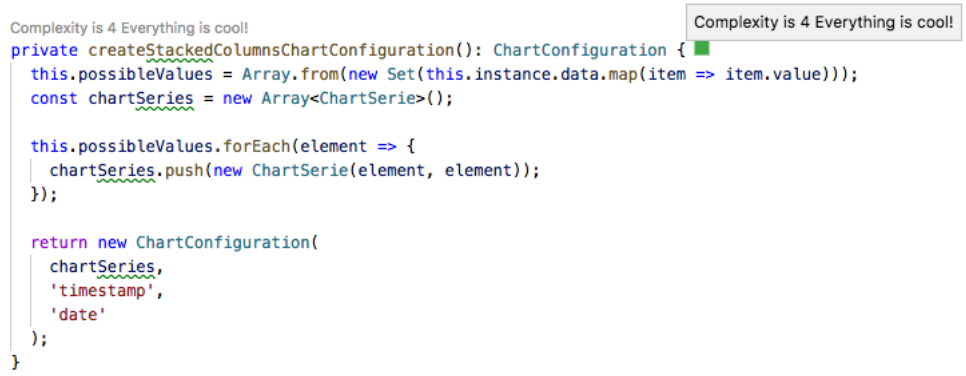


ABBILDUNG 7.1: Zyklomatische Komplexität

Kapitel 8

User Acceptance Test

Der grösste Teil der Arbeit war der Aufbau des Kommunikationskanals zwischen der Maschine und dem Frontend, was nicht durch einen User Acceptance Test geprüft werden kann. Deshalb konzentriert sich dieses Kapitel vor allem auf das Frontend.

8.1 Ablauf

Um eine Rückmeldung zu bekommen, wie und ob sich die Applikation intuitiv steuern lässt, wurde mit zwei Personen Tests durchgeführt.

Raphael Rechsteiner	HSR Informatik Student im 5. Semester.
Ozan Sahin	HSR Machinentechnik Student im 6. Semester.

TABELLE 8.2: Testpersonen

Die Benutzerführung bis zu den Maschinendetails wurde nicht getestet, da dies bereits in der Vorarbeit realisiert wurde.

8.2 Durchführung

Da die Ansichten nur mit der Rolle Servicetechniker zugänglich sind, wurden alle Tests als Servicetechniker durchgeführt.

Test Case 1	Überprüfen Sie in welchem Signalmaststatus sich die Maschine aktuell befindet.
Test Case 2	Finden Sie heraus wieviel Prozent des gestrigen Tages die Maschine im Signalmaststatus *Maintenance* war.
Test Case 3	Überprüfen Sie die letzte Fehlermeldung die von der Maschine gesendet wurde.
Test Case 4	Überprüfen Sie den aktuellen Wert des Sensors *MotorTempPart1*
Test Case 5	Lesen sie die Metainformationen zum Sensor *MotorTempPart1* aus.
Test Case 6	Lesen Sie den Wert vom Sensor *MotorTempPart1* am 10.10.2018 aus.

TABELLE 8.4: Testcases

8.3 Auswertung

8.3.1 Testperson 1: Raphael Rechsteiner

Testcase	Bemerkung	Massnahmen
TC1	Die Testperson fand die Properties in den Cards unterhalb der Details. Jedoch wurde kritisiert, dass die Darstellung nicht übersichtlich ist.	Instanz Properties in einer Tabelle mit Paging anzeigen.
TC2	Es kann nicht abgelesen werden wieviele Stunden die Maschine in diesem Status war.	Stunden zum jeweiligen Status anzeigen. Dies wird jedoch nicht umgesetzt, da ein Workaround besteht die Stunden abzulesen. 10% eine Tages mit 24h (24 * 0.1)
TC3	Die Testperson fand das Error-Log schnell und hatte keine Probleme	Keine Änderungen
TC4	*Show Sensors* wurde schnell gefunden. Es braucht einen Moment bis klar war, dass die Grafik sich eigenständig aktualisiert.	Keine Änderungen
TC5	Für die Testperson war klar, dass Sie über *Show Details* auf die Informationen kommt. Inputfelder sind nicht ansprechend, evt. in einer List anstatt Inputfelder anzeigen.	Am Design der Inputfelder werden keine Änderungen vorgenommen, da sie so designed wurden um in eine höchst mögliche Flexibilität zu gewährleisten.

TC6	Die Einheit zum Wert in der Grafik fehlt, keine Probleme.	Dies kann ohne Probleme in einem späteren Schritt umgesetzt werden und ist nicht mehr Teil dieser Arbeit.

TABELLE 8.6: Auswertung Raphael Rechsteiner

8.3.2 Testperson 2: Ozan Sahin

Testcase	**Bemerkung**	**Massnahmen**
TC1	Der Testperson war nicht ganz klar, wo Sie die Instanz Properties finden kann.	Keine Änderungen, da dies Produktwissen ist, welches nach einmaligem Gebrauch erlernt ist.
TC2	Es kann nicht abgelesen werden wieviele Stunden die Maschine in diesem Status war.	Stunden zum jeweiligen Status anzeigen.
TC3	Die Testperson erwartete Fehlermeldungen auch auf Partebene, dort fand Sie die Meldungen jedoch nicht.	Das System ist ausgelegt für Fehlermeldungen auf Partebene, diese werden jedoch noch nicht angezeigt. Die Anzeige auf Partebene kann gemäss der Implementation auf der Instanz erweitert werden.
TC4	Klare Strukturierung Die Testperson wusste wo Sie den Wert finden kann. Eine Suche mit der man die Sensoren filtern kann wäre hilfreich.	Keine Änderungen, die Suche ist eine gute Idee, konnte in dieser Arbeit aber nicht mehr umgesetzt werden.
TC5	Die Testperson hatte keine Mühe die Metainformationen zu finden.	Keine Änderungen
TC6	Der Wert konnte ohne Probleme eruiert werden.	Keine Änderungen

TABELLE 8.8: Auswertung Ozan Sahin

Kapitel 9

Schlussfolgerung und Ausblick

Dieses Kapitel beschreibt die Resultate der Arbeit und verschafft einen Überblick was alles umgesetzt werden konnte. Die in der Anforderungsanalyse ausgearbeiteten Use Cases werden nochmals erwähnt und deren Umsetzungsgrad erläutert.

Der zweite Teil beinhaltet eine detaillierte Aussicht auf allfällige Folgearbeiten, welche aus der heutigen Sicht nutzbringend implementiert werden können.

9.1 Ergebnisse

Die Arbeiten konnten grösstenteils gemäss Plan realisiert werden. Das Endprodukt überzeugt mit einem gut strukturierten User Interface, welches sich an der bisherigen Lösung orientiert. Dank einem generischen Ansatz wurde ausserdem die Grundlage für eine hohe Erweiterbarkeit gelegt, was es ermöglicht zukünftige Szenarien einfach zu implementieren.

Die Kommunikationskanäle wurden so konzipiert, dass sie auch hohe Datenlasten bewältigen können. Mittels Websocket Protokoll ist es nun auch möglich, dass der Webserver Daten an eine verbundene Maschine sendet und somit ein bidirektionaler Kommunikationsweg bereit steht.

Nachfolgend wird anhand der Use Cases genauer beschrieben, in welchem Stadium die Applikation sich heute befindet.

INSTANZÜBERSICHT

UC01	Bronze	Instanzname unter Instanzübersicht in Textform anzeigen lassen.	**Umgesetzt**
UC02	Bronze	Instanztyp unter Instanzübersicht in Textform anzeigen lassen.	**Umgesetzt**
UC03	Bronze	Seriennummer unter Instanzübersicht in Textform anzeigen lassen.	**Umgesetzt**
UC04	Bronze	Instanzsoftware-Version unter Instanzübersicht in Textform anzeigen lassen.	**Umgesetzt**
UC05	Bronze	Betriebssystem unter Instanzübersicht in Textform anzeigen lassen.	**Umgesetzt**

UC06	Bronze	Betriebssystem-Version unter Instanzübersicht in Textform anzeigen lassen.	**Umgesetzt**
UC07	Bronze	Standort unter Instanzübersicht auf Google-Maps-Karte anzeigen lassen.	**Umgesetzt**
UC08	Bronze	Signalmaststatus unter Instanzübersicht graphisch anzeigen lassen.	**Umgesetzt**
UC09	Bronze	Instanzstatus unter Instanzübersicht in Textform anzeigen lassen.	**Umgesetzt**
UC10	Silber	Instanzfehlermeldung unter Instanzlog in Textform anzeigen lassen.	**Umgesetzt**
UC11	Silber	Meldungen zu fehlenden / fehlerhaften Instanzinformationen unter Instanzübersicht in Textform anzeigen lassen.	**Umgesetzt**
UC12	Bronze	Letzter Aktualisierungszeitpunkt der Instanzinformationen unter Instanzübersicht in Textform anzeigen lassen.	**Umgesetzt**
PARTÜBERSICHT			
UC13	Bronze	Laufleistung unter Partübersicht und Detailansicht graphisch im Verlauf anzeigen lassen.	**Umgesetzt**
UC14	Bronze	Signalmaststatus unter Partübersicht und Detailansicht graphisch im Verlauf anzeigen lassen.	**Umgesetzt**
UC15	Bronze	Instanzstatus unter Partübersicht und Detailansicht graphisch im Verlauf anzeigen lassen.	**Umgesetzt**
UC16	Bronze	Umgebungstemperatur unter Partübersicht und Detailansicht graphisch im Verlauf anzeigen lassen.	**Umgesetzt**
UC17	Bronze	Schaltschranktemperatur unter Partübersicht und Detailansicht graphisch im Verlauf anzeigen lassen.	**Umgesetzt**
UC18	Bronze	Umgebungsluftfeuchtigkeit unter Partübersicht und Detailansicht graphisch im Verlauf anzeigen lassen.	**Umgesetzt**
UC19	Bronze	Schaltschrankluftfeuchtigkeit unter Partübersicht und Detailansicht graphisch im Verlauf anzeigen lassen.	**Umgesetzt**

UC20	Bronze	Servomotortemperatur unter Partübersicht und Detailansicht graphisch im Verlauf anzeigen lassen.	**Umgesetzt**
UC21	Bronze	Instanzluftdruck unter Partübersicht und Detailansicht graphisch im Verlauf anzeigen lassen.	**Umgesetzt**
UC22	Silber	Meldungen zu fehlenden / fehlerhafte Partinformationen unter Partübersicht und Detailansicht in Textform anzeigen lassen.	**Umgesetzt**
UC23	Bronze	Letzter Aktualisierungszeitpunkt der Partinformationen unter Partübersicht und Detailansicht in Textform anzeigen lassen.	**Umgesetzt**
UC24	Bronze	Zu jedem Part eine Detailansicht mit Informationen, wie Datentyp, Interval, Einheit unter Partübersicht und Detailansicht in Textform anzeigen lassen.	**Umgesetzt**
UC25	Silber	Für jede graphische Darstellung eine Einschränkung für einen bestimmen Zeitraum auswählen.	**Umgesetzt**
UC26	Silber	Für jede graphische Darstellung einen Normalwertebereich anzeigen lassen.	**Umgesetzt**
UC27	Silber	Für jeden Part einen Log mit Errorinformationen anzeigen lassen.	**Offen**
UC28	Gold	Partinformationen in der Partübersicht nach Name sortieren und anzeigen lassen.	**Offen**
UC29	Gold	Partinformationen in der Partübersicht nach Name suchen und anzeigen lassen.	**Offen**
KONFIGURATION			
UC30	Gold	Konfiguration, welche graphischen Daten miteinander im selben Diagramm angezeigt werden, über ein simples Textfile vornehmen.	**Offen**
UC31	Gold	Darstellungskonfiguration über ein UI vornehmen.	**Offen**

TABELLE 9.2: Umsetzungsstadium der Use Cases

Von 31 Uses Cases konnten 26 umgesetzt werden. In Anbetracht der kurzen Zeit ist dies ein zufriedenstellendes Ergebnis. Offene Use Cases sind zwar noch umzusetzen, aber es wurde darauf geachtet, dass diese mit möglichst geringem Aufwand implementiert werden können.

Für den **UC27** kann die Komponente verwendet werden, welche schon für den **UC10** verwendet wurde. Die Logik stimmt überein und die entsprechenden Enpoints stünden ebenfalls zur Verfügung.

UC28 und **UC29** wären etwas umfangreicher, da auch eine Suchlogik eingebaut werden müsste. Eine solche ist noch nicht vorhanden und es müssten entsprechende Libraries und Module verwendet werden, um eine funktionierende Logik zu erstellen. In Anbetracht der wenigen Sensoren, die momentan an einer Maschine installiert sind, ist dies aber kein Verlust für den Endbenutzer.

Für die Konfigurationsmöglichkeiten müssten neue Schnittstellen im Product Backbone geschaffen werden. Das UI wurde bereits so implementiert, dass die Darstellungen dynamisch gerendert werden. Dies geschieht momentan aber noch anhand der Metadateninformationen. Schöner wäre, wenn eine UI-Konfiguration mittels JSON vorgenommen werden könnte und diese dann in einer separaten Tabelle innerhalb des Product Backbones gespeichert wird.

9.2 Ausblick

Die Arbeit wird zusammen mit der Abteilung Maschinentechnik am 11. Januar 2018 präsentiert. Eine Folgearbeit soll zudem die SPA um zusätzliche Funktionalitäten ausbauen. Denkbar wäre zum Beispiel eine Visualisierung der Sensoren an der Maschine vor Ort über Augmented Reality. Der Service Techniker hätte mit dieser Lösung neu die Möglichkeit, an der Maschine zu arbeiten und mittels Mixed-Reality-Brille die Sensoren zu identifizieren und deren Daten auszuwerten. Dies würde neue Supportmöglichkeiten schaffen und den Arbeitsprozess des Service Technikers positiv beeinflussen.

Ebenfalls könnte eine Verfeinerung und Weiterführung der bisherigen Lösung ein Thema für nachfolgende Arbeiten sein. Zum einen ist die Filterfunktion des Product Backbones sehr simpel gehalten und verursacht im heutigen Fall noch eine zu grosse Netzwerklast zum Cockpit Backend. Zum anderen müsste noch ein ausgereiftes Error Handling über alle Schichten implementiert werden, welches Probleme in der Kommunikation geschickt und benutzerfreundlich abfängt.

Auch die Umsetzung der offenen Use Cases wäre empfehlenswert, besonders die Konfiguration des UIs. Hierfür müsste, wie bereits erwähnt, ein neuer Product Backbone Endpoint eingerichtet werden, welcher die Konfigurationsdaten abspeichert. Dann könnte darauf aufbauend ein UI implementiert werden, über welches die Konfiguration vorgenommen werden könnte.

Ziel ist es, dass die Firma Komax AG ein Endprodukt erhält, welches als Prototyp für weitere Arbeiten dient.

Literatur

[1] ISO/IEC JTC 1/SC 7. „Software engineering – Product quality – Part 1: Quality model“. *ISO/IEC 9126-1:2001* 1 (Juni 2001). URL: https://www.iso.org/standard/22749.html.

[2] Eyerys (2016). URL: https://www.eyerys.com/articles/web-communication-protocols.

[3] PubNub (Jan. 2015). URL: https://www.pubnub.com/blog/2015-01-05-websockets-vs-rest-api-understanding-the-difference.

[4] OPC Foundation (). URL: http://documentation.unified-automation.com/uasdkhp/1.0.0/html/platform_network.html.

[5] Luo Gang (). URL: https://chrome.google.com/webstore/detail/smart-websocket-client/omalebghpgejjiaoknljcfmglgbpocdp.

[6] Swimlane (2018). URL: https://github.com/swimlane/ngx-charts.

[7] jQWidgets (2018). URL: https://www.jqwidgets.com/jquery-widgets-demo/demos/jqxchart/index.htm.

[8] Jerome Touffe-Blin (2018). URL: http://jtblin.github.io/angular-chart.js.

[9] Fenopix (2018). URL: https://canvasjs.com/angular-charts.

[10] Wikipedia (Sep. 2014). URL: https://en.wikipedia.org/wiki/Comparison_of_JavaScript_charting_libraries.

[11] Maria Antonietta Perna (März 2016). URL: https://www.sitepoint.com/canvas-vs-svg-choosing-the-right-tool-for-the-job.

[12] OPC Foundation (2018). URL: https://opcfoundation.org.

[13] Mikael Morlund (Dez. 2018). URL: https://marketplace.visualstudio.com/items?itemName=Mikael.Angular-BeastCode.

[14] Alexander Ivanichev (Sep. 2018). URL: https://marketplace.visualstudio.com/items?itemName=alexiv.vscode-angular2-files.

[15] Sander Ledegen (Juli 2018). URL: https://marketplace.visualstudio.com/items?itemName=sanderledegen.angular-follow-selector.

[16] steoates (Nov. 2017). URL: https://marketplace.visualstudio.com/items?itemName=steoates.autoimport.

[17] Microsoft (Nov. 2018). URL: https://marketplace.visualstudio.com/items?itemName=msjsdiag.debugger-for-chrome.

[18] Eric Amodio (Dez. 2018). URL: https://marketplace.visualstudio.com/items?itemName=eamodio.gitlens.

[19] Christian Kohler (Mai 2017). URL: https://marketplace.visualstudio.com/items?itemName=christian-kohler.path-intellisense.

[20] egamma (Dez. 2018). URL: https://marketplace.visualstudio.com/items?itemName=eg2.tslint.

[21] jetbrains (). URL: https://www.jetbrains.com/resharper/.

[22] (). URL: https://en.wikipedia.org/wiki/Single_responsibility_principle.

[23] Ing. Tomas Topinka (). URL: https://plugins.jetbrains.com/plugin/4509-statistic.

[24] jetbrains (). URL: https://www.jetbrains.com/webstorm/.

Anhang A

Selbstreflexionen

A.1 Lukas Koller

Dank dem Kickoff, welches Daniel Politze organisiert hat, wurden wir bereits vor dem Semsterbeginn mit der Projektidee konfrontiert und konnten die beteiligten Parteien kennenlernen. Neben Ozan Sahin, dem Maschinenbauer der die Sensoren an der Maschine angebracht hat, lernten wir auch Lukas Kretschmar, der unsere Ansprechsperson zur Backbone Schnittstelle war, kennen.

In den ersten Wochen haben wir eine Einführung in die Backbone Schnittstelle durch Lukas Kretschmar erhalten. Etwa zur gleichen Zeit wurden die ersten Anforderungen in Zusammenarbeit mit Daniel Politze, Pius Anderhub und Ozan Sahin erarbeitet. Durch die regelmässigen Meetings mit Ozan konnten wir uns zu dritt gut abstimmen. Kommunikationsprobleme gab es aus meiner Sicht vorallem, wenn alle Beteiligten hätten involviert werden sollen. Dort fehlte zum Teil das Feingefühl, welche Informationen relevant sind für alle.

Die Teamarbeit mit Simon hat aus meiner Sicht sehr gut funktioniert. Da wir bereits Erfahrungen zusammen gesammelt haben, unteranderem im Engineering Projekt, wussten wir bereits von Beginn an, wo welche Stärken lagen. Das Herausfordernde an dieser Arbeit war für uns beide das Konzeptionelle. Schon von Beginn an wurde viel Wert auf eine generische Lösung gesetzt. Die Ausarbeitung dieser Lösung hat uns beide sehr stark herausgefordert. Dennoch hat es uns extrem viel Spass gemacht und das Endergebnis ist aus unserer Sicht sehr zufriedenstellend.

Auch die Zusammenarbeit mit Ozan verlief sehr gut. Da Ozan als Maschinenbauer eine sehr programmierlastige Bachelor Arbeit schreiben musste, unterstützen wir ihn wo wir konnten. Dank seins Engagement in seiner Arbeit, konnte er trotz fehlenden Kenntnissen in der Programmierung seinen Teil der Arbeit erfüllen.

Als Verbesserungspotential von Simon und mir sehe ich eine effizientere Arbeitsweise im Sinne von, nicht jeden möglichen Fall der in der Zukunft eintreten könnte abzudecken. Meiner Meinung nach hat uns dies beim generischen Ansatz sehr viel Zeit geraubt, da wir versuchten möglichst alles abzudecken.

Im grossen und ganzen war die Arbeit sehr erfolgreich. Wir konnten einen Prototypen bauen, der es erlaubt Echtzeitdaten einer Maschine in einer SPA zu visualisiert.

A.2 Simon Müller

Wir wurden sehr gut in die Arbeit eingeführt. Schon von Beginn an konnten wir uns einen Eindruck der Maschine und des Teams verschaffen. Dies half dabei den Zusammenhang der Beteiligten zu verstehen. Dank der Erklärung von Pius Anderhub wurde uns die Funktionalitäten der Maschine schon früh klar, was uns bei der Ausarbeitung der Aufgabenstellung unterstützte.

Weitere Gespräche mit Daniel Politze und Ozan Sahin klärten dann noch einige Unsicherheiten. Diese beinhalteten vor allem planerische Tätigkeiten, wie zum Beispiel das Ausarbeiten der Aufgabenstellung oder das Setzen der Meilensteine. Hier wurden auch bereits Herausforderungen ersichtlich, da die Bereiche Maschinentechnik und Informatik verschiedene Auffassungen von solchen Arbeitsmethodiken haben. Wir konnten uns dann schliesslich darauf einigen, dass zwei zwei Aufgabenstellungen ausgearbeitet wurden, eine individuelle und eine gemeinsame.

Die grösste Herausforderung war sicherzustellen, dass alle vom gleichen redeten und sich auf dem gleichen Stand befinden. Der Umstand, dass wir mit Betreuer und Ansprechpartner insgesamt 8 Personen waren, erschwerte die Kommunikation. So kam es oft vor, dass die involvierten Personen verschiedene Wissensstände mitbrachten. Jedoch konnten wir diese Problematik schon relativ früh ansprechen und die Lage verbesserte sich ab der Mitte der Arbeit.

Über die Arbeit kann ich sagen, dass ich umsetzen durfte, was ich in etwa erwartet habe. Ich für meinen Teil habe vor allem die Anpassungen im Backend und im Machine Communication Service vorgenommen. Dadurch kam ich auch das erste Mal in den Kontakt mit Websockets. Ich durfte mich auch genauer über de OPC/UA Standard informieren und verschiedene administrative Tätigkeiten übernehmen. Der Task mit der Ausarbeitung der Web API hat mir ausserdem sehr gut gefallen, da ich effizient alle benötigten Daten zusammenstellen konnte und somit eine sehr performante Schnittstelle bauen durfte.

Rückblickend würde ich vor allem das Tooling an gewissen Stellen ändern. Die gesamte Dokumentation wurde mit Latex in Visual Studio Code erarbeitet. Leider bietet Visual Studio Code keine wirklich gute Unterstützung von Latex. Dies führte dazu, dass repetitive Arbeiten auf Grund von langen Kompilierungszeiten sehr zeitintensiv wurden.

Desweiteren hätten wir an gewissen Stellen etwas rascher voranschreiten können. Da wir beide sehr perfektionistisch sind, wollen wir immer optimale Lösungen bauen. Dies ist jedoch sehr zeitintensiv und in gewissen Fällen auch nicht wirtschaftlich. Dort könnten wir sicherlich noch dazulernen und auch den wirtschaftlichen Nutzen einer schönen Lösung mehr gewichten.

Anhang B

Aufgabenstellung

Semester-/Bachelorarbeit HS 18/19 Abteilungen Maschinentechnik & Informatik

Aufgabenstellung "Upgrade zu einer Industrie 4.0-Maschine inklusive Aufbereitung und Visualisierung derer Daten"

Dieses Dokument beinhaltet die Aufgabenstellung für das im Rahmen einer Studien- und Bachelorarbeit geplante Upgrade einer bestehenden Maschine zu einer Industrie 4.0-Maschine inklusive Aufbereitung und Visualisierung derer Daten. Die Arbeiten wird in Kollaboration zwischen den Studiengänge Maschinentechnik und Informatik an der HSR realisiert.

Bearbeitung durch

Ozan Sahin (Maschinentechnik)	Tel. +41 79 626 86 24	Email: ozan.sahin@hsr.ch
Lukas Koller (Informatik)	Tel. +41 79 559 25 35	Email: lukas.koller@hsr.ch
Simon Müller (Informatik)	Tel. +41 76 347 47 52	Email: simon.mueller1@hsr.ch

Betreuung

Prof. Dr. Daniel P. Politze	Tel. +41 55 222 46 05	Email: daniel.politze@hsr.ch
Prof. Dr. Christian Bermes	Tel. +41 55 222 47 12	Email: christian.bermes@hsr.ch

Auftraggeber

Komax AG, Pius Anderhub Email: pius.anderhub@komaxgroup.com

Kick-off

19. September 2018, 15:00 Uhr

Abgabetermin

21. Dezember 2018, 12.00 Uhr

Ausgangslage

Aktuell sind weltweit mehrere Crimpmaschinen für Fahrzeugkabelbäume der Firma Komax AG im Einsatz. Diese Maschinen generieren fortlaufend interessante Daten, sowohl für die Komax AG wie auch deren Kunden. Stand heute sind diese Daten jedoch nicht oder nur in geringer Weise abgreifbar. Dies erschwert für die Komax AG die effiziente Unterstützung ihrer Kunden bei der Fehlersuche. Zudem gehen wichtige Erfahrungswerte verloren, die für die Komax AG von grosser Bedeutung sein können.

Zielsetzungen

Neu sollen die Maschinen der Komax AG mit Sensoren ausgestattet werden, welche Zustands- und Leistungsdaten liefern, die als Auswertunggrundlage fungieren. Dabei werden folgende Ziele gesteckt:

- Sinnvolle und auswertbare Zustands- und Leistungsdaten definieren
- Gemeinsame Schnittstelle zum Austausch der Maschinendaten entwickeln
- Daten in regelmässigen Zeitabständen über das Internet kommunizieren und im Product Cockpit anzeigen

Vorgehen

- Analyse
  - In die Problemstellung einarbeiten
  - Aufgabenstellung verfeinern
  - Gemeinsamer Projektplan erstellen
- Konzept
  - Anforderungen an die auszuwertenden Informationen definieren
  - Schnittstelleninformationen definieren
  - Zwischenspeicherung der Maschinendaten definieren
- Umsetzung
  - Prototyp realisieren und testen
  - Test des fertigen Produktes
  - Fehlerbehebungen
  - Präsentation vor Kunde

Weitere Hinweise

Individuelle Aufgabenstellungen der jeweiligen Bereichen Maschinentechnik und Informatik sind in separaten Dokumenten beschrieben.

Vertraulichkeit

Interne Informationen, Daten und Resultate sowie abgegebene Dokumente und Zeichnungen müssen vertraulich behandelt werden und dürfen nicht an Dritte weitergegeben werden. Die Ergebnisse der Arbeit gehören der Schule.

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

Unterschrift Betreuer 1: _____

Datum: _____

Unterschrift Betreuer 2: _____

Datum: _____

Unterschrift Teammitglied 1: _____

Datum: _____

Unterschrift Teammitglied 2: _____

Datum: _____

Unterschrift Teammitglied 3: _____

Datum: _____

Rapperswil, 26. September 2018 Ozan Sahin, Simon Müller, Lukas Koller

Anhang C

Sitzungsprotokolle

Thema	Kick-off
Woche	1
Datum, Zeit	19. September 2018, 15:00 - 16:30
Protokollführung	Simon Müller
	Prof. Dr. Daniel P. Politze
	Lukas Kretschmar
Sitzungsteilnehmer	Ozan Sahin
	Simon Müller
	Lukas Koller

Traktanden

- Beteiligte Personen kennenlernen
- Maschine kennenlernen
- Grundverständnis für Kundenanforderungen erlangen
- Erste Fragen klären

Beschlüsse / Ergebnisse

- Zusammen mit Ozan Aufgabenstellung verfeinern
- Backbone mit Lukas Kretschmar anschauen

Thema	Verfeinerung Aufgabenstellung 01
Woche	1
Datum, Zeit	21. September 2018, 08:30 - 09:40
Protokollführung	Simon Müller
Sitzungsteilnehmer	Prof. Dr. Daniel P. Politze Ozan Sahin Simon Müller Lukas Koller

Traktanden

- Zusammenfassen der gewünschten Anforderungen
- Ausarbeitung einer gemeinsamen Aufgabenstellung
- Offene Fragen klären

Beschlüsse / Ergebnisse

- Email "Szenario für die Studierende" von Daniel Politze erhalten Antwort von Kunde abwarten
- Denkbar wichtige Daten sind
  - Maschinenidentifikation
  - Standortkoordinate
  - Umgebungstemperatur
  - Stückzähler
  - Zustand (läuft / Fehler)
- Ozan muss Datenaufbearbeitung für das Abgreifen über das Internet noch genauer definieren

Todos

- Abklären
  - Abgabetermin bei allen gleich?
  - Schnittstelleninformationen klären:
    - Welche Sensordaten?
    - Welches Format?
    - In welchem Zeitintervall?
    - Art der Datenspeicherung
  - Gemeinsames Dokument nötig oder dürfen zwei unabhängige Dokumente mit unterschiedliche Formatierung entstehen?
- Erledigen bis 08. September 2018
  - Allgemeine Aufgabenstellung erweitern
  - Individuelle Aufgabenstellung erstellen
  - Individueller Projektplanentwurf erstellen

Thema	Einführung Backbone
Woche	2
Datum, Zeit	25. September 2018, 12:00 - 13:05
Protokollführung	Lukas Koller
Sitzungsteilnehmer	Lukas Kretschmar Lukas Koller

Traktanden

- Schnittstelle erläutern
- Einführung zum Backbone
- Offene Fragen klären

Beschlüsse / Ergebnisse

- Im Backbone wurde zusätzliche Security eingebaut:
  - SSH zertifiziert, es werden nur Requests über HTTPS bearbeitet
  - JWT Token das jeweils nach 24 Stunden erneuert werden muss
  - GUID als Identifikation der zugreifenden Applikationen
- Die Skalierbarkeit ist zum aktuellen Zeitpunkt kein Thema, da wir in der Studienarbeit nur mit einer Maschine arbeiten und wir daher nicht mit einer grossen Datenmenge rechnen müssen
- Beim persistieren der Daten wird ein timestamp (UTC) generiert
- Über die API des Backbone können im body JSON Daten übergeben werden, die direkt in dieser Struktur in einer NoSQL Datenbank gespeichert werden

Todos

- Lukas Kretschmar: Schnittstellendokumentation zusenden

Thema	Weekly mit Ozan
Woche	2
Datum, Zeit	28. September 2018, 08:30 - 10:00
Protokollführung	Lukas Koller
	Ozan Sahin
Sitzungsteilnehmer	Simon Müller
	Lukas Koller

Traktanden

- Projektplan abgleichen
- Schnittstellendefinition klären

Beschlüsse / Ergebnisse

- Am 30.11 können die ersten Produktive Tests mit Sensoren und API durchgeführt werden
- Ozan kümmert sich darum das die Daten von seinem Gerät an unsere API gesendet wird, dies könnte mit LoRaWAN umgesetzt werden

Todos

- Ozan: schickt seinen Vorschlag von Sensordaten an alle Stakeholder
- Ozan: Pflichtenheft versenden
- Ozan: Abklären ob es WLAN hat in der Eichwiesstrasse 6
- Simon und Lukas: Pflichtenheft ergänzen bis spätestens 05.10
- Alle: LoRaWAN dokumentation
- Alle: Projektplan abgleichen und fertigstellen
- Simon: Aufgabenstellung an Ozan senden

Thema	Aufgabenstellung finalisieren
Woche	2
Datum, Zeit	28. September 2018, 13:00 - 13:30
Protokollführung	Lukas Koller
	Prof. Dr. Daniel P. Politze
Sitzungsteilnehmer	Simon Müller
	Lukas Koller

Traktanden

- Aufgabenstellung und Projektplan besprechen / finalisieren

Beschlüsse / Ergebnisse

- Produktives System und Entwicklungssystem aufschalten und in der Aufgabenstellung ergänzen (kann auch auf Azure deployed werden)
- Domain Analyse ebenfalls als Punkt in der Aufgabenstellung ergänzen
- User Szenario erstellen wodurch die User Stories generiert werden können mit verschiedenen Rollen
- UI muss nicht generisch sein, kann auf unseren Case spezifisch sein. Im UI nicht das rad neu erfinden, was möglich ist umsetzen
- Validierungsszenario am schluss um die User Szenario abzudecken
- Offene Punkte und nicht erreichte Punkte in der Dokumentation klar definieren
- Reale Mockdaten aus der Aufgabenstellung entfernen
- Visualisierungskonzept in der Aufgabenstellung noch ergänzen
- Wir sind nicht an die Templates und die strukturellen Vorgaben der HSR gebunden für die Dokumentation der SA

Todos

- Simon: Abgabe und Startdatum gleich formatieren
- Simon und Lukas: Backbone Adapter konzept evt adaptieren damit meherer verschiedene Maschinen angebunden werden können, mit Lukas anschauen
- Simon: Vorgehen ergänzen um Visualisierung der Anforderungen

Thema	Weekly mit Ozan
Woche	3
Datum, Zeit	05. Oktober 2018, 08:10 - 09:10
Protokollführung	Lukas Koller
	Ozan Sahin
Sitzungsteilnehmer	Simon Müller
	Lukas Koller

Traktanden

- Anforderungen der Komax durchgehen

Beschlüsse / Ergebnisse

- Die Anforderungen der Komax AG sind von der Sensorik und der Visualisierung umsetzbar

Todos

- Simon und Lukas: Pflichtenheft ergänzen und an Ozan schicken bis 05.10 17:00
- Simon und Lukas: Anforderungen nachfragen z.B. Signalmastatus
- Simon und Lukas: Schnittstelle mit generischen Werten definieren bis 10.10 17:00
- Ozan: spätestens bis am 15.10 Struktur der Sensoren und wie die Daten ungefähr daher kommmen (type, interval, etc.)

Thema	Backbone Abklärungen
Woche	5
Datum, Zeit	17. Oktober 2018, 17:00 - 17:30
Protokollführung	Lukas Koller Lukas Kretschmar
Sitzungsteilnehmer	Simon Müller Lukas Koller

Traktanden

- Backbone claims klären
- Handling und Neugenerierung der Access token
- https nicht sicher dargestellt vom Browser

Beschlüsse / Ergebnisse

- Access token erneuern z.B. Ablauf des tokens ist im Token enthalten.

Thema	Besprechung Konzeption
Woche	5
Datum, Zeit	19. Oktober 2018, 08:10 - 08:50
Protokollführung	Simon Müller Ozan Sahin
Sitzungsteilnehmer	Simon Müller Lukas Koller

Traktanden

- Absprache von Konzeptionvorschlägen
- Austausch aktueller Stände
- Definition weiterer Vorgehensweise

Beschlüsse / Ergebnisse

- Meeting machen mit Daniel und Christian
- Konzeption von Ozan verstanden und gutgeheissen
- Eigene Konzeption vorgestellt
- Schnittstelleninformation mit Ozan besprochen und ihn um Rat gefragt –> passt so
- Könnten einen zweiten Arduino kaufen für Simulationsdaten –> lassen wir momentan noch

Todos

- Alle: Wie stellen wir sicher, dass wir wissen, welche Sensoren ausgefallen sind?
- Alle: Wie stellen wir sicher, dass wir wissen, ob Maschine noch am Netz ist?
- Alle: Meeting zusammen abmachen für Freitag 26. Oktober 2018
- Erledigen bis 08. September 2018

Thema	Besprechung Konzeption
Woche	6
Datum, Zeit	24. Oktober 2018, 10:00 - 10:30
Protokollführung	Simon Müller Daniel Politze
Sitzungsteilnehmer	Simon Müller Lukas Koller

Traktanden

- Ziele für Freitag
- Clickable Prototype und bisherige Spezifikationen zeigen

Beschlüsse / Ergebnisse

- Schauen, dass wir für Pius eine gute Visualisierung für Freitag haben
- Anpassung von Systemmastdatenanzeige als Ampel
- Navigationsstruktur möglichst generisch machen, auch im Frontend –> nicht notwendig, dass wir das Dashboard konfigurieren können
- Verschiedene Maschinen mit ähnlichen Informationen müssen visualisiert werden können
- Instanzdaten sind vorhanden in der Datenbank, müssen aber noch abgegelichen werden
- Grafische Konfiguration wäre toll, aber nicht unbedingt notwendig –> vielleicht auch per JSON-Objekt
- Standort per Sensor abgreifen und täglich aktualisieren wäre ein Mehrwert
- Error-Handling auf Ebene „Debugging“ umsetzen, im Frontend Aktualisierungszeit auch noch anzeigen
- Security nicht beachten –> einfach sinnvoll umsetzen
- Loadbalancing: Out of scope –> mit Lukas ansprechen
- OPC/UA sollte best-möglich schon verwendet werden

Todos

- Lukas und Simon: Mit Ozan Standortsensor besprechen
- Lukas: Erweiterung der Sensordatenspezifikation um PartId
- Lukas und Simon: Mit Lukas zusammen schauen, dass wir Loadbalancing anschauen
- Simon: OPC/UA nochmals evaluieren

Thema	Präsentation Konzeption
Woche	6
Datum, Zeit	26. Oktober 2018, 13:00 - 14:30
Protokollführung	Simon Müller
Sitzungsteilnehmer	Chris Schnellmann Pius Anderhub Prof. Dr. Daniel P. Politze Christian Bermes Raphael Schröder Ozan Sahin Simon Müller Lukas Koller

Traktanden

- Präsentation Konzeption

Beschlüsse / Ergebnisse

- Varianten Arduino / SPS / Grosser Panda / die "fertig"Lösung
- Variante Arduino ist die flexibelste Lösung und wird deshalb auch umgesetzt
- Komax würde gerne Sensorinformationen über Settings in der weboberfläche konfigurieren wie z.B. den Intervall des Sensors
- Genauere Daten der Maschinen sollten geliefert werden –> wir bleiben beim faken
- Zeigen des UI-Prototypes
- Parallel mehrere Sensoren in einem Diagramm anzeigen können
- Normalbereiche für die Dashboards definieren können
- Sensorrange noch mitschicken –> Achtung: Range kann von Sensor zu Sensor unterschiedlich sein
- Name des Sensors muss nicht mitgeschickt werden von Sensor zu Persistence Service
- Alternative zum UI-Gold-Case: UI Konfiguration zurück zur Maschine

Todos

- Mit Ozan abklären ob Location auch möglich ist.
- Überlegen, inwiefern wir Sensoren parallel anzeigen können
- @Pius 07.01 - 11.01 Terminfindung

Thema	Weekly mit Ozan
Woche	7
Datum, Zeit	02. November 2018, 08:10 - 08:30
Protokollführung	Lukas Koller
Sitzungsteilnehmer	Ozan Sahin Lukas Koller

Traktanden

- Stand Projekt

Beschlüsse / Ergebnisse

- Der Vorschlag von Daniel Politze von der fpga company ist nicht flexibel genug und müsste durch sie weiterentwickelt werden
- Signalmasstatus wird von Pius an Ozan geschickt
- Woche 12 (03.12 - 07.12) können die ersten tests mit den Sensoren gemacht werden
- Normalwerte für Sensoren werden manuell bei uns hinterlegt
- Kommunikation zwischen Maschine und API basiert auf einem Websocket
- Location kann umgesetzt werden, fehlerfall wenn GPS nicht gesendet werden kann beachten

Todos

- @Lukas: Normalwerte im Domain Model beachten
- @Ozan: Normalwerte für alle Sensoren liefern
- @Simon: Notwendige Informationen für den Websocket an Ozan schicken

Thema	Weekly mit Ozan
Woche	7
Datum, Zeit	02. November 2018, 15:30 - 16:30
Protokollführung	Lukas Koller
Sitzungsteilnehmer	Lukas Kretschmar Lukas Koller

Traktanden

- Abklärung bestehendes Domain Model

Beschlüsse / Ergebnisse

- ProductInstance ist nicht das selbe wie Instance auf dem machine endpoint, ProductInstance repräsentiert das Objekt auf PLM seite und machine Instance die effektive Maschine
- Ziel ist es das beide Objekte die selbe GUID haben, so wird die Referenz auf das richtige Objekt sichergestellt
- Eigenschaften der beiden können in im UI miteinander verglichen werden wie z.B. wenn sich die Location auf der machine Instance ändert, die Änderung im ProductInstance Objekt gehört jedoch nicht zum Scope
- Im InstanceRepository ist eine fixe GUID hardcodiert, dass heisst alle part informationen werden auf der selben Instance gespeichert
- Das LiteDB file befindet sich in ../API/bin/Debug/netcoreapp2.1/Modules /Adapter.Data.LiteDB/InstanceData.litedb
- UI Feedback: Eine Suche mit der man die Sensoren im Sensor Dashboard filtern kann z.B. nach Type, Part, etc.
- Grafiken dynamisch konfigurieren und generieren ist sehr viel Aufwand, daraus könnte man gut eine BA machen

Todos

- @Simon: Überprüfen wie die Websocket Verbindung mit unterbrüchen und über eine längere laufzeit ohne requests umgeht.

Thema	Weekly mit Ozan
Woche	10
Datum, Zeit	30. November 2018, 08:40 - 09:40
Protokollführung	Lukas Koller
Sitzungsteilnehmer	Ozan Sahin Lukas Koller

Traktanden

- Api request Beispiele

Beschlüsse / Ergebnisse

- Api requests werden anhand den im Mail definiertem Pattern versendet

Thema	Weekly mit Ozan
Woche	13
Datum, Zeit	13. Dezember 2018, 16:00 - 17:00
Protokollführung	Lukas Koller
	Prof. Dr. Daniel P. Politze
Sitzungsteilnehmer	Simon Müller
	Lukas Koller

Traktanden

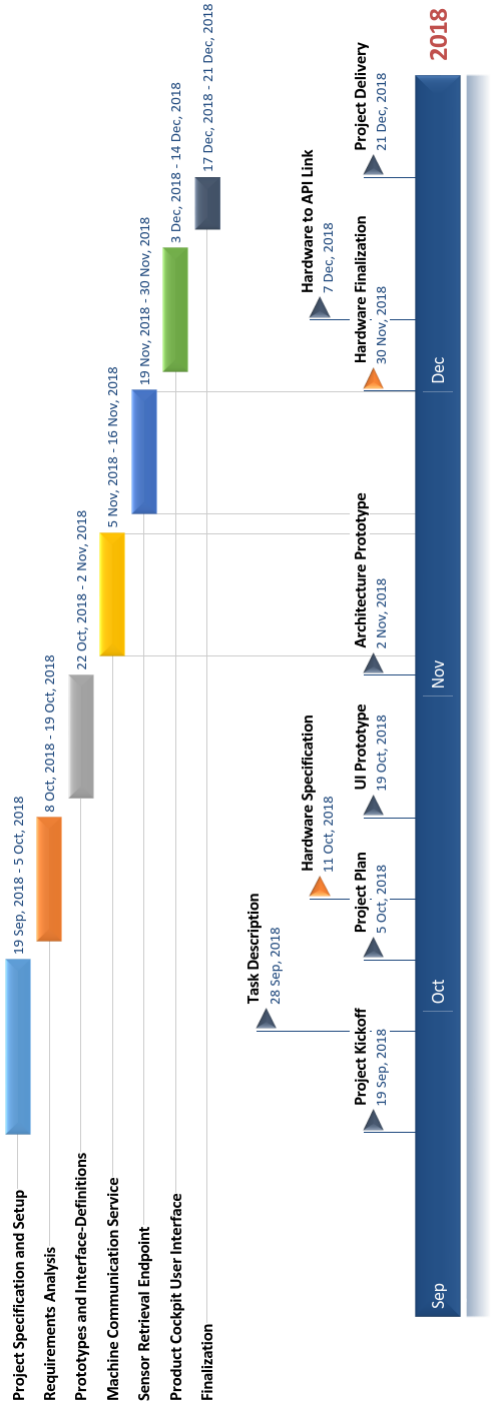
- Besprechung Projekt und fortführende Arbeiten

Beschlüsse / Ergebnisse

- akuteller Stand aus sicht von Daniel i.O. folgende Punkte sollten noch überarbeitet werden:
  - Units auf property ebene definieren
  - Usibility in der detail view verbessern
- Folgende Punkte könnten in einer fortführenden Arbeit erledigt werden
  - Überarbeitung der jetzt bestehenden Applikation, ein Beispiel dafür wäre ein CSV download der Sensor Daten
  - Daten der Maschine über eine VR Brille visualisieren

Anhang D

Projektplan



Anhang E

Prototype

Product Cockpit

6 Screens

Lukas K.

Product Cockpit

Machine Dashboard

BACK TO DASHBOARD

Machine details

Meta information

Machine name: Crimpmaschine

Machine type: n1-standard-96

Serial number: NB01234567

Status information

Machine software version — Sun Nov 21 05:56:07 2017: 14.3.123

Operating system type — Mon Jan 21 23:35:34 2017: Windows 10

Operating system version — Tue Feb 11 11:45:34 2017: 10.12.6

Signalmaststatus — Thu Aug 06 09:34:10 2018: Maintenance

Maschinenstatus — Mon Aug 21 15:34:10 2018: This sensor is no longer connected, the displayed data could be outdated

Setup

Installed marketing kits: None

Available marketing kits: Outdoor Feature, TRob+

Currently planned marketing kits: Camera Feature

SHOW SENSORS SHOW LOG

Service cases +

Filter

Creation date	Description	State
02/11/17	Desc1107	Closed
02/12/17	Desc7106	Open
20/10/17	Desc4087	Open

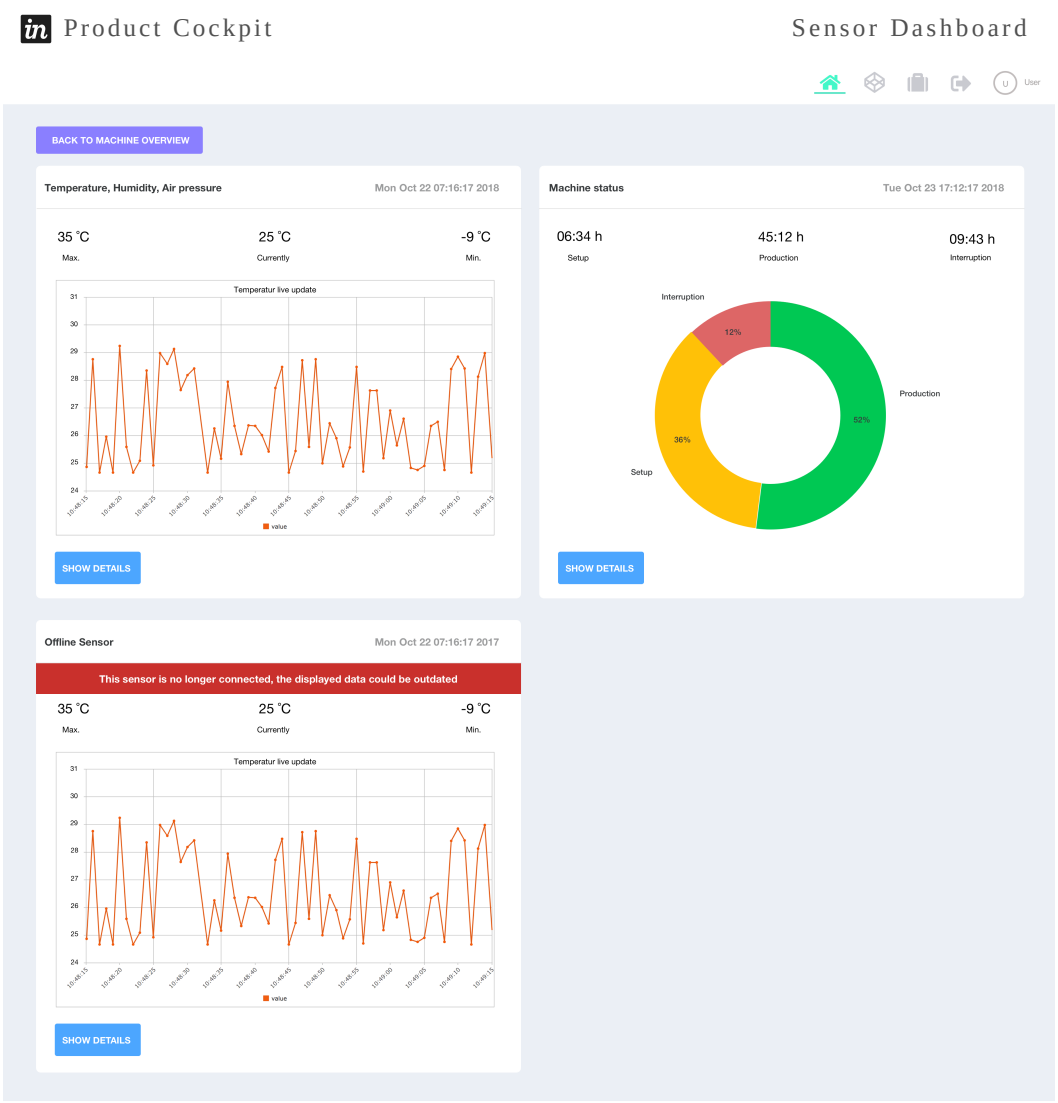
Items per page: 5 — 1 - 3 of 3

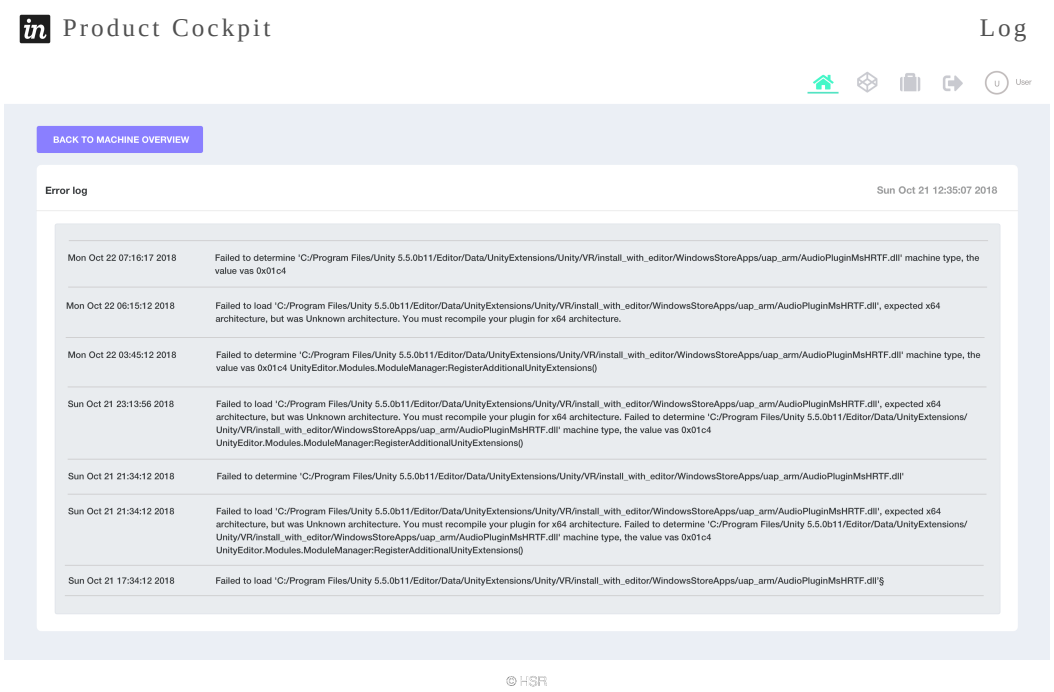
SHOW DETAILS

Bill of materials

Display options: NAME ONLY | LAST MAINTAINED | LAST MODIFIED

- Root Part
  - Part1 (Ald41)
  - Part2 (Ald42)


Product Cockpit
Sensor Dashboard
BACK TO MACHINE OVERVIEW
Temperature, Humidity, Air pressure
Mon Oct 22 07:16:17 2018
35 °C
Max.
25 °C
Currently
-9 °C
Min.
Temperatur live update
SHOW DETAILS
Machine status
Tue Oct 23 17:12:17 2018
06:34 h
Setup
45:12 h
Production
09:43 h
Interruption
Interruption
12%
Production
52%
36%
Setup
SHOW DETAILS
Offline Sensor
Mon Oct 22 07:16:17 2017
This sensor is no longer connected, the displayed data could be outdated
35 °C
Max.
25 °C
Currently
-9 °C
Min.
Temperatur live update
SHOW DETAILS


Product Cockpit
Log
User
BACK TO MACHINE OVERVIEW
Error log
Sun Oct 21 12:35:07 2018
Mon Oct 22 07:16:17 2018
Failed to determine 'C:/Program Files/Unity 5.5.0b11/Editor/Data/UnityExtensions/Unity/VR/install_with_editor/WindowsStoreApps/uap_arm/AudioPluginMsHRTF.dll' machine type, the value vas 0x01c4
Mon Oct 22 06:15:12 2018
Failed to load 'C:/Program Files/Unity 5.5.0b11/Editor/Data/UnityExtensions/Unity/VR/install_with_editor/WindowsStoreApps/uap_arm/AudioPluginMsHRTF.dll', expected x64 architecture, but was Unknown architecture. You must recompile your plugin for x64 architecture.
Mon Oct 22 03:45:12 2018
Failed to determine 'C:/Program Files/Unity 5.5.0b11/Editor/Data/UnityExtensions/Unity/VR/install_with_editor/WindowsStoreApps/uap_arm/AudioPluginMsHRTF.dll' machine type, the value vas 0x01c4 UnityEditor.Modules.ModuleManager:RegisterAdditionalUnityExtensions()
Sun Oct 21 23:13:56 2018
Failed to load 'C:/Program Files/Unity 5.5.0b11/Editor/Data/UnityExtensions/Unity/VR/install_with_editor/WindowsStoreApps/uap_arm/AudioPluginMsHRTF.dll', expected x64 architecture, but was Unknown architecture. You must recompile your plugin for x64 architecture. Failed to determine 'C:/Program Files/Unity 5.5.0b11/Editor/Data/UnityExtensions/Unity/VR/install_with_editor/WindowsStoreApps/uap_arm/AudioPluginMsHRTF.dll' machine type, the value vas 0x01c4 UnityEditor.Modules.ModuleManager:RegisterAdditionalUnityExtensions()
Sun Oct 21 21:34:12 2018
Failed to determine 'C:/Program Files/Unity 5.5.0b11/Editor/Data/UnityExtensions/Unity/VR/install_with_editor/WindowsStoreApps/uap_arm/AudioPluginMsHRTF.dll'
Sun Oct 21 21:34:12 2018
Failed to load 'C:/Program Files/Unity 5.5.0b11/Editor/Data/UnityExtensions/Unity/VR/install_with_editor/WindowsStoreApps/uap_arm/AudioPluginMsHRTF.dll', expected x64 architecture, but was Unknown architecture. You must recompile your plugin for x64 architecture. Failed to determine 'C:/Program Files/Unity 5.5.0b11/Editor/Data/UnityExtensions/Unity/VR/install_with_editor/WindowsStoreApps/uap_arm/AudioPluginMsHRTF.dll' machine type, the value vas 0x01c4 UnityEditor.Modules.ModuleManager:RegisterAdditionalUnityExtensions()
Sun Oct 21 17:34:12 2018
Failed to load 'C:/Program Files/Unity 5.5.0b11/Editor/Data/UnityExtensions/Unity/VR/install_with_editor/WindowsStoreApps/uap_arm/AudioPluginMsHRTF.dll'§

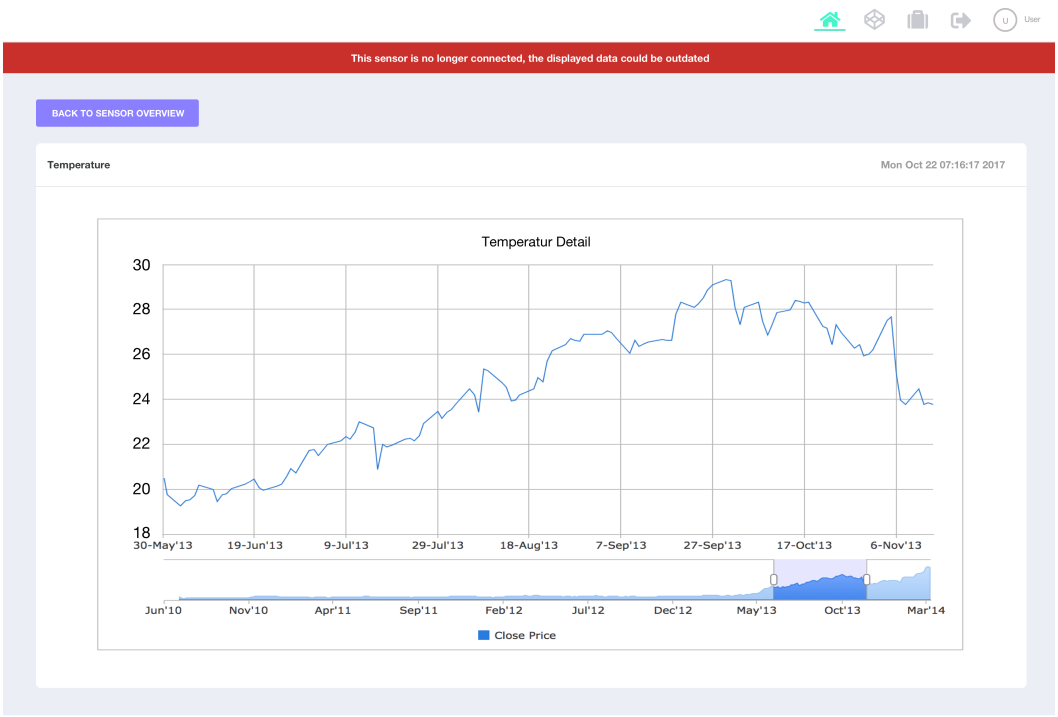
Product Cockpit

Temperatur Detail

Product Cockpit

Status Detail

Product Cockpit Offline Detail



Anhang F

Administratives

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.

Ort, Datum: Rapperswil, 21.12.18

Name, Unterschrift:

Lukas Koller

Simon Müller

Vereinbarung

1. Gegenstand der Vereinbarung

Mit dieser Vereinbarung werden die Rechte über die Verwendung und die Weiterentwicklung der Ergebnisse der Studienarbeit *Aufbereitung und Visualisierung von Maschinendaten einer Industrie4.0-Maschine* von Simon Müller und Lukas Koller unter der Betreuung von Dr. Prof. Daniel P. Politze geregelt.

2. Urheberrecht

Die Urheberrechte stehen der Studentin / dem Student zu.

3. Verwendung

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

Beilage/n:

Rapperswil, den. 21.12.18 .....
Simon Müller

Rapperswil, den. 21.12.18 .....
Lukas Koller

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

* Unterschrift des Betreuers folgt am 11.01.19. Dies ist mit Daniel Politze abgesprochen

Einverständniserklärung Publikation auf eprints.hsr.ch

☒ SA

○ BA

Titel der Arbeit: Aufbereitung und Visualisierung von Maschinendaten einer Industrie 4.0-Maschine

Team: Simon Müller, Lukas Koller

Betreuer: Prof. Dr. Daniel P. Politze

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

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

Rapperswil, 21.12.18

Name(n)	Unterschrift(en)
Lukas Koller	
Simon Müller	

Anhang G

Komax AG - Dokumente

	Statusparameter	Sichtbar für Komax	Sichtbar für Kunde	Prio
1	Maschinentyp / Seriennummer	ja	ja	1
2	Softwareversion Maschinensoftware	ja	ja	1
3	Softwareversion Betriebssystem	ja	ja	1
4	Laufleistung (Stückzahl / Betriebsst.)	ja	ja	1
5	Standortkoordinaten	ja	nein	3
6	Signalmaststatus: Rot = Störung / Grün = Produktion / Grün blinkend = Warten Bestätigung / Orange = Quality / Blau = Maintenance	ja	ja	1
7	Maschinenstatus: Auswertung Verfügbarkeit in % (Production, Setup, Interruption)	ja	ja	4
8	Temperatur Umgebung	ja	nein	2
9	Temperatur Schaltschrank (vorn / hinten)	ja	nein	2
10	Luftfeuchtigkeit Umgebung	ja	nein	2
11	Luftfeuchtigkeit Schaltschrank (vorn / hinten)	ja	nein	2
12	Temperatur Servomotoren (Bandantrieb, Schwenker, Abzug, Schneideinheit)	ja	nein	2
13	Luftdruck Maschine	ja	ja	2
14	Aktuelle oder letzte Fehlermeldung bei Maschinenstopp	ja	nein	3