HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL
FHO Fachhochschule Ostschweiz

SUNCAR

Suncar Big Data Collector & Dashboard

Bachelorarbeit

Abteilung Informatik

Hochschule für Technik Rapperswil

Frühjahrssemester 2018

Autor(en):	Tobias Stähli, Dumeni Vincenz, Jonas Wälter
Betreuer:	Manuel Bauer
Projektpartner:	SUNCAR HK AG
Experte:	Pascal Schaefer
Gegenleser:	Thomas Letsch

Aufgabenstellung

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL
FHO Fachhochschule Ostschweiz

Suncar
Big Data Collector & Dashboard

Aufgabenstellung
Diplomarbeit Dumeni Vincenz / Tobias Stähli / Jonas Wälter

Inhaltsverzeichnis

Manuel Bauer
HSR Hochschule für Technik Rapperswil ■ Oberseestrasse 10 ■ CH-8640 Rapperswil
T +41 55 222 41 11 ■ F +41 55 222 44 00 ■ office@hsr.ch ■ www.hsr.ch

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL
FHO Fachhochschule Ostschweiz

1 Auftraggeber

SUNCAR HK AG
Hinterwiden
CH-9245 Oberbüren
www.suncar-hk.com

Die SUNCAR HK AG entwickelt, baut und vermarktet autonome, batteriebetriebene Elektrofahrzeuge für den Offroad-Bereich. Das Ziel ist es, verschiedene bis anhin mit Dieselmotor betriebene Baumaschinen (z.B. Bagger) auf Elektroantrieb umzubauen und so den Dieseltreibstoff vollständig durch Solarenergie aus Photovoltaik zu ersetzen. Die von SUNCAR HK AG gebauten Elektro-Baumaschinen werden ausschliesslich mit Solarenergie betrieben.

2 Einführung & Ausgangslage

Zur Überwachung des Maschinenzustandes sowie zur Auswertung und Kontrolle der Auslegung und zur Fehlersuche wurde ein System zur Speicherung und Anzeige von Laufzeitdaten entwickelt. Die stetig wachsenden Anforderungen an das bestehende System machen eine ganzheitliche Überarbeitung notwendig. Das aktuelle System ist eine Zusammenstellung aus Python-Skripten für den Data Collector sowie MySQL/PHP für die Datenspeicherung und das Frontend. Die Anzahl der Messwerte übersteigt die Kapazität einer relationalen Datenbank und bedarf einer ganzheitlichen Neuentwicklung. Das Frontend wurde unter hohem Zeitdruck und nur für interne Zwecke entwickelt. Da die dargestellten Daten auch für den Endkunden interessant sind, soll das Frontend auch für Kunden zugänglich sein, wobei die aktuelle Lösung diesen neuen Anforderungen noch nicht genügt.

Das bestehende Gesamtsystem beinhaltet folgende Kernkomponenten:

- OpenVPN Server für eingehende Verbindungen
- Data Collector empfängt UDP Pakete, parst Messdaten und befüllt die Datenbank
- PHP-Frontend mit integrierter Business Logik zum Abruf der Daten

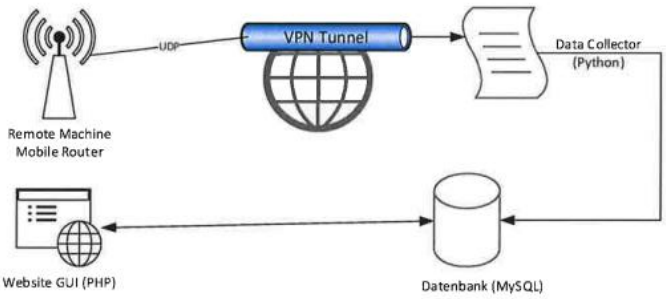


Abbildung 1: Aktuelles Gesamtsystem

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL
FHO Fachhochschule Ostschweiz

3 Voraussetzungen

Bei der Durchführung der Arbeit sind weitaus mehr als nur technische Skills gefordert. Aufgrund des Projekt-Setups sind koordinative Fähigkeiten sowie gute Kommunikationsfertigkeiten unabdingbar. Die vorliegende Aufgabenstellung beschreibt einen Bestandteil eines bestehenden Systems. Dies bringt die Notwendigkeit mit sich, nicht nur technische, sondern auch fachliche Kompetenzen mitzubringen.

4 Aufgabenstellung

Die Bachelorarbeit hat zum Ziel, die vorhandenen Teilsysteme (Data Collector, Datenbank, Frontend) in Form eines Prototyps neu zu entwickeln. Dabei soll die Architektur so modularisiert werden, dass in Zukunft einzelne Elemente einfach den neuen Anforderungen angepasst werden können. Ein weiteres Kernthema für die Architektur ist deren Skalierbarkeit.

Eine mögliche, grobe Zielarchitektur könnte wie folgt aussehen:

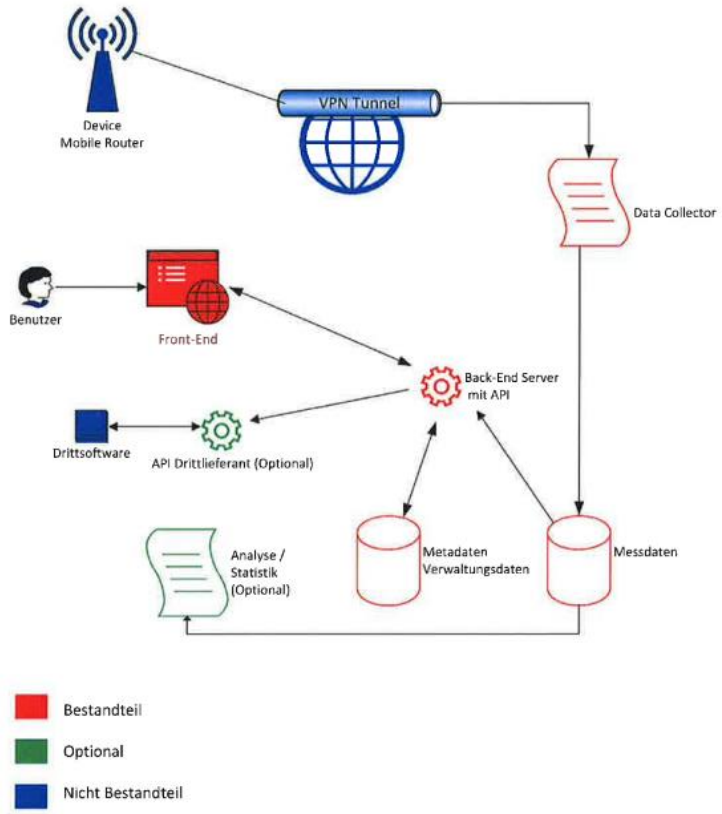


Abbildung 2: Mögliche Zielarchitektur

Manuel Bauer
HSR Hochschule für Technik Rapperswil ■ Oberseestrasse 10 ■ CH-8640 Rapperswil
T +41 55 222 41 11 ■ F +41 55 222 44 00 ■ office@hsr.ch ■ www.hsr.ch

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL
FHO Fachhochschule Ostschweiz

4.1 Funktionale Anforderungen

In den nachfolgenden Kapiteln wird der Begriff «Device» als Sammelbegriff für die eingesetzten mobilen Baumaschinen wie z.B. einen Bagger verwendet.

4.1.1 Datensammlung / Schnittstellenkonzept

Um die Daten von den Devices empfangen zu können, muss ein Schnittstellenkonzept erarbeitet und entwickelt werden. Es ist wichtig, dass das bestehende Legacy-System weiterhin unterstützt wird, da ein Software-Update auf den bereits gefertigten und ausgelieferten Devices nur mit sehr hohem Aufwand möglich wäre. Die Legacy-Devices sollen dabei über einen geeigneten Abstraktions-Layer angebunden werden.

Die Schnittstelle bietet Funktionalität zur Speicherung von IoT-Sensordaten in einer Datenbank. Beispiele der gelieferten Werte sind Temperaturmessungen, Leistungsaufnahmen, Restkapazität usw.

Abgrenzung:
Die Arbeit schliesst die Entwicklung auf den Devices aus. Alle Anpassungen «ausserhalb» der Schnittstelle müssen von den Studierenden beim Auftraggeber als Change Request eingegeben werden.

4.1.2 Datenbank

Für die Speicherung von Messdaten, Metadaten und Verwaltungsdaten ist ein entsprechend geeignetes Speicherkonzept inklusive Evaluation einer passenden Datenbank zu erarbeiten. Das Konzept soll sich auch mit der Frage auseinandersetzen, ob die angelieferten Messdaten und die für das System notwendigen Meta- und Verwaltungsdaten in unterschiedlichen Datenbanken abgelegt werden sollen.

4.1.3 Back-End API & GUI

Das zu erstellende System umfasst ein Web-Frontend, welches auf Basis einer Back-End API Messdaten darstellt sowie die Modifikation von Meta- und Verwaltungsdaten ermöglicht.

Diese Anforderungen sind auf dem GUI umzusetzen:

- CRUD-Funktionalitäten für User-, Kunden- und Device-Management
- Darstellung von Devices inkl. deren Messdaten
  - Übersicht der Messdaten
  - Grafische Aufbereitung der Messdaten
- Mandantenfähigkeit mit folgenden Spezifikationen
  - Rollen: Management, Superuser, User
  - Device: n:n-Beziehung mit Firma
  - Superuser kann User und die der Firma zugewiesenen Devices verwalten
  - Management Rolle kann alle User und alle Devices verwalten
- Die detaillierten funktionalen Anforderungen sind im Rahmen der Phase «Requirements Engineering» zu definieren
- CRUD-Operationen für Device-Typen
- CRUD-Operationen für Metadaten pro Device-Typ (Metadaten beschreiben unter anderem die Bedeutung der vom Device-Typ gelieferten Datensätze)
- Responsive Design des Front-Ends

Manuel Bauer
HSR Hochschule für Technik Rapperswil ■ Oberseestrasse 10 ■ CH-8640 Rapperswil
T +41 55 222 41 11 ■ F +41 55 222 44 00 ■ office@hsr.ch ■ www.hsr.ch

- Optionale Anforderungen
  - Device-Blacklist (Devices sind für das Management nicht ersichtlich)
  - Zeitbasierte Einschränkung der für einen User sichtbaren Messdaten (Mietmaschinen mit häufigen User-Wechseln)

4.1.4 Optionale Erweiterungen

Sollte die Aufgabenstellung schneller als geplant abgearbeitet werden können, sind nachfolgende optionale Erweiterungen denkbar:

- API für Drittanbieter, welche es erlaubt, die Device-Daten «on-demand» abzurufen
- Evaluation und Integration/Implementation eines Statistik-Modul
- Event Alerting z.B. bei Überschreitung eines Schwellenwertes (Stichwort: Predictive Maintenance)

4.2 Nicht-funktionale Anforderungen

Die nachfolgenden Nicht-funktionalen Anforderungen sind zu erfüllen:

- Die Soft- und Hardwareinfrastruktur muss für bis zu 200 Devices ausgelegt werden.
- Mit Skalierung der Infrastruktur sollen theoretisch bis zu 10'000 Devices gleichzeitig verwaltet werden können. Diese Messgrösse muss nicht getestet/gemessen werden.
- Das Web-Frontend soll auch auf mobilen Geräten abgerufen werden können, sie soll jedoch nicht nach dem «Mobile First»-Konzept entwickelt werden.
- Die Messdaten müssen 5 Jahre aufbewahrt werden.
- Das System muss nicht redundant ausgelegt werden. Die gewählte Softwarearchitektur sollte dies aber erlauben.
- Die Software muss passwortgeschützt werden und die Kommunikation über öffentliche Netzte darf nur verschlüsselt erfolgen.

4.3 Mitwirkungspflichten von Suncar HK

Die folgenden Mitwirkungspflichten sind auf Seiten der Suncar HK zu leisten. Die Verantwortung für Kommunikation und Einplanung der Ressourcen seitens Suncar HK liegt bei den Studierenden.

- Mitwirkung bei der «Requirement Engineering»-Phase
- Bereitstellung eines Dumps von 2-3 Stunden mit Messdaten eines Beispiel-Devices
- Bereitstellung eines Messdaten-Simulators
- Mitwirkung bei User Acceptance und anderen nötigen Tests
- Mitwirkung bei den regelmässigen Meetings
- Bereitstellung von Know-How über bestehende Applikation und fachspezifischem Wissen (bei Bedarf)

4.4 Resultate

Erwartete Resultate

- Technische Dokumentation
- Anforderungsanalyse
- Konzeptionierung der Systeme
- Implementation des Systems

Optionale Resultate

- Aufbau einer Continuous Integration und Staging-Umgebung

4.5 Technologien

Der Auftraggeber hat noch keine bestehende Infrastruktur, deshalb ist die Wahl der Technologien den Studierenden freigestellt. Grundsätzlich soll das System jedoch auf den folgenden Technologien aufbauen:

- Microsoft .NET Core (C#)
- React (JavaScript-Library für das Front-End)
- Datenbank: abhängig von der Evaluation/Konzeption
- Visual Studio Team Services

Abweichungen dieser Technologien müssen mit dem Auftraggeber und dem Betreuer abgesprochen werden.

5 Dokumentation

Die Wahl des Vorgehensmodells für die Projektabwicklung ist den Studierenden freigestellt. Es wird jedoch erwartet, dass die Aufgabenstellung analysiert und aufgeteilt sowie mindestens eine grobe Planung erstellt wird.

Abzugeben ist ein Bericht (1 Exemplar auf Papier plus in elektronischer Form) mit nachfolgendem Umfang:

- Titelblatt gemäss Vorlage
- Aufgabestellung
- Abstract
- Management Summary gemäss Vorlage
- Technischer Bericht
- Resultate gemäss Kapitel "4 Aufgabenstellung"
- Persönliche Berichte zu den Erfahrungen in der Arbeit
- Weitere während der Arbeit erstellte Dokumente
- Projektdokumentation welche den geplanten und effektiven Projektfortschritt nachvollziehbar aufzeigt, inkl. Termine, Meilensteinen, Aufwendungen, Projekttagebuch.
- A0-Poster (SA: elektronisch / BA: ausgedruckt)

Weitere Angaben zur Dokumentation finden sich unter:
\\hsr.ch\root\alg\skripte\Informatik\Fachbereich\Studienarbeit_Informatik
\\hsr.ch\root\alg\skripte\Informatik\Fachbereich\Bachelor-Arbeit_Informatik

Mit der Abgabe des Berichts ist ein Poster vorzubereiten, welches die wesentlichen Informationen über die Arbeit für die Ausstellung der Diplomarbeit enthält.

Manuel Bauer
HSR Hochschule für Technik Rapperswil ■ Oberseestrasse 10 ■ CH-8640 Rapperswil
T +41 55 222 41 11 ■ F +41 55 222 44 00 ■ office@hsr.ch ■ www.hsr.ch

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL
FHO Fachhochschule Ostschweiz

6 Projektmanagement

6.1 Termine

Beginn der Arbeit	19.02.2018
Abstract für Broschüre	08.06.2018
Abgabe A0-Poster	13.06.2018 10:00 Uhr
Abgabe Dokumente (archiv-i.hsr.ch)	15.06.2018 12:00 Uhr
Diplomausstellung	15.06.2018 16:00 – 20:00 Uhr
Mündliche Prüfung	06.08.2018 – 24.08.2018
Ausstellung der Poster und Diplomfeier	28.09.2018

6.2 Fortschrittsbesprechung

Alle zwei Wochen findet zu einem fixen Zeitpunkt eine Fortschrittsbesprechung (Tag/Zeit) statt. Zusätzliche Besprechungen / Diskussionen sind selbstverständlich möglich. Sollte keine Besprechung nötig sein, kann diese auch abgesagt werden.

Themen:

- Stand des Projektes
- Nächste Schritte
- Überprüfung der Meilensteine
- Diskussion fachlicher Themen (bei Bedarf)

Für jede Besprechung ist ein Kurzprotokoll zu erstellen, welches bis spätestens 2 Tage nach der Sitzung an den Betreuer gesendet werden muss.

6.3 Zwischenpräsentation

Termine für die Zwischenpräsentation und Prüfungen werden frühzeitig unter Einbezug aller Beteiligten (Verantwortlicher/Betreuer, Gegenleser, Experte, Studierende) bestimmt. Der Verantwortliche lädt zur Zwischenpräsentation und zur mündlichen Prüfung ein.

Manuel Bauer
HSR Hochschule für Technik Rapperswil ■ Oberseestrasse 10 ■ CH-8640 Rapperswil
T +41 55 222 41 11 ■ F +41 55 222 44 00 ■ office@hsr.ch ■ www.hsr.ch

Abstract

Die SUNCAR HK AG entwickelt, baut und vermarktet Elektro-Baumaschinen, welche als IoT-Devices Millionen von Sensor-Messwerten an einen zentralen Data Collector übermitteln. Diese Messdaten ermöglichen eine einfache und effiziente Überwachung der Maschinen. Da das bestehende System den stetig wachsenden Anforderungen nicht mehr gewachsen war, wurde im Rahmen dieser Bachelorarbeit eine ganzheitliche Neuentwicklung vorgenommen. Zudem durfte eine neue Webapplikation realisiert werden, weil die Endkunden ebenfalls Zugriff auf die Messdaten erhalten sollten.

Nach der anfänglichen Projektplanung und Anforderungsanalyse erfolgte die Evaluation einer für «Big Data» geeigneten Datenbanklösung. Mit InfluxDB konnte für die Messdaten eine Time Series Database eingeführt werden, welche die Anforderungen optimal erfüllt. Die restlichen Daten (Konfigurations- und Metadaten) werden in einer relationalen Datenbank (MariaDB) abgelegt. Anschliessend musste eine neue Schnittstelle für die Kommunikation zwischen Maschine und Data Collector konzipiert und implementiert werden. So konnte die bestehende, wenig skalierbare VPN-Lösung durch eine Message Queue (ActiveMQ) ersetzt und mit Client-Zertifikaten abgesichert werden. In den darauffolgenden Iterationen wurde die Entwicklung des Data Collectors und des Backends (ASP.Net Core) sowie des Frontends (React mit TypeScript) parallel vorangetrieben. Aufgrund eines idealen Projektverlaufes konnten zudem optionale Features wie eine API für Drittanbieter (ISO 15143-3) und das Event-Alerting (Kapacitor) umgesetzt werden. Alle Teilsysteme werden in Form von Docker-Container auf Microsoft Azure betrieben.

Nach insgesamt rund 1'100 Arbeitsstunden konnte dem Auftraggeber ein voll funktionsfähiger Prototyp übergeben werden. Der Auftraggeber hat sich vom Projektergebnis begeistert gezeigt und möchte in naher Zukunft mit dem neuen Gesamtsystem live gehen sowie den Endkunden die Webapplikation zur Verfügung stellen.

Management Summary

Ausgangslage

In Zeiten der Digitalisierung vernetzen sich immer öfters auch alltägliche Geräte aus dem Haushalt oder der Industrie mit dem Internet, um mit der Aussenwelt in Verbindung zu treten. Während dies bei Haushaltsgeräten oft noch als Spielerei anmutet, kann dadurch bei SUNCAR HK AG ein echter Mehrwert erzielt werden: Der Auftraggeber SUNCAR HK AG entwickelt, baut und vermarktet batteriebetriebene **Elektro-Baumaschinen**. Die Maschinen verfügen über verschiedenste Sensoren, welche **Millionen von Messwerten** an eine zentrale Sammelstelle (Data Collector) übermitteln. Diese Messdaten ermöglichen eine einfache und effiziente **Überwachung** der Maschine inklusive Auswertung und Fehleranalyse.

Abbildung 1: (SUNCAR TB216E)

Das bestehende Gesamtsystem von SUNCAR HK AG war den stetig wachsenden Anforderungen nicht mehr gewachsen und erforderte eine **ganzheitliche Neuentwicklung**. Da die Messdaten auch für die Endkunden von grossem Interesse sind, sollte zudem eine **Webapplikation** realisiert werden.

Vorgehen und Technologien

Während 17 Wochen und insgesamt rund 1'100 Arbeitsstunden wurde die Projektarbeit mit Hilfe eines **agilen und iterativen Vorgehensmodells** durchgeführt, welches sich in vielen Aspekten an Scrum orientiert. Nach der anfänglichen Projektplanung und der Anforderungsanalyse erfolgte die **Evaluation** einer für die grosse Datenmenge besser geeigneten Datenbanklösung. Mit der für Zeitserien optimierten Datenbank «InfluxDB» konnte dabei eine ideale Lösung gefunden und implementiert werden.

Anschliessend wurde ein **Schnittstellenkonzept** erarbeitet, um die bisherige Kommunikationsvorrichtung zwischen den Baumaschinen und der Sammelstelle zu ersetzen. Neu kommt eine in der Cloud betriebene Warteschlange (Message Queue) für die Messdaten zum Einsatz.

In den darauffolgenden Iterationen wurde die Entwicklung des **Data Collectors** und der **Webapplikation (Backend & Frontend)** mit ASP.NET Core und React parallel vorangetrieben. Aufgrund eines idealen Projektverlaufes konnten zudem verschiedene optionale Features in Angriff genommen werden.

Ergebnisse

Mit dem «Suncar Big Data Collector & Dashboard» wurde ein **umfassendes Gesamtsystem** entwickelt, welches über den folgenden Ablauf verfügt: Eine Baumaschine übermittelt ihre Messwerte via Message Queue an den Data Collector, welcher die Messdaten in einer dafür ausgelegten Datenbank persistiert. Dabei ist die Legacy-Kompatibilität für bereits ausgelieferte Maschinen sichergestellt.

Über die mandantenfähige Webapplikation werden den Endkunden nun die gewünschten Informationen in Form von interaktiven Messdiagrammen und einem virtuellen Maschinen-Cockpit präsentiert. Weitere Features wie die Verwaltung der Kunden, Firmen, Maschinen, etc. sowie eine externe API für Drittanbieter und das Event-Alerting (Datenanalyse) runden die Möglichkeiten des Systems ab.

Abbildung 2: Features des Gesamtsystems

Ausblick

Mit dem Abschluss dieses Projektes wird der SUNCAR HK AG ein voll funktionsfähiger **Prototyp** übergeben. Der Auftraggeber hat sich vom Projektergebnis begeistert gezeigt und möchte in naher Zukunft mit dem neuen Gesamtsystem **live gehen** und den Endkunden die Webapplikation zur Verfügung stellen.

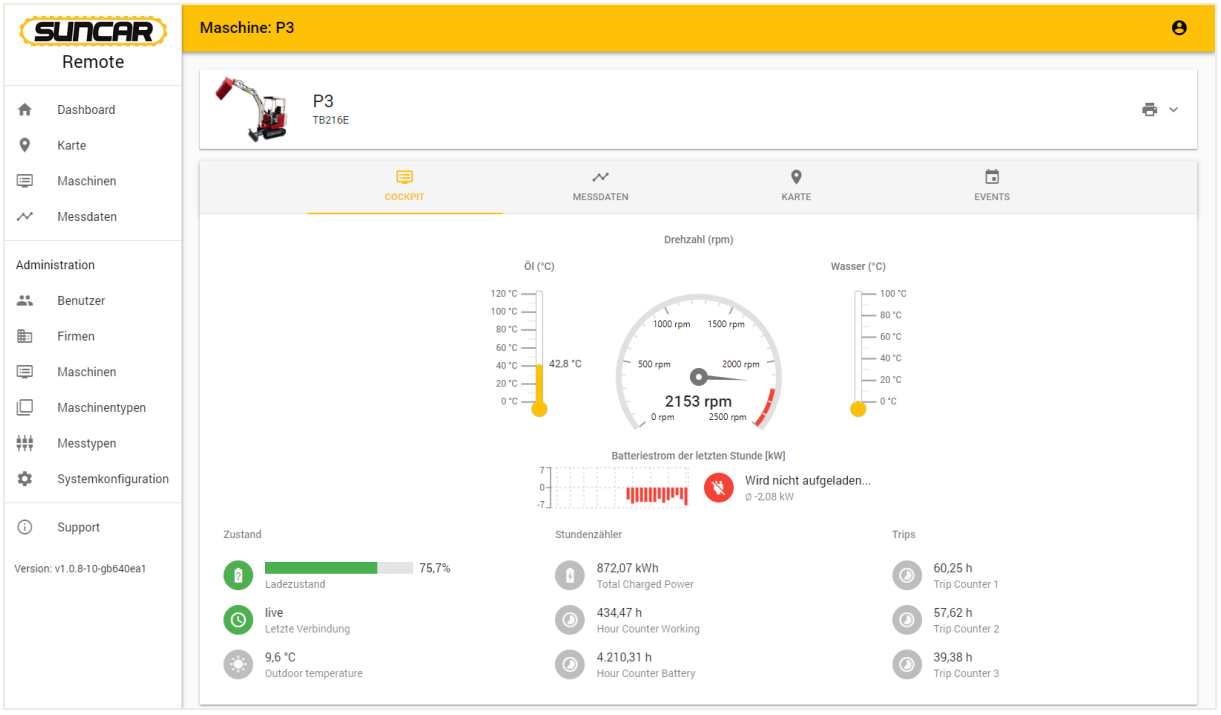


Abbildung 3: Virtuelles Maschinen-Cockpit in der Webapplikation

Inhaltsverzeichnis

1 Einleitung und Übersicht

1.1 SUNCAR HK AG

Die SUNCAR HK AG ist ein Startup, welches ursprünglich aus einem Fokusprojekt an der ETH Zürich entstanden ist und mittlerweile acht Mitarbeiter beschäftigt. Das Unternehmen entwickelt, baut und vermarktet autonome, batteriebetriebene Elektrofahrzeuge für den Offroad-Bereich. Das Ziel ist es, verschiedene bis anhin mit Dieselmotor betriebene Baumaschinen (z.B. Bagger) auf Elektroantrieb umzubauen und so den Dieseltreibstoff vollständig durch Solarenergie aus Photovoltaik zu ersetzen. Die von SUNCAR HK AG gebauten Elektro-Baumaschinen werden ausschliesslich mit Solarenergie betrieben. [1]

1.2 Problemstellung

In Zeiten der Digitalisierung vernetzen sich immer mehr Geräte aus dem Haushalt oder der Industrie mit dem "Internet der Dinge" (IoT), um mit der Aussenwelt in Verbindung zu treten. Während IoT-Geräte im Haushalt oft noch als Spielerei anmuten, kann bei SUNCAR HK durch die Nutzung von IoT ein echter Mehrwert erzielt werden.

Die Elektro-Baumaschinen von SUNCAR HK verfügen über verschiedenste Sensoren, welche dank IoT Millionen von Messwerten an einen zentralen Data Collector übermitteln. Diese Messwerte ermöglichen eine einfache und effiziente Überwachung der Maschine inklusive Auswertung und Fehleranalyse.

Der serverseitig bestehende Data Collector (Python-Skripte) und die Datenbanklösung (relationale Datenbank MySQL) werden den stetig wachsenden Anforderungen und Datenmengen nicht mehr gerecht. Die Weboberfläche zur Ansicht der Messdaten (PHP) wurde unter hohem Zeitdruck ursprünglich nur für interne Zwecke entwickelt. Da die Messdaten auch für die Endkunden von grossem Interesse sind, möchte SUNCAR HK das Frontend auch für Kunden zugänglich machen.

1.3 Suncar Big Data Collector & Dashboard

Im Zuge dieser Bachelorarbeit sollen die verschiedenen Teilsysteme ganzheitlich neu entwickelt werden. So muss ein neu zu entwickelnder **Data Collector** auch mit einer steigenden Anzahl von Elektromaschinen, Sensoren und Messwerten für einen reibungslosen Empfang aller Daten sorgen, ohne dabei die bereits ausgelieferten Legacy-Maschinen aussen vor zu lassen. Für die Speicherung aller Messdaten muss eine neue **Datenbanklösung** evaluiert und konzipiert werden, welche die Kapazitätsprobleme der bestehenden Datenbank wegen der hohen Anzahl von Messdaten beseitigt.

Anstelle des ursprünglich nur für den internen Gebrauch entwickelten Frontends soll ein modernes und benutzerfreundliches **Web GUI** (Dashboard) geschaffen werden, welches auf einem ebenfalls zu implementierenden Backend aufsetzt. Im neuen Frontend müssen einerseits die

[1] (Unternehmen: Vision, 2018)

gesammelten Messdaten begutachtet und andererseits Maschinen, Mess- und Maschinentypen sowie Kunden (Unternehmen/Benutzer) verwaltet werden können. Ein Hauptaugenmerk soll dabei auf die **Mandantenfähigkeit** gelegt werden, damit das Frontend von unterschiedlichen Kunden für ihre eigenen Maschinen genutzt werden kann, ohne dabei Einblick in andere Unternehmen und deren Maschinen zu erhalten.

Des Weiteren könnte das Gesamtsystem um zusätzliche, optionale Komponenten ausgebaut werden. So kann eine separate **API für Drittanbieter** zur Verfügung gestellt werden, worüber die Maschinen- und Messdaten direkt abgefragt und beispielsweise auf der eigenen Website eines Kunden eingebunden werden können. Für die Auswertung der Messdaten kann zudem ein Statistik-Modul integriert oder implementiert werden, welches die **Datenanalyse** vereinfachen soll. Daraus ergeben sich noch weitere Möglichkeiten, wie beispielsweise ein Event Alerting bei der Überschreitung eines Schwellenwertes oder komplexere datengestützte Analysen (Predictive Maintenance).

Abbildung 4: Features von «Suncar Big Data & Dashboard»

Für detailliertere Informationen zur vorliegenden Problemstellung und den Projektzielen sei an dieser Stelle auf die offizielle Aufgabenstellung verwiesen.

1.4 Anforderungen

Basierend auf der Aufgabenstellung wurde eine Anforderungsanalyse durchgeführt, um gemeinsam mit dem Auftraggeber die technischen Anforderungen an das Gesamtsystem zu eruieren. Dabei wurden alle Anforderungen in Form von User Stories im Product Backlog erfasst und einem übergeordneten Feature zugewiesen.

In der nachfolgenden Tabelle sind alle Features und deren User Stories aufgeführt. Eine detailliertere Beschreibung der User Stories und deren Erfüllungsgrad ist in den Dokumenten «Anforderungsspezifikation» und «Systemtest» im Anhang zu finden.

Tabelle 1: Funktionale Anforderungen in Form von Features und User Stories

Feature «Dashboard»	
↑ Dashboard	↓ Statuspage
↑ Kontaktdaten ansehen	
Feature «User-Verwaltung»	
↑ Login	↑ Mein Profil ansehen
↑ Logout	↑ Andere Profile ansehen
↑ Account aktivieren	↑ User-Liste ansehen
↑ Account löschen	↑ Mein Profil bearbeiten
↑ Passwort zurücksetzen	↑ Andere Profile bearbeiten
↑ User erfassen (als Admin)	↑ Andere Accounts löschen
↑ Super User/User erfassen	↓ Auswertung User-Aktivität
Feature «Company-Verwaltung»	
↑ Company erfassen	↑ Company bearbeiten
↑ Company ansehen	↑ Andere Company bearbeiten
↑ Andere Company ansehen	↑ Company löschen
↑ Company-Liste ansehen	
Feature «Maschinen-Verwaltung»	
↑ Maschine erfassen	↑ Virtuelles Cockpit ansehen
↑ Details von Maschine ansehen	↑ Maschinenzustand PDF
↑ Maschinen-Liste ansehen	↓ Maschinen-Blacklist
↑ Maschine bearbeiten	✱ Maschinen-Cards: Filterung
↑ Maschine löschen	✱ Cockpit: farbiger Status
↑ Aktueller Maschinen-Status ansehen	✱ Detailansicht: Maschine auswählen
↑ Aktueller Maschinen-Standort ansehen	

Feature «Maschinentyp-Verwaltung»	
↑ Maschinentyp erfassen	↑ Maschinentyp bearbeiten
↑ Details von Maschinentyp ansehen	↑ Maschinentyp löschen
↑ Maschinentypen-Liste ansehen	

Feature «Messtyp-Verwaltung»	
↑ Messtyp erfassen	↑ Messtyp löschen
↑ Details von Messtyp ansehen	↑ Messtyp-Sichtbarkeit verwalten
↑ Messtypen-Liste ansehen	↑ Messtyp an Maschinentyp zuweisen
↑ Messtyp bearbeiten	

Feature «Ownership-Verwaltung»	
↑ Maschine an Company zuweisen	↑ Maschine-Company-Zuweisung löschen
↑ Zuweisungen von Maschinen zu Company ansehen	↑ Maschine-Company-Zuweisung bearbeiten

Feature «Access-Verwaltung»	
↑ Maschinen-Access für User erfassen	↑ Maschinen-Access für User bearbeiten
↑ Eigene Maschinen-Accesses anzeigen	↑ Beliebiger Maschinen-Access für User verwalten
↑ Maschinen-Access-Liste ansehen	↑ Maschinen-Access für User löschen

Feature «Karte»	
↑ Maschinen-Karte ansehen	↓ GPS-Spur anzeigen
↑ Maschinen-Details auf Karte ansehen	✱ Maschinen-Karte: farbiger Status
↑ Karte zentrieren	✱ Maschinen-Karte: Filterung
↑ Von Maschinen-Karte zu Detailansicht navigieren	

Feature «Messdatenauswertung»	
↑ Messdaten-Diagramm ansehen	↑ Messdaten PDF
↑ Diagramm mit mehreren Messdaten ansehen	✱ Messdaten-Diagramm: Maschine auswählen
↑ Alte Messdaten ansehen	✱ CSV-Export Messdaten

Feature «Zugriff»
↑ Datenschutz

Feature «Data Collector»	
↑ Messdaten senden (neu)	↑ Data Collector
↑ Messdaten senden (Legacy)	↑ Legacy Data Collector
Feature «API für Drittanbieter»	
↓ API: Maschinenstatus lesen	↓ API: Authentifizierung
↓ API: Messdaten lesen	✱ Schnittstelle ISO 15143-3
Feature «Datenanalyse»	
↓ Datenanalysen durchführen	↓ Fehlermeldungen ansehen
↓ Benachrichtigung bei Schwellwert	↓ Fehlermeldung per Mail erhalten

Alle aufgeführten User Stories wurden durch den Auftraggeber priorisiert, wobei vier Prioritätsstufen zur Auswahl standen. SUNCAR HK hat sich jedoch für eine grobe Unterteilung zwischen «zwingend» und «optional» entschieden. Des Weiteren kamen im Verlauf des Projektes fortlaufend weitere Wünsche des Auftraggebers hinzu, welche als «Feature Requests» erfasst wurden.

Tabelle 2: Priorisierung der User Stories

Priorität	Beschreibung
↑	Zwingend (hohe Priorität)
↓	Optional (geringe Priorität)
✱	Feature Request (während der Implementation)

2 Vorgehen und Technologien

2.1 Vorgehensmodell

Um für einen reibungslosen Ablauf des Projektes zu sorgen, wurde ein agiles und iteratives Vorgehensmodell angewendet, welches sich in vielen Aspekten an Scrum[2] orientiert. Scrum eignet sich sehr gut für ein Softwareprojekt mit einer hohen Komplexität, bei dem flexibel definierte und möglicherweise wachsende Anforderungen im engen Kontakt mit dem Auftraggeber umgesetzt werden.

Aufgrund bestehender Rahmenbedingungen, der Grösse des Entwicklerteams und fehlender Scrum-Komponenten (kein Scrum Master, kein klassischer Product Owner, etc.) wurden die Konzepte von Scrum an die Rahmenbedingungen dieser Projektarbeit angepasst und in entsprechender Form angewendet.

2.2 Zeitliches Vorgehen

Die vorliegende Bachelorarbeit wurde während 17 Wochen in insgesamt rund 1'100 Arbeitsstunden durchgeführt – verteilt auf die Projektmitglieder Tobias Stähli, Dumeni Vincenz und Jonas Wälter. Die Umsetzung des Projektes erfolgte inkrementell, in dem die zur Verfügung gestandene Projektzeit in zweiwöchige Iterationen unterteilt wurden.

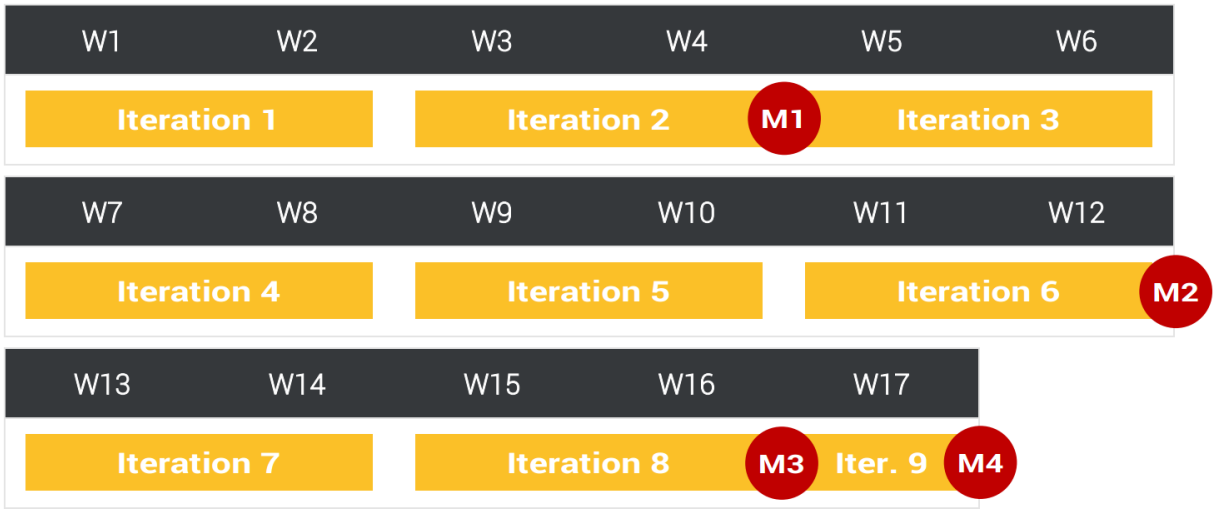


Abbildung 5: Aufteilung der Projektdauer in zweiwöchige Iterationen und Meilensteine

Des Weiteren wurden zu Projektbeginn verschiedene Meilensteine definiert, welche die Projektdauer hinsichtlich der Softwareentwicklung grob abstecken. Für eine detailliertere Beschreibung der Iterationen und Meilensteine sei an dieser Stelle auf das Dokument «Projektplan» verwiesen.

[2] (Was ist Scrum?, 2018)

2.3 Technologien

Bei der Realisierung des Projektes konnte auf verschiedenste Technologien, Programmiersprachen, Frameworks, Anbieter, etc. zurückgegriffen werden. In der nachfolgenden Tabelle sind alle eingesetzten Technologien inklusive Einsatzzweck aufgeführt.

Tabelle 3: Eingesetzte Technologien (Logos, 2018)

Frontend		
	TypeScript	Programmiersprache
	React	UI Library
	Redux	State-Verwaltung
	Material-UI	Material Design
	Cypress	End-to-End Testing
	Visual Studio Code	Entwicklungsumgebung
Backend		
	C#	Programmiersprache
	ASP.NET Core / Entity Framework Core	Framework für Web-Applikationen / OR-Mapper
	InfluxDB & Kapacitor	Time Series Database für Messdaten
	MariaDB	Relationale Datenbank für Metadaten
	Swagger	API-Dokumentation
	Apache ActiveMQ	Message Queue für Messdaten-Transport
	Visual Studio	Entwicklungsumgebung
Infrastruktur		
	Docker	Containervirtualisierung
	Microsoft Azure	Cloud Hosting
	Visual Studio Team Services	Projekt- & Quellcodeverwaltung, Deployment

3 Ergebnisse

Die nachfolgenden Unterkapitel beschreiben die implementierten Endergebnisse im Bereich der jeweiligen Teilsysteme. Zudem wird der Projektverlauf und einige Kennzahlen wie «Lines of Code» diskutiert und aufgezeigt. Detailliertere Informationen zu den Evaluationen, den Konzepten und der Softwarearchitektur können in den entsprechenden Dokumenten im Anhang eingesehen werden.

3.1 Data Collector & Schnittstellenkonzept

Als Ersatz für den bisherigen Data Collector, welcher auf verschiedenen Python-Skripten und fixen Port-Zuweisungen basiert, wurde eine neue, mehrstufige Lösung realisiert. Da die bereits ausgelieferten, weltweit verteilten Maschinen nicht ohne Weiteres auf ein neues System angepasst werden konnten, erwies sich die Umstellung als komplexes Unterfangen. Daher wurde ein umfangreiches Schnittstellenkonzept ausgearbeitet und umgesetzt.

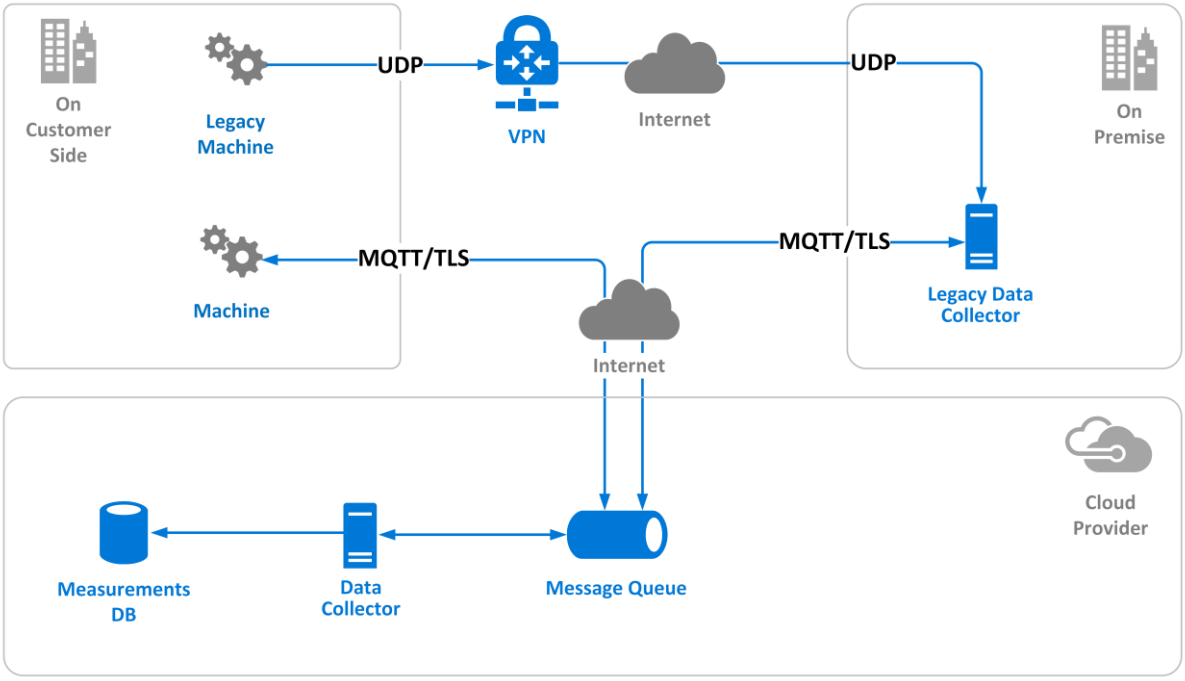


Abbildung 6: Kommunikation zwischen Maschine, Message Queue und Data Collector

Das Schnittstellenkonzept beinhaltet die funktionalen und nichtfunktionalen Anforderungen an die Schnittstelle zwischen Maschine und Data Collector, eine Analyse und Beschreibung der bestehenden Schnittstelle sowie die Evaluation und Definition der neuen Schnittstelle. Als zentrale Punkte kann dabei einerseits der Einsatz einer bidirektionalen, auf dem MQTT-Protokoll[3] basierende Message Queue und andererseits die Bestückung der Maschinen mit Client-Zertifikaten zugunsten der Sicherheit hervorgehoben werden.

[3] (MQTT Protocol, 2018)

Um die Legacy-Kompatibilität mit den bereits ausgelieferten Maschinen aufrecht zu erhalten, wurde zusätzlich ein Legacy Data Collector implementiert. Dieser übernimmt gegenüber den Maschinen die Funktion des bisherigen Data Collectors und reicht die Messdaten gemäss dem erarbeiteten Schnittstellenkonzept via Message Queue an den neuen Data Collector weiter.

Während die Message Queue und der Data Collector in der Cloud betrieben werden, ersetzt der Legacy Data Collector den bisherigen Data Collector auf dem lokalen Server von SUNCAR HK. Dies war erforderlich, weil die im Schnittstellenkonzept erwähnte VPN-Infrastruktur nicht bis in die Cloud erweitert werden sollte. Die beiden Data Collectors wurden als ASP.NET-Applikationen in C# entwickelt. Als Message Queue kommt aktuell ActiveMQ zum Einsatz, welche jedoch bei Bedarf ohne Weiteres durch eine andere Implementation des MQTT-Protokolls ersetzt werden kann.

Für weitere Informationen und Definitionen der Kommunikation zwischen Maschine und Data Collector sei an dieser Stelle auf das Dokument «Schnittstellenkonzept» verwiesen.

3.2 Datenbanklösung

Die bisherige relationale Datenbank sah sich mit massiven Kapazitätsproblemen aufgrund der enormen Datenmengen der Messwerte konfrontiert. Im Zuge der Projektarbeit wurde daher eine neue zweiteilige Datenbanklösung evaluiert und konzipiert.

3.2.1 Measurements DB

Für die Persistierung der beträchtlichen Anzahl an Messdaten wurde eine «Time Series Database» von InfluxDB[4] eingeführt. Time-Series-Datenbanken sind für die Verwaltung von Zeitserien optimiert, wobei zu jedem Datenpunkt auch ein zugehöriger Zeitstempel als Index abgelegt wird. Durch diese Eigenschaften ist InfluxDB ideal für den Einsatz in der vorliegenden Problemstellung geeignet.

3.2.2 Metadata DB

Nebst der NoSQL-Datenbank für die Messdaten wird eine durch den Auftraggeber vorgegebene MariaDB als klassische relationale Datenbank eingesetzt, welche die restlichen Daten (Konfigurations- und Metadaten) beinhaltet.

3.3 Cloud Provider & Continuous Deployment

Da der Auftraggeber zuvor noch keine Dienstleistungen über einen Cloud Provider bezogen hatte, musste ein geeigneter Provider evaluiert und ein entsprechendes Lösungskonzept erarbeitet werden. Dabei wurden verschiedene Lösungsansätze und die Kosten der jeweiligen Anbieter detaillierter betrachtet.

[4] (InfluxDB, 2018)

Nach der Abwägung aller Alternativen und unter Berücksichtigung der Kosten wurde entschieden, mindestens während der Projektphase eine virtuelle Maschine auf Microsoft Azure mit einer manuellen Installation von Docker als Basis für das System einzusetzen. Alle Teilsysteme, welche in der Cloud betrieben werden müssen, wurden daher in Form von Docker-Containern aufgesetzt. Durch den Einsatz dieser Containerlösung ist sowohl die Skalierung als auch die Portabilität gewährleistet. So können die einzelnen Docker-Container bei Bedarf ohne beträchtlichen Aufwand beispielsweise zu einem anderen Cloud-Anbieter oder in eine verwaltete Docker-Umgebung portiert werden.

Mit Hilfe von Visual Studio Team Services wurde zudem eine eigene Staging-/Deployment-Umgebung konzipiert und aufgebaut. So werden nun alle Änderungen auf dem Master-Branch automatisch auf eine Staging-Umgebung deployt, sofern sie zuvor alle automatisierten Code-Formatierungs-Checks, Unit Tests und End-to-End-Tests erfolgreich durchlaufen haben. Die Änderungen können dann auf der Staging-Umgebung begutachtet und per Knopfdruck auch auf das produktive System deployt werden.

3.4 Backend

Mit dem neuen Backend wurde eine zentrale Komponente entwickelt, welches mehrere Umsysteme zusammenhält und miteinander verbindet. So greift das Backend auf die Messdaten in der Measurements DB und die Konfigurationsdaten in der Metadata DB zu, um diese dem Frontend über eine interne Web-API zur Verfügung zu stellen. Für die Ablage von Bilddateien und den Versand von E-Mails wurden zudem ein Blob Storage und ein Email Gateway angebunden, welche wie das gesamte Backend ebenfalls in der Cloud betrieben werden.

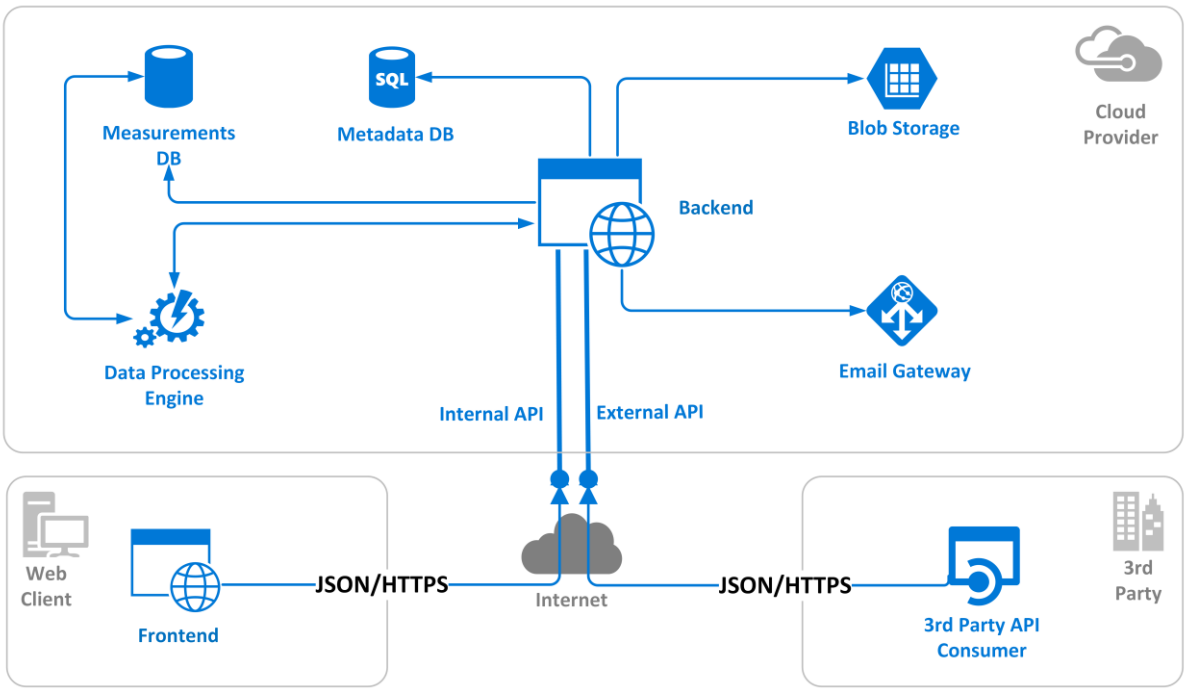


Abbildung 7: Systemübersicht des Backends/Frontends

Das Backend wurde als ASP.NET Core Web API entwickelt, welche in die drei Schichten «WebAPI», «BusinessLayer» und «DataAccess» unterteilt ist. Der Zugriff auf die Metadata DB (MariaDB) erfolgt dabei mit Hilfe von Entity Framework Core, während für Abfragen der Measurements DB (InfluxDB) die separate .NET Library (InfluxData.Net, 2018) im Einsatz steht.

3.5 Frontend

Als ein weiteres Herzstück des Gesamtsystems kann das neue Frontend bezeichnet werden, ist es doch die direkte Schnittstelle zu den Endkunden von SUNCAR HK. Mit Hilfe von TypeScript, React und Redux konnte ein modernes Web GUI geschaffen werden, welches dank Material Design und Responsive Design für eine hohe Benutzerfreundlichkeit sorgt.

Über das Frontend stehen den Endkunden und den Administratoren zusammengefasst die folgenden Möglichkeiten zur Verfügung:

- Virtuelles Maschinen-Cockpit (live)
- GPS-Verlauf einer Maschine
- Karten-Ansicht aller eigenen Maschinen
- Analyse von Messdaten mit Hilfe von Messdiagrammen
- Supportanfrage
- Verwaltung von Benutzern, Firmen, Maschinen, Maschinen- und Messtypen, Maschine-Firma-Zuweisungen (Ownership), Benutzerzugriffe und der Systemkonfiguration

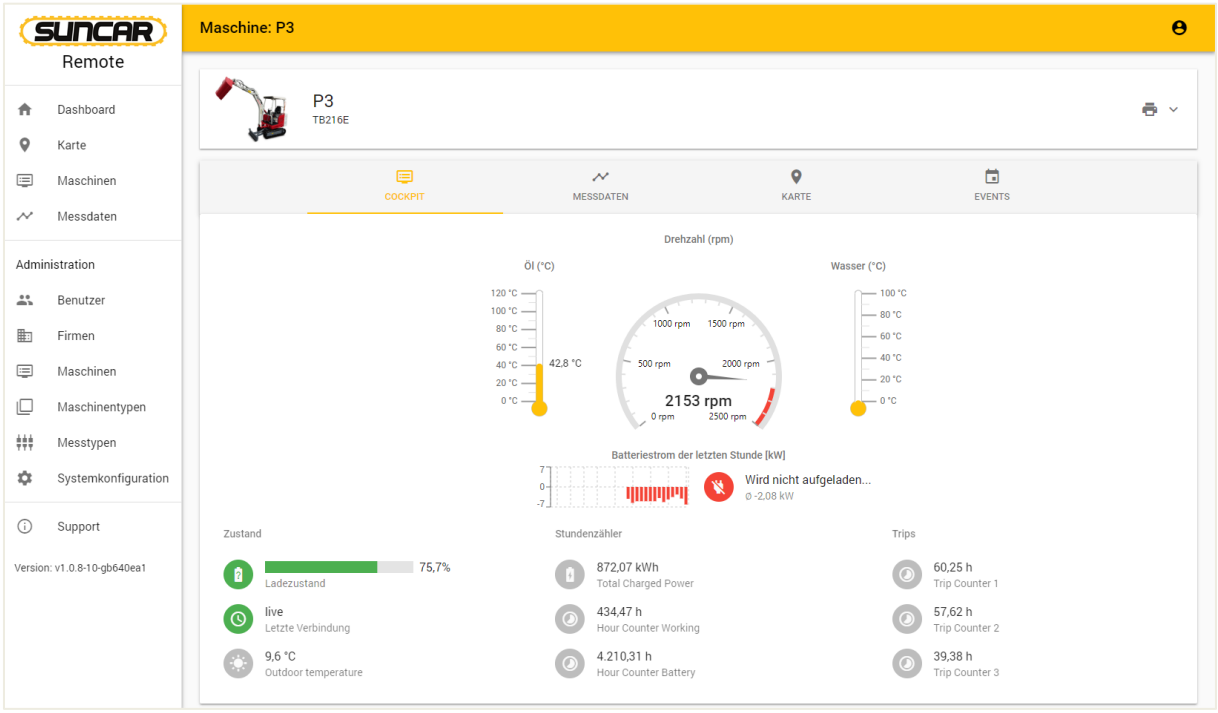


Abbildung 8: Virtuelles Maschinen-Cockpit im Frontend

Aufgrund der geforderten Mandantenfähigkeit wurde ein umfassendes Berechtigungssystem mit verschiedenen Benutzerrollen (User, Superuser, Distributor, Administrator) entwickelt, welches sowohl im Frontend als auch im Backend integriert ist. So wird dafür gesorgt, dass ein Benutzer nur die Daten sehen und verwalten kann, auf welche er Zugriff hat. Für die detaillierte Berechtigungsübersicht sei an dieser Stelle auf das Dokument «Domainanalyse» verwiesen.

Um die Funktionalität und das Responsive Design der Weboberfläche sowie die Backend-API automatisiert und fortlaufend überprüfen zu können, wurden zudem grafische End-to-End-Tests basierend auf dem Test Runner «Cypress»[5] implementiert.

3.6 API für Drittanbieter

Neben der internen API für den Zugriff über das Frontend konnte auch eine externe API für Drittanbieter implementiert werden, damit ein Endkunde Informationen zu seinen Maschinen beispielsweise direkt auf der eigenen Website präsentieren kann. Mit hohem Aufwand wurde dabei die umfassende ISO-Norm ISO/TS 15143-3 «Earth-moving machinery and mobile road construction machinery – Worksite data exchange – Part 3: Telematics data»[6] umgesetzt.

Über diese neue API können nun einerseits statische Maschinendaten wie das Installationsdatum oder die Marke des Herstellers und andererseits aktuelle, fortlaufende Daten wie der Standort, die Arbeitsstunden oder der Motorstatus abgefragt werden. Für eine detaillierte Beschreibung der Umsetzung der ISO-Norm sei an dieser Stelle auf das Dokument «Umsetzung ISO 15143-3» verwiesen.

3.7 Datenanalyse

Als letztes Teilsystem wurde ein Statistik-Modul in das Gesamtsystem integriert. Dabei handelt es sich um die Data Processing Engine «Kapacitor»[7], welche vom Hersteller der InfluxDB zusätzlich entwickelt wurde und deshalb ideal auf die eingesetzte Datenbanklösung abgestimmt ist. Mit Hilfe von Kapacitor können der Datenstream von InfluxDB verarbeitet und allfällige Alerts ausgelöst werden.

So kann nun im Frontend eine Eventvorlage für einen Maschinentyp erstellt und für eine spezifische Maschine aktiviert werden. Wenn ein Schwellenwert der Eventvorlage überschritt wird, stehen drei mögliche, kombinierbare Aktionen zur Auswahl.

Als erste Möglichkeit kann der ausgelöste Alert als Event in die InfluxDB geschrieben und so den Benutzern der betroffenen Maschine im Frontend angezeigt werden. Zusätzlich kann der Alert auch in Form eines normalen Messwertes bei den restlichen Messdaten abgelegt und dadurch im Frontend beispielsweise im Messdatendiagramm analysiert werden. Zuletzt kann der Alert über eine interne API an das Backend weitergereicht werden, welches allen betroffenen Benutzern eine Benachrichtigung per E-Mail zukommen lässt. Falls ein Benutzer keine E-Mail-Benachrichtigungen erhalten möchte, kann er diese Option bei Bedarf wieder deaktivieren.

[5] (Cypress, 2018)

[6] (ISO/TS 15143-3:2016, 2016)

[7] (Kapacitor, 2018)

3.8 Statistische Daten zum Projektverlauf

In rund 1'100 Arbeitsstunden wurden insgesamt 1'080 Commits im Source Code Repository gemacht, was im Schnitt fast einem Commit pro Stunde entspricht. Insgesamt wurden 207 Pull Requests mit total 11 Releases auf das produktive System durchgeführt. Gesamthaft wurden gut 53'000 Zeilen Code geschrieben, wovon gut 10'000 Zeilen generierter Code ist.

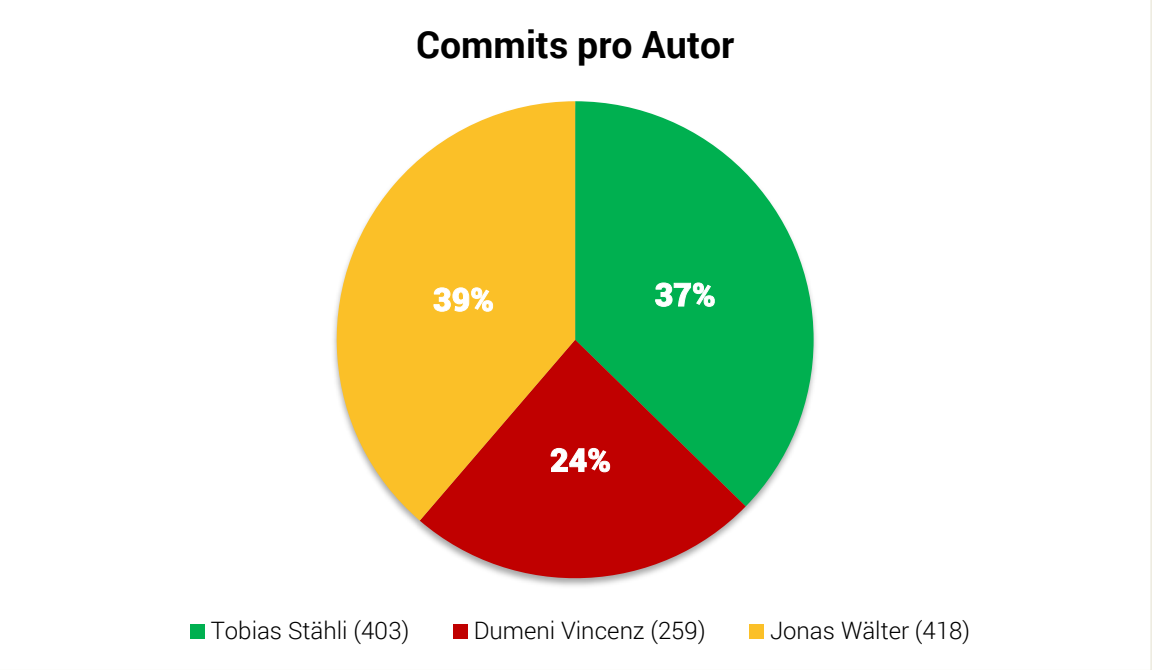


Abbildung 9: Anzahl Commits pro Autor

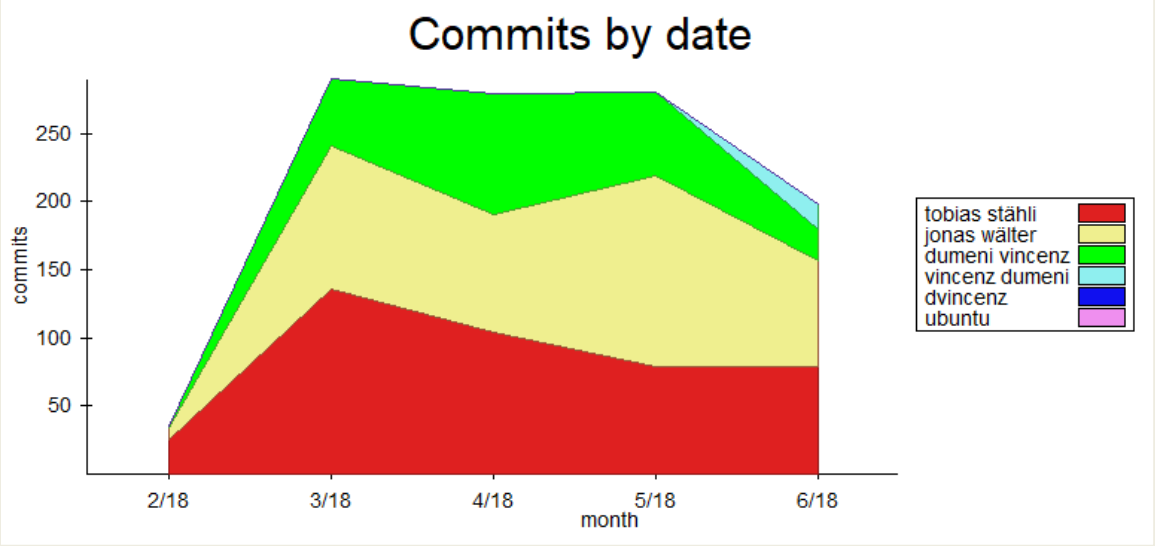


Abbildung 10: Anzahl Commits im Projektverlauf

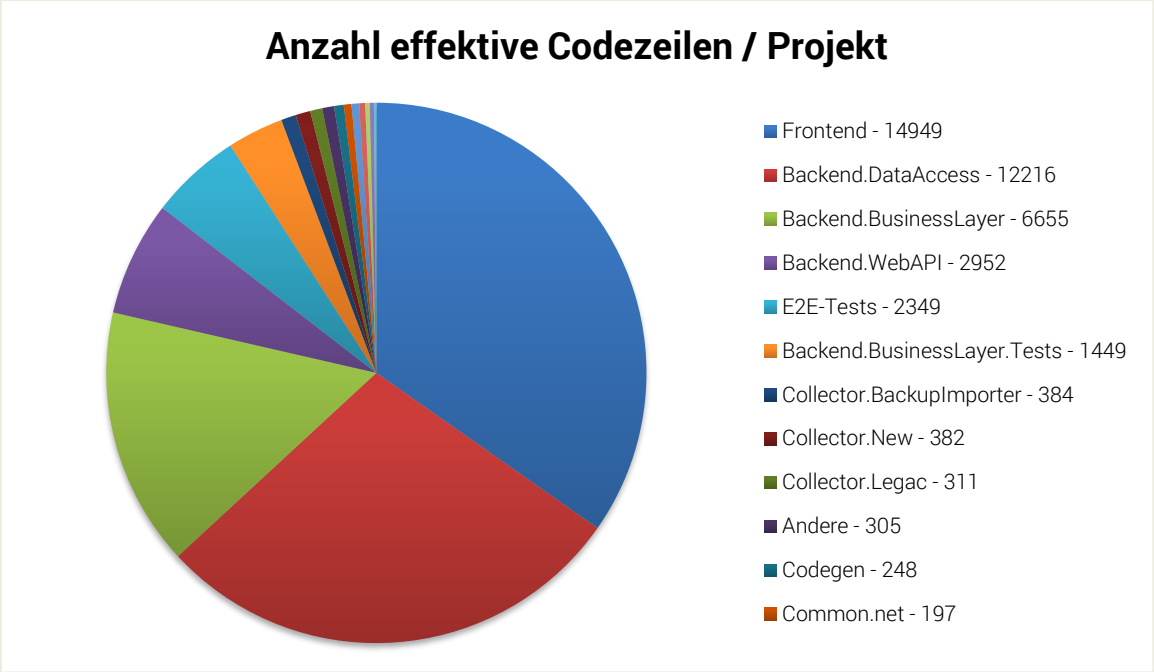


Abbildung 11: Aufteilung Codezeilen pro Unterprojekt

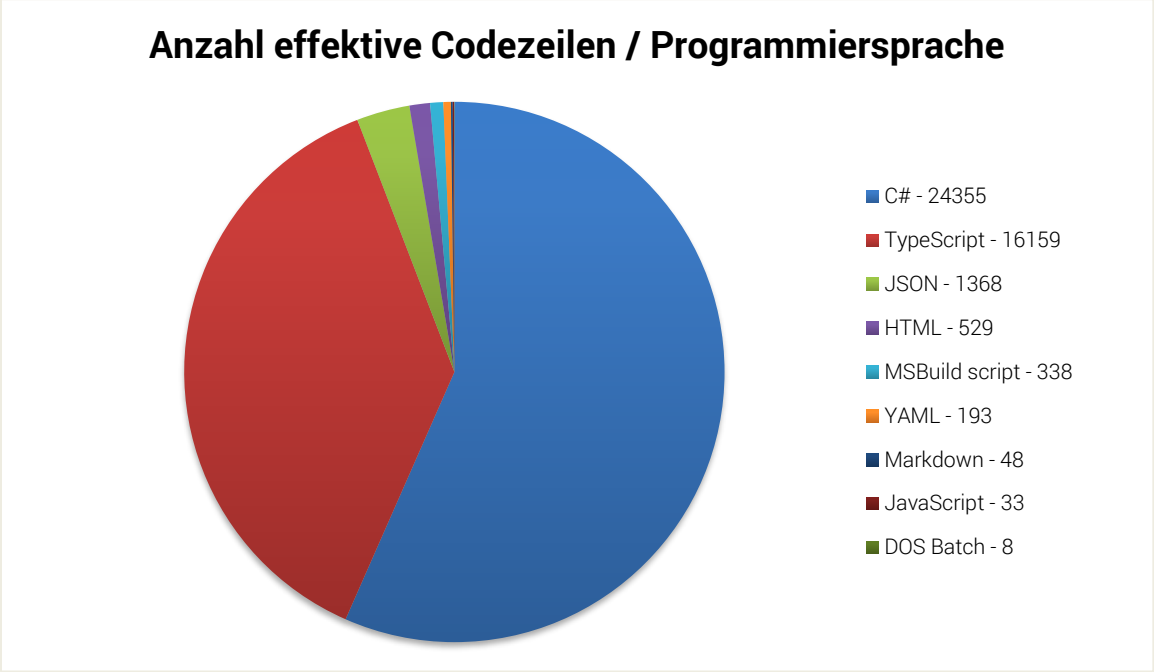


Abbildung 12: Aufteilung Codezeilen pro Programmiersprache

4 Schlussfolgerungen

Nach einer 17-wöchigen Projektdauer kann die Bachelorarbeit nun abgeschlossen werden – Zeit das Erreichte zu bewerten und Schlussfolgerungen anzustellen.

4.1 Bewertung der Ergebnisse

In den nachfolgenden Unterkapiteln werden die Ergebnisse bewertet und die zwingenden, optionalen, zusätzlichen sowie nicht erfüllten Resultate beschrieben.

4.1.1 Zwingende Resultate

In der Aufgabenstellung und im Zuge der durchgeführten Anforderungsanalyse wurden die folgenden zwingenden Resultate eruiert und festgehalten. Wie der Tabelle vorweggenommen werden kann, wurden alle zwingenden Resultate ausnahmslos umgesetzt.

Tabelle 4: Die zwingenden Resultate

Zwingende Resultate
✓ Datensammlung / Schnittstellenkonzept
Der bisherige Data Collector wurde erfolgreich durch ein neues System inklusive Message Queue und Legacy-Kompatibilität ersetzt. Als Grundlage dafür wurde ein umfassendes Schnittstellenkonzept erarbeitet.
✓ Datenbank
Anstelle der bisherigen relationalen Datenbank wurde eine zweistufige Datenbanklösung evaluiert und in Betrieb genommen, welches die Anforderungen der vorliegenden Problemstellung optimal erfüllt.
✓ Backend-API & Web GUI
Als Herzstück des Gesamtsystems wurde ein Backend und Frontend realisiert, welche allen Anforderungen der Aufgabenstellung gerecht wird. (Darstellung der Messdaten, Modifikation der Metadaten, Mandantenfähigkeit, etc.)
✓ Software-/Hardwareinfrastruktur, Skalierbarkeit & Redundanz
Dank dem Einsatz einer Containerlösung für alle Teilsysteme können die Skalierbarkeit und Redundanz gewährleistet werden. Ein Performance-Test mit rund 3.2 Millionen Messdaten über einen Zeitraum von knapp 17 Minuten ist reibungslos und ohne Probleme verlaufen.

✓	**Responsive Design, Browsersupport**
	Das gesamte Frontend-Design wurde stets mit Rücksicht auf verschiedene Bildschirmgrössen entwickelt und fortlaufend für die Mobile-Ansicht optimiert. Durch den Einsatz von End-to-End-Tests wird sichergestellt, dass das Frontend auch auf kleinen Displays bedient werden kann. Zudem lässt sich das Frontend in allen geforderten Browser problemlos bedienen.
✓	**5 Jahre Aufbewahrung von Messdaten**
	Dank InfluxDB besteht kein Kapazitätsproblem mehr in der Datenbank, wodurch Messdaten auch über eine Zeitdauer von fünf Jahren hinaus aufbewahrt werden können.
✓	**Datensicherheit: Passwortschutz & Verschlüsselung**
	Für den Zugriff auf das Frontend und das Backend ist eine Authentifizierung mittels E-Mailadresse und Passwort notwendig. Zusätzlich erfolgt der gesamte öffentliche Datenverkehr verschlüsselt.
✓	**Zwingende User Stories gemäss Anforderungsspezifikation**
	Alle zwingenden User Stories gemäss Kapitel 1.4 wurden vollumfänglich umgesetzt.

4.1.2 Optionale Resultate

Neben den zwingenden Resultaten wurden in der Aufgabenstellung und der Anforderungsspezifikation auch einige optionale Resultate definiert, welche in der nachfolgenden Tabelle aufgeführt und bewertet werden. Wie bereits bei den zwingenden Resultaten konnten auch bei den optionalen Resultaten alle Ziele erreicht werden.

Tabelle 5: Die optionalen Resultate

Optionale Resultate	
✓	**API für Drittanbieter**
	Nachdem in der Aufgabenstellung lediglich eine simple API zur Abfrage von Maschinendaten gefordert wurde, kam während der Projektphase beim Auftraggeber der Wunsch auf, die neue ISO-Norm zu berücksichtigen. Mit hohem Zusatzaufwand konnte die umfangreiche ISO-Norm schlussendlich erfolgreich umgesetzt werden.
✓	**Evaluation und Integration/Implementation eines Statistik-Moduls Event-Alerting z.B. bei Überschreitung eines Schwellenwertes**
	Mit der Integration von Kapacitor konnten zwei zusätzliche optionale Resultate kombiniert werden. Durch die Integration der Data Processing Engine steht SUNCAR HK nun ein tiefgreifendes Statistik-Modul zur Verfügung. Als Beispielanwendung wurde zudem das Event-Alerting umgesetzt und inklusive Verwaltung direkt in das Frontend integriert.

Aufbau einer Continuous Integration und Staging-Umgebung

Um die Entwicklung und das Deployment zu vereinfachen, wurde bereits zu Beginn des Projektes eine automatisierte Staging-Umgebung mit einem DEV- und einem PROD-System aufgebaut. Änderungen auf dem Master-Branch werden nach erfolgreichen Tests automatisch in das DEV-System deployt und können anschliessend per Knopfdruck auch in das PROD-System deployt werden.

Optionale Resultate gemäss Anforderungsspezifikation

Die nachfolgenden optionalen User Stories konnten umgesetzt werden:

- Statuspage zu den laufenden Systemen (Datenbanken) auf dem Dashboard
- Auswertung der User-Aktivität (Integration von Google Analytics)
- Maschinen-Blacklist, um beliebige Maschinen für normale Benutzer auszublenden
- GPS-Spur einer Maschine anzeigen
- Zeitbasierte Einschränkungen der für einen User sichtbaren Messdaten
- API für Drittanbieter: Authentifizierung mittels OAuth 2.0
- API für Drittanbieter: Maschinenstatus lesen
- API für Drittanbieter: Messdaten lesen
- Datenanalyse: Datenanalysen durchführen
- Datenanalyse: Benachrichtigung bei Schwellwert

4.1.3 Zusätzliche Resultate

Da im Zuge der Entwicklung neue Wünsche des Auftraggebers aufkamen oder vom Projektteam aus als sinnvoll erachtete Erweiterungen implementiert wurden, konnten zusätzliche Resultate erzielt werden.

Tabelle 6: Die zusätzlichen Resultate

Zusätzliche Resultate
Fake Client Um eine laufende Baumaschine zu simulieren und so den Data Collector sowie den Legacy Data Collector zu testen, wurde ein entsprechender «Fake Client» implementiert, welcher das Senden von beliebigen Messdaten und das Abspielen von aufgezeichneten Messdaten (via CSV-File) erlaubt.
Projekt «Codegen» Damit im Backend und im Frontend die gleichen Datentransferobjekte/Models zur Verfügung stehen, wurde ein eigenes Codegen-Projekt implementiert, welches auf die Eigenheiten des Systems abgestimmt ist. Der Codegenerator liest die Datentransferobjekte des Backends (C#-Klassen) aus und erstellt basierend darauf übereinstimmende TypeScript-Models für das Frontend.

Backup Importer / Archivdaten

Da die bisherige Datenbank durch die neue Datenbanklösung ersetzt wird, wäre es für SUNCAR HK wünschenswert, wenn die bestehenden Archivdaten von der alten in die neue Datenbank migriert werden können. Um dieses Vorgehen zu vereinfachen, wurde ein entsprechender Backup Importer implementiert. Dieser Backup Importer konnte zugleich auch für die durchgeführten Stresstests verwendet werden.

Grafana

Für eine direkte Überwachung der InfluxDB wurde in einem zusätzlichen Docker-Container Grafana[8] in das Gesamtsystem eingebunden. Grafana ist ein Monitoring/Analytics-Tool unter anderem für InfluxDB, welches eine vom Frontend unabhängige Überwachung der Messdaten erlaubt.

Evaluation Cloud-Lösung/-Provider

Da der Auftraggeber zu Beginn des Projektes noch keine Cloud-Dienstleistungen im Einsatz hatte, wurde eine entsprechende Evaluation der Cloud-Lösung/-Provider durchgeführt.

Unit Testing

Obwohl in der Aufgabenstellung die Implementation von Unit Test nicht enthalten ist, wurden die unterschiedlichen Schichten des Backends, der Data Collectors, etc. mit entsprechenden Unit Tests abgedeckt.

End-to-End Testing

Neben den Unit Tests wurden zusätzlich mit Hilfe von Cypress Integration-Tests implementiert. Dabei wird einerseits die Bedienung des Frontends mit verschiedenen Bildschirmgrössen und andererseits direkt die API-Endpunkte des Backends getestet. So kann die Bedienbarkeit des Frontends und die Einhaltung des Berechtigungskonzeptes sichergestellt werden.

Benutzerrolle «Distributor»

Im Verlauf des Projektes kam beim Auftraggeber der Wunsch auf, die in der Aufgabenstellung definierten Benutzerrollen um die zusätzliche Rolle «Distributor» (eingeschränkter Administrator) zu erweitern. Obwohl diese Anpassung einen grossen Aufwand und viele Abhängigkeiten mit sich brachte, konnte sie erfolgreich implementiert werden.

[8] (Grafana, 2018)

Internationalisierung

Heutzutage zeichnet eine professionelle Webapplikation unter anderem aus, dass diese in verschiedenen Anzeigesprachen angeboten wird. So wurde das gesamte Frontend von Grund auf mit Englisch und Deutsch als verfügbare Anzeigesprachen ausgestattet. Die dadurch geschaffene Grundlage ermöglicht das Hinzufügen von weiteren Sprachen, ohne dass abgesehen von den Übersetzungen weitere aufwändige Codeänderungen vorgenommen werden müssen.

Feature Requests

Während der gesamten Projektphase hatte SUNCAR HK Zugriff auf den aktuellen Stand des Frontends und hat dies fleissig getestet. Dadurch kamen fortlaufend neue Wünsche auf, welche allesamt ebenfalls umgesetzt werden konnten – obwohl einige Feature Requests erst nach Ablauf der gesetzten Frist eingetroffen sind.

Die nachfolgenden Feature Requests (User Stories) wurden umgesetzt:

- Maschinen-Cards: Filterung (nach Maschine/Maschinentyp/Verbindungsstatus)
- Cockpit: farbiger Status (letzte Verbindung)
- Maschinen-Detailansicht: Maschine auswählen (für einfachere Navigation)
- Maschinen-Karte: farbiger Status (letzte Verbindung)
- Maschinen-Karte: Filterung (nach Maschine/Maschinentyp/Verbindungsstatus)
- Messdaten-Diagramm: Maschine auswählen (für einfachere Navigation/Vergleich)
- CSV-Export Messdaten
- und viele weitere, kleinere Wünsche des Auftraggebers

Produktive Daten & Produktives System

Der Auftraggeber zeigte sich bereits früh in der Projektphase begeistert vom neuen System und hat daher den bestehenden Data Collector nach einigen Wochen aufgerüstet, damit die produktiven Messdaten direkt zum neuen Data Collector weitergeleitet werden. Somit enthielt das PROD-System des neuen Frontends tatsächlich produktive Daten – live. So kam es dann auch, dass das Frontend noch während der Entwicklung von Mitarbeitenden von SUNCAR HK rege genutzt wurde. Der Auftraggeber ging gar noch einen Schritt weiter und stellte das neue Frontend noch während der Projektphase einigen seiner Endkunden vor.

4.1.4 Nicht erfüllte Resultate

Der Erläuterung der zwingenden und optionalen Resultate kann entnommen werden, dass sämtliche Anforderungen erfüllt wurden. Es gibt jedoch trotzdem einige User Stories, welche bewusst nicht umgesetzt wurden.

Tabelle 7: Die nicht umgesetzten User Stories

User Story	Begründung für die Nicht-Umsetzung
Details von Messtyp ansehen	Eine Detailansicht für einen Messtyp macht keinen Sinn, weil alle vorhandenen Daten bereits in der Übersichtsliste aller Messtypen angezeigt werden.

Fehlermeldungen ansehen Fehlermeldung per Mail erhalten	Die Übermittlung von Fehlercodes der Maschine an den Data Collector ist aktuell ein Experiment des Auftraggebers, welches möglichen Änderungen unterworfen ist. Da sich ein Ende des Experiments während der Projektphase nicht mehr abzeichnete, wurden diese beiden User Stories in Absprache mit dem Auftraggeber zurückgestellt und nicht implementiert.

4.1.5 Risikomanagement

Dank einem ausführlichen und realistischem Risikomanagement konnten riskante Aufgabenteile ermittelt, priorisiert und bereits im Architekturprototyp eruiert werden. So konnten früh allfällige technische Hürden erkannt und angegangen werden.

Trotz dem Teilausfall eines Entwicklungsrechners, diversen kleineren Unterbrüchen im Azure Web Portal sowie Problemen beim initialen Deployment auf einen Cloud Provider konnten alle Meilensteine zeitgerecht und mit dem geplanten Zeitaufwand erreicht werden. Detaillierte Informationen können dem Dokument «Risikomanagement» entnommen werden.

4.1.6 Fazit

4.1.6.1 Umfang

Wie in den vorangehenden Kapiteln aufgezeigt wurde, konnte in den vergangenen 17 Wochen ein sehr umfangreiches Gesamtergebnis erzielt werden. Bei der Durchsicht der zahlreichen funktionalen und nichtfunktionalen Anforderungen wurde bereits zu Beginn des Projektes klar, dass die Umsetzung aller User Stories eine echte Herausforderung darstellen wird. Umso erfreulicher ist daher die Feststellung, dass nicht nur alle zwingenden und optionalen Ziele erreicht wurden, sondern darüber hinaus eine beträchtliche Anzahl von zusätzlichen Features und Erweiterungen realisiert werden konnte.

Diese ausserordentliche Leistung ist primär auf eine gute Projektplanung, einen idealen Projektverlauf, eine harmonierende Teamarbeit sowie eine reibungslose Kommunikation mit dem Auftraggeber und dem Betreuer zurückzuführen.

4.1.6.2 Technologien

Während der Durchführung des Projektes bot sich die Möglichkeit, mit vielen verschiedenen, neuen und noch wenig bekannten Technologien zu experimentieren und diese als kleiner Bestandteil in das Gesamtsystem zu integrieren. Als Beispiel kann dabei der Einsatz von InfluxDB und Kapacitor hervorgehoben werden. Abgesehen von verschiedenen Technologien wurden in dieser Projektarbeit auch verschiedene moderne IT-Themen angeschnitten und miteinander verknüpft – von «IoT» über «Big Data» und «NoSQL» bis hin zu «Predictive Maintenance».

Mit der Umsetzung des Gesamtsystems ist eine auf SUNCAR HK abgestimmte Lösung gelungen, welche verschiedenste Technologien und Systeme miteinander kombiniert, um gemeinsam einen echten Mehrwert für das Unternehmen zu erzielen.

4.2 Ausblick

Mit dem Projektabschluss dieser Bachelorarbeit wird dem Auftraggeber ein voll funktionsfähiges Gesamtsystem übergeben. Während die Arbeit für das Projektteam damit beendet ist, beginnt die Weiterarbeit und Inbetriebnahme vom «Suncar Big Data Collector & Dashboard» für SUNCAR HK. So ist es an der Zeit, auf zukünftige Arbeiten und Erweiterungsmöglichkeiten vorauszublicken.

4.2.1 Go-live

Wie in der Aufgabenstellung festgehalten, wurde im Zuge der Projektarbeit ein Prototyp entwickelt. Der Auftraggeber zeigt sich jedoch begeistert vom Projektergebnis und möchte in naher Zukunft mit dem neuen Gesamtsystem live gehen und den Endkunden das Frontend zur Verfügung stellen.

4.2.2 Zusätzliche Anzeigesprachen

Wenn das Frontend für die Endkunden aufgeschaltet wird, könnten neben den bestehenden Anzeigesprachen Englisch und Deutsch weitere Sprachen erforderlich werden. Mit der Integration der Internationalisierung über alle Teilsysteme hinweg wurde dazu eine ideale Basis geschaffen. So kann eine neue Sprache ohne grossen Aufwand analog zu den bestehenden Sprachen implementiert werden.

4.2.3 Wartung und Weiterentwicklung

Nachdem die Entwicklung der ersten Version von Backend und Frontend nun abgeschlossen ist, muss diese in Zukunft gewartet und bei Bedarf weiterentwickelt werden. Dieser Aufgabe ist sich SUNCAR HK bewusst und hat deshalb noch während der Projektphase eine neue Stelle ausgeschrieben. Die Tatsache, dass ein Startup mit acht Mitarbeitenden dafür eine zusätzliche Stelle schafft, zeigt den bedeutenden Stellenwert des neuen Systems bei SUNCAR HK.

Bezüglich Weiterentwicklung des Backends und des Frontends werden die Ideen auch in Zukunft nicht ausgehen. So stehen beispielsweise im breiten Feld der Datenanalyse und der Events unzählige Ergänzungsmöglichkeiten offen.

4.2.4 Erweiterung der Schnittstelle

Durch die Ausarbeitung des Schnittstellenkonzeptes und der Implementation des Data Collectors inklusive Message Queue konnte eine Basis für eine Erweiterung der Kommunikation zwischen Maschine und Data Collector geschaffen werden. Die bidirektionale Message Queue wird aktuell nur für die Übertragung der Messdaten und die Zeitsynchronisierung genutzt. Diese Infrastruktur kann jedoch auch für weitere Einsatzzwecke genutzt werden. So ist es für SUNCAR HK durchaus vorstellbar, dass in Zukunft auch ein direkter Remote-Zugriff auf die Maschine realisiert werden könnte.

5 Abbildungsverzeichnis

6 Tabellenverzeichnis

7 Literaturverzeichnis

Cypress. (2018). Abgerufen am 11. Juni 2018 von Cypress.io: https://www.cypress.io/

Grafana. (2018). Abgerufen am 11. Juni 2018 von GrafanaLabs: https://grafana.com/

InfluxData.Net. (2018). Abgerufen am 11. Juni 2018 von GitHub: https://github.com/pootzko/InfluxData.Net

InfluxDB. (2018). Abgerufen am 11. Juni 2018 von InfluxData: https://www.influxdata.com/time-series-platform/influxdb/

ISO/TS 15143-3:2016. (Dezember 2016). Abgerufen am 11. Juni 2018 von ISO: https://www.iso.org/standard/67556.html

Kapacitor. (2018). Abgerufen am 11. Juni 2018 von InfluxData: https://www.influxdata.com/time-series-platform/kapacitor/

Logos. (2018). Abgerufen am 5. Juni 2018 von seeklogo: https://seeklogo.com/

MQTT Protocol. (2018). Abgerufen am 11. Juni 2018 von MQTT: http://mqtt.org/

SUNCAR TB216E. (5. Mai 2018). Abgerufen am 11. Juni 2018 von SUNCAR HK: http://suncar-hk.com/images/picts/medienmitteilung-tb260e.jpg

Unternehmen: Vision. (2018). Abgerufen am 21. Februar 2018 von SUNCAR HK AG: http://www.suncar-hk.com/de/unternehmen/vision.php

Was ist Scrum? (2018). Abgerufen am 21. Februar 2018 von Scrum-Master: http://scrum-master.de/Was_ist_Scrum

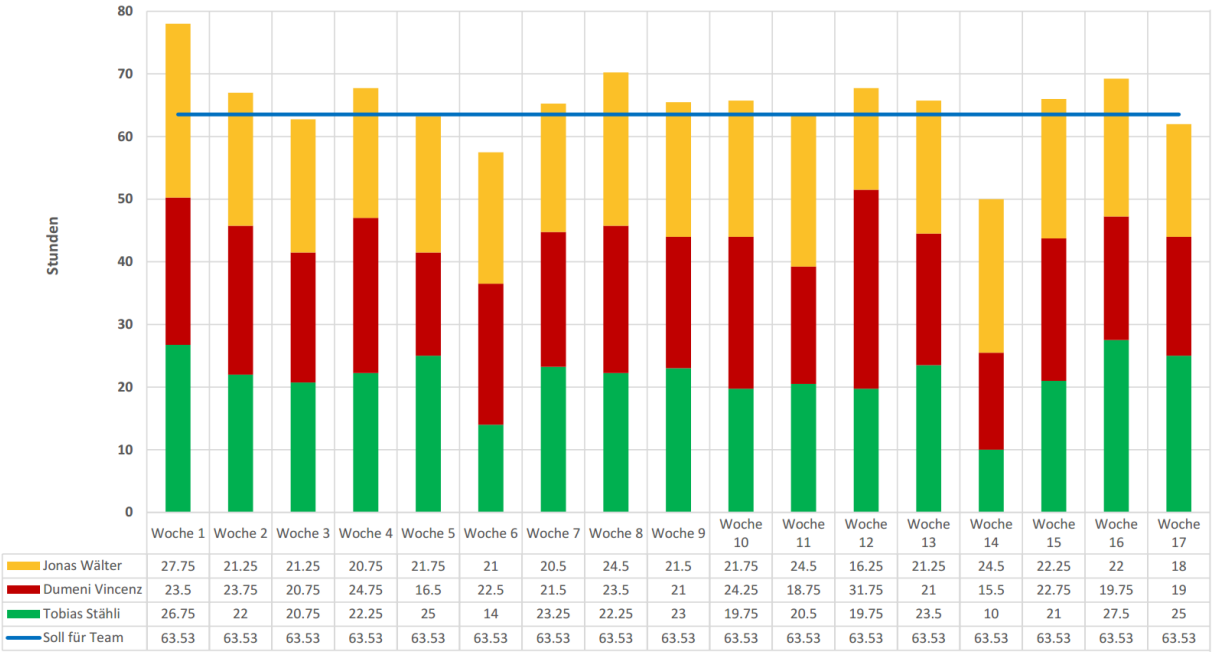
8 Anhang

8.1 Zeitauswertung

Die Verwaltung aller Arbeitspakete inklusive Zeiterfassung erfolgte fortlaufend auf Visual Studio Team Services, was nun verschiedene Auswertungen erlaubt. Für die detaillierte Zeiterfassung sei an dieser Stelle direkt auf VSTS oder aber auf das separate Dokument «Timesheet Report» verwiesen.

8.1.1 Aufwand pro Woche

Die Auswertung des Arbeitsaufwands pro Woche und Teammitglied zeigt sich im nachfolgenden Diagramm.



	Woche 1	Woche 2	Woche 3	Woche 4	Woche 5	Woche 6	Woche 7	Woche 8	Woche 9	Woche 10	Woche 11	Woche 12	Woche 13	Woche 14	Woche 15	Woche 16	Woche 17
Jonas Wälter	27.75	21.25	21.25	20.75	21.75	21	20.5	24.5	21.5	21.75	24.5	16.25	21.25	24.5	22.25	22	18
Dumeni Vincenz	23.5	23.75	20.75	24.75	16.5	22.5	21.5	23.5	21	24.25	18.75	31.75	21	15.5	22.75	19.75	19
Tobias Stähli	26.75	22	20.75	22.25	25	14	23.25	22.25	23	19.75	20.5	19.75	23.5	10	21	27.5	25
Soll für Team	63.53	63.53	63.53	63.53	63.53	63.53	63.53	63.53	63.53	63.53	63.53	63.53	63.53	63.53	63.53	63.53	63.53

Abbildung 13: Arbeitsaufwand pro Woche und Teammitglied

Im Diagramm ist ersichtlich, dass teilweise kleineren Zeitschwankungen aufgetreten sind aufgrund von im Voraus bekannten oder kurzfristigen Absenzen. So hat beispielsweise in Woche 14 eine internationale Konferenz zu einer geringeren Arbeitskapazität geführt. Indem die dadurch fehlenden Arbeitsstunden jeweils bereits vorgeholt oder schnellstmöglich nachgeholt wurden, konnten grössere Abweichungen verhindert werden.

8.1.2 Aufwand pro Teammitglied

In der nachfolgenden Tabelle wird die Gesamtarbeitszeit zudem pro Teammitglied aufgeschlüsselt. Dabei zeigt sich, dass eine gute Balance untereinander gehalten werden konnte und alle Teammitglieder etwa gleich viel Einsatz für das Projekt geleistet haben. Die vorgegebene Arbeitszeit von 360 Stunden wurde von jedem Teammitglied leicht überschritten.

Tabelle 8: Gesamtarbeitsaufwand pro Teammitglied

Teammitglied	SOLL	IST
Tobias Stähli	360 h	366.25 h
Dumeni Vincenz	360 h	370.50 h
Jonas Wälter	360 h	370.75 h
Total	1'080 h	1'107.50 h

8.1.3 Aufwand pro Tätigkeitsgruppe

Eine weitere Auswertung betrifft den Aufwand für die während der Projektplanung definierten Tätigkeitsgruppen. Dazu wurden alle Arbeitspakete in eine der aufgeführten Tätigkeitsgruppen eingeteilt, worauf sich der geschätzte Aufwand einer Tätigkeitsgruppe mit der tatsächlich geleisteten Arbeitszeit vergleichen lässt.

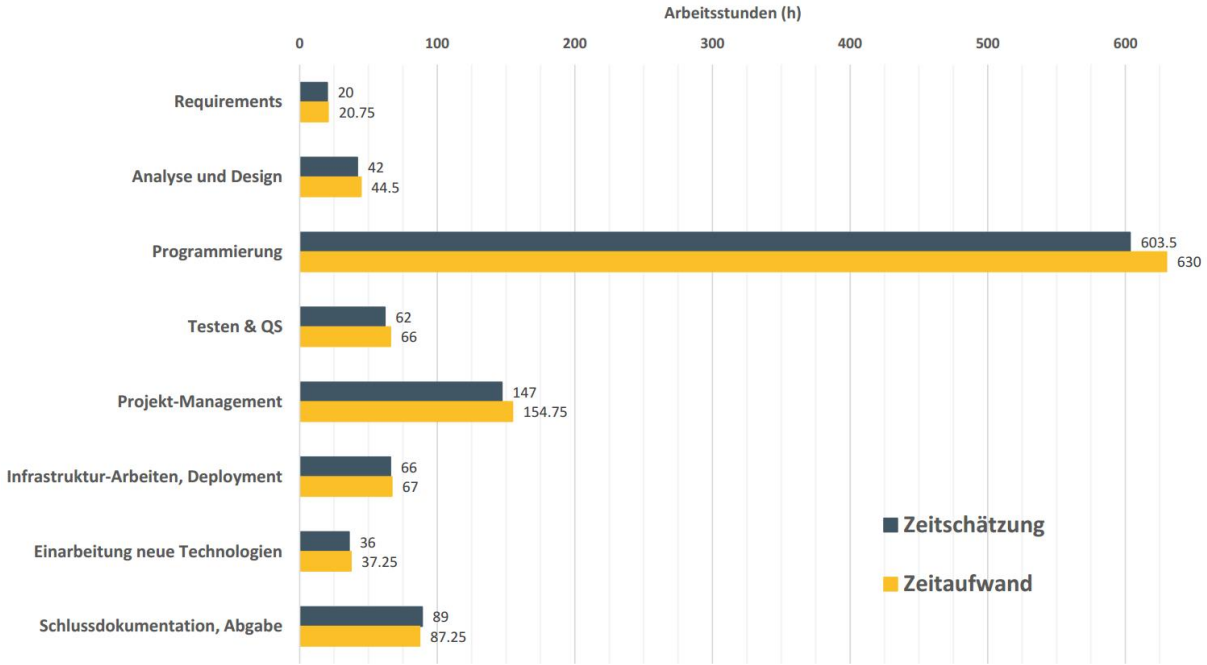


Abbildung 14: Zeitschätzung vs. Zeitaufwand pro Tätigkeitsgruppe

Das Diagramm zeigt, dass die Zeitschätzung und der tatsächliche Zeitaufwand keine grossen Abweichungen aufweisen. Wie erwartet nahm die Programmierung mit Abstand am meisten Arbeitszeit in Anspruch. An zweiter Stelle folgt das Projekt-Management, welches unter anderem die Kommunikation im Team sowie mit dem Betreuer und dem Auftraggeber beinhaltet.

8.2 Dokumente der Projektdurchführung/Softwareentwicklung

1. Projektplan
2. Risikomanagement
3. Anforderungsspezifikation
4. Personas
5. Schnittstellenkonzept
6. Datenbank-Evaluation
7. Cloud-Evaluation
8. Domainanalyse
9. Softwarearchitekturdokument
10. Umsetzung Schnittstelle nach ISO 15143-3
11. Systemtest
12. Continuous Integration/Continuous Delivery & Deployment (CI/CD)
13. Betriebsdokumentation Backup/Restore
14. Glossar

Projektplan

Projektmitglieder
Stähli Tobias
Vincenz Dumeni
Wälter Jonas

Änderungsgeschichte

Datum	Version	Änderung	Autor
21.02.2018	1.0	Erstellung des Dokuments, Projektübersicht/-organisation, Management Abläaufe	Jonas Wälter
23.02.2018	1.1	Infrastruktur, Qualitätsmassnahmen	Jonas Wälter
26.02.2018	1.2	Risikomanagement	Tobias Stähli
28.02.2018	1.3	Review / orthografische Korrekturen	Dumeni Vincenz
28.02.2018	1.4	Review / orthografische Korrekturen	Tobias Stähli

Inhaltsverzeichnis

1 Einführung

1.1 Beschreibung

Dieser Projektplan liefert grundlegende Informationen zum Projekt «Suncar Big Data Collector & Dashboard» betreffend Planung, Organisation und Aufbau der Arbeit. Dieses Dokument dient daher als Projektübersicht und Referenz für nachfolgende Projektdokumente.

1.2 Gültigkeitsbereich

Die Gültigkeit dieses Dokuments beschränkt sich auf die gesamte Projektdauer der Bachelorarbeit im Frühjahrssemester 2018. Allfällige spätere Anpassungen oder Ergänzungen im Dokument werden in der Änderungsgeschichte festgehalten.

1.3 Referenzen

Als Nachschlagewerk für unklare Begriffe sowie Abkürzungen sei an dieser Stelle auf das Glossar verwiesen, welches fortlaufend nachgeführt wird. In der nachfolgenden Tabelle sind alle weiteren referenzierten Dokumente und Links aufgelistet.

Thema	Referenz
Aufgabenstellung	OneDrive: \01_Aufgabenstellung\Aufgabenstellung.docx
Dokumentvorlage	OneDrive: \00_Administration\Template.docx
Glossar	OneDrive: \00_Administration\Glossar.docx
Projektplan	OneDrive: \02_Projektplan\Projektplan.docx
Risikomanagement	OneDrive: \02_Projektplan\Risikomanagement.xlsx
Logo Suncar HK	http://www.suncar-hk.com/

2 Projektübersicht

2.1 Auftraggeber

SUNCAR HK AG
Hinterwiden
CH-9245 Oberbüren
www.suncar-hk.com

Die SUNCAR HK AG ist ein Startup, welches ursprünglich aus einem Fokusprojekt an der ETH Zürich entstanden ist und mittlerweile acht Mitarbeiter beschäftigt. Das Unternehmen entwickelt, baut und vermarktet autonome, batteriebetriebene Elektrofahrzeuge für den Offroad-Bereich. Das Ziel ist es, verschiedene bis anhin mit Dieselmotor betriebene Baumaschinen (z.B. Bagger) auf Elektroantrieb umzubauen und so den Dieseltreibstoff vollständig durch Solarenergie aus Photovoltaik zu ersetzen. Die von SUNCAR HK AG gebauten Elektro-Baumaschinen werden ausschliesslich mit Solarenergie betrieben.[1]

2.2 Suncar Big Data Collector & Dashboard

Alle von der SUNCAR HK AG umgerüsteten Elektromaschinen verfügen als IoT-Device über einen mobilen Internetzugang, um diverse Sensordaten in bestimmten Intervallen an einen zentralen Server übermitteln zu können. Der serverseitig bestehende Data Collector und die relationale Datenbank werden den stetig wachsenden Anforderungen und Datenmengen nicht mehr gerecht. Der neu zu entwickelnde **Suncar Big Data Collector** soll daher auch bei einer steigenden Anzahl von Elektromaschinen für einen reibungslosen Empfang aller Sensordaten sorgen, ohne dabei die vorhandenen Legacy-Devices aussen vor zu lassen. Mit einer neuen **Datenbanklösung** (inklusive Speicherkonzept und Evaluation) werden die Kapazitätsprobleme der bestehenden Datenbank wegen der hohen Anzahl von Messdaten der Vergangenheit angehören.

Um die gesammelten Messdaten begutachten und auswerten zu können, wird die bestehende und ursprünglich nur für den internen Gebrauch vorgesehene Webapplikation durch ein neues, modernes **Web GUI (Frontend)** ersetzt. Das neue Frontend ermöglicht auf Basis einer Backend-API die Verwaltung von Maschinen, Mess- und Maschinentypen sowie Kunden (Unternehmen/ Benutzer). So können die Kunden der SUNCAR HK AG in Zukunft die aktuellen und zurückliegenden Messdaten aller ihrer Maschinen bequem über die neue Weboberfläche beobachten und analysieren. Ein Hauptaugenmerk wird dabei auf die **Mandantenfähigkeit** gelegt, damit das Frontend von unterschiedlichen Kunden für ihre eigenen Maschinen genutzt werden kann, ohne jedoch Einblick in andere Unternehmen und deren Maschinen zu erhalten.

[1] (Unternehmen: Vision, 2018)

Des Weiteren könnte das Gesamtsystem um zusätzliche, optionale Komponenten ausgebaut werden. So kann eine separate API für Drittanbieter zur Verfügung gestellt werden, worüber die Maschinen- und Messdaten direkt abgefragt und beispielsweise auf der eigenen Website eines Kunden eingebunden werden können. Für die Auswertung der Messdaten kann zudem ein Statistik-Modul integriert oder implementiert werden, welches die Datenanalyse vereinfachen soll. Daraus ergeben sich noch weitere Möglichkeiten, wie beispielsweise ein Event Alerting bei der Überschreitung eines Schwellenwertes (Predictive Maintenance).

Abbildung 1: Features von «Suncar Big Data & Dashboard»

2.3 Zweck und Ziel

Im aktuellen System übersteigt die Anzahl der Messdaten die Kapazität der vorhandenen Datenbank-Lösung und der Data Collector ist aufgrund seiner Funktionsweise nur auf eine begrenzte Anzahl Devices ausgelegt. Des Weiteren wurde bereits ein Frontend entwickelt, welches jedoch den Anforderungen eines Kundenportals noch nicht gerecht wird.

Das Ziel dieses Projektes ist eine ganzheitliche Neuentwicklung der bestehenden Teilsysteme in Form eines Prototyps, damit das Gesamtsystem für das zu erwartende Wachstum in der Zukunft gerüstet ist. Zudem soll das neue Frontend zusätzliche Möglichkeiten für die Kunden bieten und dadurch sowohl den Kunden als auch dem Auftraggeber einen Mehrwert bringen.

2.4 Lieferumfang

Als Resultate dieses Projektes werden am Projektende die folgenden Komponenten abgeliefert:

Produkt

- Anforderungsspezifikation
- Personas
- Domainanalyse
- Softwarearchitekturdokumentation
- Datenbank-/Schnittstellenkonzept
- Implementation der Teilsysteme
- Testspezifikation/-Protokolle

Projektmanagement

- Projektplan
  (inklusive Risikomanagement)
- Projektmonitoring
  - Zeiterfassung/-auswertung
  - Soll/Ist-Vergleiche
  - persönliche Berichte
- Technische Dokumentation
- Erklärung zur eigenständigen Durchführung der Arbeit

3 Projektorganisation

3.1 Organisationsstruktur / Verantwortlichkeiten

Für die Durchführung des Projektes wird auf eine flache Organisationsstruktur zurückgegriffen, wobei alle wesentlichen Entscheidungen im Projektteam getroffen werden. Das Projektteam setzt sich aus den Studierenden Tobias Stähli, Dumeni Vincenz und Jonas Wälter zusammen. Alle Teammitglieder sind einander gleichgestellt und beteiligen sich gleichermassen in allen Bereichen, wobei die Verantwortung für die einzelnen Bereiche jeweils an ein Projektmitglied übergeben wird. Die Betreuung des Projektes übernimmt Manuel Bauer.

Tobias Stähli

Verantwortlichkeiten:

- Data Collector
- Backend
- Testing
- Code Review/Qualität

Dumeni Vincenz

Verantwortlichkeiten:

- Infrastruktur / Deployment
- Datenbank
- Kommunikation mit Kunde

Jonas Wälter

Verantwortlichkeiten:

- Frontend
- Projektadministration (VSTS)
- Dokumentation Review/Qualität

3.2 Externe Schnittstellen

Während der Durchführung des Projektes gelten die nachfolgenden Ansprechpersonen:

Name	Zuständigkeit
Manuel Bauer Betreuer/Dozent	• Beratung während der Projektphase • Bewertung der Projektarbeit
Daniel Vincenz SUNCAR HK AG	• CTO von SUNCAR HK AG • Auftraggeber
Jürg Horisberger SUNCAR HK AG	• Entwicklung des aktuellen Systems • Unterstützung für fachliche Fragen

4 Management Abläufe

4.1 Kostenvoranschlag

Die Projektphase dauert vom 19. Februar bis am 15. Juni 2018, wobei jedes Projektmitglied insgesamt 360 Arbeitsstunden leistet. Durch die Verteilung auf diese 17 Wochen ergibt sich ein wöchentlicher Arbeitsaufwand von ungefähr 21.18 Stunden pro Person.

Projektstart	Montag, 19. Februar 2018
Projektende	Freitag, 15. Juni 2018
Projektdauer	17 Wochen
Anzahl Projektmitglieder	3 Personen
Arbeitsstunden insgesamt	1'080 h
Arbeitsstunden pro Woche und Person	ca. 21.18 h

Die erforderlichen Resultate sind in dieser Projektdauer vollständig zu erbringen. Sofern der Prototyp mit den zwingenden Anforderungen schneller als geplant fertig gestellt werden kann, können die optionalen Erweiterungen gemäss der Aufgabenstellung in Angriff genommen werden. Falls die Umsetzung des Projektes die Projektdauer zu übertreffen droht, wird der Projektumfang in Absprache mit dem Auftraggeber und dem Betreuer entsprechend angepasst.

4.2 Vorgehensmodell

Für die Durchführung des Projektes wird ein **agiles und iteratives Vorgehensmodell** angewendet, welches sich in vielen Aspekten an Scrum[2] orientiert. Scrum eignet sich sehr gut für ein Softwareprojekt mit einer hohen Komplexität, bei dem flexibel definierte und möglicherweise wachsende Anforderungen im engen Kontakt mit dem Auftraggeber umgesetzt werden.

Aufgrund bestehender Rahmenbedingungen, der Grösse des Entwicklerteams und fehlender Scrum-Komponenten (Scrum Master, klassischer Product Owner, etc.) werden die Konzepte von Scrum – von den Artefakten (Backlogs) bis hin zu den Aktivitäten (Meetings) – an die Rahmenbedingungen dieser Projektarbeit angepasst und in entsprechender Form angewendet.

[2] (Was ist Scrum?, 2018)

4.3 Arbeitspakete / Zeiterfassung

Wie von Scrum übernommen, werden die unterschiedlichen Software-Anforderungen in Form von **User Stories** (anstelle von Use Cases) im **Product Backlog** als priorisierte und dynamische Anforderungsliste verwaltet. Alle zur Erfüllung einer User Story erforderlichen Arbeitspakete werden in Form von **Tasks** erfasst und im jeweiligen **Sprint Backlog** verwaltet.

Die Verwaltung dieser Artefakte und Arbeitspakete sowie die Zeiterfassung erfolgen direkt im Work-Bereich von **Visual Studio Team Services (VSTS)**. Für die Zeiterfassung wird zusätzlich die VSTS-Extension «Timetracker»[3] eingesetzt. Diese Weblösung erlaubt eine agile Arbeitsplanung mit Hilfe von Iterationen und Backlogs, um die Planung immer aktuell zu halten und auf unvorhergesehene Ereignisse reagieren zu können.

Visual Studio Team Services – Bereich «Work»	
URL	https://suncar-hk.visualstudio.com/remote/_backlogs/board/Stories
Login	Mittels eingerichtetem Microsoft Account

Die Tasks werden zeitlich geschätzt und einer der nachfolgenden Tätigkeitsgruppen zugewiesen. Ausserdem erfolgt die Zeiterfassung auf Task-Ebene, um eine fortlaufende Zeitauswertung zu ermöglichen. Die mit einem Stern gekennzeichneten VSTS Activity Codes entsprechen nicht den Standard-Activities in VSTS und wurden manuell erstellt.

	Tätigkeitsgruppe	VSTS Activity Code
1)	**Requirements**	Requirements
2)	**Analyse und Design**	Design
3)	**Programmierung**	Development
4)	**Testen & QS**	Testing
5)	**Projekt-Management**	ProjectManagement*
6)	**Infrastruktur-Arbeiten, Deployment**	Deployment
7)	**Einarbeitung neue Technologien**	KnowHow*
8)	**Schlusspräsentation, Aufarbeitung Schlussdokumentation**	Documentation

[3] (Timetracker, 2018)

4.4 Zeitliche Planung

4.4.1 Iterationen

Die Umsetzung des Projektes erfolgt inkrementell, in dem die zur Verfügung stehende Projektzeit in zweiwöchige Iterationen unterteilt werden. Des Weiteren wurden verschiedene Meilensteine definiert, welche die Projektdauer hinsichtlich der Softwareentwicklung grob abstecken.

Iteration 1	W 1	19.02. - 25.02.18	**Beschreibung:** Projektplanung, Anforderungsanalyse, Infrastruktur, Entwicklungsumgebung
	W 2	26.02. - 04.03.18	
Iteration 2	W 3	05.03. - 11.03.18	**Beschreibung:** Prototyp (Durchstich)
	W 4	12.03. - 18.03.18	
18.03.18	M 1	Durchstich	Proof of Concept (Änderungen möglich)
Iteration 3	W 5	19.03. - 25.03.18	**Beschreibung:** Implementation gemäss User Stories
	W 6	26.03. - 01.04.18	
Iteration 4	W 7	02.04. - 08.04.18	**Beschreibung:** Implementation gemäss User Stories
	W 8	09.04. - 15.04.18	
Iteration 5	W 9	16.04. - 22.04.18	**Beschreibung:** Implementation gemäss User Stories
	W 10	23.04. - 29.04.18	
Iteration 6	W 11	30.04. - 06.05.18	**Beschreibung:** Implementation gemäss User Stories
	W 12	07.05. - 13.05.18	
17.05.18	P 1	Zwischenpräsentation	
Iteration 7	W 13	14.05. - 20.05.18	**Beschreibung:** Implementation gemäss User Stories
	W 14	21.05. - 27.05.18	
27.05.18	M 2	Feature Freeze	Keine neuen Features
Iteration 8	W 15	28.05. - 03.06.18	**Beschreibung:** Refactoring, Bugfixing, Dokumentation
	W 16	04.06. - 10.06.18	
10.06.18	M 3	Code Freeze	Keine weiteren Änderungen am Sourcecode
Iteration 9	W 17	11.06. - 15.06.18	**Beschreibung:** Abschlussarbeiten, Projektabgabe
15.06.18	M 4	Projektabschluss	Abgabe der Projektarbeit

4.4.1 Aktivitäten

Scrum verfügt über verschiedene Aktivitäten wie «Daily Scrum», «Sprint Planning», «Sprint Review» und «Sprint Retrospective», welche auf die vorliegenden Rahmenbedingungen adaptiert und in zusammengefasster Form wie folgt durchgeführt werden.

Aktivität	Beschreibung
Daily Meeting	Zu Beginn von jedem Arbeitstag findet ein 15-minütiges Teammeeting statt, um sich gegenseitig über die aktuellen Aufgaben und Probleme auszutauschen.
Iteration Planning und Retrospective	In einem Projektteam-Meeting am Ende einer Iteration wird die Arbeit im Team rekapituliert und gleichzeitig auch die nächste Iteration geplant. Im Zuge der Iterationsplanung werden die in der nächsten Iteration zu erledigende User Stories ausgewählt (basierend auf der Priorisierung des Auftraggebers), detaillierter beschrieben (mit Akzeptanzkriterien) und in entsprechende Tasks (inkl. Zeitschätzung) unterteilt.
Meeting mit Betreuer und/ oder Auftraggeber	Für Meetings mit dem Betreuer und dem Auftraggeber steht ein wöchentlicher Termin zur Verfügung, die Einladungen wurden allen Beteiligten versendet. In diesen Meetings werden beispielsweise der Fortschritt und das Endresultat einer Iteration besprochen oder Kundendemos durchgeführt. Falls aktuell keine zu behandelnden Traktanden vorliegen, findet in der entsprechenden Woche kein Meeting statt. Es wird fortlaufend im Voraus entschieden, ob ein Meeting notwendig ist, ob die Teilnahme des Betreuers und/oder des Auftraggebers erforderlich ist und ob das Meeting physisch oder via Skype-Telefonkonferenz durchgeführt werden soll.

Wenn nebst dem wöchentlich eingeplanten Meeting zusätzliche Absprachen mit dem Auftraggeber und dem Betreuer nötig sind oder Fragen auftauchen, so kann dies über die dazu eingerichtete Skype-Gruppe erfolgen.

4.4.1.1 Protokollierung

Bei allen Meetings mit dem Betreuer und/oder dem Auftraggeber wird ein Sitzungsprotokoll geführt, welches die getroffenen Beschlüsse und die offenen Punkte festhält. Beim Daily Meeting und der Iterationsplanung wird stattdessen auf ein Sitzungsprotokoll verzichtet. Allfällige getroffene Entscheide werden stattdessen im entsprechenden VSTS-Work Item dokumentiert. Bei der Iterationsplanung dient der Backlog sowie die Notizen in den Work Items als Outcome des Meetings.

Für das Sitzungsprotokoll wurde eine entsprechende Dokumentvorlage erstellt, welche für alle Protokolle zu verwenden ist. Das Protokoll soll dabei helfen, Missverständnisse zu vermeiden und Unklarheiten aus dem Weg zu räumen. Wie der Aufgabenstellung zu entnehmen ist, muss das Kurzprotokoll bis spätestens zwei Tage nach dem Meeting an alle Sitzungsteilnehmer gesendet werden.

4.4.2 Arbeitstage

Die Arbeitszeit der Projektteammitglieder verteilt sich wie folgt auf eine Arbeitswoche:

	Montag	Dienstag	Mittwoch	Donnerstag	Freitag
Tobias Stähli	Keine Zeit (Teilzeitarbeit)	Keine Zeit (Teilzeitarbeit)	Zeit für Projektarbeit	Zeit für Projektarbeit	Zeit für Projektarbeit
Dumeni Vincen	Keine Zeit (Teilzeitarbeit)	Keine Zeit (Teilzeitarbeit)	Teilzeitarbeit/ Projektarbeit	Zeit für Projektarbeit	Zeit für Projektarbeit
Jonas Wälter	Keine Zeit (Teilzeitarbeit)	Keine Zeit (Teilzeitarbeit)	Zeit für Projektarbeit	Zeit für Projektarbeit	Zeit für Projektarbeit

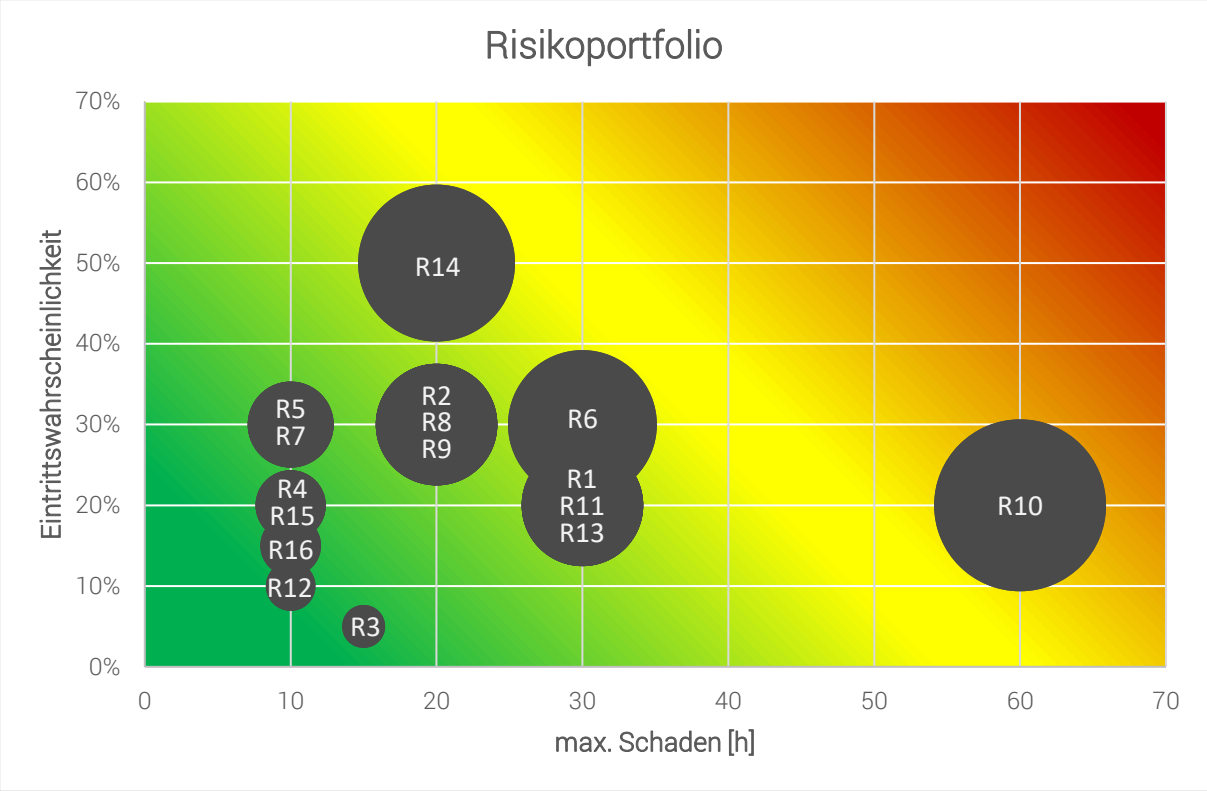
Bei Tobias Stähli und Jonas Wälter stehen jeweils drei Arbeitstage für die Projektarbeit zur Verfügung, was einem täglichen Arbeitspensum von 7.06 Stunden entspricht. Dumeni Vincenz wird infolge eines höheren Teilzeitarbeitspensums mittwochs teilweise keine Zeit für die Projektarbeit aufbringen können. Stattdessen wird er sich in den entsprechenden Wochen jeweils am Wochenende einen Tag reservieren und/oder ein höheres Tagespensum einplanen.

4.5 Releases

Beginnend ab der zweiten Iteration wird jeweils am Ende einer Iteration ein entsprechender Release erstellt. Dabei wird die Nummerierung aus einer Haupt- und zwei Nebenversionsnummern zusammengesetzt. Während sich die hintere Nebenversionsnummer bei jedem Release erhöht, wird die vordere Nebenversionsnummer nur bei weitreichenden Änderungen und die Hauptversionsnummer nur bei einer kompletten Überarbeitung erhöht. So beginnt die Nummerierung der Releases mit der Version 1.0.1.

5 Risikomanagement

Eine Risikoidentifikation und Risikoanalyse ist in Form einer detaillierten Auflistung der Risiken mit gewichtetem Schaden und Informationen zur Vorbeugung und Verhalten beim Eintreten im Dokument «Risikomanagement» zu finden.



R1	Anforderungen	R9	Programmiersprachen/Tooling
R2	Anforderungen von Auftraggeber unklar	R10	Skalieren auf mehr Daten
R3	SW-Projektmanagement-Tool	R11	Technische Machbarkeit
R4	IDE	R12	Ausfall Entwickler-Notebook
R5	Software-Qualität	R13	Cloud Provider Features
R6	Externe Komponenten	R14	Cloud Provider Management UI
R7	Dokumentation der eingesetzten Technologien	R15	Cloud Provider Hosting Ausfall
R8	Kommunikation	R16	Ausfall Projektmanagement-Tool

5.1 Umgang mit Risiken

Wenn erst reagiert wird, wenn das Risiko eintritt, ist es in der Regel zu spät und der Schaden ist unter Umständen grösser als er sein müsste. Da in diesem Fall nebst dem eingetretenen Risiko auch mit dem zusätzlichen Zeitdruck mangels fehlender Reserven umgegangen werden muss, wirkt sich fehlendes Risikomanagement doppelt auf das Projekt aus.

Wir pflegen in diesem Projekt einen proaktiven Umgang mit allen im Projekt erkannten Risiken und bringen mögliche Blocker und Probleme so früh als möglich zur Sprache.

5.2 Risikoplanung / Risikobewertung

Bei der Iterationsplanung wird jeweils mit entsprechenden Reserven gearbeitet, um Risiken frühzeitig erkennen und entsprechend behandeln zu können.

Nach jeder Iteration wird der aktuelle Stand der Risiken überprüft und falls nötig korrigiert. Nicht eingetretene Risiken werden entsprechend markiert. Im Falle eines eingetretenen Risikos wird die definierte Massnahme umgesetzt und anschliessend wird das Risiko neu beurteilt.

6 Infrastruktur

Für die Durchführung des Projektes und die Implementierung der einzelnen Teilsysteme wird auf die im Folgenden beschriebene Infrastruktur zurückgegriffen.

6.1 Entwicklungsumgebung

Für die Arbeit am Projekt verfügen alle Teammitglieder über ein eigenes Notebook mit den nachfolgenden Entwicklungsumgebungen, welche für alle projektbezogenen Arbeiten verwendet werden. Die Entwicklung wird durch die Tatsache erleichtert, dass alle Teammitglieder über die gleichen Versionen betreffend Betriebssystem (Windows 10) und Toolchain verfügen.

Entwicklungsumgebung	Einsatzzweck
Microsoft Visual Studio 2017	Data Collector und Backend
Microsoft Visual Code	Frontend

6.2 Quellcode-Verwaltung

Die Verwaltung des Quellcodes inklusive Versionsverwaltung erfolgt in einem Code Repository direkt in VSTS mit dem Versionsverwaltungssystem Git.

Visual Studio Team Services – Bereich «Code»	
URL	https://suncar-hk.visualstudio.com/_git/remote
Login	Mittels eingerichtetem Microsoft Account

6.3 Deployment

Auch die Continuous Integration und das Deployment werden in Form von Build-/Release-Definitionen direkt in VSTS abgebildet. Die Entwicklungsumgebung der Cloud-Lösung (Hosting der Infrastruktur, Hosting-Methode) sind abhängig von der Konzeptionierung der Systemarchitektur und der Evaluation einer Datenbankarchitektur sowie eines Cloud Providers, welche ebenfalls Bestandteil des Projektes sind. Wenn möglich soll dabei auf Angebote von «Platform as a Service» (PaaS) oder andernfalls «Container as a Service» (CaaS) gesetzt werden.

6.4 Datenablage / Kommunikation

Für die Ablage von Projektdokumenten wird Microsoft OneDrive in Form eines teamintern freigegebenen Projektverzeichnisses verwendet. Die Kommunikation im Projektteam und gegenüber dem Betreuer sowie dem Auftraggeber erfolgt mittels Skype von Microsoft.

7 Qualitätsmassnahmen

Damit sich das Endsystem dieses Projektes mit einer hohen Qualität auszeichnen kann, werden die nachfolgenden Massnahmen getroffen.

Massnahme	Zeitraum	Ziel
Teamkommunikation	Fortlaufend	Austausch über Projektstand, mögliche Fehler frühzeitig erkennen
Meetings mit Betreuer und Auftraggeber	Wöchentlich (nach Bedarf)	Fortschrittbesprechung inkl. Demos, Klärung von offenen Fragen
Code Reviews	Pull Requests	Erhöhte Code-Qualität durch Testabdeckung und Einhaltung der Code Style Guidelines.
Automatisierte Tests	Fortlaufend	Fehlereliminierung durch erhöhte Code-Qualität

7.1 Definition of Done

Für jede Iteration werden die zu implementierenden User Stories unter den Teammitgliedern aufgeteilt mit dem Ziel, alle zugeteilten User Stories bis zum Iterationsende vollständig zu realisieren. Noch nicht vollständig implementierte User Stories müssen in die nächste Iteration weitergezogen werden. Bei der Entscheidung über die Vollständigkeit einer User Story wird die nachfolgende **Definition of Done** zur Hilfe gezogen, welche dem Scrum-Prinzip[4] entnommen wurde.

Definition of Done
• Alle Akzeptanzkriterien sind erfüllt
• Der Code ist fertiggestellt und im VSTS-Repository eingecheckt
• Entsprechende Unit Tests wurden erstellt und sind erfolgreich
• Die Code Guidelines wurden eingehalten
• Die Dokumentation im Sourcecode ist ausreichend
• Ein Pull Request wurde erstellt und von einem anderen Teammitglied akzeptiert

7.2 Dokumentation

Alle während dem Projekt entstandenen Dokumente werden zentral auf **Microsoft OneDrive** abgelegt. Durch die Unterstützung von parallelem Arbeiten an einem Dokument mittels Microsoft Office können Versionskonflikte verhindert werden. Die integrierte Versionsverwaltung ermöglicht das Vergleichen und Verfolgen von Änderungen. Vor der Abgabe von Dokumenten werden diese jeweils einem Review unterzogen.

[4] (Was ist die "Definition of Done" (DoD)?, 2018)

7.3 Projektmanagement

Wie den vorherigen Ausführungen entnommen werden kann, werden die Arbeitspakete, der Sourcecode, die Builds/Releases und die Zeiterfassung mit Hilfe von **Visual Studio Team Services (VSTS)** verwaltet. So kann in allen Belangen auf ein etabliertes Projektmanagementsystem zurückgegriffen werden. Alle Teammitglieder sind angehalten, die aufgewendete Arbeitszeit und den Status aller Tasks und User Stories fortlaufend zu erfassen/anzupassen, damit der aktuelle Projektfortschritt jederzeit sichtbar bleibt.

Für die Verwaltung des Status eines **Tasks** wird der in VSTS standardmässig definierte **Workflow** für die Prozessvariante «Agile» verwendet, welcher in der nachfolgenden Abbildung aufgezeigt wird. Neu erstellte Tasks erhalten den Status «New», während der Implementation den Status «Active» und nach der Implementation und erfolgreichen Unit Tests den Status «Resolved». Nun wird die Implementation im Zuge des Pull Requests durch ein anderes Teammitglied einem Review unterzogen und im Erfolgsfall der Status «Closed» zugewiesen.

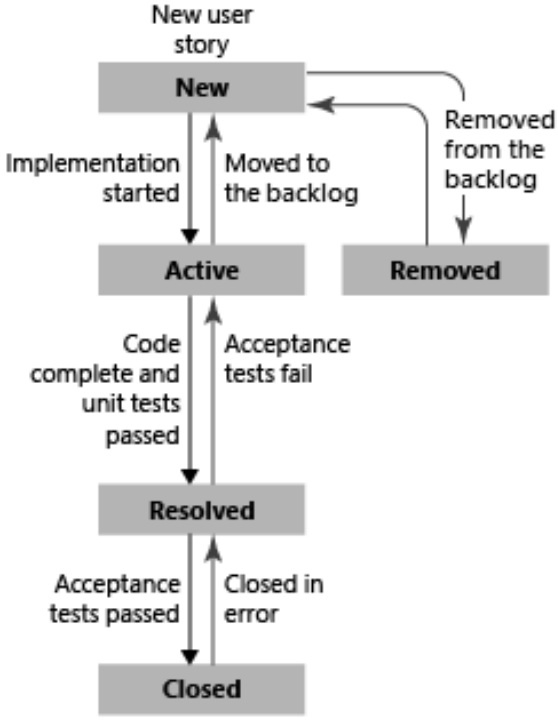


Abbildung 2: VSTS-Workflow für agile Prozesse[5]

Bei **User Stories** verläuft der Prozess ähnlich, aber abhängig von den Tasks. Sobald eine User Story der aktuellen Iteration zugewiesen wird, wechselt der Status von «New» zu «Active». Erst nachdem alle der entsprechenden User Story zugewiesenen Tasks abgeschlossen sind (Status «Closed»), gilt eine User Story ebenfalls als beendet (Status «Closed»).

[5] (Workflow states and state categories, 2017)

7.4 Entwicklung

Der Sourcecode aller Teilsysteme wird in einem privaten **VSTS-Repository** verwaltet, auf welches nur die Teammitglieder, der Auftraggeber sowie der Betreuer Zugriff erhalten.

7.4.1 Vorgehen

Jedes Teammitglied verfügt über eine lokale Kopie des Remote Repositorys auf VSTS und kann so auch offline daran weiterarbeiten. Für jede User Story wird jeweils ein **neuer Branch** erstellt, worin die entsprechenden Code-Änderungen mit aussagekräftigen Commit-Messages (inklusive Verlinkung der zugehörigen Task-ID) festgehalten werden.

Nach der vollständigen Implementation einer User Story kann diese durch einen **Pull Request** vom betreffenden Branch in den Master-Branch vorgeschlagen werden. Ein anderes Teammitglied muss nun einen Code Review durchführen und den Pull Request entweder akzeptieren oder mit einem «Reject» an den Autor zurückweisen. Mit Hilfe solcher Code Reviews nach dem Vier-Augen-Prinzip wird stets die Qualität des Sourcecodes sichergestellt.

7.4.2 Code Style Guidelines

Um einen sauberen und konsistenten Sourcecode sicherstellen zu können, sollen entsprechende Code Style Guidelines berücksichtigt und eingehalten werden. Für die Programmierung des Data Collectors und des Backends mit C# gelten die **Codekonventionen für C# von Microsoft**[6], wobei dazu eine ReSharper Style-Konfiguration erstellt und von allen Teammitgliedern eingesetzt wird. Mit ReSharper können Verletzungen des Code Styles über einzelne Files, Projekte oder ganze Solutions per Knopfdruck eliminiert werden. Des Weiteren wird eine separate Build Definition in VSTS konfiguriert, welche automatisiert dafür sorgt, dass ein Pull Request erst nach einem erfolgreichen ReSharper-Check in den Master-Branch gemergt werden kann.

Bei der Programmierung des Frontends (TypeScript/JavaScript) wird der **Google JavaScript Code Style Guide**[7] in Kombination mit der Regelwerk **tslint-microsoft-contrib**[8] von Microsoft berücksichtigt. Die Einhaltung dieser Richtlinien wird in der Entwicklungsumgebung und im Zuge des Deployment-Vorgangs automatisch mit Hilfe von **TSLint**[9] überprüft. Dazu wird TSLint direkt in Visual Studio Code integriert und kann so allfällige Formatierungsfehler fortlaufend verhindern.

[6] (Codekonventionen für C#, 2015)

[7] (Google JavaScript Style Guide, 2018)

[8] (tslint-microsoft-contrib, 2018)

[9] (TSLint, 2018)

7.4.3 Unit Testing

Die Funktionalität aller Komponenten wird systematisch und fortlaufend durch sinnvolle Unit Tests überprüft. Banale Unit Tests ohne wirklichen Mehrwert, wie beispielsweise für das Testen von Konstruktoren, werden nicht erstellt. Daher wird an dieser Stelle auch auf eine Zielvorgabe der Code Coverage oder Ähnlichem verzichtet. Ein Pull Request darf nur erstellt werden, wenn alle Unit Tests erfolgreich durchgelaufen sind. Dies wird ebenfalls während dem automatisierten Deployment-Prozess überprüft.

8 Literaturverzeichnis

Codekonventionen für C#. (20. Juli 2015). Abgerufen am 22. Februar 2018 von Microsoft Docs: https://msdn.microsoft.com/de-de/library/ff926074.aspx

Google JavaScript Style Guide. (2018). Abgerufen am 22. Februar 2018 von Google on GitHub: https://google.github.io/styleguide/jsguide.html

Timetracker. (2018). Abgerufen am 21. Februar 2018 von Visual Studio Marketplace: https://marketplace.visualstudio.com/items?itemName=Berichthaus.TfsTimetracker

TSLint. (2018). Abgerufen am 22. Februar 2018 von TSLint: https://palantir.github.io/tslint/

tslint-microsoft-contrib. (2018). Abgerufen am 22. Februar 2018 von Microsoft on GitHub: https://github.com/Microsoft/tslint-microsoft-contrib

Unternehmen: Vision. (2018). Abgerufen am 21. Februar 2018 von SUNCAR HK AG: http://www.suncar-hk.com/de/unternehmen/vision.php

Was ist die "Definition of Done" (DoD)? (2018). Abgerufen am 22. Februar 2018 von SCRUMevents: https://www.scrum-events.de/was-ist-die-definition-of-done-dod.html

Was ist Scrum? (2018). Abgerufen am 21. Februar 2018 von Scrum-Master: http://scrum-master.de/Was_ist_Scrum

Workflow states and state categories. (29. September 2017). Abgerufen am 22. Februar 2018 von Microsoft Docs: https://docs.microsoft.com/en-us/vsts/work/customize/workflow-and-state-categories

Risikomanagement Suncar Big Data Collector & Dashboard

Gewichteter Schaden:	80.25

Nr	Titel	Beschreibung	max. Schaden [h]	Eintrittswahr-scheinlichkeit	Gewichteter Schaden	Vorbeugung	Verhalten beim Eintreten
R1	**Anforderungen**	Neue oder ständig wechselnde Anforderungen	30	20%	**6**	Iteratives Projektmodell und regelmässige Standbesprechungen mit Auftraggeber	Anforderungen aufnehmen, schätzen, priorisieren und einplanen
R2	**Anforderungen von Auftraggeber unklar**	Die gegebenen Anforderungen an die Applikation wurden nicht korrekt verstanden oder bieten zu viel Spielraum.	20	30%	**6**	Möglichst frühe Besprechung der Anforderungsspezifikation und des Projektplanes mit Betreuer und Auftraggeber, um allfällige Missverständnisse im Entstehen zu entdecken	Besprechen der Unklarheiten bei wöchentlichen Meetings oder direkt im Skype-Gruppenchat
R3	**SW-Projektmanagement-Tool**	Visual Studio Team Services entspricht nicht den projektspezifischen Bedürfnissen	15	5%	**0.75**	Abklären/Konfigurieren der detailierten Funktionalitäten von VSTS so früh wie möglich	Wechsel auf Jiira als Bug-Tracking-System und GitHub als Sourcecode-Verwaltung
R4	**IDE**	Fehlende Kenntnisse oder Kompatibilität der eingesetzten Entwicklungswerkzeuge	10	20%	**2**	Möglichst früh alle Entwicklungswerkzeuge im Team einrichten und testen	Evaluierung anderer Entwicklungswerkzeuge
R5	**Software-Qualität**	Nichteinhaltung der Coding Guidelines, Qualitätssicherung und Konfigurationsmanagement	10	30%	**3**	Alle halten sich an die Guidelines, eingeplante Code Reviews; Einsatz von Linting-Tools und Continous Integration; Definition of Done enthält Anforderungen an Codequalität;	Zusätzliche Reviews und Sensibilisierung der Entwickler
R6	**Externe Komponenten**	Fehlfunktion externer Komponenten/Bibliotheken	30	30%	**9**	Bei der Evaluation externer Komponenten/Bibliotheken wird auf die Verbreitung und den Support geachtet	Wechsel mit Evaluation zu anderen Bibliotheken
R7	**Dokumentation der eingesetzten Technologien**	Fehlende Dokumentationen der Technologien	10	30%	**3**	Bei der Evaluation der Technologien wird auf die Dokumentation geachtet	Evaluierung anderer Technologien und wechseln
R8	**Kommunikation**	Unzureichender Informationsfluss im Projektteam	20	30%	**6**	Daily Meetings und klare Zuweisung der Verantwortungen, zeitnahe Dokumentation der Arbeitsstandes im zugehörigen VSTS Work-Item, um doppelte Arbeit zu verhindern	Zusätzliche Meetings ansetzen, Teambuilding-Event durchführen
R9	**Programmiersprachen/ Tooling**	Fehlende Kenntnisse der eingesetzten Programmier-sprachen, Tools und Entwicklungsumgebungen	20	30%	**6**	Architekturdurchstich so früh als möglich; Experten für einzelne Technologien im Team definieren	Fehlendes Know-How aufbauen
R10	**Skalieren auf mehr Daten**	Die gewählten Technologien skalieren nicht wie erwartet.	60	20%	**12**	Von Beginn an mit möglichst realitätsnahen Messdaten arbeiten; Einsatz von Stresstest-Tools, welche hohe Last akurat simulieren; Systemarchitektur wird von Beginn an für die Skalierung und den Betrieb in einer Cloud-Umgebung entworfen;	Prüfen, ob Ressourcenprobleme durch Optimierungen des Codes (Profiling), vertikales Skalieren oder allenfalls horizontales Skalieren gelöst werden können
R11	**Technische Machbarkeit**	Bei der Analyse/Konzept stellt sich heraus, dass eine Anforderung aufgrund von Wiedersprüchen oder technischer/logischer Einschränkungen nicht wie gewünscht lösbar ist.	30	20%	**6**	Evaluierung meherer Lösungsansätze; Je nach Konzept/Variante einen Prototypen bauen, um Potenzial und Machbarkeit abschätzen zu können; Prototypen mit möglichst realitätsnahen Daten testen.	Ausweichen auf andere Lösungen, Funtionalität in Absprache mit Betreuer und Auftraggeber einschränken
R12	**Ausfall Entwickler-Notebook**	Ein Notebook eines Teammitglieds fällt aus (Defekt, Verlust, Diebstahl), wobei bereits getätigte Arbeit und Daten verloren gehen können.	10	10%	**1**	Organisation eines Ersatz-Notebooks für jedes Teammitglied, regelmässige Speicherung von Dokumenten/Code auf OneDrive/VSTS	Ersatz-Notebook und Entwicklungsumgebung einrichten
R13	**Cloud Provider Features**	Der evaluierte Cloud Provider unterstützt eine Anforderung unseres Deployments nicht oder nur ungenügend	30	20%	**6**	Vom Beginn an direkt auf Ziel-Provider deployen; kritische Funktionen/Features so früh als möglich ermitteln und auf Cloud Provider implementiern; Einsatz von Container-Lösung (Docker), um allfälligen Umzug so schmerzlos als möglich zu machen	Ausweichen auf einen anderen Cloud-Anbieter
R14	**Cloud Provider Management UI**	Cloud Provider Web Manmagement Tools sind vorübergehend nicht verfügbar oder mit Fehlern behaftet und verzögern die Arbeit am Deployment	20	50%	**10**	Erfahrung zeigt, dass i.d.R. mit Ausfall des Web Managements zu rechnen ist, jedoch nicht der dahinterliegenden Infrastruktur; So viel als möglich automatisieren, sodass Zugriff auf Web Management Tools nicht regelmässig notwendig ist	Warten auf Behebung durch Cloud Provider, Eröffnen eines Tickets bei Cloud Provider
R15	**Cloud Provider Hosting Ausfall**	Cloud Plattform ist durch Problem auf Seiten von Cloud Provider vorübergehend nicht verfügbar	10	20%	**2**	Architektur von Beginn an auf Ausfälle einzelner Komponenten ausrichten; Möglichkeit zur Redundanz einplanen; Anforderungen an Uptime erfassen und mit SLA/Garantien von Cloud Provider abgleichen; Wenn nötig SLA oder höhere Verfügbarkeit bei Cloud Provider anstreben; Monitoring der Verfügbarkeit einzelner Teilkomponentem auf VSTS einrichten	Warten auf Behebung durch Cloud Provider, Eröffnen eines Tickets bei Cloud Provider, Prüfen, ob allenfalls durch den Ausfall eine Gutschrift entstanden ist (Anspruch muss i.d.R. durch Kunden beim Cloud Provider gemeldet werden)
R16	**Ausfall Projektmanagement-Tool**	Die Umgebung von Visual Studio Team Services ist nicht verfügbar, die Verwaltung der Arbeitspakete (inkl. Zeiterfassung) sowie der Zugriff auf das Code-Repository sind nicht möglich	10	15%	**1.5**	Regelmässiger Export aller Arbeitspakete (inkl. Zeiterfassung) in ein Excel-Dokument; Entwicklung mit einer lokalen Kopie des Code-Repositorys, Iterationsplanung so erstellen, dass Ausweichsarbeiten im Falle von VSTS Downtime möglich sind	Verwaltung der Arbeitspakete und die Entwicklung erfolgen lokal bis VSTS wieder verfügbar ist, anschliessend werden die Daten (Zeiterfassung im Excel, Code Changes) wieder importiert
Summe			**335**		**80.25**		

Änderungsgeschichte

Datum	Version	Änderung	Autor
26.02.2018	1.0	Erstellung des Dokuments, Erfassung der Risiken	Tobias Stähli
28.02.2018	1.1	Hinzufügen Cloud Provider Risiko in Absprache mit Dumeni Vincenz	Tobias Stähli
28.02.2018	1.2	Review / orthografische Korrekturen	Jonas Wälter
12.06.2018	1.3	Überprüfung der Risiken, Dokumentation eingetretener Risiken	Tobias Stähli, Dumeni Vincenz, Jonas Wälter

Eingetretene Risiken

Risiko	Beschreibung	Vorgehen	Schaden
R1	Fortlaufend neue Anforderungen und Wünsche seitens des Auftraggebers während des Projektverlaufs	Die neuene Anforderungen wurden jeweils aufgenommen, zuerst intern auf ihre Machbarkeit überprüft und anschliessend den Auftraggeber gebeten, die neu erfasste User Story zu priorisieren. Dies führte zu etwas mehr Planungsaufwand, da jedoch dem Auftraggeber klar war, dass wir nur eine beschränkte Zeit für unser Projekt zur Verfügung haben, zeigte er auch Verständnis, wenn wir ein Feature mit einer Begründung ablehnten.	**6**
R2	Umsetzung der externen API gemäss ISO unklar, da ISO viel Interpretationsspielraum lässt	Erstellen eines Dokumentes zur geplanten Umsetzung der API mit Vorschlägen, wie die Abbildung aus Sicht des Projektteams aussehen soll. Dieses wurde anschliessend vom Auftraggeber gegengelesen und die offenen Fragen konnten geklärt werden.	**3**
R5	Coding Styleguides (TypeScript/C#) wurden nicht immer eingehalten	Einsatz von Build-Task-Step, der das Mergen eines PRs bei Verstoss gegen Styleguides verhinderte; Sensibilisierung aller Projektmitarbeiter auf den Styleguide	**2**
R6	Fehlende Typings für verwendete TypeScript Module	Erstellen von PR auf entsprechendes Repository und Weiterarbeit auf Fork des Repositories bis entsprechender Merge Request akzeptiert wurde	**3**
R10	Unklarheit bei Stresstest, wo Bottleneck liegt, da keine der Komponente ausgelastet schien	Aufsetzen des Systems auf einem anderen Cloud Provider (Google Cloud) und Wiederholung der Tests. Es konnte eruiert werden, dass der Bottleneck ausserhalb des Systems liegen musste; vermutlich wurde der Netzwerk-Traffic durch den Provider künstlich gedrosselt;	**4**
R12	Technischer Defekt am Display des Entwicklernotebooks von Tobias Stähli (unbrauchbar ohne externer Monitor)	Da Notebook noch über Vor-Ort-Garantie verfügte, konnte das Problem innert 3 Tagen behoben werden; Wärenddessen wurde auf einem Ersatz-Notebook weitergearbeitet; Aufwand zum Aufsetzen von Docker for Windows schlug mit 0.5h zu Buche.	**0.5**
R13	Scheinbar aufgehängte VMs bei hoher Belastung durch Build-Server	Bei mehreren parallel laufenden Buildvorgängen schien die VM (System & Build-Agents) teilweise nur noch lesend auf die Festplatte zuzugreifen; Weil dadurch viel RAM gebraucht wurde, geriet die VM teilweise in einen Zustand, in dem die Funktion des DEV- & PROD-Systems beeinträchtigt war; Als Lösung wurde der Build Agent auf eine separate VM ausgelagert;	**6**
R14	Azure Portal Teilausfall bei Fehlersuche	Teilweise war das Portal von Azure nicht oder nur sehr langsam aufrufbar. Dies hat die Fehlersuche teilweise erschwert oder verlangsamt.	**2**

Totaler eingetretener Schaden: 26.5

Anforderungsspezifikation

Projektmitglieder
Stähli Tobias
Vincenz Dumeni
Wälter Jonas

Änderungsgeschichte

Datum	Version	Änderung	Autor
01.03.2018	1.0	Erstellung des Dokuments	Tobias Stähli
02.03.2018	1.1	Nichtfunktionale Anforderungen	Tobias Stähli
02.03.2018	1.2	Aktualisierung VSTS-Export	Tobias Stähli
07.03.2018	1.3	Review / orthografische Korrekturen	Jonas Wälter
10.06.2018	1.4	Aktualisierung VSTS-Export	Jonas Wälter

Inhaltsverzeichnis

1 Einführung

1.1 Beschreibung

Dieses Dokument beschreibt die detaillierten Anforderungen an das während dieser Bachelorarbeit zu entwickelnde System. Im Rahmen dieser Arbeit wird auch ein Schnittstellenkonzept für eine besser skalierende Schnittstelle zwischen Maschine und Data Collector definiert. Die Anforderungen an diese neue Schnittstelle werden separat im Dokument «Schnittstellenkonzept» erfasst.

1.2 Gültigkeitsbereich

Die Gültigkeit dieses Dokuments beschränkt sich auf die gesamte Projektdauer der Bachelorarbeit im Frühjahrssemester 2018. Allfällige spätere Anpassungen oder Ergänzungen im Dokument werden in der Änderungsgeschichte festgehalten.

1.3 Referenzen

Als Nachschlagewerk für unklare Begriffe sowie Abkürzungen sei an dieser Stelle auf das Glossar verwiesen, welches fortlaufend nachgeführt wird. In der nachfolgenden Tabelle sind alle weiteren referenzierten Dokumente und Links aufgelistet.

Thema	Referenz
Aufgabenstellung	OneDrive: \01_Aufgabenstellung\Aufgabenstellung.docx
Dokumentvorlage	OneDrive: \00_Administration\Template.docx
Glossar	OneDrive: \00_Administration\Glossar.docx
Projektplan	OneDrive: \02_Projektplan\Projektplan.docx
Schnittstellenkonzept	OneDrive: \04_Architektur_Design\Schnittstellenkonzept.docx
Personas	OneDrive: \ 03_Anforderungsanalyse\Personas.docx
Logo Suncar HK	http://www.suncar-hk.com/

2 Allgemeine Beschreibung

2.1 Produktfunktion

«Suncar Big Data Collector & Dashboard» ist ein Gesamtsystem zur Sammlung von Messdaten verschiedenster Elektromaschinen über einen Data Collector inklusive Auswertungs- und Verwaltungsmöglichkeiten, welche den Endkunden der SUNCAR HK AG über ein Frontend-Dashboard zur Verfügung gestellt werden.

Eine allgemeine Beschreibung der Funktionalität vom Gesamtsystem kann dem Projektplan entnommen werden. Die funktionalen und nichtfunktionalen Anforderungen sind in den entsprechenden Kapiteln in diesem Dokument detaillierter beschrieben.

2.2 Benutzermerkmale

Die Benutzergruppe des Frontends beschränkt sich auf die Mitarbeiter von Unternehmen, welche Elektromaschinen der SUNCAR HK AG in Betrieb haben. Ein interessierter Benutzer kann sich nicht eigenhändig für das System registrieren, stattdessen muss ein neuer Benutzeraccount durch einen Superuser des betreffenden Unternehmens oder durch einen Administrator erstellt werden.

Für eine detailliertere Beschreibung der Merkmale von Beispielbenutzern sei an dieser Stelle auf die «Personas» verwiesen.

2.3 Einschränkungen

Das Dashboard wird als Webplattform betrieben und ist dementsprechend abhängig von der bestehenden Internetverbindung des Benutzers. Die Realisierung einer Mobile App ist nicht Bestandteil des Projektes. Weitere Einschränkungen im Rahmen der Anforderungsspezifikation sind im Kapitel «Nichtfunktionale Anforderungen» detailliert beschrieben.

3 Funktionale Anforderungen

Dieser Abschnitt dokumentiert die funktionalen Anforderungen an das zu entwickelnde System mit Hilfe von User Stories. Die User Stories werden direkt in Visual Studio Team Services (VSTS) erfasst und verwaltet.

Für die funktionalen Anforderungen an die neue Schnittstelle zwischen Maschine und Data Collector sei an dieser Stelle auf das Schnittstellenkonzept verwiesen.

3.1 Aktoren (Rollen)

Die funktionalen Anforderungen greifen auf verschiedene Aktoren zurück, welche in der nachfolgenden Tabelle kurz beschrieben werden.

Aktor	Beschreibung
Standard User	Ein Standard User hat keine Verwaltungsrechte und nutzt das Frontend hauptsächlich, um die für ihn sichtbaren Maschinen und deren Messdaten anzusehen.
Superuser	Ein Superuser hat alle Möglichkeiten des Standard Users und kann als zusätzlich sein Unternehmen und die zugewiesenen User verwalten. (Unternehmens-Administrator)
Administrator	Ein Administrator hat alle Rechte und Möglichkeiten des Systems und kann so auch die Maschinen, Maschinentypen und Messtypen verwalten.
User	Als User gelten alle Standard User, Superuser und Administratoren.
Maschine	Eine Maschine kann dem Legacy Data Collector oder dem Data Collector ihre Messdaten übermitteln.
Legacy Data Collector	Ein Legacy Data Collector empfängt die Messdaten von einer Maschine und leitet diese an den Data Collector weiter.
Data Collector	Ein Data Collector empfängt die Messdaten von einer Maschine oder dem Legacy Data Collector und verarbeitet/persistiert diese.

Nachtrag:

Im Verlauf der Implementierungsphase kam beim Auftraggeber der Wunsch auf, die in der Aufgabenstellung definierten Benutzerrollen um den zusätzlichen Aktor **«Distributor»** zu erweitern, welcher über spezielle Berechtigungen für den Industriepartner «Huppenkothen» verfügt. Ein Distributor ist dabei grundsätzlich ein Administrator mit gewissen Einschränkungen (kein Schreibzugriff auf Maschinen, Maschinetypen und Messtypen).

3.2 User Stories

Zur Abbildung der funktionalen Anforderungen wurden insgesamt 85 User Stories definiert und jeweils einem von 14 Features zugewiesen. Für eine grobe Übersicht zu allen User Stories enthalten die nachfolgenden Unterkapitel einen VSTS-Export der User Stories für jedes Feature.

Alle User Stories wurden in der dazu üblichen Form («Als %Aktor% möchte ich …») kurz und knapp definiert sowie zusätzlich mit Akzeptanzkriterien versehen. Aus Gründen der Übersichtlichkeit wird auf die Darstellung der Akzeptanzkriterien in diesem Dokument verzichtet. Die Akzeptanzkriterien können entweder direkt in den User Stories auf VSTS oder aber im Dokument «Systemtest» nachgeschlagen werden.

Nachtrag:

Da der Aktor «Distributor» erst nachträglich als Anforderung aufkam, ist er in der kurzen Beschreibung der User Stories nicht enthalten. Die Akzeptanzkriterien hingegen wurden entsprechend angepasst. Wie bei den Aktoren beschrieben, entspricht der Distributor in vielen User Stories einem Administrator.

3.2.1 Feature «Dashboard»

↑ Dashboard

Als angemeldeter User möchte ich eine Einstiegsseite sehen, welche die für mich wichtigen Informationen zusammengefasst darstellt.

↓ Statuspage

Als angemeldeter Administrator möchte ich sehen, ob alle systemrelevanten Dienste ordnungsgemäss funktionieren.

↑ Kontaktdaten ansehen

Als angemeldeter User möchte ich die Kontaktdaten für den Maschinensupport ansehen. (E-Mail/Telefonnummer/Formular)

3.2.2 Feature «User-Verwaltung»

↑ Login

Als User möchte ich mich anmelden.

↑ Logout

Als angemeldeter User möchte ich mich abmelden.

↑ Account aktivieren

Als User möchte ich ein E-Mail erhalten, um den Account zu aktivieren und ein Passwort festlegen zu können.

↑ Account löschen

Als angemeldeter User möchte ich meinen Account löschen.

↑ Passwort zurücksetzen

Als User möchte ich mein Passwort zurücksetzen.

↑ User erfassen (als Admin)

Als angemeldeter Administrator möchte ich einen neuen User erfassen (inkl. Zuweisung Company).

↑ Superuser/User erfassen

Als angemeldeter Superuser möchte ich einen neuen Superuser oder Standard User für meine Company erfassen.

↑ Mein Profil ansehen

Als angemeldeter User möchte ich mein Profil ansehen

↑ Andere Profile ansehen

Als angemeldeter User möchte ich das Profil von einem anderen User ansehen, wenn ich dazu berechtigt bin.

↑ User-Liste ansehen

Als angemeldeter Superuser oder Administrator möchte ich eine Auflistung aller für mich sichtbaren User ansehen.

↑ Mein Profil bearbeiten

Als angemeldeter User möchte ich meine Profilangaben bearbeiten.

↑ Andere Profile bearbeiten

Als angemeldeter User möchte ich die Profilangaben von einem anderen User bearbeiten, wenn ich dazu berechtigt bin.

↑ Andere Accounts löschen

Als angemeldeter User möchte ich einen anderen Account löschen, wenn ich dazu berechtigt bin.

↓ Auswertung User-Aktivität

Als angemeldeter Administrator möchte ich eine Auswertung der User-Aktivität ansehen/ generieren.

3.2.3 Feature «Company-Verwaltung»

↑ Company erfassen

Als angemeldeter Administrator möchte ich eine neue Company erfassen.

↑ Company ansehen

Als angemeldeter Superuser möchte ich meine Company ansehen.

↑ Andere Company ansehen

Als angemeldeter Administrator möchte ich die Details (inkl. zugewiesener Users) einer beliebigen Company ansehen.

↑ Company-Liste ansehen

Als angemeldeter Administrator möchte ich eine Auflistung aller Companies ansehen.

↑ Company bearbeiten

Als angemeldeter Superuser möchte ich die Informationen meiner Company bearbeiten.

↑ Andere Company bearbeiten

Als angemeldeter Administrator möchte ich die Informationen einer beliebigen Company bearbeiten.

↑ Company löschen

Als angemeldeter Administrator möchte ich eine beliebige Company löschen.

3.2.4 Feature «Maschinen-Verwaltung»

↑ Maschine erfassen

Als angemeldeter Administrator möchte ich eine neue Maschine erfassen (inkl. Zuweisung Maschinentyp).

↑ Details von Maschine ansehen

Als angemeldeter User möchte ich Details (inkl. Maschinentyp) einer für mich sichtbaren Maschine ansehen.

↑ Maschinen-Liste ansehen

Als angemeldeter User möchte ich eine Auflistung aller für mich sichtbaren Maschinen ansehen.

↑ Maschine bearbeiten

Als angemeldeter Administrator möchte ich die Informationen einer Maschine bearbeiten.

↑ Maschine löschen

Als angemeldeter Administrator möchte ich eine Maschine löschen.

↑ Aktueller Maschinen-Status ansehen

Als angemeldeter User möchte ich den Live-Status einer für mich sichtbaren Maschine ansehen.

↑ Aktueller Maschinen-Standort ansehen

Als angemeldeter User möchte ich den aktuellen Standort einer für mich sichtbaren Maschine ansehen.

↑ Virtuelles Cockpit ansehen

Als angemeldeter User möchte ich das Virtuelle Cockpit mit aktuellen Messdaten einer für mich sichtbaren Maschine ansehen.

↑ Maschinenzustand PDF

Als angemeldeter User möchte ich ein PDF oder Ausdruck generieren, der den aktuellen Maschinenzustand enthält. (min. Standort, Stundenzähler)

↓ Maschinen-Blacklist

Als angemeldeter Administrator möchte ich Maschinen als "blacklisted" definieren, damit sie im GUI nicht abrufbar sind.

Maschinen-Cards: Filterung

Als angemeldeter User möchte ich die aufgeführten Maschinen nach Maschine, Maschinentyp und Status (letzte Verbindung) filtern können.

Cockpit: farbiger Status

Als angemeldeter User möchte ich auf dem Cockpit eine farbliche Unterscheidung der letzten Verbindung zur Maschine sehen.

Detailansicht: Maschine auswählen

Als angemeldeter User möchte ich direkt in der Detailansicht einer Maschine eine andere Maschine auswählen können.

3.2.5 Feature «Maschinentype-Verwaltung»

Maschinentyp erfassen

Als angemeldeter Administrator möchte ich einen neuen Maschinentyp erfassen.

Details von Maschinentyp ansehen

Als angemeldeter Administrator möchte ich Details eines beliebigen Maschinentyps ansehen.

Maschinentypen-Liste ansehen

Als angemeldeter Administrator möchte ich eine Auflistung aller Maschinentypen ansehen.

Maschinentyp bearbeiten

Als angemeldeter Administrator möchte ich die Informationen eines Maschinentyps bearbeiten.

Maschinentyp löschen

Als angemeldeter Administrator möchte ich einen Maschinentyp löschen.

3.2.6 Feature «Messtyp-Verwaltung»

Messtyp erfassen

Als angemeldeter Administrator möchte ich einen neuen Messtyp erfassen.

Details von Messtyp ansehen

Als angemeldeter Administrator möchte ich Details eines beliebigen Messtyps ansehen.

Messtypen-Liste ansehen

Als angemeldeter Administrator möchte ich eine Auflistung aller Messtypen ansehen.

Messtyp bearbeiten

Als angemeldeter Administrator möchte ich die Informationen eines Messtyps bearbeiten.

Messtyp löschen

Als angemeldeter Administrator möchte ich einen Messtyp löschen.

↑ Messtyp-Sichtbarkeit verwalten

Als angemeldeter Administrator möchte ich für einen Messtyp definieren, ab welcher Benutzerrolle dieser sichtbar sein soll.

↑ Messtyp an Maschinentyp zuweisen

Als angemeldeter Administrator möchte ich die Messtypen eines Machinentyps verwalten (ansehen, zuweisen, löschen).

3.2.7 Feature «Ownership-Verwaltung»

↑ Maschine an Company zuweisen

Als angemeldeter Administrator möchte ich eine Maschine einer Company zuweisen (inkl. Start- und optional Enddatum).

↑ Zuweisungen von Maschinen zu Company ansehen

Als angemeldeter Administrator möchte ich eine Auflistung aller Maschine-Company-Zuweisungen pro Maschine und pro Company ansehen.

↑ Maschine-Company-Zuweisung bearbeiten

Als angemeldeter Administrator möchte ich die Zuweisung einer Maschine zu einer Company bearbeiten.

↑ Maschine-Company-Zuweisung löschen

Als angemeldeter Administrator möchte ich die Zuweisung einer Maschine zu einer Company entfernen.

3.2.8 Feature «Access-Verwaltung»

↑ Maschinen-Access für User erfassen

Als angemeldeter Superuser möchte ich einen neuen Access-Zeitrahmen für einen User und eine Maschine meiner Company erfassen.

↑ Eigene Maschinen-Accesses anzeigen

Als angemeldeter User möchte ich meine eigenen aktuellen Access-Zeitrahmen auf allen für mich sichtbaren Maschinen ansehen.

↑ Maschinen-Access-Liste ansehen

Als angemeldeter Administrator möchte ich eine Auflistung aller Accesses sehen (Filterung nach User/Maschine, aktuelle & abgelaufene)

↑ Maschinen-Access für User bearbeiten

Als angemeldeter Superuser möchte ich den Access-Zeitrahmen eines Users auf eine Maschine meiner Company bearbeiten.

↑ Beliebiger Maschinen-Access für User verwalten

Als angemeldeter Administrator möchte ich beliebige Access-Zeitrahmen bearbeiten/löschen/erstellen (analog Superuser).

↑ Maschinen-Access für User löschen

Als angemeldeter Superuser möchte ich den Access eines Users auf eine Maschine meiner Company entfernen.

3.2.9 Feature «Karte»

↑ Maschinen-Karte ansehen

Als angemeldeter User möchte ich alle für mich sichtbaren Maschinen auf einer Karte ansehen (inkl. Symbol für Maschinentyp).

↑ Maschinen-Details auf Karte ansehen

Als angemeldeter User möchte ich die wichtigsten Messdaten (SOC, Drehzahl, Fehler) einer Maschine direkt auf der Karte sehen (auch mobile).

↑ Karte zentrieren

Als angemeldeter User möchte ich den Kartenausschnitt via Button auf alle meine Maschinen zentrieren.

↑ Von Maschinen-Karte zu Detailansicht navigieren

Als angemeldeter User möchte ich über die Karte zur Detailansicht einer Maschine navigieren.

↓ GPS-Spur anzeigen

Als angemeldeter User möchte ich die GPS-Spur einer Maschine über einen definierten Zeitraum auf der Karte ansehen.

✱ Maschinen-Karte: farbiger Status

Als angemeldeter User möchte ich auf der Karte eine farbliche Unterscheidung der Markers gemäss der letzten Verbingung zur Maschine sehen.

✱ Maschinen-Karte: Filterung

Als angemeldeter User möchte ich die auf der Karte angezeigten Maschine nach Maschine, Maschinentyp und Status (letzte Verbindung) filtern können.

3.2.10 Feature «Messdatenauswertung»

↑ Messdaten-Diagramm ansehen

Als angemeldeter User möchte ich alle Logdaten einer für mich sichtbaren Maschine in einem dynamischen Diagramm ansehen.

↑ Diagramm mit mehreren Messdaten ansehen

Als angemeldeter User möchte ich mehrere Messdaten im gleichen Diagramm ansehen, um Korrelationen erkennen zu können.

↑ Alte Messdaten ansehen

Als angemeldeter User möchte ich auch alte Messdaten ansehen können.

↑ Messdaten PDF

Als angemeldeter User möchte ich ein PDF oder Ausdruck generieren, der die aktuell angezeigten Messdaten enthält.

✱ Messdaten-Diagramm: Maschine auswählen

Als angemeldeter User möchte ich bei der Anzeige des Messdaten-Diagramms eine andere Maschine auswählen, ohne die angezeigten Messtypen nochmals neu auswählen zu müssen.

✱ CSV-Export Messdaten

Als angemeldeter User möchte ich die im Diagramm angezeigten Messdaten in ein CSV-File exportieren und herunterladen.

3.2.11 Feature «Zugriff»

↑ Datenschutz

Als User möchte ich sicher sein, dass nur berechtigte User auf die Daten zugreifen können und der Transfer verschlüsselt ist.

3.2.12 Feature «Data Collector»

↑ Messdaten senden (neu)

Als neue Maschine möchte ich dem Data Collector meine Messdaten übermitteln.

↑ Messdaten senden (Legacy)

Als bereits ausgelieferte Maschine möchte ich dem Legacy Data Collector meine Messdaten übermitteln.

↑ Data Collector

Als Data Collector möchte ich die erhaltenen Messdaten verarbeiten und persistieren.

↑ Legacy Data Collector

Als Legacy Data Collector möchte ich dem Data Collector die erhaltenen Messdaten weiterleiten.

3.2.13 Feature «API für Drittanbieter»

↓ API: Maschinenstatus lesen

Als API User möchte ich den aktuellen Status einer Maschine auslesen, falls ich dazu berechtigt bin.

↓ API: Messdaten lesen

Als API User möchte ich Messdaten einer Maschine lesen, falls ich dazu berechtigt bin.

↓ API: Authentifizierung

Als API User möchte ich mich über eine API authentifizieren, um anschliessend Token-basiert, weitere API-Abfragen absetzen zu können.

Schnittstelle ISO 15143-3

Als API User möchte ich meine Daten über die in ISO 15143-3 normierte Schnittstelle abrufen.

3.2.14 Feature «Datenanalyse»

Datenanalysen durchführen

Als angemeldeter User möchte ich eine Datenanalyse durchführen, um statistische Aussagen über die Daten machen zu können.

Benachrichtigung bei Schwellwert

Als angemeldeter User möchte ich einen Schwellwerte für ein Messtyp einrichten, bei dessen Überschreitung ich eine Benachrichtigung erhalte.

Fehlermeldungen ansehen

Als angemeldeter Administrator möchte ich ansehen, ob eine Maschine einen Fehler/Warnung generiert hat und wann dies geschehen ist. (Event-basierte Fehlermeldungen)

Fehlermeldung per Mail erhalten

Als Administrator möchte ich per E-Mail informiert werden, sollte ein Fehler/Warnung einer Maschine auftreten.

3.3 Priorisierung

Wie die Verwaltung der User Stories erfolgt auch deren Priorisierung direkt in VSTS, wobei vier verschiedene Prioritätsstufen zur Wahl stehen. Alle aufgeführten User Stories wurden durch den Auftraggeber priorisiert, wobei Suncar HK nur eine grobe Unterteilung zwischen «zwingend» und «optional» vorgenommen hat. Des Weiteren kamen im Verlauf des Projektes fortlaufend weitere Wünsche des Auftraggebers hinzu, welche als «Feature Requests» erfasst wurden.

Priorität	VSTS-Priorität	Beschreibung
↑	1 – 2	Zwingend (hohe Priorität)
↓	3 – 4	Optional (geringe Priorität)
✱	1 – 4	Feature Request (während der Implementation)

4 Nichtfunktionale Anforderungen

Genauso wichtig wie die funktionalen Anforderungen und meistens mit immenser Auswirkung auf die Softwarearchitektur sind die nichtfunktionalen Anforderungen an ein Projekt. Da sich die nichtfunktionalen Anforderungen nur schwierig auf einzelne User Stories abbilden lassen, werden diese zentral in diesem Abschnitt festgehalten.

Für die nichtfunktionalen Anforderungen an die neue Schnittstelle zwischen Maschine und Data Collector sei an dieser Stelle auf das Schnittstellenkonzept verwiesen.

4.1 NFR-01: Skalierbarkeit

Die Software- und Hardwareinfrastruktur muss für bis zu 200 Devices ausgelegt sein. Eine Skalierung der Infrastruktur soll theoretisch die gleichzeitige Verwaltung von 10'000 Devices ermöglichen. Diese Messgrösse muss jedoch nicht getestet/gemessen werden.

4.2 NFR-02: Internationalisierung

Als professionelle Webplattform soll das Frontend über eine wählbare Anzeigesprache verfügen («Internationalization»). Während der Projektphase müssen jedoch nebst Englisch und Deutsch keine weiteren Sprachen hinzugefügt werden.

4.3 NFR-03: Responsive Design für Mobile Devices

Das gesamte Frontend soll auch auf Mobile Devices intuitiv und ohne Darstellungsprobleme bedient werden können. Die Entwicklung des Frontends muss jedoch nicht nach dem «Mobile-First»-Ansatz erfolgen.

4.4 NFR-04: Browsersupport

Das Frontend soll primär für die Darstellung in aktuellen Browsern (Google Chrome, Mozilla Firefox, Safari, Microsoft Edge) optimiert werden. Die Kompatibilität mit veralteten und nicht weit verbreiteten Browsern ist nicht erforderlich.

4.5 NFR-05: Datensicherheit

Das System muss vor unberechtigtem Zugriff geschützt werden. Für den Zugriff auf das Frontend ist zwingend ein registrierter User erforderlich.

Ein angemeldeter User darf nur auf die für ihn freigegebenen Ressourcen (Maschinen, Company, User, etc.) zugreifen können, die Trennung der Mandanten soll stets gewährleistet sein.

Des Weiteren sollen sowohl die Measurements-Datenbank als auch die Metadata-Datenbank nicht direkt aus dem Internet erreichbar sein. Jede Kommunikation über öffentliche Netze darf nur verschlüsselt erfolgen. Zudem dürfen alle Passwörter nur verschlüsselt in der Datenbank abgelegt werden.

4.6 NFR-06: Modulare Architektur

Bei der Konzipierung und Implementierung des Gesamtsystems soll auf eine modulare Systemarchitektur geachtet werden, damit einzelne Komponenten bei Bedarf auf einfache Art und Weise ausgetauscht oder erweitert werden können.

4.7 NFR-07: Aufbewahrung Messdaten

Alle Messdaten in der Measurements-Datenbank sollen mindestens fünf Jahre aufbewahrt werden.

4.8 NFR-08: Redundante Softwarearchitektur

Das Gesamtsystem muss nicht redundant ausgelegt werden. Die Softwarearchitektur soll jedoch so gewählt werden, dass eine redundante Auslegung erlaubt wird.

4.9 NFR-09: Benutzerfreundliches GUI

Das Frontend soll so konzipiert und strukturiert aufgebaut sein, dass der Einarbeitungsaufwand für einen User gering ist und keine Bedienungsanleitung erforderlich ist.

Um dem Benutzer eine moderne und gewohnte Umgebung zu bieten, soll sich das Frontend-Design an den Vorgaben der Design-Guideline «Material Design» orientieren.

Alle Eingabedaten sollen sowohl clientseitig während der Eingabe in Formularen, als auch serverseitig vor der Persistierung validiert und allenfalls mit verständlichen Fehlermeldungen beantwortet werden.

4.10 NFR-10: Branchenspezifische Normen

Bei der Entwicklung der API für Drittanbieter soll die ISO-Norm 15143-3 für «Earth-moving machinery and mobile road construction machinery» eingehalten werden.

Personas

Projektmitglieder
Stähli Tobias
Vincenz Dumeni
Wälter Jonas

Änderungsgeschichte

Datum	Version	Änderung	Autor
02.03.2018	1.0	Erstellung des Dokuments	Jonas Wälter
16.03.2018	1.1	Vervollständigung der Personas	Jonas Wälter
25.04,2018	1.2	Erweiterung gemäss Meeting	Jonas Wälter

Zweck

In diesem Dokument sind verschiedene Personas definiert, welche als Grundlage für die weitere Analyse und Definition der Anforderungen dienen.

Wie den nichtfunktionalen Anforderungen entnommen werden kann, soll besonderen Wert auf ein benutzerfreundliches Frontend gelegt werden (NFR-09). Dazu ist es während der Entwicklung notwendig, sich in die zukünftigen Benutzer aller Zielgruppen und ihre Verhaltensmuster hineinzuversetzen.

In der Anforderungsanalyse haben sich die nachfolgenden drei Zielgruppen ergeben, welche im System schlussendlich in Form von Benutzerrollen abgebildet werden.

- User
- Superuser
- Administrator

Im Verlauf des Projektes kam auf Wunsch des Auftraggebers mit dem **«Distributor»** noch eine zusätzliche Benutzerrolle hinzu, welche in den Personas nicht abgebildet wird. Für eine Übersicht zu den Benutzerrollen und ihren Berechtigungen sei an dieser Stelle auf die Domainanalyse verwiesen.

Eine Persona wiederspiegelt einen typischen Vertreter einer bestimmten Zielgruppe inklusive Verhaltensweise, Bedürfnisse sowie Ziele und gibt der Zielgruppe dadurch ein Gesicht. So kann allen Teammitgliedern ein einheitliches Verständnis der Zielgruppen vermittelt und eine Grundlage für zukünftige Entwicklungs- und Designentscheide geschaffen werden.[1]

Durch die Definition der Personas soll folglich sichergestellt werden, dass die Zielgruppen und die Benutzerfreundlichkeit während der gesamten Entwicklungsphase berücksichtigt werden.

Referenzen

Thema	Referenz
Profilbilder	https://randomuser.me/photos
Anforderungsspezifikation	OneDrive: \03_Anforderungsanalyse\ Anforderungsspezifikation.docx
Domainanalyse (Rollen und Berechtigungen)	OneDrive: \03_Anforderungsanalyse\Domainanalyse.docx

[1] (Personas, 2018)

Personas

Marco Heller

Der gewissenhafte Bauführer

E-Mailadresse	marco.heller@suncar-hk.com
Geburtsdatum	13.07.1985 (32 Jahre)
Passwort	Marco1985!
Familienstand	Verheiratet seit 4 Jahren, 1 Tochter
Beruf / Zuständigkeiten	Langjähriger Mitarbeiter eines Bauunternehmens und als Bauführer verantwortlich für die Baumaschinenflotte des Unternehmens und damit auch für die SUNCAR-Baumaschinen
Ziele	Marco möchte stets über den Zustand seiner SUNCAR-Baumaschinen Bescheid wissen, da er für deren reibungsloses Funktionieren verantwortlich ist. Da er oft auf den Baustellen unterwegs ist, möchte er die benötigten Informationen auf einer einfachen Art und Weise direkt über das Smartphone abrufen können. Damit die Baumaschinen am nächsten Tag mit vollen Batterien starten können, möchte er jeden Abend von zuhause aus kurz überprüfen, ob alle Baumaschinen aufgeladen werden.
Fragen	• Funktionieren meine Baumaschinen einwandfrei? • Wie lange reicht die Batterie einer Maschine noch? • Wie steht es mit den anderen Messwerten und wie haben sich die Daten im Verlaufe der Zeit verändert?
Informatikkenntnisse	Marco besitzt ein Samsung Smartphone und im Unternehmen steht im zudem ein Notebook zur Verfügung. Mit der Bedienung beider Geräte und der Navigation im Internet bekundet er keine Mühe.

Benutzerrolle

Marco Heller ist ein typischer Vertreter der Benutzerrolle **«User»**. Er benötigt primär Lesezugriff auf die ihm zugewiesenen Maschinen, um deren Status überprüfen und die Messdaten analysieren zu können. Weiterführende Zugriffsrechte sind hingegen nicht notwendig, um seinen Zuständigkeiten nachzukommen.

Karin Freuler

Die Maschinen-Verwalterin

E-Mailadresse	karin.freuler@suncar-hk.com
Geburtsdatum	03.12.1976 (41 Jahre)
Passwort	19Karin76#
Familienstand	Verheiratet seit 15 Jahren, 2 Söhne & 1 Tochter
Beruf / Zuständigkeiten	Als Kauffrau im Sekretariat eines Bauunternehmens tätig und in dieser Funktion für die Verwaltung der Baumaschinen zuständig
Ziele	Karin möchte die SUNCAR-Baumaschinen des Unternehmens verwalten – inklusive Userverwaltung und Zugriffsberechtigungen. Da SUNCAR-Baumaschinen auch an Drittunternehmen vermietet werden, legt sie grossen Wert darauf, dass diese Unternehmen nur während der Mietperiode Informationen der Maschine einsehen können. Karin arbeitet im Büro an einem Notebook und möchte die Einstellungen betreffend Maschinen, User und Zugriffsberechtigungen mit möglichst wenig Zeitaufwand ändern können.
Fragen	• Welche Maschinen und Benutzer sind in meiner «Verwaltungshoheit»? • Welche Benutzer haben in welchem Zeitraum Zugriff auf welche Maschinen und wie kann ich das ändern? • Wo befinden sich die Maschinen aktuell?
Informatikkenntnisse	Karin besitzt noch ein altmodisches Handy ohne Internetanschluss und nutzt daher vorwiegend ihr Familien-Notebook. Als Kauffrau kommt sie sehr gut und schnell mit Informatikmitteln zurecht. Sie nervt sich jedoch, wenn eine Website unverständlich aufgebaut ist und sie dadurch viel Zeit verliert.

Benutzerrolle

Karin Freuler entspricht vollumfänglich der Benutzerrolle **«Superuser»**. Zusätzlich zu den Berechtigungen der Rolle «User» braucht sie Schreibzugriff auf ihre eigene Firma sowie deren Benutzer und Maschinen, um ihre Verwaltungsaufgaben erledigen zu können.

Thomas Sommer

Der Administrator

E-Mailadresse	thomas.sommer@suncar-hk.com
Geburtsdatum	28.10.1992 (25 Jahre)
Passwort	!s(T%s7e3>Z
Familienstand	Ledig
Beruf / Zuständigkeiten	Nach dem Bachelorabschluss als Maschineningenieur nun für die SUNCAR HK AG tätig und für die Administration des Kunden-Webportals zuständig
Ziele	Thomas möchte alle Maschinen/-typen, Messtypen/-daten, Unternehmen, Benutzer, etc. zentralisiert und so einfach wie möglich verwalten. Bei Support-Anfragen von Kunden möchte er zudem eine Auswertung der betreffenden Maschine durchführen, um den Problemen auf den Grund gehen zu können. Er arbeitet normalerweise mit einem Computer am Arbeitsplatz, möchte die Webplattform aber auch während dem Arbeitsweg im Zug auf seinem Tablet nutzen können.
Fragen	• Wie kann ich neue Maschinen, Maschinentypen, Messtypen, etc. im System anlegen? • Welche Maschinen haben sich seit einer längeren Zeit nicht mehr gemeldet? • Welche Benutzer haben sich schon lange nicht mehr angemeldet? • Was könnte gemäss den Messdaten das Problem bei einer defekten Baumaschine sein?
Informatikkenntnisse	Thomas ist sehr technikaffin und legt sich immer die neusten Smartphones, Tablets und Notebooks zu. Dementsprechend hat er auch keinerlei Probleme im Umgang mit den Geräten und dem Internet.

Benutzerrolle

Thomas Sommer ist ein klassisches Beispiel der Benutzerrolle **«Administrator»**. So ist er auf Lese- und Schreibrechte in allen vorhandenen Bereichen angewiesen, um das Gesamtsystem administrieren zu können.

Literaturverzeichnis

Personas. (2018). Abgerufen am 24. April 2018 von usability.de: https://www.usability.de/leistungen/methoden/personas.html

Schnittstellenkonzept

Projektmitglieder
Stähli Tobias
Vincenz Dumeni
Wälter Jonas

Änderungsgeschichte

Datum	Version	Änderung	Autor
22.02.2018	1.0	Erstellung des Dokuments	Tobias Stähli
23.02.2018	1.1	Erfassen Ist-Schnittstelle	Tobias Stähli
28.02.2018	1.2	Erfassen Anforderungen an Schnittstelle	Tobias Stähli
02.03.2018	1.3	Dokumentation neue Schnittstelle	Tobias Stähli
07.03.2018	1.4	Review / orthografische Korrekturen	Jonas Wälter
23.03.2018	1.5	Anpassungen Client-Zertifikate-Verteilung	Tobias Stähli

Inhaltsverzeichnis

1 Einführung

1.1 Beschreibung

Dieses Dokument beschreibt die Schnittstelle zwischen Maschinen und dem Data Collector. Dabei werden die Anforderungen an die Schnittstelle analysiert und die bestehende Schnittstelle sowie deren Abdeckungsgrad der Anforderungen erläutert.

Anschliessend werden die neue Schnittstelle und das Vorgehen zur Ersetzung bzw. parallele Betreiben der bestehenden und der neuen Schnittstellen definiert.

1.2 Gültigkeitsbereich

Die Gültigkeit dieses Dokuments beschränkt sich auf die gesamte Projektdauer der Bachelorarbeit im Frühjahrssemester 2018. Allfällige spätere Anpassungen oder Ergänzungen im Dokument werden in der Änderungsgeschichte festgehalten.

1.3 Referenzen

Als Nachschlagewerk für unklare Begriffe sowie Abkürzungen sei an dieser Stelle auf das Glossar verwiesen, welches fortlaufend nachgeführt wird. In der nachfolgenden Tabelle sind alle weiteren referenzierten Dokumente und Links aufgelistet.

Thema	Referenz
Aufgabenstellung	OneDrive: \01_Aufgabenstellung\Aufgabenstellung.docx
Dokumentvorlage	OneDrive: \00_Administration\Template.docx
Glossar	OneDrive: \00_Administration\Glossar.docx
Projektplan	OneDrive: \02_Projektplan\Projektplan.docx
Anforderungsanalyse	OneDrive: \03_Anforderungsanalyse\ Anforderungsspezifikation.docx
Logo Suncar HK	http://www.suncar-hk.com/

2 Anforderungen

Die nachfolgenden Anforderungen beziehen sich auf die Übertragung der Messdaten zwischen einer Maschine und dem Data Collector. Die Rechenkapazitäten auf der Maschine sind stark eingeschränkt, da es sich um ein Embedded System handelt. Konkret dürfen aber in Absprache mit dem Auftraggeber folgende Voraussetzungen für das Schnittstellenkonzept angenommen werden:

- Die Maschinen verfügen über einen TCP-Stack.
- Die Maschinen können eine mit SSL gesicherte Verbindung aufbauen.
- Die Maschinen verfügen **nicht** unbedingt über eine Real Time Clock, können aber mindestens die relative Zeit während des Betriebes messen.
- Die Maschinen verfügen über begrenzte CPU und RAM Ressourcen. Zudem ist der Sekundärspeicher begrenzt.

2.1 Funktionale Anforderungen

Die Schnittstelle muss über direkte oder indirekte Wege mindestens die nachfolgenden Informationen übertragen. Um eine ungefähre Abschätzung des Datenvolumens in den nichtfunktionalen Anforderungen durchführen zu können, wird an dieser Stelle ein Datentyp respektive eine Grösse des Informationselementes angenommen.

Bezeichnung	Datentyp
Maschinen ID	4 Byte unsigned int
Messwert Typ	4 Byte unsigned int
Zeitstempel auf Sekunde	8 Byte unsigned long
Messwert	8 Byte double

2.2 Nichtfunktionale Anforderungen

Einige nichtfunktionale Anforderungen sind direkt aus der Aufgabenstellung ablesbar. Zusätzlich wurden die direkt ablesbaren Anforderungen mit solchen basierend auf der Analyse des Netzwerktraces einer laufenden Maschine über den Zeitraum von einer Stunde ergänzt. Ferner wurden einige Punkte direkt mit dem Auftraggeber abgeklärt und wo nötig festgelegt.

Gemäss den Messungen werden durchschnittlich 2.33 Datenwerte pro Sekunde pro Maschine versendet. Um allfällige zukünftige Anforderungen an die Anzahl Datenwerte pro Sekunde abdecken zu können, wird der gemessene Wert grosszügig aufgerundet. Daher wird für alle nachfolgenden Abschätzungen mit einem durchschnittlichen Datenaufkommen von 10 Datenwerten/Sekunde/Maschine gerechnet. Gemäss Auftraggeber ist dies sicherlich zu hoch geschätzt, da bereits die 2.33 Datenwerte pro Sekunde höher sind als eigentlich gewollt.

2.2.1 NFR-01: Skalierbarkeit

Die Schnittstelle soll ihm Rahmen dieses Projektes auf 200 aktive Maschinen ausgelegt werden. Allerdings soll eine Planung zur Skalierung der Schnittstelle auf 10'000 gleichzeitig aktive Maschinen erarbeitet werden. Es sollen mindestens 10 Messwerte/Sekunde/Maschine übertragen werden können. Die Schnittstelle soll also eine möglichst gute Netzwerkauslastung bieten und protokollbedingten Overhead wo möglich in einem sinnvollen Rahmen minimieren.

2.2.2 NFR-02: Ressourcenschonende Client-Implementation

Die Übertragung einzelner Messdaten soll im Normalfall innert zwei bis drei Sekunden nach deren Erzeugung auf der Maschine abgeschlossen sein, um einen Puffer auf Maschinenseite möglichst klein zu halten.

Falls Bibliotheken oder existierende Protokolle eingesetzt werden, müssen Implementationen in einer hardwarenahen Programmiersprache (C oder C++) verfügbar sein.

2.2.3 NFR-03: Übertragungssicherheit

Die Pakete sollen sicher übermittelt und im Falle eines verlorenen Netzwerkpaketes erneut übertragen werden. Der Empfang der einzelnen Pakete soll von der Gegenstelle quittiert werden. Die Reihenfolge der einzelnen Messwerte muss auf der Empfängerseite rekonstruierbar sein.

2.2.4 NFR-04: Datensicherheit

Es dürfen nur authentifizierte Maschinen Daten über die Schnittstelle übertragen. Die Datenübertragung soll durch das öffentliche Internet erfolgen können und daher entsprechend vor Manipulationen und Mitlesern geschützt werden.

3 Bestehende Schnittstelle

Dieser Abschnitt beschreibt die bestehende Schnittstelle, über welche die bereits im Betrieb stehenden Maschinen gegenwärtig ihre Messdaten übermitteln.

3.1 Definition

Jeder einzelnen Maschine wird ein eindeutiger UDP-Port zugewiesen, über welchen die Datenpakete verschickt werden. Jeder Messwert wir in einem separaten UDP-Paket übertragen. Weil die auf den Maschinen verwendeten CPUs bzw. das verwendete Betriebssystem zur Zeit der Implementation über keinen eingebauten TCP-Stack verfügten, war UDP die einzige mögliche Variante, um von der Maschine Kontakt zur Aussenwelt herzustellen.

3.1.1 Protokoll

Der UDP-Payload ist ein ASCII-encodierter Text, dessen Format durch die nachfolgende EBNF-Grammatik[1] beschrieben wird:

```
UDP_Payload = "!", MeasurementType, ",", MeasurementValue, ",", ";";
MeasurementType =     unsigned_decimal_integer;
MeasurementValue =    signed_decimal_float;
```

Die Bedeutung der tatsächlichen Werte der MeasurementType ist für das Protokoll irrelevant und wird in der Metadatenbank der Applikation festgehalten.

Der UDP-Payload und die zugehörige EBNF-Grammatik lassen sich anhand der nachfolgenden Beispiele von UDP-Paketen aufzeigen.

ASCII-Darstellung	Bytes	Bedeutung
!3,109.6,;	0x21, 0x33, 0x2c, 0x31, 0x30, 0x39, 0x2e, 0x36, 0x2c, 0x3b	**MeasurementType** = 3 **MeasurementValue** = 109.6 ➔ Maschine meldet aktuelle Spannung der Hauptbatterie von 109.6 V
!5,-8.6,;	0x21, 0x35, 0x2c, 0x2d, 0x38, 0x2e, 0x36, 0x2c, 0x3b	**MeasurementType** = 5 **MeasurementValue** = -8.6 ➔ Maschine meldet aktuellen Leistungsbezug des Motors von 8.6 kW

[1] (Message Queuing Telemetry Transport, 2018)

Die nachfolgende Abbildung zeigt das tatsächliche UDP-Paket inklusive IP Header des zweiten Beispiels in Wireshark.

```
> Frame 2538: 37 bytes on wire (296 bits), 37 bytes captured (296 bits)
  Raw packet data
v Internet Protocol Version 4, Src: 10.8.0.110 (10.8.0.110), Dst: 10.8.0.1 (10.8.0.1)
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 37
    Identification: 0x0000 (0)
  > Flags: 0x02 (Don't Fragment)
    Fragment offset: 0
    Time to live: 63
    Protocol: UDP (17)
    Header checksum: 0x274a [validation disabled]
    [Header checksum status: Unverified]
    Source: 10.8.0.110 (10.8.0.110)
    Destination: 10.8.0.1 (10.8.0.1)
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
v User Datagram Protocol, Src Port: msfw-array (2174), Dst Port: msfw-array (2174)
    Source Port: msfw-array (2174)
    Destination Port: msfw-array (2174)
    Length: 17
    Checksum: 0xe394 [unverified]
    [Checksum Status: Unverified]
    [Stream index: 0]
v Data (9 bytes)
    Data: 21352c2d382e362c3b
    Text: !5,-8.6,;
    [Length: 9]
```

Abbildung 1: UDP-Paket inklusive IP Header

3.1.2 Übertragung

Um die Datensicherheit zu gewährleisten und die übertragungsbedingten Paketverluste zu minimieren, wurde zwischen Maschine und Data Collector ein VPN über TCP eingerichtet. Zu diesem Zweck wird noch auf der Maschine ein GSM-Router eingesetzt. Alle Maschinen befinden sich im gleichen logischen Subnetz.

Die Identifikation der Maschine erfolgt über den verwendeten UDP-Port. Deshalb ist jeder Maschine genau ein UDP-Port zugewiesen. Die Übersetzung (Mapping) von einem UDP-Port zur entsprechenden Maschine erfolgt im Data Collector.

Als Zeitpunkt der Messung wird im Data Collector der Eingang des UDP-Paketes verwendet. Es findet keine Korrektur von allfälligen latenzbedingten Verfälschungen statt. Der Empfang eines Messwertes wird nicht quittiert. Sobald die Maschine das UDP-Paket versendet hat, wird eine erfolgreiche Übermittlung angenommen.

3.2 Diskussion

Die bisherige Schnittstelle kann sowohl die funktionalen als auch die nichtfunktionalen Anforderungen nicht mehr genügend erfüllen. In diesem Abschnitt wird etwas detaillierter auf die Probleme der aktuellen Schnittstelle mit den nichtfunktionalen Anforderungen eingegangen.

3.2.1 NFR-01: Skalierbarkeit

Die Schnittstelle würde rein von der Datenmenge her recht gut auf eine höhere Anzahl Maschinen bzw. Messdaten pro Maschine skalieren, da sie sehr simpel implementiert ist und rein netzwerktechnisch wenig Anforderungen stellt. Da jedoch jede einzelne Maschine ihren eigenen UDP-Port zugewiesen hat, muss bei jeder Maschine ein separater Port hinterlegt werden. Zudem muss der Data Collector auf den neuen Port bzw. die dazugehörige Maschine konfiguriert werden. Fehlkonfigurationen (z.B. Doppelbenutzung eines UDP-Ports) werden nicht erkannt und können zu fehlerhaften und inkonsistenten Einträgen in der Messdatenbank führen.

Als weitere Schwachstelle ist zu bedenken, dass im Falle von einer höheren Anzahl Maschinen, irgendwann die Grenze des möglichen Datendurchsatzes auf einem Netzwerk-Interface-Controller erreicht wird. Dann muss zwingend eine zweite Instanz des Data Collectors eingesetzt werden. Da jedoch kein Mechanismus zur dynamischen Konfiguration der Ziel-IP bzw. des Ziel-Ports implementiert wurde, ist dieses Skalierungsvariante mit sehr hohem Aufwand verbunden.

3.2.2 NFR-02: Ressourcenschonende Client-Implementation

Diese Anforderung ist auf Kosten der anderen Anforderungen dieser Schnittstelle sehr gut erfüllt, da UDP im Gegensatz zu TCP beinahe ohne State und damit ohne Ressourcenanforderungen auskommt.

3.2.3 NFR-03: Übertragungssicherheit

Da ein Teil der Übertragung der Messdaten mit UDP implementiert ist, kann nicht garantiert werden, dass auf dem Übertragungsweg keine Pakete verloren gehen. Dies wird zwar durch den Einsatz des VPN-Tunnels vermindert, trotzdem kann beispielsweise die Netzwerk-Schnittstelle des Data Collectors Pakete verwerfen, falls z.B. die Speicherung der Daten in der Messdatenbank einmal länger dauern sollte. Zudem ist der Data Collector als «Single Point of Failure» ein Risikofaktor, der bei einem Ausfall dazu führt, dass Messdaten verloren gehen. Mit diesem Datenverlust ist auch bei einem geplanten Stopp des Data Collectors (z.B. wegen Wartungsarbeiten am Hostsystem oder dem Einspielen einer neuen Software-Version) zu rechnen.

3.2.4 NFR-04: Datensicherheit

Mittels des VPN-Tunnels wird die Sicherheit der Datenübertragung gewahrt. Solange die VPN-Zugangsdaten auf sicherem Wege verteilt werden (z.B. Einrichten direkt vor Ort durch einen Techniker), kann auch die Datenintegrität garantiert werden. Dieser Ansatz skaliert vielleicht noch bis zu mehreren zehn oder hundert Maschinen, aber spätestens ab tausend Maschinen kann nicht mehr jeder einzelne eingesetzte VPN-Router durch eine absolut vertrauenswürdige Person eingerichtet und allenfalls ersetzt werden. Wenn also das Konzept des einen Subnetzes ohne zusätzliche Verschlüsselung und Authentifizierung hochskaliert wird, dann sind Sicherheitsprobleme betreffend Datensicherheit, Integrität und Verfügbarkeit vorprogrammiert.

4 Neue Schnittstelle

Im Zuge dieses Projektes wird eine neue Schnittstelle konzipiert und implementiert. Neu soll die Übertragung der Messdaten mittels eines standardisierten und speziell für Mess- bzw. Sensordaten entwickelten Message Brokers umgesetzt werden. Dabei wir auf den OASIS-Standard Message Queuing Telemetry Transport (MQTT)[2] gesetzt, welcher einen leichtgewichtigen und auf TCP aufbauenden Message Broker definiert. MQTT kennt drei verschiedene Arten von Quality of Service (QoS) für die zu übertragenden Nachrichten. Für die vorliegende Problemstellung wird bei der Übertragung das QoS-Level 1 (At least once delivery) verwendet.

4.1 Definition

Jede Maschine implementiert einen MQTT-Client und verbindet sich mit einem zentralen Message Queuing Server. Die Adresse dieses Servers ist in Form einer URI auf der Maschine hinterlegt. Zudem wird jede Maschine mit einer individuellen nummerischen ID konfiguriert.

Jede Maschine sollte sich auf das Topic /timestamp registrieren. Auf diesem Topic wird alle 30 Sekunden die aktuelle Zeit vom Data Collector versendet. Dabei wird das QoS-Level 0 (At most once delivery) verwendet. Die Maschine übernimmt die versendete Zeit für die Messwerte, welche an das Topic /measurement_data gesendet werden. Falls Messwerte vor dem ersten Empfang des Data Collector-Zeitstempels versendet werden, wird der Zeitstempel 0 mitgeschickt. In diesem Fall ersetzt der Data Collector den Wert 0 durch den Zeitstempel bei Eingang des Netzwerkpaketes.

Die Maschinen senden ihre Messwerte an das Topic /measurement_data mit QoS-Level 1 (At least once delivery). Dazu verwenden sie das im Abschnitt 4.1.2 definierte Payload-Protokoll. Um die Anzahl der Netzwerkpakete und den protokollbedingten Overhead zu reduzieren, kann die Übertragung mehrere Messwerte in der gleichen Message zusammengefasst werden. Eine Maschine kann so zum Beispiel alle 10 Sekunden die Messwerte seit der letzten Übertragung versenden. Das Payload-Protokoll unterstützt das explizite Angeben von Zeitinformationen in Form eines Referenz-Zeitstempels und einem allfälligen Offset pro mitgesendetem Messwert.

4.1.1 Transport Security und Client Authentifizierung

Die Absicherung des Datenverkehrs zwischen Message Broker und Maschine erfolgt mittels TLS. Der Message Broker Service verfügt dazu über ein Zertifikat, welches den Maschinen bei der Installation der Firmware als vertrauenswürdig hinterlegt wird. Dabei wird das sogenannte Certificate Pinning der ausstellenden Zertifizierungsstelle (CA) eingesetzt. Das Pinning findet auf die ausstellende CA statt, da somit im Falle der Kompromittierung des Message Broker Certificates nur das entsprechende Zertifikat ausgewechselt werden muss und kein Update aller Maschinen notwendig ist.

[2] (Message Queuing Telemetry Transport, 2018)

Die Maschinen authentifizieren sich gegenüber dem Message Broker mit einem bei der Inbetriebnahme der Maschine generierten und von einer vom Auftraggeber betriebenen CA signierten Clientzertifikat. Die Verwaltung und Signierung der Clientzertifikate bzw. der entsprechenden CA wird durch SUNCAR HK sichergestellt.

Der in TLS verwendete Algorithmus zur symmetrischen Verschlüsselung wird vom Message Broker festgelegt und soll den Anforderungen aus NFR-04 (Datensicherheit) entsprechen. Es soll ein Schlüsselaustauschverfahren eingesetzt werden, welches Perfect Forward Secrecy (PFS) unterstützt. Damit wird im Falle einer Kompromittierung von Schlüsselmaterial das Ausmass des Schadens in Grenzen gehalten.

4.1.2 Protokoll von Topic /timestamp

Jedes versendete Datenpaket besteht aus der Big-Endian 8 Byte-Darstellung der Anzahl vergangener Millisekunden seit dem 1. Januar 1970. Die folgende Tabelle enthält einige Timestamp-Beispiele:

Timestamp	Zeitpunkt	Bytes
1000	01.01.1970 00:00:01	[0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0xE8]
1520164972476	04.03.2018 12:02:52	[0x00, 0x00, 0x01, 0x61, 0xF0, 0xE4, 0xA7, 0xBC]
7258118400000	01.01.2200 00:00:00	[0x00, 0x00, 0x06, 0x99, 0xE9, 0x91, 0xA8, 0x00]

4.1.3 Payload von /measurement_data

Der Payload von /measurement_data ist ein ASCII-encodierter Text, dessen Format durch die folgende EBNF-Grammatik beschrieben wird. ASCII wurde gewählt, um das Parsen und die Erzeugung des Payloads so einfach wie möglich zu halten und die Lesbarkeit des Payloads zu verbessern. Mit einem kompakten binären Encoding wie beispielsweise Googles «Protocol Buffer»[3] oder «ANS1 Packed Encoding Rules»[4] könnten zwar noch einige Bytes gespart werden, allerdings auf Kosten der Komplexität während der Entwicklung und bei der Generierung des Message Payloads auf der Maschine.

```
MQTT_Payload =          MachineIdentifier, ";", InitialTimestamp,
                        ";", Measurements ;
MachineIdentifier =     unsigned_decimal_integer ;
InitialTimestamp =      unsigned_decimal_integer ;
Measurements =          Measurement, { ",", Measurement }
```

[3] (Protocol Buffers, 2018)

[4] (ANS.1 encoding rules, 2002)

```
Measurement =           [ TimestampAddition, "!" ], Measurement-
                        Type, ":", MeasurementValue ;
MeasurementType =       unsigned_decimal_integer ;
MeasurementValue =      signed_decimal_float ;
TimestampAddition =     unsigned_decimal_integer ;
```

Die nachfolgende Tabelle beinhaltet eine kurze Beschreibung der einzelnen Variablen der EBNF-Grammatik.

Teil	Bedeutung
MQTT_Payload	Das Top-Level Informationselement, welches die in dieser Message versendeten Messwerte beinhaltet.
MachineIdentifier	Ein von der SUNCAR HK AG definierter, eindeutiger nummerischer ID-Wert für die Maschine, auf welcher die nachfolgenden Messwerte ursprünglich gemessen wurden.
InitialTimestamp	Ein vollständiger Unix-Timestamp, welcher die Millisekunden seit dem 1.1.1970 enthält. Falls die Maschine noch nie eine Nachricht auf dem Topic /timestamp empfangen hat, muss dieses Feld auf 0 gesetzt werden. Es wird dann auf Empfängerseite der Timestamp bei Eingang des dazugehörigen Netzwerkpaketes angenommen.
Measurements	Auflistung der in dieser Message übertragenen Messwerte
Measurement	Ein einzelner Messwert, bestehend aus einem Typ, einem tatsächlichen Wert und einem optionalen Additionswert zum Timestamp.
MesurementType	Gibt den Typ der gemessenen Information an: Eindeutiger, vorzeichenloser Ganzzahlwert in Dezimaldarstellung
MeasurementValue	Gemessener Wert als Gleitkommazahl in Dezimaldarstellung.
TimestampAddition	Dieser optionale, positive Ganzzahlwert in Dezimaldarstellung gibt einen Zusatzwert in Millisekunden für diesen und alle folgenden Measurements bzw. dessen Timestamps bezüglich des Wertes in InitialTimestamp in dieser Nachricht an. Dabei addieren sich mehrere TimestampAdditions. Fehlt dieser Wert, dann wird 0 angenommen, was gleichbedeutend ist mit demselben Zeitstempel für das aktuelle Measurement und dem vorhergehenden Measurement. Handelt es sich dabei um das erste Measurement in dieser Message, dann ist der InitialTimestamp zu verwenden.

Der Payload und die zugehörige EBNF-Grammatik lassen sich anhand der nachfolgenden Beispiele aufzeigen.

ASCII-Darstellung	Bedeutung
432;1519818326846;3:109.8,100!3:109.5,31!5:-8.6	Maschine 432 meldet Messwerte um UTC Wed Feb 28 2018 11:45:26. Es werden drei Messwerte übertragen: **MesurementType** = 3, **Zeitstempel** = 1519818326846 + 0, **MeasurementValue** = 109.8 **MesurementType** = 3, **Zeitstempel** = 1519818326846 + 100, **MeasurementValue** = 109.5 **MesurementType** = 5, **Zeitstempel** = 1519818326846 + 100 + 31, **MeasurementValue** = -8.6
222;0;3:100,5:-34.3,5000!3:98.0	Maschine 222 meldet drei Messwerte zum gleichen Zeitpunkt. Da der InitialTimestamp 0 ist, wird auf Empfangsseite der Empfangszeitpunkt angenommen. **MesurementType** = 3, **Zeitstempel** = <Empfangszeit>, **MeasurementValue** = 100.0 **MesurementType** = 5, **Zeitstempel** = <Empfangszeit>, **MeasurementValue** = -34.3 **MesurementType** = 3, **Zeitstempel** = <Empfangszeit> + 5 Sekunden, **MeasurementValue** = 98.0

4.2 Diskussion

Die neue Schnittstelle soll die Probleme der alten Schnittstelle beheben und die Anforderungen besser erfüllen. Dieser Abschnitt diskutiert die Erfüllung der nichtfunktionalen Anforderungen, erklärt die getroffenen Design-Entscheide und differenziert die eingegangenen Tradeoffs. Weitere in Betracht gezogene, aber anschliessend verworfene Varianten sind in Abschnitt 4.3 zu finden.

4.2.1 NFR-01: Skalierbarkeit

Die neue Schnittstelle skaliert sowohl horizontal als auch vertikal. Durch die Aufteilung in Message Broker und Data Collector können beide Komponenten unabhängig voneinander skaliert werden. Falls ein weiterer Message Broker nötig wird, können die Maschinen mittels DNS Loadbalancing zufällig oder nach dem Round Robin-Prinzip auf die vorhandenen Message Broker verteilt werden.

Sollte die Message Queue auf dem einzelnen Message Broker nicht mehr genügend schnell. abgearbeitet werden können, dann kann eine weitere Data Collector-Instanz gestartet werden und die Queue kann somit schneller abgearbeitet werden.

Dadurch, dass die gesamte Kommunikation über einen Port läuft, ist bei der Inbetriebnahme einer neuen Maschine keine Anpassung mehr auf dem Data Collector notwendig. Die Maschine muss nur noch im Dashboard erfasst und dem Kunden zugewiesen werden.

4.2.2 NFR-02: Ressourcenschonende Client-Implementation

Ein wesentliches Designziel des MQTT-Standard ist es, auch Geräte zu unterstützen, welche über wenig Ressourcen verfügen. Eine Liste der verfügbaren Referenz-Implementationen ist unter (MQTT: Libraries, 2018) zu finden.

Der Ressourcenbedarf des notwendigen SSL/TLS- bzw. TCP-Layers ist nicht zu unterschätzen. Da jedoch jede Maschine genau eine gleichzeitige Verbindung offenhalten muss und Messwerte gebündelt in Paketen verschickt werden können, ist der dadurch bedingte CPU-Overhead verkraftbar.

4.2.3 NFR-03: Übertragungssicherheit

Die Übertragungssicherheit wird durch die Verwendung der korrekten QoS-Levels sichergestellt. Da MQTT keine klassische Message Queue implementiert, muss bei der Konfiguration des verwendeten Message Brokers darauf geachtet werden, dass die von den Maschinen gesendeten Messages auch tatsächlich gepuffert werden, sollte einmal kein Data Collector auf das /measurement_data Topic subscribed sein. Wenn jedoch die letzte Registrierung auf ein Topic mit CleanSession False und die Nachricht als QoS-Level 1 oder Level 2 versendet wurde, ist der Message Broker gemäss OASIS-Standard[5] Abschnitt 3.1.2.4 (Clean Session) für die sichere Aufbewahrung uns Auslieferung verantwortlich.

4.2.4 NFR-04: Datensicherheit

Durch die Verwendung von SSL/TLS mit Server und Client-Authentifizierung wird die Datensicherheit und die Integrität der Daten sichergestellt. Durch Clientzertifikate können «Man-in-the-Middle» Angriffe abgewendet werden. Das Betreiben der für die Clientzertifikate notwendigen Certificate Authority ist mit dem Auftraggeber abgesprochen und nicht Teil dieser Arbeit.

[5] (MQTT Version 3.1.1, 2014)

4.3 Alternativen

Die oben beschriebene Schnittstelle wurde bei einer Abklärung möglicher Varianten als die beste Alternative evaluiert. Nachfolgend werden die ebenfalls in Betracht gezogenen Varianten beschrieben und mit der gewählten Lösung verglichen. Eine gute Beschreibung möglicher Protokolle und Verfahren wurde von Andrew Foster[6] verfasst..

4.3.1 Eigene TCP/TLS Lösung

Mit einer eigenen Implementation (aufsetzend auf TLS/SSL und TCP) könnte die Performance und der Ressourcen-Verbrauch auf der Maschine so klein wie möglich gehalten werden. Allerdings müsste die Thematik des quittierten Versendens sowie eine Pufferung auf Seiten des Data Collectors auch selbst implementiert werden.

Nach Rücksprache mit dem Auftraggeber wurde klar, dass die Ressourcen der Maschine nicht so knapp sind bzw. sein werden, dass das allerletzte «Stück» Performance und Speicher gespart werden muss. Mit MQTT wird ein Standard eingesetzt, welcher sich Einfachheit und Ressourcenschonung auf die Fahne geschrieben hat.

4.3.2 Eigene UDP Lösung

Ein Ausbau der bestehenden, auf UDP basierten Lösung wurde relativ früh verworfen, da insbesondere NFR-03 (Übertragungssicherheit) und NFR-04 (Datensicherheit) nur schwer bzw. mit viel Aufwand abzudecken sind. Zudem durfte nach Absprache mit dem Auftraggeber ein TCP-Stack bzw. die Fähigkeit, TLS/SSL-Verbindungen aufzubauen, vorausgesetzt werden.

4.3.3 CoAP

Mit dem «Constrained Application Protocol» (CoAP)[7] wurde ein Standard geschaffen, welcher auf die Datenübertragung zwischen Geräten mit eingeschränkten Ressourcen und schlechten Übertragungskanälen ausgelegt ist. Unter anderem werden Multicasts und Device2Device-Kommunikation unterstützt. CoAP baut auf UDP auf und verwendet optional DTLS zur gesicherten Datenübertragung.

Die neue Schnittstelle könnte sowohl mit CoAP als auch mit MQTT gut implementiert werden. Da jedoch keine der expliziten Stärken von CoAP verwendet werden und MQTT insgesamt kleiner und einfacher aufgebaut ist, wurde diese Variante nicht weiterverfolgt. Zudem wurde dem Projektteam vom Auftraggeber der Einsatz von MQTT nahegelegt.

---

[6] (Foster, 2013)

[7] (Z. Shelby, ARM, K. Hartke, & C. Bormann, 2014)

4.3.4 AMPQ als Message Queue

Alternativ zu MQTT könnte auch das Advanced Message Queuing Protocol (AMQP)[8] als Protokoll zwischen Maschine und Data Collector eingesetzt werden. So unterstützt beispielsweise der Azure IoT Hub sowohl MQTT als auch AMQP[9]. Das «Advanced» in der Bezeichnung lässt jedoch erahnen, dass es sich hier um ein komplexeres Protokoll handelt. MQTT deckt die Anforderungen gut ab und ist sehr simpel gehalten, daher wurde nicht auf AMQP gesetzt.

4.3.5 Alternative Möglichkeiten zur Client-Authentifizierung

Um die Clients gegenüber dem Message Broker zu authentifizieren, gibt es von Seiten TLS bzw. MQTT mehrere Varianten:

1. Keine Authentifizierung
2. Benutzername/Passwort-basierte Authentifizierung
3. Authentifizierung mittels Client-Zertifikat

Variante 1 scheidet von Beginn an aus, da NFR-04 (Datensicherheit) vorschreibt, dass keine Datenmanipulation stattfinden darf. Wenn der Message Broker die Clients nicht authentifiziert, kann ein Angreifer ohne Probleme eine Maschine simulieren und so gefälschte Daten in das System einschleusen.

Zwischen Variante 2 und 3 spielen Skalierungsüberlegungen eine sehr wichtige Rolle. Variante 2 würde bedingen, dass der Message Broker eine Liste mit gültigen Benutzername/Passwort-Kombinationen pflegen und prüfen muss. Dies ist zwar mit einigen zehn bzw. hundert Clients noch vorstellbar, wenn aber die unter NFR-01 genannten Skalierungsanforderungen in Betracht gezogen werden, ist das keine gangbare Lösung. Man stelle sich vor, die Passwort/Benutzerliste würde kompromittiert: In diesem Fall müssten alle 10'000 Maschinen mit neuen Zugangsdaten ausgestattet werden.

Variante 3 bedingt, dass jede Maschine mit einem individuellen Client-Zertifikat ausgerüstet wird. Da jedoch schon in Variante 2 jede Maschine mit individuellen Zugangsdaten ausgestattet werden muss, ist der Mehraufwand vor allem im Aufbau einer Public Key Infrastructure (PKI), welche die Client-Zertifikate erstellt/signiert, zu suchen. Zudem muss ein Verfahren zum Ersetzen des Client-Zertifikates definiert werden bzw. wie allfällige als kompromittiert erklärte Client-Zertifikate erkannt werden. Dazu gibt es zwei Möglichkeiten: Einerseits das Führen einer Certificate Revocation List (CRL) und andererseits das Verwenden von Online Certificate Status Protocol (OCSP). Da der Message Broker als Einziger die Clients validiert, ist die simplere Variante einer CRL genügend.

[8] (AMQP v1.0 - Final, 2011)

[9] (Overview of the Azure IoT Hub service, 2018)

5 Umgang mit Legacy-Schnittstellen

Die bereits ausgelieferten und betriebenen Maschinen können nicht ohne Weiteres auf die neue Schnittstelle aktualisiert werden. Daher muss ein Legacy Data Collector implementiert werden, der noch die alte, UDP-basierte Schnittstelle unterstützt. Damit die Datenhaltung in der Messdatenbank so einfach wie möglich gehalten werden kann, werden die alten UDP-Pakete auf die neue Schnittstelle übertragen. Dabei werden die Informationselemente von der alten Schnittstelle auf die neue Schnittstelle wie folgt abgebildet:

Informationselement	Abbildung
MaschinenID	Ein Mapping-File auf dem Legacy Data Collector erfasst die Zuordnung zwischen UDP-Port und MaschinenID.
Zeitstempel	Der Eingangszeitpunkt des UDP-Paketes ist der Zeitstempel für den entsprechenden Messwert. Um die Netzwerkbelastung zu optimieren, werden die Messwerte vom Legacy Data Collector gebündelt (z.B. alle 10 Sekunden) an den neuen Data Collector geschickt.
MesasurementType	Wird aus dem UDP-Paket übernommen
MeasurementValue	Wird aus dem UDP-Paket übernommen
Client-Zertifikat	Werden auf dem Legacy Data Collector hinterlegt und gemäss Mapping-File übernommen bzw. der Legacy Data Collector weist sich gegenüber dem neuen Data Collector mit dem entsprechenden Client-Zertifikat aus.

6 Literaturverzeichnis

AMQP v1.0 - Final. (7. October 2011). Abgerufen am 7. März 2018 von Advanced Message Queuing Protocol: http://www.amqp.org/sites/amqp.org/files/amqp.pdf

ANS.1 encoding rules. (Juli 2002). Abgerufen am 7. März 2018 von International Telecommunications Union (ITU): https://www.itu.int/ITU-T/studygroups/com17/languages/X.691-0207.pdf

Foster, A. (November 2013). *Messaging Technologies for the Industrial Internet and the Internet of Things.* Abgerufen am 7. März 2018 von PrismTech: http://www.prismtech.com/sites/default/files/documents/MessagingComparsionNov2013USROW_vfinal.pdf

Message Queuing Telemetry Transport. (2018). Abgerufen am 7. März 2018 von MQTT: http://mqtt.org

MQTT Version 3.1.1. (29. October 2014). Abgerufen am 7. März 2018 von OASIS: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html

MQTT: Libraries. (12. Februar 2018). Abgerufen am 7. März 2018 von GitHub: https://github.com/mqtt/mqtt.github.io/wiki/libraries

Overview of the Azure IoT Hub service. (29. Januar 2018). Abgerufen am 7. März 2018 von Microsoft Azure: https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-what-is-iot-hub

Protocol Buffers. (2018). Abgerufen am 7. März 2018 von Google Developers: https://developers.google.com/protocol-buffers

Z. Shelby, ARM, K. Hartke, & C. Bormann. (Juni 2014). *The Constrained Application Protocol (CoAP).* Abgerufen am 7. März 2018 von IETF: https://tools.ietf.org/html/rfc7252

Datenbank-Evaluation

Projektmitglieder
Stähli Tobias
Vincenz Dumeni
Wälter Jonas

Änderungsgeschichte

Datum	Version	Änderung	Autor
23.02.2018	1.0	Erstellung des Dokuments	Dumeni Vincenz
07.03.2018	1.1	Review / orthografische Korrekturen	Tobias Stähli
07.03.2018	1.2	Review / orthografische Korrekturen	Jonas Wälter
04.04.2018	1.3	Anpassungen gemäss Meeting	Jonas Wälter

Inhaltsverzeichnis

1 Einführung

1.1 Beschreibung

Dieses Dokument beschreibt die Evaluation eines geeigneten Datenbankmodells für die Erfüllung der geforderten funktionalen und nichtfunktionalen Anforderungen.

1.2 Gültigkeitsbereich

Die Gültigkeit dieses Dokuments beschränkt sich auf die gesamte Projektdauer der Bachelorarbeit im Frühjahrssemester 2018. Allfällige spätere Anpassungen oder Ergänzungen im Dokument werden in der Änderungsgeschichte festgehalten.

1.3 Referenzen

Als Nachschlagewerk für unklare Begriffe sowie Abkürzungen sei an dieser Stelle auf das Glossar verwiesen, welches fortlaufend nachgeführt wird. In der nachfolgenden Tabelle sind alle weiteren referenzierten Dokumente und Links aufgelistet.

Thema	Referenz
Aufgabenstellung	OneDrive: \01_Aufgabenstellung\Aufgabenstellung.docx
Dokumentvorlage	OneDrive: \00_Administration\Template.docx
Glossar	OneDrive: \00_Administration\Glossar.docx
Projektplan	OneDrive: \02_Projektplan\Projektplan.docx
Anforderungsanalyse	OneDrive: \03_Anforderungsanalyse\ Anforderungsspezifikation.docx
Logo Suncar HK	http://www.suncar-hk.com/

2 Anforderungen

Die Applikation hat in erster Linie die Anforderung, Messdaten zu speichern. Es fallen sehr viele Messdaten an, welche über längere Zeit persistent gehalten und ausgewertet werden wollen. Zudem ist die Performance der Anfragen relevant. So sollen Abfragen mit steigender Menge an Daten nicht massiv verlangsamt werden.

Neben den Messdaten müssen zusätzlich Konfigurationsdaten sowie Metadaten gespeichert werden. Diese Datenmenge ist überschaubar und wird in einem geringen Rahmen bleiben.

3 Evaluation: Datenbanklösung

Wie in den Anforderungen sehr gut ersichtlich ist, bestehen zwei unterschiedliche Anforderungen bzw. zwei unterschiedliche Datencharaktere: einerseits die Messdaten und andererseits die Konfigurationsdaten der Applikation. Daher macht es Sinn, für diese zwei Probleme unterschiedliche Speichersysteme zu verwenden.

3.1 Messdaten

Die Messdaten sind Daten, welche in grosser Mengen gespeichert werden müssen. Dabei sind immer Datenpaare «Messwert & Messtyp» vorhanden, welche einer Maschine zugewiesen werden. Alle Auswertungen finden entweder zeitbasiert mit Aggregationen statt oder es wird der zuletzt gemessen Wert benötigt. Die Daten dürfen auch verzögert konsistent sein, weil keine Daten verändert, sondern nur geschrieben und gelesen werden.

3.1.1 NoSQL-Datenbanklösungen

Die genannten Anforderungen lassen sich nur sehr schwer mit einer konventionellen, relationalen SQL-Datenbank erfüllen. In den letzten Jahren konnten sich NoSQL-Datenbanklösungen für diese Art der Datenspeicherung etablieren. NoSQL-Systeme haben das Ziel, die Verfügbarkeit sowie Partitionstoleranz zu maximieren, während die Transaktionskonsistenz vernachlässigt werden kann.

3.1.2 CAP-Theorem

Das bekannte CAP-Theorem von Eric Brewer[1] in der nachfolgenden Abbildung zeigt auf, dass von den drei Eigenschaften Verfügbarkeit (Availability), Konsistenz (Consistency) und Ausfalltoleranz (Partition Tolerance) eine beliebige Datenbanklösung nur deren zwei gleichzeitig erfüllen kann. Weil die Messdaten nach dem Einfügen nicht mehr editiert werden müssen, kann auf die Eigenschaft «Konsistenz» verzichtet werden.

[1] (Grundlagen Cloud Computing: CAP-Theorem, 2011)

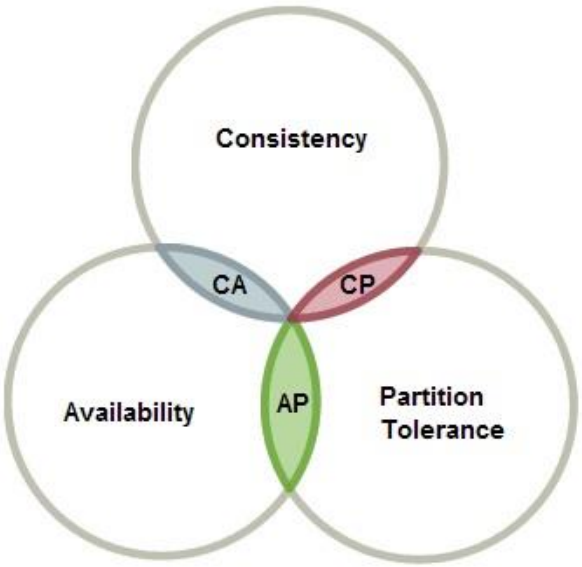


Abbildung 1: Das CAP Theorem zeigt die möglichen Eigenschaften von Datenbanken[2]

3.1.3 Time-Series-Datenbanken

Die Messdaten in dieser Problemstellung haben noch eine andere Eigenschaft: sie sind immer zeitorientiert. Unter diesem Umstand bietet es sich an, die Zeit als Index zu nutzen, um möglichst schnelle und effiziente Auswertungen zu ermöglichen. Genau für diesen Zweck wurden sogenannte zeitbasierte Datenbanken («Time-Series Database») entwickelt. Oft bringen solche Time-Series-Datenbanken bereits eingebaute Aggregationsmöglichkeiten mit sich, was das Handling der Daten massiv vereinfachen kann.

Da die durch Time-Series-Datenbanken zur Verfügung gestellte Funktionalität die Anforderungen der vorliegenden Problemstellung ideal abdecken kann, werden keine weiteren Datenbanklösungen detaillierter in Betracht gezogen. Stattdessen wird in einem weiteren Evaluationsschritt das zu verwendende Produkt unter den bestehenden Time-Series-Datenbanken evaluiert.

3.2 Konfigurationsdaten

Für die Speicherung der Konfigurationsdaten der Applikation (Meta- und Stammdaten) kann ohne Weiteres auf eine simple relationale Datenbank als weitverbreitetste und bewährte Methode zurückgegriffen werden.

[2] (CAP Theorem, 2014)

4 Evaluation: Produkt

4.1 Time-Series-Datenbank

Die Technik der zeitorientierten Speicherung von Daten ist nicht neu. So wurde der erste Release der RRDtool Datenbank[3], einer der Vorreiter von modernen Time-Series-Datenbanken, bereits im Jahr 1999 freigegeben. Für die Evaluation der für die vorhandene Problemstellung geeignetsten zeitserienbasierten Datenbanken auf dem Markt wurden die fünf bekanntesten und gemäss (Ranking of Time Series DBMS, 2018) mächtigsten Produkte unter die Lupe genommen, welche in der nachfolgenden Tabelle aufgeführt sind. Dabei wurde darauf geachtet, dass die Datenbanken als Open Source oder ohne Lizenzkosten verfügbar sind, um hohe Kosten zu vermeiden. Des Weiteren sollte die Datenbank auch lokal und nicht nur als Service in der Cloud verfügbar sein, um die Abhängigkeit zu einem bestimmten Cloud-Provider zu minimieren und die lokale Entwicklung zu vereinfachen.

Die Auswahl des Produktes wurde aufgrund eines Ausschlussverfahren getätigt. Dazu wurde die nachfolgende Tabelle um eine Zusammenfassung mit den wichtigsten Entscheidungsmerkmalen ergänzt.

Datenbank	Kurzbeschreibung
RRDtool	- Die Community scheint relativ inaktiv zu sein - Etablierte Datenbank mit sehr viel Erfahrung (seit 1999 auf dem Markt) - Keine direkte Anbindung zu .NET Core
Prometheus	- Ist stark auf die Administration und das Monitoring von Events und Messdaten in Serverlandschaften ausgerichtet - Der Fokus der Datenbank entspricht nicht ganz jenem der vorliegenden Problemstellung
InfluxDB	- Junge Open Source Datenbank - SQL-ähnliche Abfragen mit .NET Core Libraries - Aggregationsmöglichkeiten - Kostenpflichtige Clusterfunktionalität - Unterstützung von Alerting und Machine Learning Algorithmen - Wird aktiv weiterentwickelt und kommerziell durch InfluxData betrieben
Graphite	- Monitoring Tool mit einer integrierten Implementation einer Time-Series-Datenbank - Verfehlt das Ziel der vorliegenden Problemstellung klar
OpenTSDB	- Skalierbare Time-Series-Datenbank (Open Source) von Yahoo - Keine direkte Anbindung zu .NET Core - Kaum Entwicklungsaktivität (Sourcecode auf GitHub) seit Ende 2016

[3] (ANNOUCNE: RRDtool 1.0.0, 1999)

4.1.1 Entscheidung

Im Ausschlussverfahren der analysierten Datenbanken fiel der Entscheid schlussendlich auf das Produkt InfluxDB. Ausschlaggeben waren die bereits vorhandene Libraries für .NET Core, die innovative Gemeinschaft, welche eine Weiterentwicklung vorantreibt sowie auch die Features für Aggregationen, Alerting und die Möglichkeit, Algorithmen zu implementieren. Leider ist das Clustering der Datenbank nicht Open Source. Eine einzelne Instanz basiert jedoch auf der Open Source Version.

Des Weiteren hat auch die Performance überzeugt: Ein durchgeführter Test hat gezeigt, dass innerhalb von 30 Minuten mehr als 10 Millionen Datensätzen in eine InfluxDB in einem Docker-Container mit 1 GB RAM geschrieben werden können. Für die Laufzeit des Projekts reicht somit eine einfache Instanz der Datenbank aus. Sollte die Applikation zu einem späteren Zeitpunkt hochverfügbar werden, so kann die Datenbank auch als Service direkt bei InfluxData[4] bezogen werden. Wenn eine solch hohe Anzahl Transaktionen und die Hochverfügbarkeit später erforderlich werden, sollten die überschaubaren monatlich anfallenden Kosten kein Hindernis darstellen.

4.2 Relationale Datenbank

Die Evaluation einer relationalen Datenbank für die Speicherung der Konfigurationsdaten erübrigt sich, weil MariaDB als zu verwendendes Produkt durch den Auftraggeber vorgegeben wird. Durch den Einsatz von MariaDB (Open Source) kann die Plattformunabhängigkeit gewahrt, Kosten gespart und das bereits vorhandene Know-How bezüglich MySQL/MariaDB effizient eingesetzt werden.

5 Betrieb

5.1 InfluxDB

Der Betrieb der InfluxDB könnte als kostenpflichtiger Service direkt bei InfluxData bezogen werden. Diese Datenbanken sind clustered und hochverfügbar, jedoch relativ teuer im Vergleich mit einer eigenen Docker-Instanz bei einem Cloud Provider. Da vor allem während der Projektlaufzeit die Hochverfügbarkeit nicht notwendig ist, wurde entschieden, eine eigene Instanz in einem Docker-Container des Cloud Providers zu betreiben. Für die Entwicklung kann eine InfluxDB-Instanz auch als lokaler Docker-Container betrieben werden.

5.2 MariaDB

Die MariaDB wird von vielen Clouddienstleistern direkt als Service angeboten oder kann analog zur InfluxDB als Docker-Instanz in der Cloud oder lokal betrieben werden. Eine clustered Version ist zu diesem Zeitpunkt nicht notwendig, wird jedoch von MariaDB unterstützt, falls dies zu einem späteren Zeitpunkt notwendig sein sollte.

[4] (InfluxCloud Plans, 2018)

6 Literaturverzeichnis

ANNOUCNE: RRDtool 1.0.0. (16. Juli 1999). Abgerufen am 7. März 2018 von RRDtool: https://lists.oetiker.ch/pipermail/rrd-announce/1999-July/000007.html

CAP Theorem. (21. Februar 2014). Abgerufen am 7. März 2018 von IBM developerWorks: https://www.ibm.com/developerworks/community/blogs/IMSupport/resource/BLOGS_UPLOADED_IMAGES/CAP_Theorem.jpg

Grundlagen Cloud Computing: CAP-Theorem. (29. August 2011). Abgerufen am 4. April 2018 von codecentrix Blog: https://blog.codecentric.de/2011/08/grundlagen-cloud-computing-cap-theorem/

InfluxCloud Plans. (2018). Abgerufen am 7. März 2018 von influxdata: https://cloud.influxdata.com/plan-picker

Ranking of Time Series DBMS. (März 2018). Abgerufen am 7. März 2018 von DB-Engines: https://db-engines.com/en/ranking/time+series+dbms

Cloud-Evaluation

Projektmitglieder
Stähli Tobias
Vincenz Dumeni
Wälter Jonas

Änderungsgeschichte

Datum	Version	Änderung	Autor
23.02.2018	1.0	Erstellung des Dokuments	Dumeni Vincenz
08.03.2018	1.1	Review / orthografische Korrekturen	Jonas Wälter
04.04.2018	1.2	Review / orthografische Korrekturen	Tobias Stähli

Inhaltsverzeichnis

1 Einführung

1.1 Beschreibung

Dieses Dokument beschreibt die gewählte Cloudarchitektur für den Betrieb des Gesamtsystems sowie die Evaluation eines geeigneten Cloud Providers.

1.2 Gültigkeitsbereich

Die Gültigkeit dieses Dokuments beschränkt sich auf die gesamte Projektdauer der Bachelorarbeit im Frühjahrssemester 2018. Allfällige spätere Anpassungen oder Ergänzungen im Dokument werden in der Änderungsgeschichte festgehalten.

1.3 Referenzen

Als Nachschlagewerk für unklare Begriffe sowie Abkürzungen sei an dieser Stelle auf das Glossar verwiesen, welches fortlaufend nachgeführt wird. In der nachfolgenden Tabelle sind alle weiteren referenzierten Dokumente und Links aufgelistet.

Thema	Referenz
Aufgabenstellung	OneDrive: \01_Aufgabenstellung\Aufgabenstellung.docx
Dokumentvorlage	OneDrive: \00_Administration\Template.docx
Glossar	OneDrive: \00_Administration\Glossar.docx
Projektplan	OneDrive: \02_Projektplan\Projektplan.docx
Anforderungsspezifikation	OneDrive: \03_Anforderungsanalyse\ Anforderungsspezifikation.docx
Logo Suncar HK	http://www.suncar-hk.com/

2 Anforderungen

Vom Auftraggeber wurde gefordert, eine kosteneffiziente Betriebslösung zu evaluieren, welche herstellerunabhängig entwickelt werden kann. Dabei sollen die Skalierbarkeit und eine einfache Wartbarkeit nicht ausser Acht gelassen werden. Die Lösung soll für die aktuelle Problemstellung angemessen sein und die bestehenden Best Practices in der Entwicklung berücksichtigen. Die Teilsysteme dürfen auch in einer Cloud ausserhalb der Schweiz betrieben werden, müssen aber entsprechend den Sicherheitsvorgaben verschlüsselt kommunizieren.

2.1 Funktionale Anforderungen

Die nachfolgende Auflistung beinhaltet alle funktionalen Anforderungen an die Cloud-Architektur. Dabei beziehen sich die Anforderungen in erster Linie auf den Betrieb aller Komponenten des Gesamtsystems, welche in der Systemarchitektur enthalten sind.

- Betrieb der folgenden Komponenten:
  - MQTT Middleware (Message Queue)
  - Data Collector (ASP.NET Core Projekt)
  - Measurements DB (Time Series Datenbank)
  - Metadata DB (relationale Datenbank)
  - Backend (ASP.NET Core Web API)
  - Storage (Speicherung von Bildern)
  - Email Gateway
  - Frontend (React Projekt mit statischen Files)
- Automatisches Deployment
- Einfaches Erstellen von weitere Instanzen (z.B. Staging-Umgebung)

2.2 Nichtfunktionale Anforderungen

Die nichtfunktionalen Anforderungen an die Cloud-Infrastruktur können von den globalen nichtfunktionalen Anforderungen abgeleitet werden, welche der Anforderungsspezifikation entnommen werden können. Die nachfolgende Auflistung enthält die nichtfunktionalen Anforderungen an den Cloud Provider bzw. an das Betriebsmodell.

- Vertikale Skalierbarkeit
- Horizontale Skalierbarkeit, um Kosten effizient zu sparen
- Individuelle Skalierbarkeit der einzelnen Komponenten
- Verschlüsselte Kommunikation
- Kein öffentlicher Zugriff auf die Datenbanken
- «Cloud ready»-Systemarchitektur (keine Legacy Softwarearchitektur)
- Herstellerunabhängige Softwareentwicklung wo immer möglich
- Einfacher lokaler Betrieb der notwendigen Systemkomponenten, um eine effiziente Entwicklung zu ermöglichen

Um eine sinnvolle Skalierung für bis zu 10'000 Maschinen zu erreichen, ist eine horizontale Skalierung das ideale Werkzeug. Wenn alle Komponenten sauber in abgekoppelte Services aufgeteilt werden, können die einzelnen Komponenten bei Bedarf individuell skaliert werden. Eine solche Architektur führt dazu, dass jede Komponente «stateless» sein muss. Ob oder wann eine horizontale Skalierung wirklich notwendig ist, kann zu diesem Zeitpunkt nicht bestimmt werden. Mit einer horizontalen Skalierung können jedoch die Anforderungen sicher erreicht werden. Aus diesem Grund wird grossen Wert daraufgelegt, die Softwarearchitektur und das Betriebsmodell entsprechend zu evaluieren und zu entwickeln.

3 Lösungskonzept

Um die oben erwähnte Anforderungen abzudecken, gibt es unterschiedliche Lösungskonzepte. Damit die erforderlichen Skalierungseigenschaften erreicht werden können, ist es notwendig, die Komponenten containerbasiert zu betreiben. Eine containerbasierte Lösung entspricht den heutigen Anforderungen für eine moderne Softwarearchitektur und bietet die Möglichkeit einer horizontalen Skalierung. Die klassische Installation von Komponenten auf einem Server entspricht nicht mehr dem «State of the Art» der heutigen Betriebsarchitektur und erschwert die geforderte Skalierung.

3.1 Platform as a Service (PaaS)

Das PaaS-Konzept ist ein ideales Werkzeug, um Softwarekomponenten einfach, kostengünstig, ausfallsicher und skalierbar zu hosten. Bei einer PaaS-Umgebung wird eine Plattform durch den Cloud Provider zur Verfügung gestellt, wobei man sich nicht mehr um die Infrastruktur, das Patching des Betriebssystems und die Konfiguration von sonstiger Software kümmern muss. Der Cloud Provider übernimmt diese Aufgaben und betreibt die sogenannten «Apps» in einer hochverfügbaren, hochskalierbaren und meistens geteilten (shared) Umgebung.

3.2 Container as a Service (CaaS)

Mit CaaS bieten Cloud Dienstleister eine verwaltete (managed) Containerplattform an, auf welcher beliebige Container (meistens Docker) betrieben werden können. In der Regel sind dies dedizierte Umgebungen, bei welchen der Cloud Provider die Verantwortung für das Betriebssystem sowie die Containersoftware inklusive Load Balancing und weiteren Diensten übernimmt. Solche Lösungen sind oft erheblich teurer für eine geringe Anzahl an Containern, weil dafür mehrere dedizierte Server gestartet werden müssen, um die Ausfallsicherheit und alle Dienste zur Verfügung stellen zu können. Für das Deployment von eigene Container ist dazu noch eine Container-Registry notwendig, in welche die Container-Images von Entwicklern hochgeladen werden können.

3.1 Infrastructure as a Service (IaaS)

Bei «Infrastructure as a Service» übernimmt der Cloud Provider nur die Verantwortung über die Infrastruktur, wobei lediglich eine virtuelle Maschine zur Verfügung gestellt wird. In dieser virtuellen Maschine ist der Kunde völlig frei, muss sich jedoch auch selbst um das Betriebssystem (z.B. Patching) kümmern sowie natürlich um die benötigte Software.

3.2 Auswahl der idealsten Betriebsplattform

Die beschriebenen Angebotsklassen (PaaS, CaaS und IaaS) sind nicht offiziell spezifiziert bzw. standardisiert. Dieser Umstand hat zur Folge, dass jeder Clouddienstleister seine Services so aufbaut und anbietet, wie er dies möchte und damit nicht immer vollumfänglich einer Angebotsklasse entspricht. Daher können die Cloud-Lösungen von verschiedenen Anbietern nie auf eine einfache Art und Weise miteinander verglichen werden. Oft unterscheiden sich die Dienste nur in einem kleinen, aber entscheidenden Detail. Red Hat, der Weltmarktführer im Bereich Open Source, hat in einem Bericht die verschiedenen PaaS- und CaaS-Plattformen miteinander verglichen.

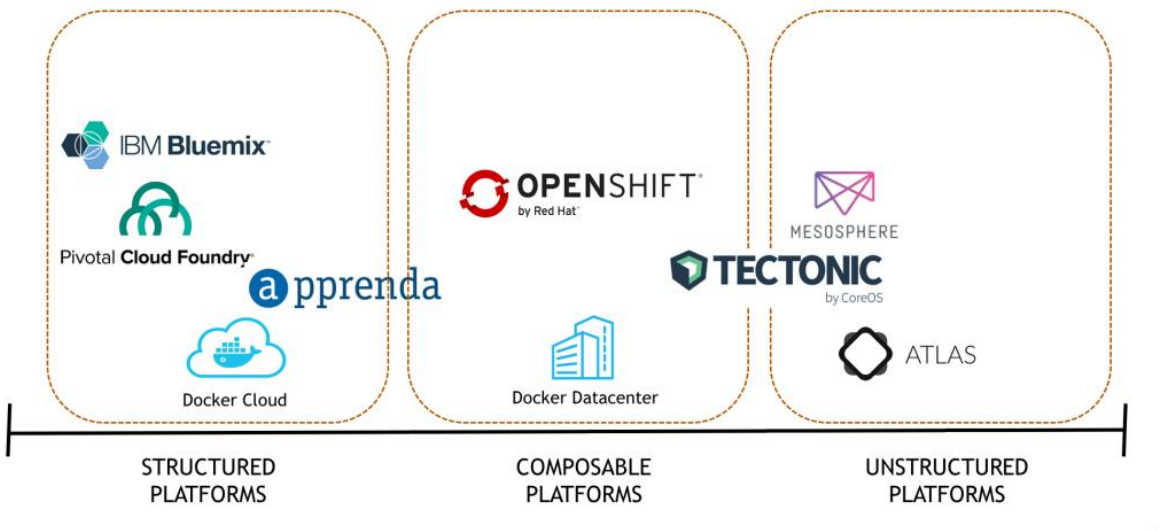


Abbildung 1: Vergleich von PaaS- und CaaS-Plattformen [1]

Für die vorliegende Problemstellung ist eine PaaS-Lösung der optimalste Ansatz. Als Softwareentwickler und Betreiber des Softwaresystems müssen weder das Patching des Betriebssystems noch die Lastverteilung oder sonstige Infrastruktur-Dienste verwaltet werden. Dazu können die entsprechenden «Web Apps» (Microsoft Azure, Amazon Web Services) oder «Apps» (Google Cloud) relativ preiswert betrieben und sehr gut skaliert werden.

4 Cloud Provider / Lösungsansätze

Wie bereits angesprochen, lassen sich die Dienstleistungen und Angebote der Cloud Provider untereinander nicht abschliessend vergleichen. Die unterschiedlichen Leistungen und Preismodelle sind die Hauptursache dieser Tatsache.

So wird beispielsweise bei Swisscom der Preis linear zu den benötigten Ressourcen skaliert, während bei Microsoft Azure entsprechende «Services» mit verschiedenen Möglichkeiten betreffend Festplattenspeicher, RAM, CPU-Leistung, etc. gekauft werden müssen. Bei Amazon

[1] (Comparing PaaS and Container Platforms, 2016)

Web Services (AWS) gelten die Preise für EC2 und S3, wobei zuerst herausgefunden werden muss, was dahintersteckt.

So fährt man beispielsweise mit ein bis zwei kleine Apps mit dem Preismodell von Swisscom viel besser als bei AWS oder Microsoft Azure. Wenn die Apps jedoch hochskaliert werden, explodieren die Kosten förmlich.

Des Weiteren erschweren Rabatte und Aktionen bei gewissen Clouddienstleistern den Preisvergleich zusätzlich. Wenn beispielsweise Ressourcen für ein Jahr reserviert werden (Gegenteil von dynamischen Ressourcen), gewährt ein Dienstleister Rabatte von bis zu 75%.

Wenn wie in der vorliegenden Problemstellung mehrere Komponenten deployt werden müssen, können die effektiven Kosten grundsätzlich erst nach dem Deployment einigermassen bestimmt werden. In einem früheren Stadium sind schlicht zu viele Variablen vorhanden, um die Cloud Provider abschliessend miteinander vergleichen zu können. Je nach Architektur können die Kosten schnell um den Faktor zehn oder auch zwanzig höher sein.

Aus diesem Grund erfolgt die Wahl des Cloud Providers in erster Linie nicht über den Preis, sondern über die verfügbare Funktionalität der Cloud-Umgebung. Dennoch wurde natürlich parallel dazu darauf geachtet, dass die Kosten nicht ausser Kontrolle geraten. So soll das Gesamtsystem während der Entwicklungszeit (minimale Last, keine Redundanz notwendig) mit Kosten von maximal CHF 50.– pro Monat betrieben werden können, was den aktuellen Kosten für den Serverbetrieb der Legacy-Applikation bei SUNCAR HK entspricht. Basierend auf dem Leistungskatalog der Cloud Provider kann überprüft werden, ob die Anforderungen wie gewünscht erfüllt werden können. Ob die einzelnen Komponenten und ihre Kommunikation untereinander tatsächlich wie gewünscht konfigurierbar ist und funktioniert, stellt sich dann jedoch erst bei einem tatsächlichen Testaufbau heraus.

Die nachfolgenden Unterkapitel enthalten daher die verschiedenen Ansätze, welche weiterverfolgt und getestet wurden.

4.1 Swisscom Cloud

Die folgenden Eigenschaften sprechen für den Einsatz der Swisscom Cloud, welche auf der Cloud-Plattform «Cloud Foundry» basiert:

- Schweizer Dienstleister
- Alle notwendigen Apps im Angebot (.NET Core, statische Website)
- «RabbitMQ» für Message Queuing vorhanden
- «MariaDB» als relationale Datenbank on Demand
- Eigene Services auf Docker-Container möglich (notwendig für InfluxDB)
- Preis für minimale Last: ca. CHF 40.– pro Monat

Ein Testaufbau hat gezeigt, dass die Systemarchitektur bis auf ein entscheidendes Detail funktioniert: Swisscom bietet leider keinen NFS-Speicher an, welcher als persistenter Speicher an einen Docker-Container gemountet werden kann. Um diesem Problem entgegenzuwirken, stellt Swisscom eine Anleitung zur Verfügung, wie ein NFS-Speicher eines anderen Dienstleisters eingebunden werden kann.

Im Zuge des Testaufbaus konnte jedoch kein kompatibler Anbieter mit einem NFS-Speicher on Demand gefunden werden, der sich in der Nähe der Rechenzentren von Swisscom (Zürich und

Bern) befindet. Die nächstgelegenen Rechenzentren (von AWS) liegen in Frankfurt. Der Umstand, dass CPU und RAM der InfluxDB in Zürich/Bern liegen und der NFS-Speicher der Datenbank in Frankfurt, führt leider zu extremen Leistungseinbussen bei der Datenbank-Performance.

Neben der Distanz führen vermutlich auch die Latenz wegen unterschiedlichen Netzwerkgeräten beider Dienstleister und die für diesen Fall nicht explizit ausgelegt Infrastruktur zu den Performance-Einbussen. Somit ist Swisscom als Cloud Provider nicht für die vorliegende Problemstellung geeignet und wird nicht weiterverfolgt.

4.2 OpenShift

OpenShift ist eine Container Application Plattform von Red Hat und verfügt über ähnliche Eigenschaften und Funktionalität wie Cloud Foundry. Beide Lösungen verfügen über eine ähnliche PaaS-Software und sind Open Source. Red Hat bietet OpenShift in ihren eigenen Rechenzentren als PaaS-Plattform an. Die Rechenzentren befinden sich jedoch leider ausschliesslich in der USA und nicht in Europa, was zu grossen Latenzzeiten führt.

Mit APPUiO bietet ein offizieller Schweizer Cloud Provider die gleiche Funktionalität wie Red Hat (OpenShift) an und dies in einem Rechenzentrum in der Schweiz. Mit CHF 159.– monatlich (exkl. dedizierter Speicher) liegen die Kosten jedoch weit ausserhalb der Vorgaben.

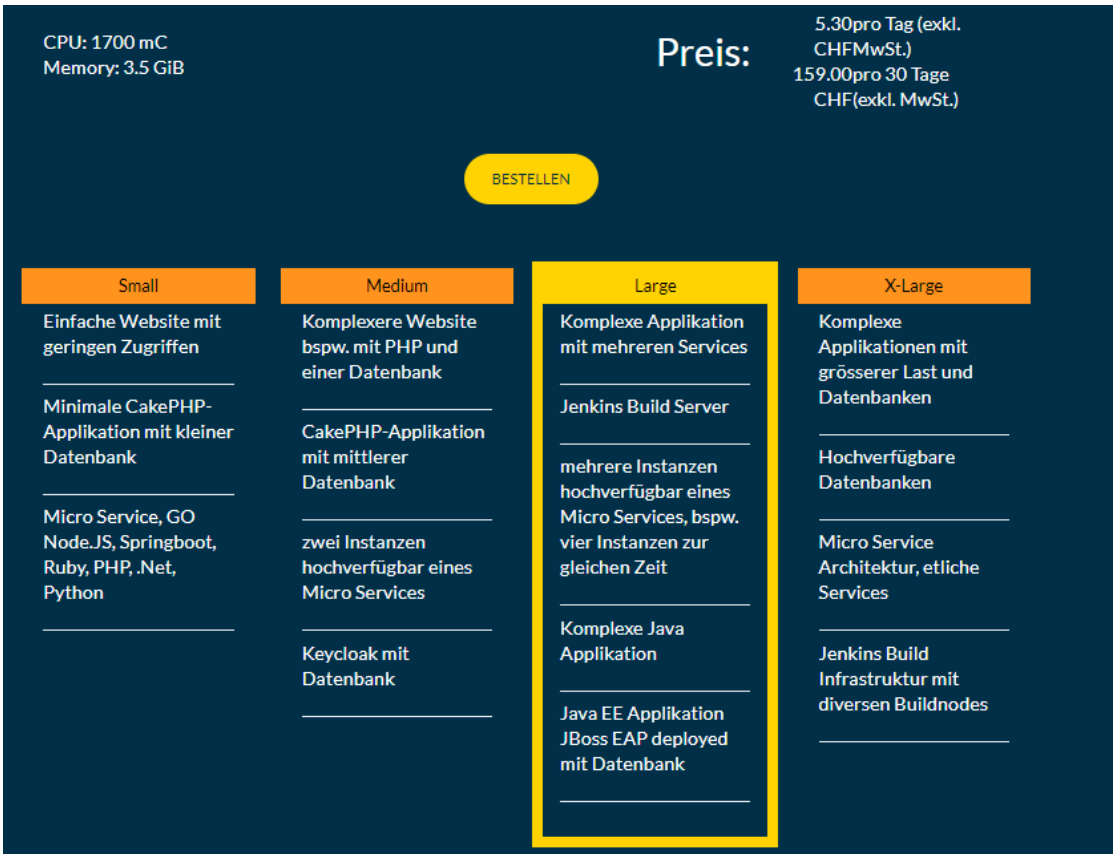


Abbildung 2: Kosten bei APPUiO[2]

[2] (Public Platform APPUiO, 2018)

4.3 Microsoft Azure

Microsoft Azure ist neben AWS und Google Marktführer im Cloud-Betrieb. Eine Analyse hat ergeben, dass alle drei Provider die Anforderungen des Projektes theoretisch erfüllen können. Da mit Visual Studio Team Services für die Continuous Integration/Deployment bereits ein Produkt von Microsoft eingesetzt wird, ist es naheliegend, mit Microsoft Azure jene Cloud-Umgebung ins Visier zu nehmen, welche am besten mit VSTS harmoniert.

In Microsoft Azure können das Backend und der Data Collector sehr einfach als Web App deployt werden, für das Frontend ist ebenfalls ein Web App oder sogar nur ein Blob Storage ausreichend. Für die von Microsoft Azure zur Verfügung gestellte MariaDB muss leider ein separater Sever gestartet (und bezahlt) werden, alternativ kann die Datenbank auch als «Custom»-Docker selbst implementiert werden. Des Weiteren muss auch InfluxDB als separater Server oder als Docker-Container betrieben werden, um Kosten zu sparen.

4.3.1 Docker

In Microsoft Azure stehen verschieden Möglichkeiten zur Verfügung, um ein Docker-Container zu betreiben. Die nachfolgende Tabelle enthält die untersuchten Ansätze inklusive Beschreibung und Diskussion. Neben den aufgelisteten, naheliegendsten Alternativen gäbe es noch etliche weitere Möglichkeiten.

Ansatz	Beschreibung/Diskussion
Docker als Web App	Die Einbindung eines persistenten Speichers ist nicht möglich bzw. der persistente Speicher kann nur in das Home-Verzeichnis vom Docker-Container gemappt werden, was nicht brauchbar ist.
Dedizierter Kubernetes-Cluster	Ein verwalteter Kubernetes-Cluster wird dediziert zur Verfügung gestellt (Container Service AKS). Um von der Ausfallsicherheit profitieren zu können, sind mindestens drei Hosts notwendig, was für die vorliegende Problemstellung absolut oversized ist. Wenn für das Deployment nicht die empfohlene Version, sondern kleinere Hosts verwendet werden, schlägt das Deployment des Clusters fehl. Zudem befindet sich dieses Angebot noch im Beta-Stadium (Preview) und kann nicht als stabil gewertet werden.
VM mit manueller Docker-Installation	Eine virtuelle Maschine ist die kostengünstigste Variante, wobei alle Apps als Docker-Container betrieben werden können. Die VM und Docker sind einfach zu installieren/konfigurieren und zu betreiben, es ist jedoch keine Ausfallsicherheit vorhanden.
Docker-Instanz	Dies ist sehr ähnlich aufgebaut wie eine Web App mit eigenem Docker, wobei hier mehr Konfigurationsmöglichkeiten zur Verfügung stehen (z.B. Mapping von persistentem NFS-Speicher). Dieses Angebot ist jedoch einiges teurer als beispielsweise eine VM mit denselben Ressourcen (CPU, RAM, Festplattenspeicher). Zudem ist diese Variante ebenfalls noch ein Preview mit fehlender Dokumentation/Beispiele.
Docker for Azure (CE)	Dieses Angebot beinhaltet eine VM mit vorinstalliertem Docker und einigen Tools (CLI, GUI), um die Docker-Containers zu administrieren. Da es sich dabei jedoch um eine ältere «Classic VM» handelt, ist sie nicht

	kompatibel mit den virtuellen Netzwerken, wodurch keine sinnvolle Kommunikation mit anderen Azure-Ressourcen möglich ist.
Docker EE for Azure	Bei dieser Variante steht ein Docker Swarm mit mindestens zwei Hosts für den Betrieb von Docker zur Verfügung als Alternative zur manuell installierten Docker-Version. Dieses Angebot entspricht damit der Docker Enterprise Edition.

4.3.2 Entscheidung

Ursprünglich war geplant, alle Komponenten als Web Apps und die InfluxDB-Instanz entweder ebenfalls als Web App oder als Docker-Container zu betreiben. Beim Testaufbau hat sich jedoch herausgestellt, dass dieser Ansatz so nicht implementiert werden kann, weil die Web Apps nicht an ein Virtual Network angeschlossen werden können. Für die Einbindung in ein Virtual Network müsste stattdessen «Enterprise Web Apps» genutzt werden, was eine dedizierte Umgebung mit Kosten von über CHF 1'000.– pro Monat voraussetzt. Aus Kostengründen ist dies kein gangbarer Weg. Somit hätte die Kommunikation zwischen Data Collector/Backend und den Datenbanken über das öffentliche Internet erfolgen müssen, was der nichtfunktionalen Anforderung NFR-05 (Datensicherheit) aus der Anforderungsspezifikation widerspricht.

Nach der Abwägung aller Alternative wurde unter Einbeziehung der Kosten entschieden, mindestens während der Projektphase eine virtuelle Maschine mit einem Ubuntu Server und einer manuellen Installation von Docker als Basis für das System einzusetzen. So bleiben die Kosten in einem überschaubaren Rahmen und die Installation gestaltet sich simpel.

Wenn zu einem späteren Zeitpunkt zunehmende Anforderungen betreffend Hochverfügbarkeit und automatische Skalierung aufkommen, könnten alle Docker-Container ohne grossen Mehraufwand beispielsweise in eine Kubernetes-Umgebung verschoben werden. In diesem Fall kann auch das Hosting der InfluxDB als Enterprise-Version direkt beim Hersteller InfluxData in Betracht gezogen werden.

Mit diesem Systemaufbau werden während der Entwicklungsphase einige Abstriche bezüglich Skalierbarkeit gemacht, um Kosten zu sparen. Durch die Nutzung von Docker-Container ist das System jedoch so ausgelegt, dass einer späteren vertikalen Skalierung nichts im Wege steht.

Neben Microsoft Azure gilt es stets, auch Amazon Web Services und Google Cloud als stärkste Alternativen in der Cloud-Branche zu berücksichtigen. Weil mit dem zuvor beschriebenen Systemaufbau von Microsoft Azure alle Anforderungen an einen Cloud Provider erfüllt werden können und der Evaluationsaufwand in einem zeitlich vernünftigen Rahmen gehalten werden soll, wurde auf eine detaillierte Analyse inklusive Testaufbau der weiteren Cloud Provider verzichtet. Mit dem Einsatz von Docker-Container könnte theoretisch zu einem späteren Zeitpunkt ohne Weiteres zu einem anderen Cloud Provider gewechselt werden.

4.4 Shared PaaS-Umgebung

Im Zuge der Evaluation hat sich gezeigt, dass «shared PaaS-Umgebungen» oft grössere Einschränkungen haben und/oder relativ teuer sind. Weitere Abklärungen haben ergeben, dass oft die Security der entscheidende Faktor für diesen Umstand ist. Standardmässig funktioniert Docker nicht ordnungsgemäss, wenn es nicht als Root gestartet wird. Wenn es dann einem Angreifer gelingt, aus Docker «auszubrechen», können sehr schnell Root-Rechte auf dem Host erlangt werden. Somit könnte ein Angreifer beispielsweise mit einem Zero-Day-Kernel-Exploit auf einer shared Docker-Umgebung relativ einfach Zugriff auf alle Instanzen auch von anderen Kunden erhalten.

4.5 Blob Storage für Frontend

Für das Frontend wurde ursprünglich ein Blob Storage angedacht, welcher extrem tiefe Kosten mit sich ziehen würde. Der Testaufbau hat dann jedoch einige Einschränkungen und Herausforderungen aufgezeigt. So liefert der Blob Storage Service beispielsweise nicht automatisch die Indexseite (index.html) beim Zugriff auf ein Verzeichnis. Des Weiteren können keine Fehlerseiten definiert werden, es wird lediglich eine entsprechende XML-Meldung vom Blob Storage zurückgeliefert. Um diese Probleme zu umgehen, wird das Frontend ebenfalls als Web App in einem eigenen Docker-Container instanziiert.

5 Cloud-Architektur

Diese Cloud-Evaluation hat zur Wahl von Microsoft Azure als Cloud Provider geführt. Bei einem Testaufbau konnten die vorgegebenen Anforderungen erfüllt werden. Für eine detaillierte Beschreibung der endgültigen Cloud-Architektur, welche auf Microsoft Azure betrieben wird, sei an dieser Stelle auf das separate Dokument «Continuous Integration/Continous Delivery» verwiesen.

6 Literaturverzeichnis

Comparing PaaS and Container Platforms. (2. Juli 2016). Abgerufen am 8. März 2018 von OPENSHIFT: https://www.openshift.com/container-platform/compare-openshift.html

Public Platform APPUiO. (2018). Abgerufen am 8. März 2018 von APPUiO: https://appuio.ch/public.html

Domainanalyse

Projektmitglieder
Stähli Tobias
Vincenz Dumeni
Wälter Jonas

Änderungsgeschichte

Datum	Version	Änderung	Autor
02.03.2018	1.0	Erstellung des Dokuments	Jonas Wälter
07.03.2018	1.1	Ergänzungen der Klassenspezifikation	Jonas Wälter
07.03.2018	1.2	Review / orthografische Korrekturen	Tobias Stähli
09.03.2018	1.3	Anpassungen gemäss Meeting	Jonas Wälter
11.04.2018	1.4	Ergänzung Konzept «SystemConfig»	Tobias Stähli
05.05.2018	1.5	Hinzufügen Benutzerrolle «Distributor»	Tobias Stähli
12.06.2018	1.6	Ergänzung Konzept «KapacitorTemplate»	Jonas Wälter

Inhaltsverzeichnis

1 Einführung

1.1 Beschreibung

Dieses Dokument beschreibt die Domainanalyse und liefert einen Überblick über alle Konzepte des Systems und deren gegenseitigen Beziehungen. Des Weiteren ist in diesem Dokument eine detaillierte Übersicht aller Berechtigungen enthalten.

1.2 Gültigkeitsbereich

Die Gültigkeit dieses Dokuments beschränkt sich auf die gesamte Projektdauer der Bachelorarbeit im Frühjahrssemester 2018. Allfällige spätere Anpassungen oder Ergänzungen im Dokument werden in der Änderungsgeschichte festgehalten.

1.3 Referenzen

Als Nachschlagewerk für unklare Begriffe sowie Abkürzungen sei an dieser Stelle auf das Glossar verwiesen, welches fortlaufend nachgeführt wird. In der nachfolgenden Tabelle sind alle weiteren referenzierten Dokumente und Links aufgelistet.

Thema	Referenz
Aufgabenstellung	OneDrive: \01_Aufgabenstellung\Aufgabenstellung.docx
Dokumentvorlage	OneDrive: \00_Administration\Template.docx
Glossar	OneDrive: \00_Administration\Glossar.docx
Projektplan	OneDrive: \02_Projektplan\Projektplan.docx
Anforderungsspezifikation	OneDrive: \03_Anforderungsanalyse\ Anforderungsspezifikation.docx
Personas	OneDrive: \03_Anforderungsanalyse\Personas.docx
Softwarearchitekturdokument	OneDrive: \04_Architektur_Design\ Softwarearchitekturdokument.docx
Logo Suncar HK	http://www.suncar-hk.com/

2 Domainmodell

In diesem Abschnitt werden die wichtigsten Konzepte im Gesamtsystem von «Suncar Big Data Collector & Dashboard» beschrieben sowie deren Beziehungen untereinander erläutert.

Für detailliertere Informationen zur dynamischen Interaktion und zur Erstellung der genannten Konzepte sei an dieser Stelle auf das Softwarearchitekturdokument verwiesen.

2.1 Strukturdiagramm

Das nachfolgende UML-Diagramm bietet eine Übersicht zu den Konzepten im System.

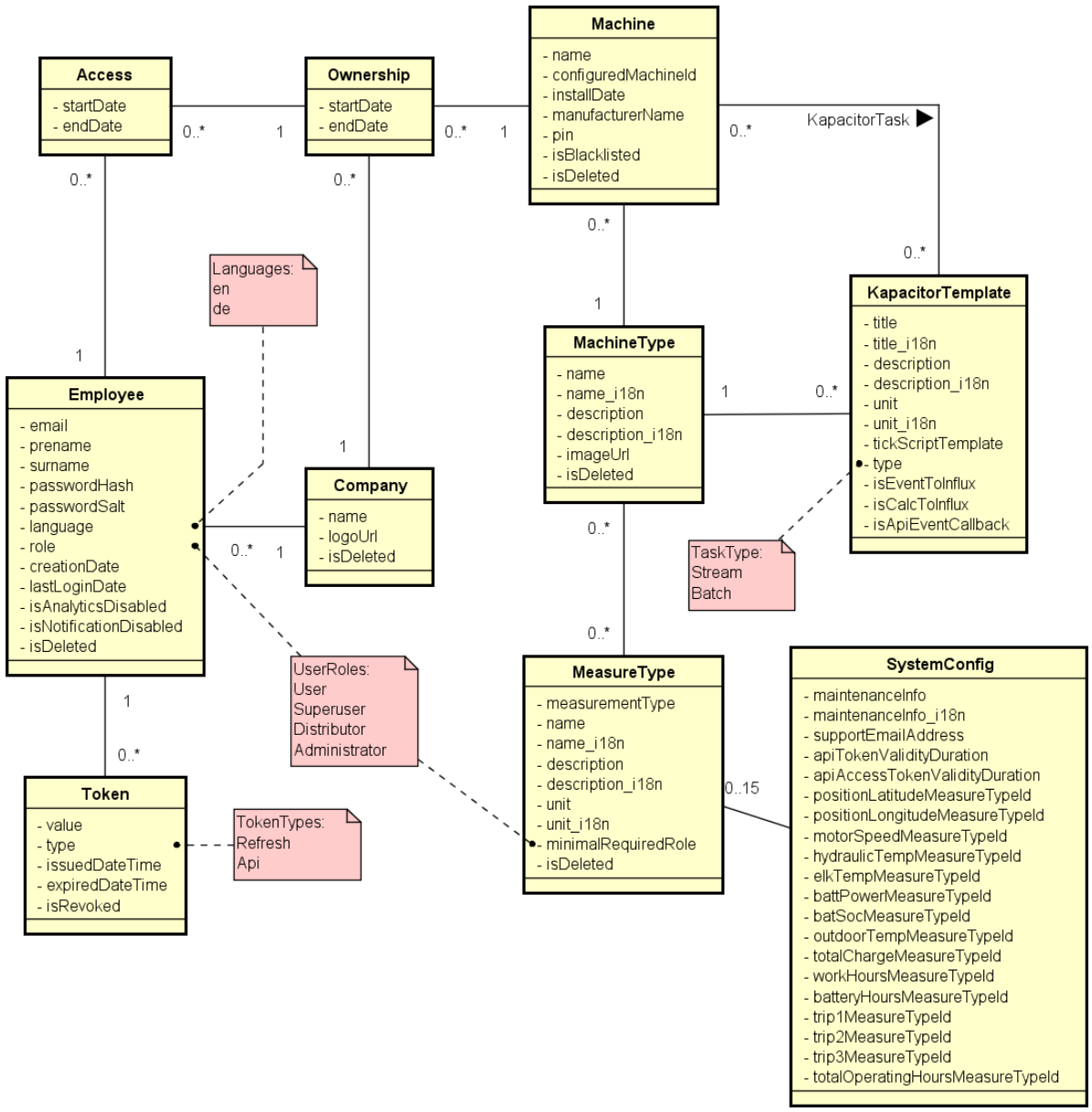


Abbildung 1: Strukturdiagramm – die wichtigsten Konzepte

2.2 Klassenspezifikation

In den nachfolgenden Unterkapiteln sind alle Konzepte aufgeführt und inklusive ihrer Attribute beschrieben.

Alle Konzepte von Stammdaten, welche die Grundinformationen des Systems beinhalten (Employee, Company, Machine, MachineType, MeasureType), wurden um ein Attribut **isDeleted** ergänzt. So wird bei einer zu löschenden Konzept-Instanz lediglich dieses Attribut auf «true» gesetzt, statt die Instanz tatsächlich zu löschen. Die restlichen Konzepte hingegen verfügen nicht über dieses zusätzliche Attribut und werden im Bedarfsfall tatsächlich gelöscht.

2.2.1 Employee

Dieses Konzept bezeichnet einen Mitarbeiter beziehungsweise dessen Useraccount im System. Wenn eine Person in der Realität Mitglied von mehreren Unternehmen (Companies) ist oder die Maschinen von mehreren Unternehmen sehen möchte, so erhält diese Person zwei Employee-Instanzen.

Attribut	Beschreibung
email	E-Mailadresse des Mitarbeiters
prename	Vorname des Mitarbeiters
surname	Nachnahme des Mitarbeiters
passwordHash	«Gehashtes» Passwort inklusive Salt des Mitarbeiters
passwordSalt	Zufälliger HMAC-Key, welcher im Hash-Algorithmus benötigt wird
language	Anzeigesprache des Frontends für den Mitarbeiter: en | de
role	Benutzerrolle: User | Superuser | Distributor | Administrator
creationDate	Zeitstempel der Erstellung des Useraccounts
lastLoginDate	Zeitstempel des letzten Logins
isAnalyticsDisabled	**false (Default):** Nutzerverhalten dieses Useraccounts wird mit Google Analytics erfasst **true:** Es wird kein personalisiertes Nutzerverhalten aufgezeichnet
isNotification-Disabled	**false (Default):** Mitarbeiter möchte E-Mail-Benachrichtigungen zu aufgetretenen Events erhalten **true:** Es werden keine E-Mail-Benachrichtigungen versendet
isDeleted	**false (Default):** Mitarbeiter existiert **true:** Mitarbeiter wurde gelöscht

2.2.2 Company

Dieses Konzept bezeichnet ein Unternehmen, welches als Kunde der SUNCAR HK AG im System erfasst ist.

Attribut	Beschreibung
name	Name des Unternehmens
logoUrl	Pfad zum Logo des Unternehmens auf dem Blob Storage
isDeleted	**false (Default):** Unternehmen existiert **true:** Unternehmen wurde gelöscht

2.2.3 Machine

Dieses Konzept bezeichnet eine Maschine (z.B. Elektrobagger), deren Messdaten im Frontend angezeigt werden können.

Attribut	Beschreibung
name	Bezeichnung der Maschine
configured-MachineId	Auf der Maschine konfigurierte ID, welche zur Identifikation der zugehörigen Messwerte benötigt wird
installDate	Installationsdatum der Maschine (gemäss ISO 15143-3)
manufacturerName	Hersteller der Maschine (gemäss ISO 15143-3)
pin	PIN/VIN der Maschine (gemäss ISO 15143-3)
isBlacklisted	**false (Default):** Maschine ist im Frontend sichtbar **true:** Maschine wird im Frontend nicht angezeigt
isDeleted	**false (Default):** Maschine existiert **true:** Maschine wurde gelöscht

2.2.4 Ownership

Dieses Konzept beschreibt den Besitz/Eigentum einer Maschine für ein Unternehmen (inklusive Zeitrahmen).

Attribut	Beschreibung
startDate	Datum, ab wann eine Maschine einer Company zugewiesen und im Frontend sichtbar ist
endDate	Datum, bis wann eine Maschine einer Company zugewiesen und im Frontend sichtbar ist (unbefristete Zuweisung bei fehlendem Datum)

2.2.5 Access

Dieses Konzept beschreibt den Zugriff eines Mitarbeiters auf einen Ownership und somit auf eine Maschine, welche dem Unternehmen des Mitarbeiters zugewiesen ist – inklusive Zeitrahmen. Ein Superuser benötigt keine Access-Instanz für den Zugriff auf eine Maschine, es gilt der Zeitrahmen des Ownerships. Ein Administrator hat Zugriff auf alle Maschinen – auch ohne Access-Instanz.

Attribut	Beschreibung
startDate	Datum, ab wann eine Maschine für einen Mitarbeiter sichtbar ist
endDate	Datum, bis wann eine Maschine für einen Mitarbeiter sichtbar ist (unbefristete Zuweisung bei fehlendem Datum)

2.2.6 MachineType

Dieses Konzept bezeichnet einen Maschinentyp, der von der SUNCAR HK AG hergestellt und angeboten wird.

Attribut	Beschreibung
name	Standard-Bezeichnung des Maschinentyps
name_i18n	Translation-Struktur (JSON) mit der Bezeichnung des Maschinentyps in allen verfügbaren Sprachen
description	Standard-Beschreibung des Maschinentyps
description_i18n	Translation-Struktur (JSON) mit der Beschreibung des Maschinentyps in allen verfügbaren Sprachen
imageUrl	Pfad zum Foto des Maschinentyps auf dem Blob Storage
isDeleted	**false (Default):** Maschinentyp existiert **true:** Maschinentyp wurde gelöscht

2.2.7 MeasureType

Dieses Konzept bezeichnet die Beschreibung eines in den Messdaten enthaltenen Sensortyps.

Attribut	Beschreibung
measurementType	Nummerischer Identifier, mit welchem dieser Messdatentyp von einer Maschine gemeldet wird
name	Standard-Bezeichnung des Messdatentyps
name_i18n	Translation-Struktur (JSON) mit der Bezeichnung des Messdatentyps in allen verfügbaren Sprachen
description	Standard-Beschreibung des Messdatentyps
description_i18n	Translation-Struktur (JSON) mit der Beschreibung des Messdatentyps in allen verfügbaren Sprachen
unit	Standard-Masseinheit des Messdatentyps
unit_i18n	Translation-Struktur (JSON) mit der Masseinheit des Messdatentyps in allen verfügbaren Sprachen
minimalRequired-Role	Minimal erforderliche Benutzerrolle, um auf diesen Messdatentyp zugreifen zu können
isDeleted	**false (Default):** Messdatentyp existiert **true:** Messdatentyp wurde gelöscht

2.2.8 Token

Dieses Konzept beschreibt einen Security-Token, welcher im Zusammenhang mit der Authentifizierung im Frontend oder bei der API für Drittanbieter benötigt wird.

Attribut	Beschreibung
value	Inhalt des Tokens
type	Einsatzzweck des Tokens: Refresh | Api
issuedDateTime	Zeitstempel der Ausgabe des Tokens
expiredDateTime	Zeitstempel, bis wann der Token gültig ist
isRevoked	**false (Default):** Token wurde noch nicht widerrufen **true:** Token wurde widerrufen

2.2.9 KapacitorTemplate

Dieses Konzept beschreibt eine Eventvorlage für die Datenverarbeitung mit Hilfe von Kapacitor und stellt damit die Grundlage für das Event-Alerting dar.

Attribut	Beschreibung
title	Standard-Titel der Eventvorlage
title_i18n	Translation-Struktur (JSON) mit dem Title der Eventvorlage in allen verfügbaren Sprachen
description	Standard-Beschreibung der Eventvorlage
description_i18n	Translation-Struktur (JSON) mit der Beschreibung der Eventvorlage in allen verfügbaren Sprachen
unit	Standard-Masseinheit der Eventvorlage
unit_i18n	Translation-Struktur (JSON) mit der Masseinheit der Eventvorlage in allen verfügbaren Sprachen
tickScriptTemplate	TICK-Skript zur Datenverarbeitung in Kapacitor
type	Art der Datenverarbeitung: Stream | Batch
isEventToInflux	**false (Default):** keine Aktion **true:** Beim Überschreiten eines Schwellwerts wird ein Event in der InfluxDB gespeichert
IsCalcToInflux	**false (Default):** keine Aktion **true:** Beim Überschreiten eines Schwellwerts wird ein Pseudo-Messwert in der InfluxDB gespeichert
IsCalcToInflux	**false (Default):** keine Aktion **true:** Beim Überschreiten eines Schwellwerts wird eine interne Backend-API zum Mailversand aufgerufen

2.2.10 SystemConfig

Dieses Konzept bezeichnet die Konfigurationsparameter für das System.

Attribut	Beschreibung
maintenanceInfo	Standard-Wartungsmeldung für das System
maintenanceInfo_i18n	Translation-Struktur (JSON) mit der Wartungsmeldung in allen verfügbaren Sprachen
supportEmailAddress	E-Mailadresse zum Empfang von Supportanfragen
apiTokenValidityDuration	Gültigkeitsdauer eines API Tokens (in Tagen)
apiAccessTokenValidityDuration	Gültigkeitsdauer eines API Access Tokens (in Minuten)
	Zuweisung der zu verwendenden Messdatentypen für folgenden Zwecke:
positionLatitudeMeasureType	Breitengrad (GPS)
positionLongitudeMeasureType	Längengrad (GPS)
motorSpeedMeasureType	Drehmoment
hydraulicTempMeasureType	Öl-Temperatur
elkTempMeasureType	Wassertemperatur
battPowerMeasureType	Batterie-Leistung
batSocMeasureType	Ladezustand
outdoorTempMeasureType	Aussentemperatur
totalChargeMeasureType	Gesamtakkuenergie
workHoursMeasureType	Arbeitsstunden
batteryHoursMeasureType	Batteriestunden
trip1MeasureType	Trip 1
trip2MeasureType	Trip 2
trip3MeasureType	Trip 3
totalOperatingHoursMeasureType	Total Betriebsstunden

2.3 Beziehungen

In der nachfolgenden Auflistung sind alle Konzept-Beziehungen aufgeführt und mit einer Beschreibung versehen.

Beziehung	Beschreibung
Employee – Company	Ein Employee ist Mitglied von genau einem Unternehmen, wobei ein Unternehmen beliebig viele Employees zugewiesen haben kann.
Token – Employee	Ein Token gehört genau einem Mitarbeiter, wobei ein Mitarbeiter über beliebig viele Tokens verfügen kann.
Company – Ownership – Machine	Ein Unternehmen kann beliebig viele Maschinen besitzen, wobei eine Maschine ebenfalls von beliebig vielen Unternehmen besitzt werden kann. Dieser Besitz (Ownership) wird zeitlich festgehalten (Start-/Enddatum).
Employee – Access – Ownership	Ein Mitarbeiter hat Zugriff (Access) auf beliebig viele Maschinen im Besitz seines Unternehmens (Ownership), wobei dies auch in die umgekehrte Richtung gilt. Innerhalb des Zeitrahmens des Ownerships kann der Zugriff für den Mitarbeiter zeitlich weiter eingegrenzt werden (Start-/Enddatum). Der effektive Zugriffs-Zeitrahmen ergibt sich von MAX(Ownership-Start, Access-Start) bis MIN(Ownership-End, Access-End).
Machine – MachineType	Eine Maschine kann genau einem Maschinentyp zugewiesen werden, wobei ein Maschinentyp beliebig viele zugewiesene Maschinen vorweisen kann.
MachineType – MeasureType	Ein Maschinentyp kann beliebig viele Messdatentypen zugewiesen erhalten, welche die Bedeutung der Sensortypen beschreiben. Umgekehrt kann ein Messdatentyp ebenfalls beliebig vielen Maschinentypen zugewiesen werden.
KapacitorTemplate – MachineType	Eine Eventvorlage kann für genau einen Maschinentyp erfasst werden, wobei ein Maschinentyp über beliebig viele Eventvorlagen verfügen kann.
Machine – Kapacitortemplate	Eine Eventvorlage kann für beliebig viele Maschinen des zugewiesenen Maschinentyps aktiviert werden, wodurch in Kapacitor ein Task erstellt wird. Umgekehrt können auch bei einer Maschine beliebig viele Eventvorlagen des zugewiesenen Maschinentyps aktiviert werden.
SystemConfig – MeasureType	Im Frontend werden bis zu 15 spezifische Messdatentypen angezeigt (z.B. GPS-Koordinaten oder Cockpit-Daten). In der SystemConfig wird festgehalten, welcher Messdatentyp welchem logischen Informationselement zugeordnet ist

3 Berechtigungsübersicht

Die verschiedenen Benutzerrollen (Attribut «role» im Konzept «Employee») haben einen grossen Einfluss auf die Benutzung des Systems. Die nachfolgende Tabelle enthält daher die unterschiedlichen Berechtigungen der Benutzerrollen, um die Zugriffsrechte klar abzugrenzen.

3.1 Employee-Verwaltung

CRUD	Aktion	User	Super	Dist	Admin
Create	• Neuer Mitarbeiter (Superuser/User) für eigene Company anlegen	✗	✓	✓	✓
	• Neuer Mitarbeiter (tiefere oder gleichberechtigte Rolle) für beliebige Company anlegen	✗	✗	✓	✓
Read	• Eigenes User-Profil ansehen	✓	✓	✓	✓
	• Profil von User der eigenen Company ansehen	✗	✓	✓	✓
	• Profil von beliebigem User ansehen	✗	✗	✓	✓
Update	• Eigenes User-Profil bearbeiten	✓	✓	✓	✓
	• Profil von User der eigenen Company bearbeiten	✗	✓	✓	✓
	• Profil von beliebigem User bearbeiten	✗	✗	✓	✓
Delete	• Eigener Useraccount löschen	✓	✓	✓	✓
	• Account von User der eigenen Company löschen	✗	✓	✓	✓
	• Beliebiger Useraccount löschen	✗	✗	✓	✓

3.2 Company-Verwaltung

CRUD	Aktion	User	Super	Dist	Admin
Create	• Neue Company anlegen	✗	✗	✓	✓
Read	• Eigene Company ansehen	✓	✓	✓	✓
	• Beliebige Company ansehen	✗	✗	✓	✓
Update	• Eigene Company bearbeiten	✗	✓	✓	✓
	• Beliebige Company bearbeiten	✗	✗	✓	✓
Delete	• Beliebige Company löschen	✗	✗	✓	✓

3.3 Machine-Verwaltung

CRUD	Aktion	User	Super	Dist	Admin
Create	• Neue Machine anlegen	✗	✗	✗	✓
Read	• Machine mit gültigem Access (für eigenen User) und Ownership (für eigene Company) ansehen im Zeitraum MAX(A.Start, O.Start) – MIN(A.End, O.End)	✓	✓	✓	✓
	• Machine mit gültigem Ownership (für eigene Company) ansehen im Zeitraum von Ownership	✗	✓	✓	✓
	• Beliebige Machine ansehen	✗	✗	✓	✓
	• Ausgeblendete Maschinen ansehen	✗	✗	✗	✓
Update	• Machine der eigenen Company bearbeiten (Attribut «name»)	✗	✓	✓	✓
	• Beliebige Machine bearbeiten	✗	✗	✗	✓
	• Machine ausblenden («isBlacklisted»)	✗	✗	✗	✓
Delete	• Beliebige Machine löschen	✗	✗	✗	✓

3.4 MachineType-Verwaltung

CRUD	Aktion	User	Super	Dist	Admin
Create	• Neuer MachineType anlegen	✗	✗	✗	✓
Read	• MachineType einer für mich sichtbaren Machine ansehen	✓	✓	✓	✓
	• Beliebiger MachineType ansehen	✗	✗	✗	✓
Update	• Beliebiger MachineType bearbeiten	✗	✗	✗	✓
Delete	• Beliebiger MachineType löschen	✗	✗	✗	✓

3.5 MeasureType-Verwaltung

CRUD	Aktion	User	Super	Dist	Admin
Create	• Neuer MeasureType anlegen	✗	✗	✗	✓
Read	• MeasureTypes einer für mich sichtbaren Machine ansehen (via MachineType)	✓	✓	✓	✓
	• Beliebiger MeasureType ansehen	✗	✗	✗	✓
Update	• Beliebiger MeasureType bearbeiten	✗	✗	✗	✓
Delete	• Beliebiger MeasureType löschen	✗	✗	✗	✓

3.6 Ownership-Verwaltung

CRUD	Aktion	User	Super	Dist	Admin
Create	• Neuer Ownership anlegen	✗	✗	✓	✓
Read	• Ownerships der eigenen Company ansehen	✓	✓	✓	✓
	• Beliebiger Ownership ansehen	✗	✗	✓	✓
Update	• Beliebiger Ownership bearbeiten	✗	✗	✓	✓
Delete	• Beliebiger Ownership löschen	✗	✗	✓	✓

3.7 Access-Verwaltung

CRUD	Aktion	User	Super	Dist	Admin
Create	• Neuer Access für User der eigenen Company anlegen	✗	✓	✓	✓
	• Neuer Access für beliebigen User anlegen	✗	✗	✓	✓
Read	• Eigene Accesses ansehen	✓	✓	✓	✓
	• Access von einem User der eigenen Company ansehen	✗	✓	✓	✓
	• Beliebiger Access ansehen	✗	✗	✓	✓
Update	• Access von einem User der eigenen Company bearbeiten	✗	✓	✓	✓
	• Beliebiger Access bearbeiten	✗	✗	✓	✓
Delete	• Access von einem User der eigenen Company löschen	✗	✓	✓	✓
	• Beliebiger Access löschen	✗	✗	✓	✓

3.8 KapacitorTemplate-Verwaltung

CRUD	Aktion	User	Super	Dist	Admin
Create	• Neue Eventvorlage anlegen	✗	✗	✗	✓
Read	• Eventvorlage von einem für mich sichtbaren MachineType ansehen	✓	✓	✓	✓
	• Beliebige Eventvorlage ansehen	✗	✗	✗	✓
Update	• Beliebige Eventvorlage bearbeiten	✗	✗	✗	✓
Delete	• Beliebige Eventvorlage löschen	✗	✗	✗	✓

3.9 SystemConfig-Verwaltung

CRUD	Aktion	User	Super	Dist	Admin
Read	• Systemkonfiguration abfragen	✓	✓	✓	✓
Update	• Systemkonfiguration bearbeiten	✗	✗	✗	✓

4 Datenspeicherung

4.1 Metadata-Datenbank

Basierend auf dem Domainmodell wurde mit Hilfe des Vorgehens «Entity Framework Code First» direkt das Datenbank-Schema für die MariaDB generiert, welches sich wie folgt präsentiert.

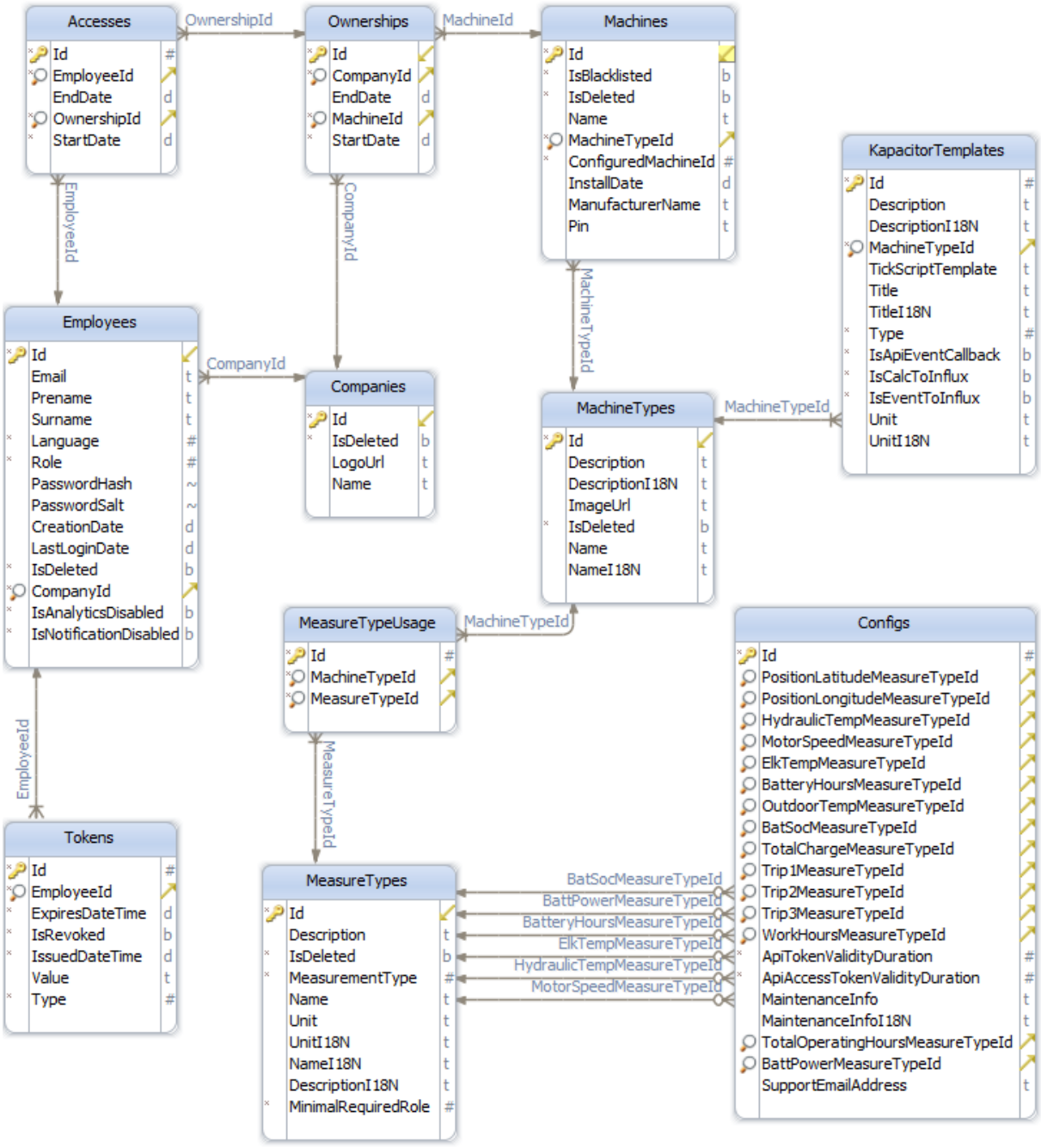


Abbildung 2: Datenmodell (MariaDB)

Aus Gründen der Übersichtlichkeit wurden einige Referenzen zwischen den Tabellen «Configs» und «MeasureTypes» ausgeblendet.

4.2 Measurements-Datenbank

Die Speicherung der Messdaten erfolgt in einer InfluxDB, einer schemalosen Time-Series-Datenbank. Eine InfluxDB besteht anstelle von Tabellen aus sogenannten «**Serien**», welche Messpaare mit unterschiedlichen Tags und Fields enthalten. Ein «**Tag**» ist dabei ein indexiertes Feld, während ein «**Field**» den Messwert enthält und nicht indexiert wird. Sowohl Tags als auch Fields können beliebige Zeichenfolgen sein. Dabei gilt es zu beachten, dass Gruppierungen oder Filterungen nur auf Tags durchgeführt werden können. Dem gegenüber können Aggregationen (z.B. Durchschnitt der Messwerte über eine bestimmte Zeitdauer) nur auf Fields angewendet werden. Als primärer Index gilt immer der **Zeitstempel**, es gibt keinen Eintrag in der Datenbank ohne Zeitangabe.

Für die vorliegende Problemstellung reicht eine einzige Serie «measurement_data» mit allen Messwerten (als Fields) und den dazugehörenden Zeitstempeln (als primärer Index) aus. Für die Identifikation der betroffenen Maschine wird die konfigurierte Kennzahl der Maschine als Tag mitgespeichert. Die Unterscheidung der verschiedenen Messtypen (Measure Types) erfolgt mit Hilfe der entsprechenden Fields. Die nachfolgende Tabelle beschreibt das Schema eines Eintrags in der Serie «measurement_data» mit den zugehörigen Tags und Fields.

Tabelle 1: Schema der Serie «measurement_data»

Property	Beschreibung	Beispiel
time [timestamp]	Zeitstempel der Messung als Unix Timestamp (in Nanosekunden)	1522526278003000000
machine_id [int]	Kennzahl der Maschine als Tag	1
measure_type_1 [double]	Messwert-Field, wenn es sich dabei um den Messtyp 1 handelt	109.6
measure_type_2 [double]	Messwert-Field, wenn es sich dabei um den Messtyp 2 handelt	-5.4
...	...	...
measure_type_N [double]	Messwert-Field, wenn es sich dabei um den Messtyp N handelt	0.75

Softwarearchitekturdokument

Projektmitglieder
Stähli Tobias
Vincenz Dumeni
Wälter Jonas

Änderungsgeschichte

Datum	Version	Änderung	Autor
02.03.2018	1.0	Erstellung des Dokuments	Jonas Wälter
07.05.2018	1.1	Dokumentation Backend	Jonas Wälter
03.06.2018	1.2	Dokumentation Frontend	Jonas Wälter
07.06.2018	1.3	Hinzufügen von 12-Factor Apps	Tobias Stähli
08.06.2018	1.4	Dokumentation Sicherheitsmassnahmen	Tobias Stähli
12.06.2018	1.5	Div. Ergänzungen im Dokument	Dumeni Vincenz
13.06.2018	1.6	Review Dokument	Tobias Stähli

Inhaltsverzeichnis

1 Einführung

1.1 Beschreibung

Das Softwarearchitekturdokument beschreibt die Architektur des Gesamtsystems inklusive aller wichtigen Design-Entscheide. Dabei sind eine Systemübersicht, die detaillierte Beschreibung der einzelnen Module, die ergriffenen Massnahmen zur Wahrung der Sicherheit der Daten und der Applikation, Informationen zur Verwendung von Prozessen und Threads sowie eine Analyse der Applikation anhand des «The Twelve-Factor App»-Modells enthalten.

1.2 Gültigkeitsbereich

Die Gültigkeit dieses Dokuments beschränkt sich auf die gesamte Projektdauer der Bachelorarbeit im Frühjahrssemester 2018. Allfällige spätere Anpassungen oder Ergänzungen im Dokument werden in der Änderungsgeschichte festgehalten.

1.3 Referenzen

Als Nachschlagewerk für unklare Begriffe sowie Abkürzungen sei an dieser Stelle auf das Glossar verwiesen, welches fortlaufend nachgeführt wird. In der nachfolgenden Tabelle sind alle weiteren referenzierten Dokumente und Links aufgelistet.

Thema	Referenz
Aufgabenstellung	OneDrive: \01_Aufgabenstellung\Aufgabenstellung.docx
Dokumentvorlage	OneDrive: \00_Administration\Template.docx
Glossar	OneDrive: \00_Administration\Glossar.docx
Projektplan	OneDrive: \02_Projektplan\Projektplan.docx
Anforderungsspezifikation	OneDrive: \03_Anforderungsanalyse\ Anforderungsspezifikation.docx
Domainanalyse	OneDrive: \04_Architektur_Design\ Domainanalyse.docx
Schnittstelle nach ISO 15143-3	OneDrive: \03_Anforderungsanalyse\ Schnittstelle nach ISO 15143-3.docx
Schnittstellenkonzept	OneDrive: \04_Architektur_Design\ Schnittstellenkonzept.docx
Logo Suncar HK	http://www.suncar-hk.com/

2 Systemübersicht

Das Gesamtsystem vom «Suncar Big Data Collector & Dashboard» wurde basierend auf der Aufgabenstellung, den Erkenntnissen der Anforderungsanalyse sowie der Domainanalyse entworfen. Die nachfolgende Abbildung zeigt eine Übersicht zur Architektur des Gesamtsystems inklusive der Zusammenarbeit aller Teilsysteme.

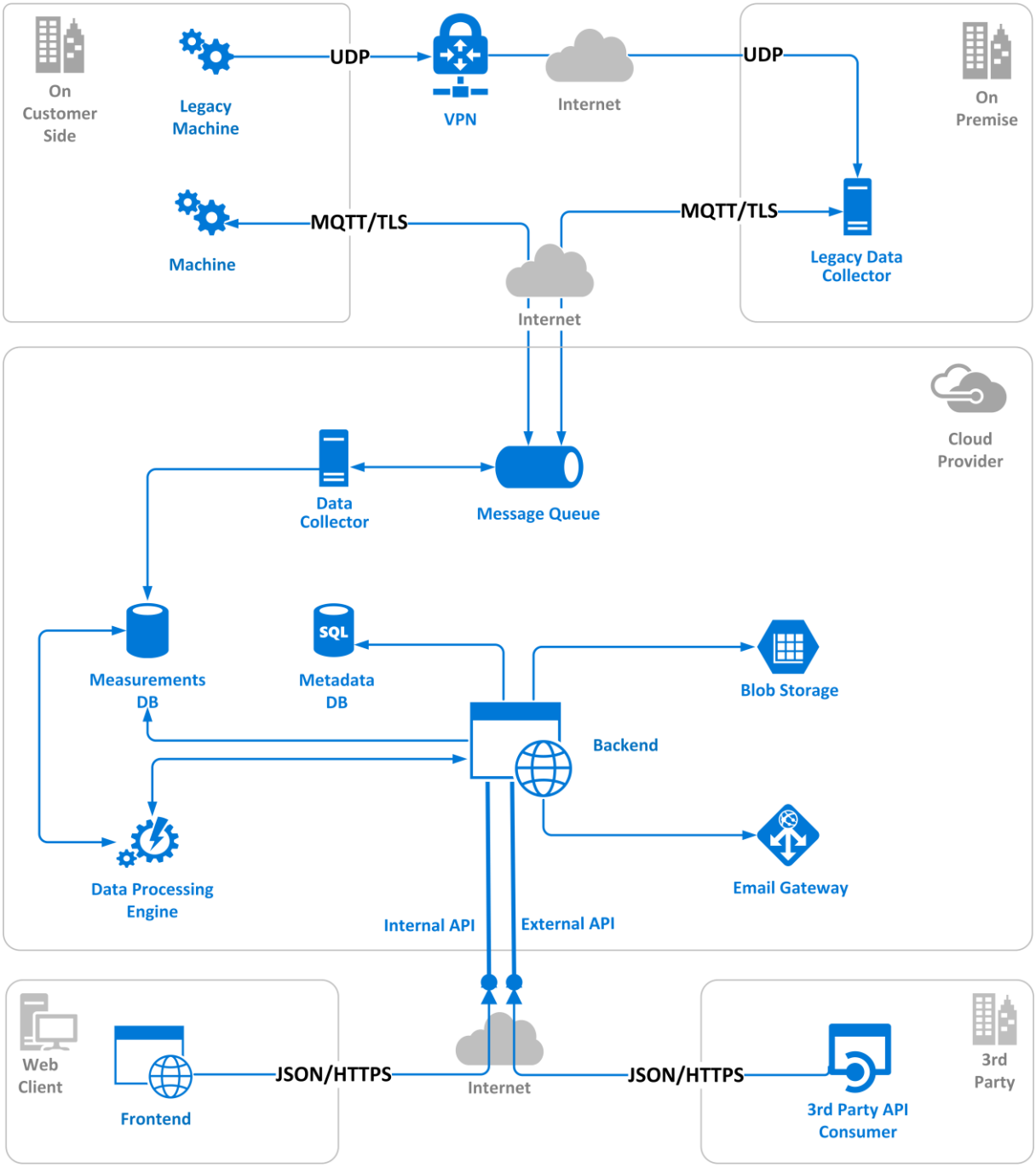


Abbildung 1: Systemübersicht «Suncar Big Data Collector & Dashboard»

Die folgende Tabelle enthält die einzelnen Komponenten des Gesamtsystems inklusive einer kurzen Beschreibung.

Komponente	Beschreibung
Legacy Machine	Eine bereits ausgelieferte, in Betrieb stehende Maschine, welche nicht ohne Weiteres aktualisiert werden kann
Machine	Eine für das neue System konfigurierte Maschine
Legacy Data Collector	Empfang der Messdaten von einer **Legacy Machine** und Weiterleitung an den Data Collector
Message Queue	Bidirektionale Pufferung der Messdaten und Zeitstempel
Data Collector	Empfang der Messdaten einer **Machine** oder via **Legacy Data Collector** und Persistierung in der Measurements DB
Measurements DB	InfluxDB zur Speicherung der Messdaten (Time Series-Datenbanklösung)
Metadata DB	MariaDB zur Speicherung der Meta- und Stammdaten (relationale Datenbanklösung)
Backend	Backend-Komponente, welche die Datenbasis für das Frontend sowie die External API zur Verfügung stellt.
Blob Storage	Speicherung der Bilder für Maschinentypen und Firmen
Email Gateway	Versand von E-Mails für die Applikation
Data Processing Engine	Komponente zur Datenverarbeitung und -analyse (Kapacitor)
Frontend	Grafische Benutzerschnittstelle für die Endkunden
3rd Party API Consumer	Applikationen von Dritten, welcher über eine externe API auf das System zugreifen

3 Logische Architektur

Das entwickelte System lässt sich grob in die Teilbereiche «Backend» und «Frontend» zerlegen. Das Backend setzt sich dabei aus dem Data Collector und der Web API mit jeweiligen Tests zusammen. Neben dem Frontend existieren noch die End-to-End Tests, welche das Gesamtsystem automatisiert testen. In den nachfolgenden Abschnitten werden die verschiedenen Bereiche des Gesamtsystems detailliert beschrieben.

3.1 Backend und Data Collector

Die logische Struktur des Backends und des Data Collectors besteht aus verschiedenen Projekten, welche in die drei Bereiche «backend», «collector» und «shared» aufgeteilt wurden.

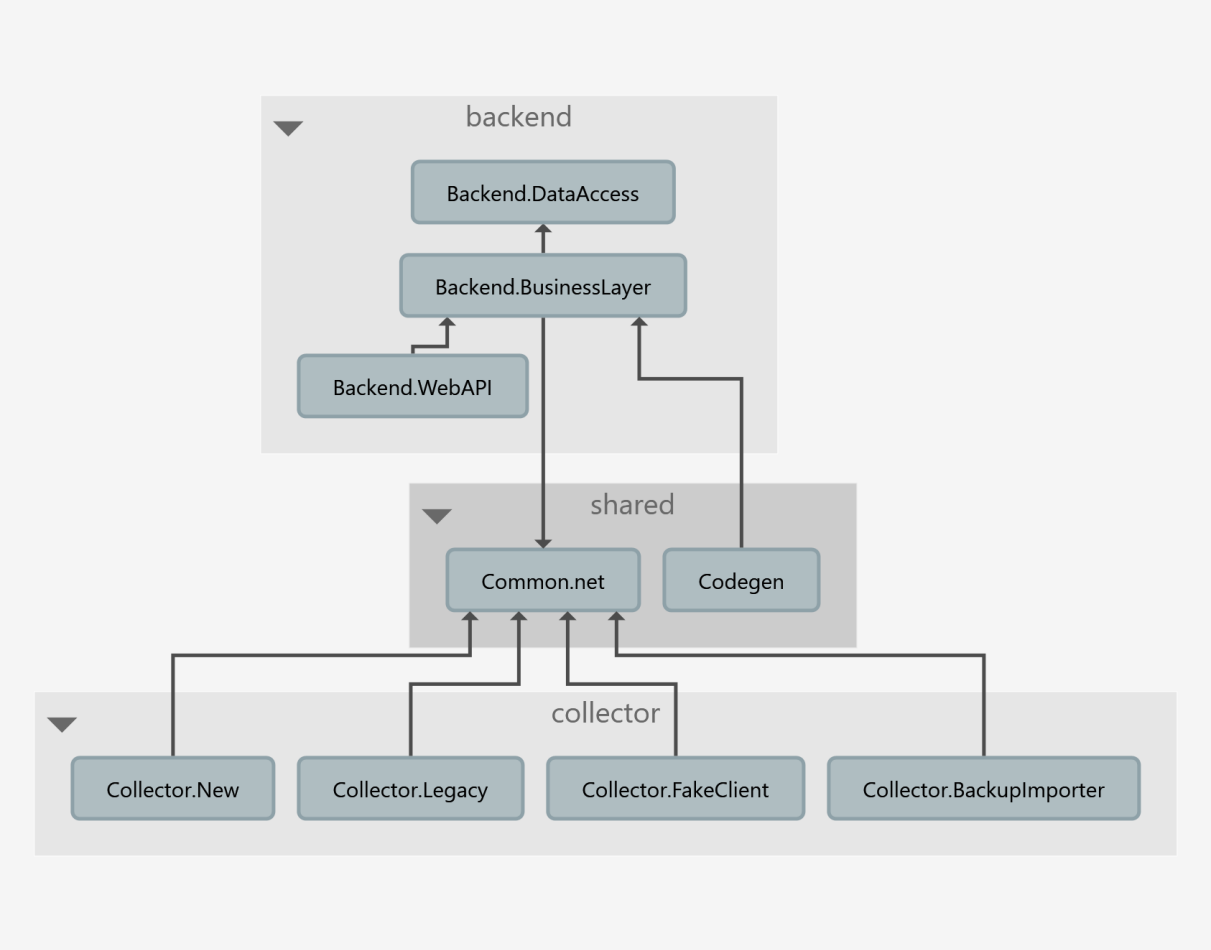


Abbildung 2: Architekturdiagramm «Backend» und «Data Collector»

Wie dem Architekturdiagramm entnommen werden kann, konnten einige Funktionen, welche sowohl im «backend» als auch im «collector» benötigt werden, in den Bereich «shared» ausgelagert werden (Projekt «Common.net»).

3.1.1 Backend.DataAccess

Im Projekt Backend.DataAccess erfolgt die Abwicklung des Datenbankzugriffs mit Hilfe von Microsoft Entity Framework Core.

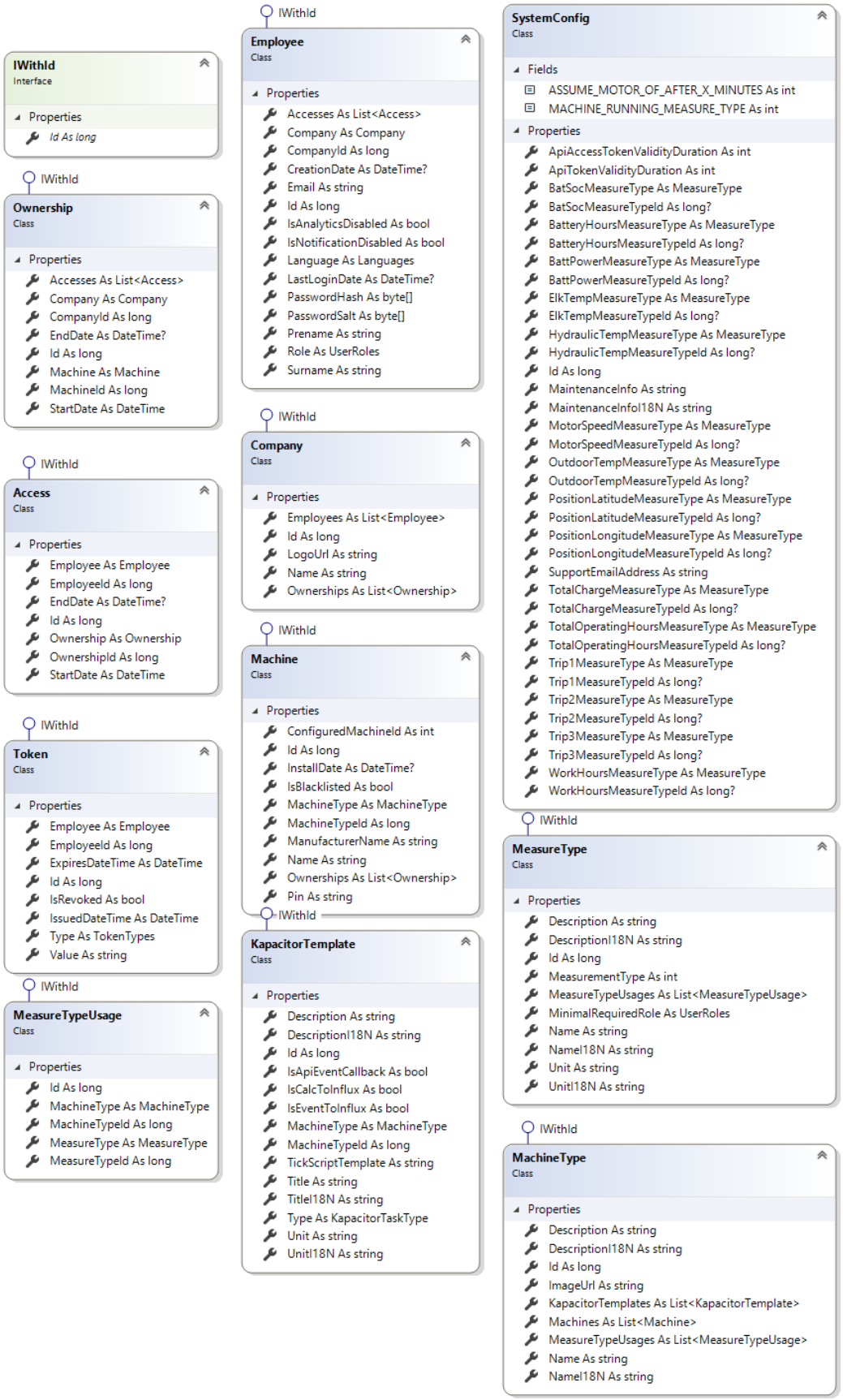


Abbildung 3: Klassendiagramm «Backend.DataAccess» (Entities)

Wie dem ersten Teil des Klassendiagramm entnommen werden kann, sind in diesem Projekt primär alle Entities angesiedelt, welche bereits in der Domainanalyse beschrieben wurden. Durch das Vorgehen «Entity Framework Code First»[1] wird die Datenbank inklusive Tabellen direkt aus den im Code definierten Entities generiert, wobei spätere Änderungen an den Entities mittels Migrationen in die Datenbank übernommen werden.

Alle Entities ausser «SystemConfig» implementieren das Interface «IWithId» und verfügen somit über eine Property «Id», was die Voraussetzung für den Einsatz des generischen CrudServices (Projekt «Backend.BusinessLayer») darstellt. Einige Entities verfügen zudem über Enum-Properties, welche ebenfalls in diesem Projekt enthalten sind und im zweiten Teil des Klassendiagramms aufgeführt sind.

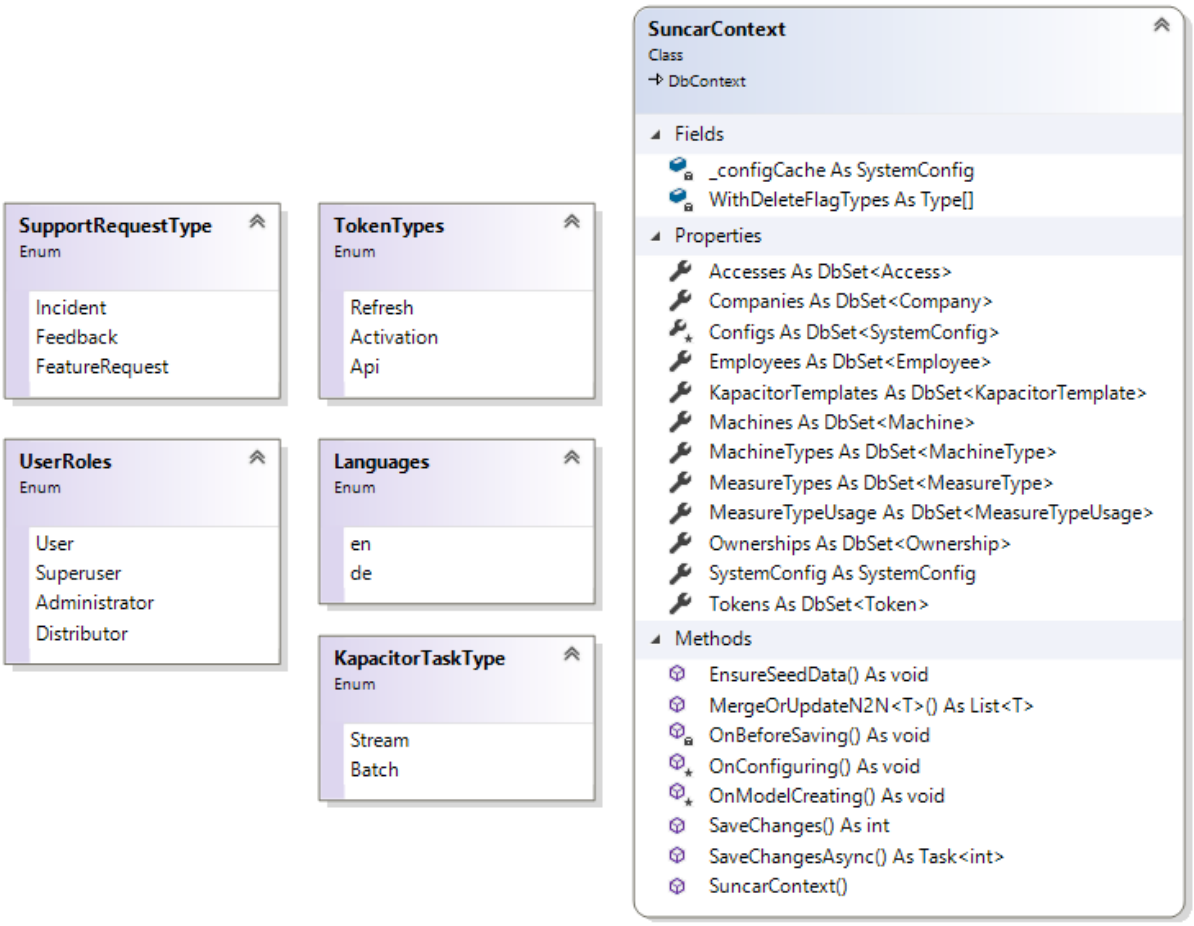


Abbildung 4: Klassendiagramm «Backend.DataAccess» (Enums, Context)

Eine weitere wichtige Komponente in diesem Projekt ist der Entity Framework Context «SuncarContext», welcher die folgenden Aufgaben erledigt:

Anders als alle anderen Entities wird die «SystemConfig» vom Context nicht als DBSet, sondern als einfache Property zur Verfügung gestellt und mit einem Caching ergänzt. Der Grund dafür

[1] (What is Code-First?, 2018)

ist, dass immer nur eine gültige «SystemConfig» vorhanden sein darf, diese nur sehr selten geändert wird und sie bei der Vorbereitung von InfluxDB-Aufrufen stets benötigt wird.

Ferner sorgt der SuncarContext für die Umsetzung des «Soft Delete»-Ansatzes: Im Gegensatz zu den Konzepten in der Domainanalyse fehlen in den Entities das isDeleted-Flag. Stattdessen enthält das Field «WithDeleteFlagTypes» alle Entities, welche über das isDeleted-Flag verfügen sollen. In der Methode «OnModelCreating» werden die entsprechenden Entities mit dem ModelBuilder um diese Property und einen Query Filter ergänzt. Der Query Filter[2] sorgt dafür, dass bei beliebigen SELECT-Abfragen nur noch Datensätze zurückgeliefert werden, welche nicht als gelöscht markiert sind. Zusätzlich wird in der Methode «OnBeforeSaving» dafür gesorgt, dass bei DELETE-Anfragen der entsprechenden Entities keine Datensätze gelöscht werden, sondern lediglich das isDeleted-Flag gesetzt wird.[3]

Da Entity Framework Core in der verwendeten Version 2.0 noch keine Unterstützung für Many-to-Many-Beziehungen enthält, wurde die Helpermethode «MergeOrUpdateN2N» hinzugefügt. Diese Methode vereinfacht die Verwaltung der für n-zu-n-Beziehung nötigen Zwischentabelle.

Abschliessend wird mit der Methode «EnsureSeedData» sichergestellt, dass beim ersten Starten des Backends initiale Datensätze (Employee, Company, MeasureTypes, MachineType, Machine und SystemConfig) erstellt werden, falls noch keine existieren.Backend.BusinessLayer

Das Projekt Backend.BusinessLayer enthält die gesamte Geschäftslogik in Form von verschiedenen Services. Des Weiteren ist hier das Entity-Datentransferobjekt-Mapping angesiedelt, welches mit Hilfe von AutoMapper[4] erfolgt.

3.1.1.1 Datentransferobjekte

Einen grossen Teil aller Klassen in diesem Projekt bezeichnen die verschiedenen Datentransferobjekte der Entities. Dabei können für jedes Entity bis zu vier verschiedene Datentransferobjekte existieren, welche die folgenden Einsatzzwecke haben:

DTO	Beschreibung und Einsatzzweck
…BaseDto	Basisklasse mit Properties, welche in allen vorhandenen Unterklassen benötigt werden
…OverviewDto	Unterklasse mit den wichtigsten Properties, welche grundsätzlich bei API-Aufrufen für GetAll, Add und Update zurückgeliefert werden
…DetailDto	Unterklassen mit weiteren und detaillierteren Properties, welche grundsätzlich bei API-Aufrufen für Get zurückgeliefert werden
…EditDto	Unterklasse mit den für die Bearbeitung eines Entity notwendigen Properties für die API-Aufrufe Add und Update

[2] (Global Query Filters, 2017)

[3] (Entity Framework Core: Soft Delete using Query Filters, 2017)

[4] (AutoMapper, 2018)

Als Beispiel dieser Konstellation kann das Klassendiagramm der Datentransferobjekte für das Entity «Machine» aus der nachfolgenden Abbildung herangezogen werden.

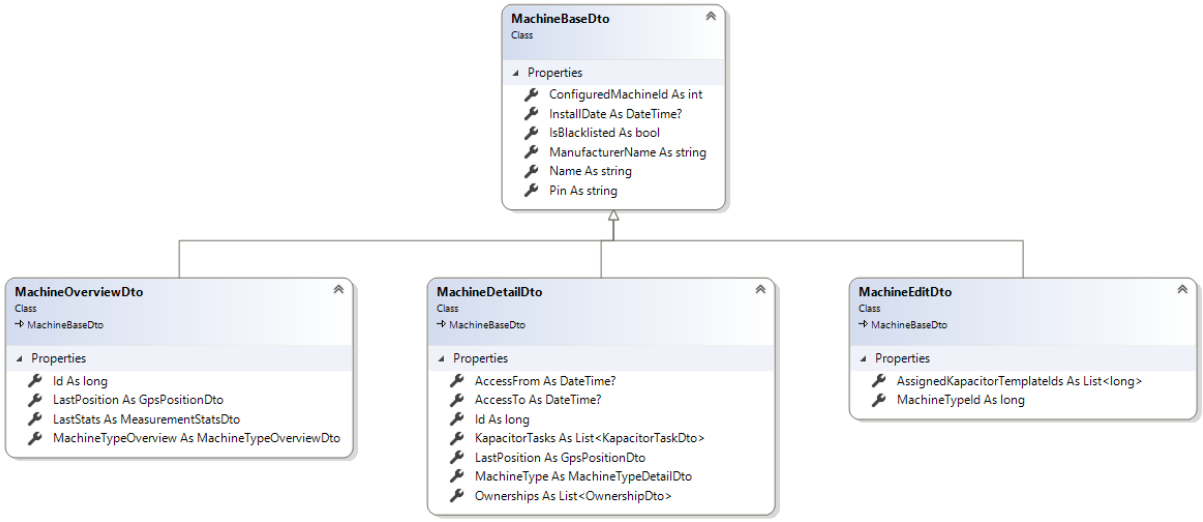


Abbildung 5: Klassendiagramm «Backend.BusinessLayer» (Datentransferobjekte für Entity)

Wie der Beschreibung der verschiedenen Unterklassen entnommen werden kann, wird durch diese Aufteilung dafür gesorgt, dass bei jedem API-Aufruf ein passendes Datentransferobjekt angefordert/zurückgeliefert wird, welches nur die tatsächlich benötigten Properties enthält.

Die Mapping-Konfiguration für die Konvertierung zwischen Entities und Datentransferobjekten erfolgt in der Klasse «SuncarDtoMappings», welche dem Backend als Singleton zur Verfügung steht.

Eine Ausnahme in der Klassen-Benennung bilden die Datentransferobjekte «User*», welche der Entity «Employee» entsprechen. Die Unterscheidung ist auf den folgenden Umstand zurückzuführen: Aus Domainsicht und nach Absprache mit dem Auftraggeber handelt es sich bei einem Nutzer des Frontends um einen Mitarbeiter, der einem einzigen Unternehmen angehört. Wenn eine Person Mitarbeiter von mehreren Unternehmen ist, sind somit zwei voneinander unabhängige Accounts notwendig. Gegen aussen soll das Konzept des «Employees» jedoch wie im Internet allgemein üblich als «User» auftreten.

Neben den von Entities abhängigen Datentransferobjekten enthält das Projekt weitere Datentransferobjekte und andere Klassen, welche in der nachfolgenden Tabelle kurz beschrieben werden.

Klasse	Beschreibung
AccessUtil	Hilfsklasse zur Überprüfung von Constraints
MachineTimeslot	Zeitrahmen, in welchem auf eine Maschine und deren Messdaten zugegriffen werden darf
QueryConstraint	Einschränkung, ob abhängig vom User, dessen Benutzerrolle und Firma, der Maschinen-Blacklist, etc. eine Ressource gelesen werden darf
UpdateConstraint	Einschränkung, ob abhängig vom User, dessen Benutzerrolle und Firma, der Maschinen-Blacklist, etc. eine Ressource verändert werden darf
DashboardDto	Informationen, welche auf dem Dashboard angezeigt werden

InfluxStatistics-Dto	Statistiken der InfluxDB für das Dashboard
MachineStatisti-csDto	Statistiken der Maschinen (aus der InfluxDB) für das Dashboard
MachineType-StatisticsDto	Statistiken der Maschinentypen (aus der InfluxDB) für das Dashboard
RdbmsStatistics-Dto	Statistiken der MariaDB für das Dashboard
BadAuth-ResponseDto	Für die externem ISO 15143-3 konforme API muss im Zuge der OAuth 2.0 Authentifizierung eine gesonderte BadAuthResponse erstellt werden.
Iso15143	Aus den XSD-Schema der ISO 15143-3 Norm generierte Response-Klassen
IsoFleetSnap-shotDto	Informationen zu einem Fleet-Snapshot für die externe ISO 15143-3 konforme API
TokenResponse-Dto	Für die externem ISO 15143-3 konforme API muss im Zuge der OAuth 2.0 Authentifizierung eine gesonderte TokenResponse erstellt werden.
KapacitorAlertDto	Alert, welcher vom Kapacitor empfangen wird
KapacitorEvent-Dto	Event, welcher aus der InfluxDB gelesen und an das Frontend geliefert wird
KapacitorTaskDto	Task aus dem Kapacitor (für Tasks gibt es kein Entity in der relationalen Datenbank)
GPSPositionDto	GPS-Position (Latitute und Longitude) mit Zeitstempel
MeasurementDto	Messung mit Zeitstempel und Messwerten
Measurement-ListDto	Messwerte-Liste mit einem Zeitrahmen und einer MeasurementDto-Liste
Measurement-StatsDto	Aktuelle Messwert-Statistiken für das virtuelle Cockpit
Measurement-StatsIds	Definition der für das virtuelle Cockpit zu verwendenden Messwerten
PasswordBase-Dto	Basisklasse von PasswordOverviewDto und PasswordEditDto
PasswordEditDto	Datentransferobjekt für das Ändern des Passworts
Password-OverviewDto	Datentransferobjekt für das Zurücksetzen des Passworts
ApiTokenDto	API-Token für den Zugriff auf die ISO 15143-3 konforme externe API
AuthenticationDto	Antwort auf ein erfolgreiches Login mit User, Access- und Refresh-Token
DownloadToken-Dto	Token für den Download der Messdaten via Messdiagramm
EmailDto	E-Mail-Inhalt zur Übergabe an den EmailService
I18NValues	Dictionary für die mehrsprachigen Properties der Datentransferobjekte
LoginDto	Login-Informationen (E-Mail und Passwort)

PagedEntities	Generische Liste mit einer Teilmenge von Elementen und der Gesamtanzahl aller Elemente für das serverseitige Paging
SupportRequest-Dto	Support-Anfrage mit Betreff, Nachricht und Supporttyp
TokensDto	Token-Paar (Refresh-Token und abgelaufener Access-Token) zur Generierung eines neuen Access-Tokens
ListParams	Parameter für das serverseitige Filtering, Sorting und Paging

3.1.1.2 CRUD-Services

Ein weiterer wichtiger Bestandteil des Business-Layers bilden die verschiedenen Services. Für die Realisierung von Dependency Injection ist es erforderlich, dass alle Services ein entsprechendes Interface implementieren. Durch diese Interfaces wird festgelegt, welche Methoden der durch Dependency Injection zur Verfügung gestellten Services genutzt werden können. Aus Gründen der Übersichtlichkeit wird an dieser Stelle auf die detaillierte Darstellung der Service-Interfaces im Klassendiagramm verzichtet. Das implementierte Interface ist jedoch stets bei der jeweiligen Service-Klasse ersichtlich.

Da viele Services primär die CRUD-Operationen (AddAsync, GetAsync, GetAllAsync, UpdateAsync, DeleteAsync) des jeweiligen Entitys beinhalten, konnte mit dem CrudService eine Basisklasse geschaffen werden, welcher alle asynchronen CRUD-Operationen generisch abdeckt. Dabei sind auch das Filtering, Sorting und Paging bereits in der Basisklasse integriert.

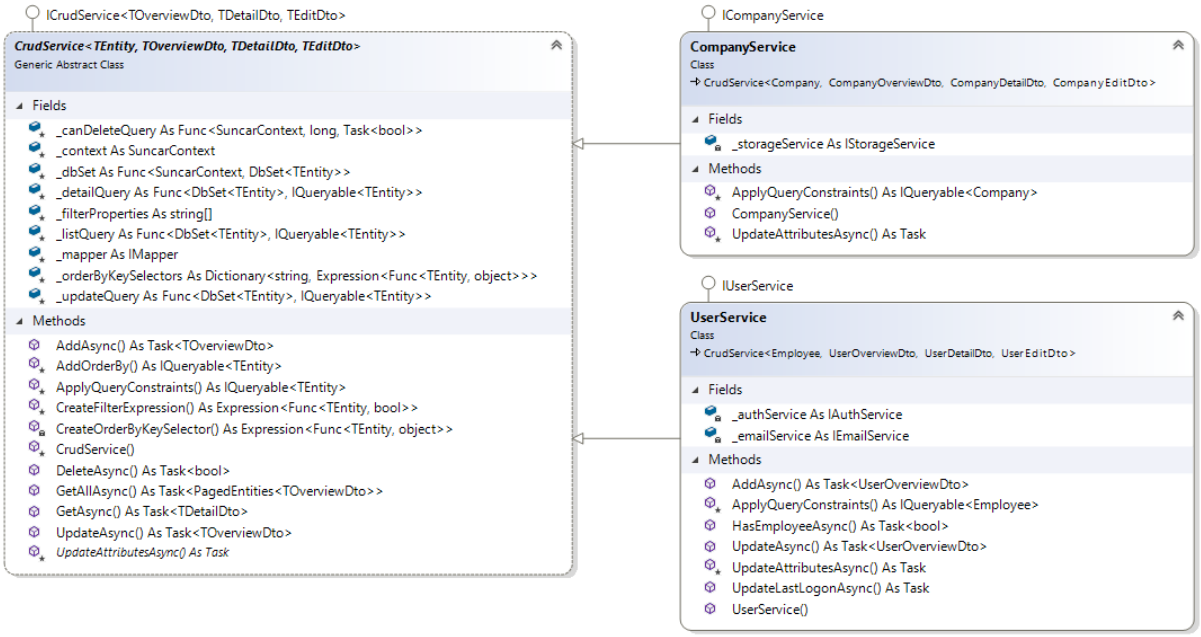


Abbildung 6: Klassendiagramm «Backend.BusinessLayer» (CRUD-Service & Unterklassen)

Wie am Beispiel der Unterklassen «CompanyService» und «UserService» zu sehen ist, können die Methoden der Basisklasse durch eine Unterklasse übersteuert werden. Lediglich die Methode «UpdateAttributes», welche beim Hinzufügen und Bearbeiten eines Entitys für die Aktua-

lisierung der Entity-Properties zuständig ist, muss zwingend von allen Unterklassen implementiert werden. Aus Gründen der Übersichtlichkeit wird an dieser Stelle auf die Darstellung der restlichen CRUD-Services verzichtet.

3.1.1.3 Kapacitor Events

Wie der Systemübersicht in Kapitel 2 entnommen werden kann, interagiert eine «Data Processing Engine» mit dem Backend und stellt die Möglichkeit zur Verfügung, basierend auf den kontinuierlich eintreffenden Messdaten Berechnungen und Analysen durchzuführen. Bei dieser Komponente handelt es sich um das InfluxDB-Addon «Kapacitor».

Kapacitor unterstützt zwei verschiedene Möglichkeiten der Datenanalyse: Beim sogenannten «Stream-Processing» werden die Daten zeitgleich mit dem Schreiben in der Datenbank analysiert. Je nach verwendetem Skript werden bestimmte Alerts generiert oder abgeleitete Pseudo-Messwerte in die InfluxDb zurückgeschrieben. Beim «Batch-Processing» werden in regelmässigen Abständen die konfigurierten Algorithmen angewendet und die definierten Aktionen durch den Kapacitor initiiert.

Für die Programmierung der Algorithmen wird eine «Domain Specific Language» (DSL) namens TICKScript eingesetzt. Die Applikation selber implementiert keine kompletten TICKScripts, sondern ermöglicht es dem Anwender, TICKScripts als Eventvorlagen im Frontend zu erfassen und dadurch die entsprechenden Tasks für die Berechnungen im Kapacitor zu aktiveren. Dabei werden drei vordefinierte Optionen für die Rückgabe von Resultaten in den TICKScripts unterstützt:

1. Pseudo-Messwerte in InfluxDB:
   Die Ergebnisse der Berechnungen können in Form von Pseudo-Messtypen in die InfluxDB geschrieben und im Frontend wie normale Messwerte angezeigt werden.
2. Events in InfluxDB:
   Die Ergebnisse der Berechnungen können in Form von Events in die InfluxDB geschrieben werden un im Frontend auch als solche angezeigt werden.
3. Alerts via API Callback:
   Die Ergebnisse der Berechnungen können über eine separate API an das Backend übermittelt werden um E-Mail-Benachrichtigungen versenden zu können.

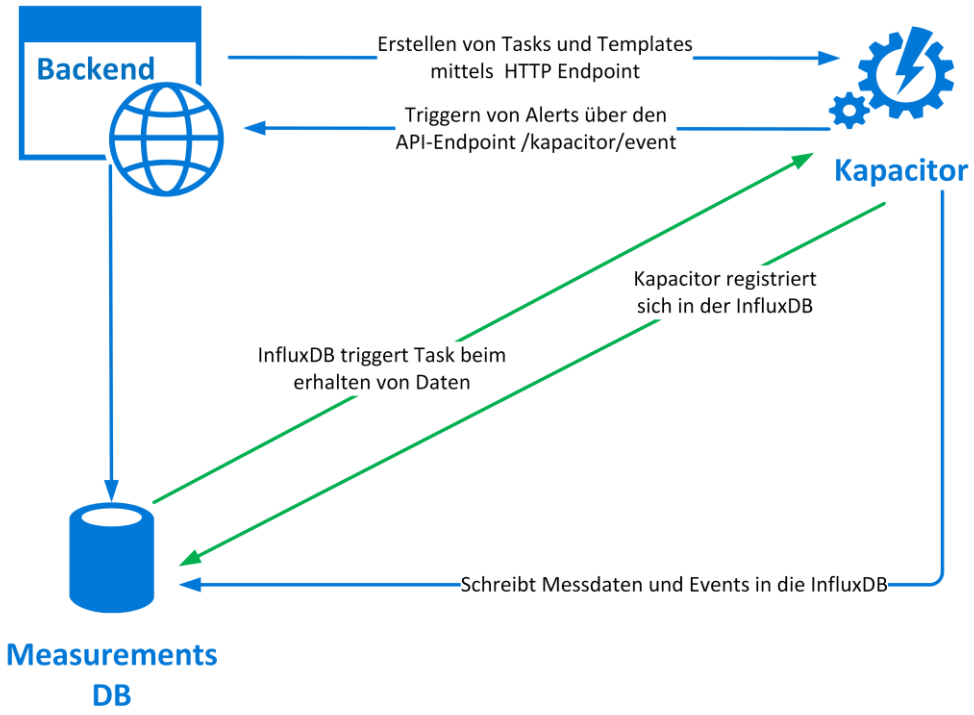


Abbildung 7: Zusammenspiel Backend, Kapacitor und Measurements DB

Im obigen Diagramm ist die Zusammenarbeit vom Kapacitor, dem Backend und der InfluxDB aufgezeigt. Dabei sind auf Seiten des Backends die nachfolgend beschriebenen Klassen für die Logik bezüglich Kapacitor verantwortlich.

Klasse	Beschreibung
KapacitorApi-Client	Anbindung an den Kapacitor mit der Library «InfluxData.Net»
ForbiddenTick-ScriptException	Exception, welche bei falscher Syntax eines TickScripts geworfen wird
EventServcie	Service, welcher einerseits dem Frontend die Events aus der InfluxDb zur Verfügung stellt und andererseits die Alerts vom Kapacitor verarbeitet (Initiierung des Versands der E-Mail-Benachrichtigungen)
Kapacitor-TemplateService	Verwaltung von Kapacitor-Templates (Eventvorlagen): Während die Eventvorlage in der relationalen Datenbank gehalten wird, muss das darin enthaltene Kapacitor-Template direkt im Kapacitor abgelegt werden.
KapacitorTask-Service	Verwaltung der Kapacitor-Tasks: Kapacitor-Tasks werden direkt im Kapacitor verwaltet und sind dort einem entsprechenden Kapacitor-Template zugewiesen. Ein Task ist eine konkrete Instanz eines Kapacitor-Templates und ist stets mit der Maschinen-Id parametriert, auf welche sich die Datenanalyse bezieht.

3.1.1.4 Weitere Services und Klassen

Nebst den CRUD-Services sind noch weitere Services und andere Klassen Bestandteil der Geschäftslogik, welche in der nachfolgenden Tabelle ersichtlich und kurz beschrieben sind.

Klasse	Beschreibung
AccessService OwnershipService	CRUD-Operationen für Entities «Access» und «Ownership», welche nicht durch den CrudService abgedeckt werden, weil es sich dabei um Verbindungs-Entities handelt
AuthService	Operationen bezüglich Login, Logout, Passwort- und Token-Validierung, etc.
AzureStorage-Service LocalFileStorage-Service	Beide Services implementieren das Interface IStorageService zur Ablage von Bildern. Im produktiven System wird der AzureStorageService genutzt, welcher die Bilddateien in einem Azure Blob Storage ablegt. Im DEBUG-Modus hingegen wird aus Effizienzgründen und um nicht jedem Entwickler einen eigenes Blob Storage erstellen zu müssen, der LocalFileStorageService genutzt. Dieser legt die Bilder, wie der Name erahnen lässt, in einem lokalen Verzeichnis ab.
AzureStorage-Options	Konfigurationsobjekt für den AzureStorageService
Dashboard-Service	Get-Operation der Daten für das Frontend-Dashboard
DictionaryExpressionExtension	Hilfsmethoden zum Erstellen eines Dictionary, welcher die für die Filterung verwendbaren Properties eines Entitys enthält.
EmailService	Operationen zum Versand von E-Mails für die Account-Aktivierung, das Zurücksetzen des Passworts, Support-Anfragen und Events mit Hilfe von «Sendgrid» (verwendeter E-Mail Gateway)
EmailOptions	Konfigurationsobjekt für den EmailService
InfluxClient	Anbindung an die InfluxDB mit Hilfe der Library «InfluxData.Net»
InfluxConnection-Config	Konfigurationsobjekt für den Datenbank-Namen der InfluxDB
JwtOptions	Konfigurationsobjekt für das Secret des JWT-Tokens
Maintenance-Service	Wartungsoperationen für die InfluxDB, den Blob Storage, etc.
Measurements-Service	Operationen zur Abfrage von Messdaten aus der InfluxDB via InfluxClient
SystemConfig-Service	Get-/Set-Operationen für die Systemkonfiguration
SuncarService-DIExtensions	Extension-Methode, welche die vorhandenen Services via Dependency Injection injiziert
TokenUtilities	Extension-Methoden bezüglich Token/Passwort, welche im AuthService und im UserService benötigt werden

3.1.2 Backend.WebAPI

Das Projekt «Backend.WebAPI» stellt die eigentliche API zur Verfügung und ist somit die Schnittstelle gegen aussen. Die verschiedenen API-Endpunkte werden dabei durch verschiedene Controller implementiert.

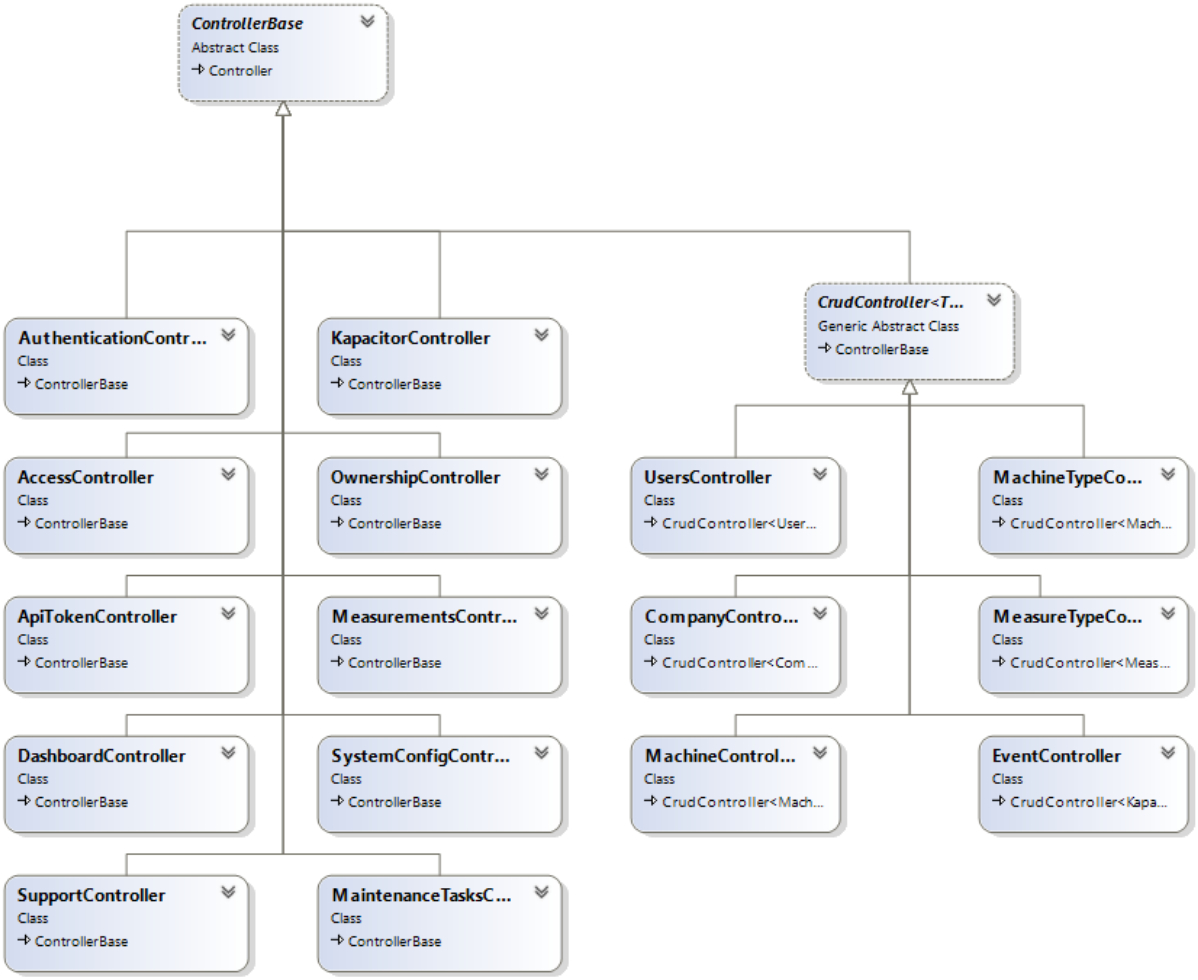


Abbildung 8: Klassendiagramm «Backend.WebAPI» (Controller)

Wie das Klassendiagramm zeigt, steht mit der Klasse «ControllerBase» für alle Controller eine Basisklasse zur Verfügung, welche alle für die Berechtigungsüberprüfung notwendigen Properties und Methoden bereits beinhaltet.

3.1.2.1 CRUD-Controller

Analog zur Basisklasse «CrudService» (im Projekt «Backend.BusinessLayer») konnte auch bei den zugehörigen Controllern die CRUD-Grundfunktionalität in die generische Basisklasse «CrudController» ausgelagert und den erbenden Controller zur Verfügung gestellt werden. So müssen sich die Unterklassen lediglich noch um die Controller-spezifische Berechtigungsüberprüfung kümmern und können anschliessend die entsprechende Internal-Methode der Basisklasse aufrufen.

3.1.2.2 Weitere Controller und Klassen

Nebst den CRUD-Controller sind noch weitere Controller und Klassen Bestandteil dieses Bereiches, welche in der nachfolgenden Tabelle aufgeführt und kurz beschrieben werden.

Klasse	Beschreibung
AccessController OwnershipController	CRUD-API-Endpunkte für Entities «Access» und «Ownership», welche nicht durch den CrudController abgedeckt werden, weil es sich dabei um Verbindungs-Entities handelt
Authentication-Controller	API-Endpunkte bezüglich Login, Logout, Passwort- und Token-Validierung, etc.
DashboardController	API-Endpunkt für das Dashboard
EventController	API-Endpunkte bezüglich Templates und Events von Kapacitor
Maintenance-TasksController	API-Endpunkte für Wartungsoperationen bezüglich InfluxDB, Blob Storage, etc.
Measurements-Controller	API-Endpunkte für die Abfrage von Messdaten
SupportController	API-Endpunkt für eine Supportanfrage
SystemConfig-Controller	API-Endpunkte für die Verwaltung der Systemkonfiguration
DateTimeJson-Converter	Die Klasse wird in der ASP.NET Core Pipeline registriert und ist somit für die (De)Serialisierung von DateTime-Objekten zuständig. Dabei stellt sie sicher, dass alle DateTime-Objekte als Unix Timestamp mit Millisekunden-Genauigkeit serialisiert werden. So kann garantiert werden, dass das Frontend alle Zeiten in der lokalen Zeitzone anzeigt.
TestEnvironment	Hilfsklasse für das End to End Testing. Aktiviert das schnelle und einfache Ersetzen der gesamten Datenbank. Damit können einzelne End to End Tests zu Beginn die Datenbank auf einen definierten und kontrollierten Stand zurücksetzen.
Startup	Startup-Klasse der Web API, welche für die Konfiguration der Services, die CORS-Einstellungen, die Swagger-Dokumentation, den JWT-Authentication-Service, etc. zuständig ist

3.1.2.3 API für Drittanbieter (ISO-15143)

Die Implementation der ISO-15143-3 konformen Schnittstelle wurde in den Namespace «Backend.WebAPI.Controllers.Iso15143» ausgelagert. Da ISO-15143-3 die Authentifizierung mittels OAuth 2.0 vorschreibt, musste die Authentifizierung anders als in der internen API implementiert werden. Detaillierte Informationen zur Umsetzung dieser API können dem Dokument «Schnittstelle nach ISO 15143-3» entnommen werden. Für die Umsetzung dieser API für Drittanbieter werden die nachfolgenden Klassen benötigt.

Klasse	Beschreibung
ApiTokenController	API-Endpunkte zur Verwaltung der API-Tokens (erstellen, widerrufen, löschen, etc.)
TokenController	Der im Zuge der OAuth 2.0 Implementation notwendige Authorisation Server, welcher basierend auf dem länger gültigen API-Token ein für kurze Zeit gültiger Access-Token erstellt, der anschliessend für den API-Zugriff verwendet werden kann
IsoApiControllerBase	Basis-Funktionalitäten, welche sowohl vom FleetController als auch vom TimeSeriesController verwendet werden
FleetController	FleetEndpoint gemäss ISO 15143-3 Kapitel 8.2.1: Liefert Daten als zeitlichen Schnappschuss zurück
TimeSeriesController	TimeSeriesEndpoint gemäss ISO 15143-3 Kapitel 8.3: Liefert Daten zu den Maschinen über eine Zeitspanne

3.1.3 Common.net

In diesem Projekt sind alle Hilfsklassen und Enums enthalten, welche in mehreren Projekten aus den Bereichen «backend» und «collector» benötigt werden.

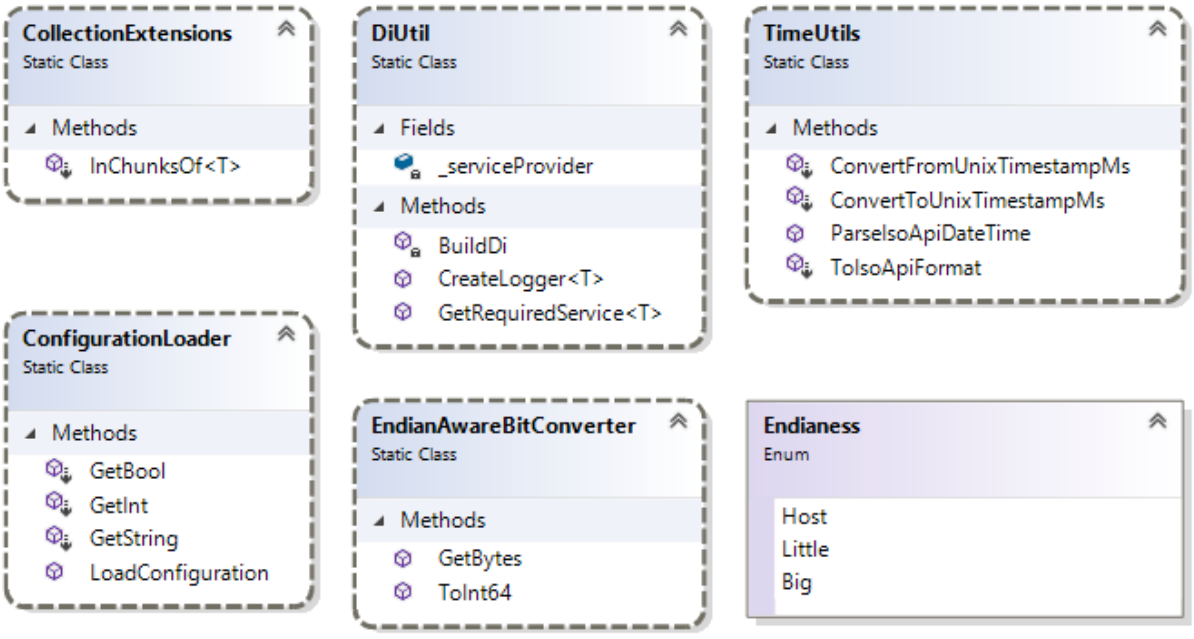


Abbildung 9: Klassendiagramm «Common.net»

Die nachfolgende Tabelle enthält eine kurze Beschreibung aller Klassen gemäss dem Klassendiagramm des «Common.net»-Projektes:

Klasse	Beschreibung
CollectionExtensions	Diese Klasse enthält die Extension-Methode «InChunksOf» zur Aufteilung und Iteration einer Liste in mehrere Teillisten mit anzugebender Grösse, was von den Services AzureStorageService, LocalFileStorageService und MaintenanceService auf Seiten des Bereichs «backend» sowie vom Backup-Import im Bereich «collector» benötigt wird.
ConfigurationLoader	Auslesen und Parsen von App-Konfigurationen
DIUtil	Einrichtung von Dependency Injection inklusive Logging im Bereich «collector»
EndianAwareBitConverter	De-/Serialisierung von long nach byte-Array unter Berücksichtigung einer definierten Byte-Reihenfolge (Endianness); Wird innerhalb des Data Collectors verwendet, da die Byte-Reihenfolge des MQTT-Payload gemäss «Schnittstellenkonzept» vorgegeben ist.
TimeUtils	Mit Hilfe zweier Extension-Methoden erfolgt die Konvertierung von einem DateTime-Objekt in ein Unix-Timestamp und umgekehrt (vor einem InfluxDB-Insert bzw. nach einem InfluxDB-Select)

3.1.4 Codegen

Wie dem Kapitel 3.1.1.1 entnommen werden kann, sind im Backend diverse Datentransferobjekte als C#-Klassen enthalten, welche für die API-basierte Kommunikation mit dem Frontend benötigt werden. Das Frontend muss daher über eigene TypeScript-Models verfügen, welche mit den jeweiligen Datentransferobjekten übereinstimmen sollten. Nebst dem Frontend werden im separaten Ent-to-End-Testing-Projekt ebenfalls TypeScript-Models der Datentransferobjekte und zusätzlich auch der Entities benötigt.

Das Codegen-Projekt übernimmt daher die Generierung von TypeScript-Models aus den Datentransferobjekten und Entities des Backends und sorgt somit für konsistente Datenobjekte in allen beteiligten Projekten.

Dieses Projekt ist sehr einfach gehalten und konnte mit wenigen Zeilen Code erstellt werden. Obwohl Tools existieren (beispielsweise NSwag[5]), welche ähnliche Code-Generierung erlauben, wurde aus Gründen der zusätzlichen Komplexität, die ein solches Tool mit sich bringen würde, entschieden, die eigene, hoch spezialisierte Implementation zu verwenden. Im Projektverlauf wurde das Tool mehrfach um neue Funktionen erweitert, was sehr einfach möglich war, da die Code-Generierung keine generalisierten Fälle enthält. Bei der Verwendung eines externen Tools hätte im schlimmsten Fall eine Anforderung nicht oder nur mit sehr hohem Aufwand abgedeckt werden können.

[5] (NSwag, 2018)

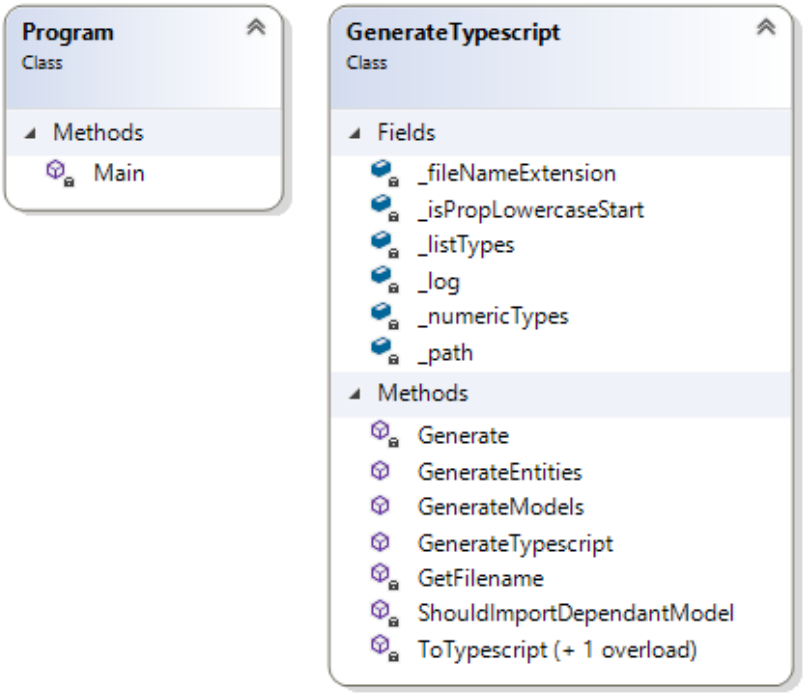


Abbildung 10: Klassendiagramm «Codegen»

Die Klasse «GenerateTypescript» ermöglicht mit den zwei entsprechenden Methoden die Generierung von Models und Entities, wobei benötigten Datentransferobjekte und Entities ebenfalls direkt in der Klasse aufgeführt sind. Somit erfolgt die Verwaltung der zu generierenden Klassen an einer zentralen Stelle in den beiden Methoden «GenerateEntities» und «GenerateModels».

Bei der Ausführung des Projektes, werden schlussendlich die folgenden Tasks durch die Main-Klasse erledigt:

1. Generierung der Datentransferobjekte für das Frontend
2. Generierung der Datentransferobjekte für das End-to-End Testing
3. Generierung der Entities für das End-to-End Testing

3.1.5 Collector.New

Dieses Projekt beinhaltet den Data Collector, welcher Messdaten über eine Message Queue von einer Maschine oder dem Legacy Data Collector empfängt und in der Measurements DB persistiert.

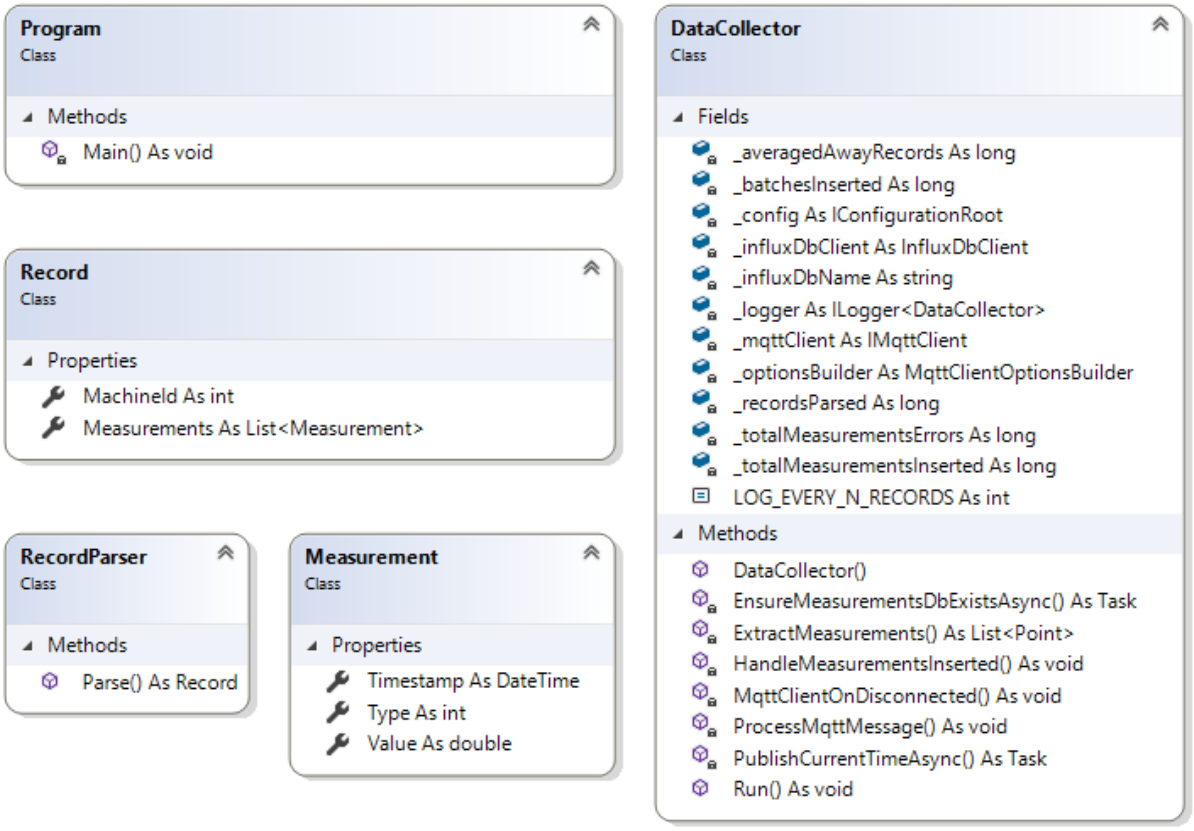


Abbildung 11: Klassendiagramm «Collector.New»

Die im obigen Diagramm ersichtlichen Klassen haben dabei die folgenden Einsatzzwecke:

Klasse	Beschreibung
Program	Start des Data Collectors
Measurement	Datentransferobjekt einer Messung mit Zeitstempel, Messtyp und Wert
Record	Datentransferobjekt einer Messübertragung mit der ID der betroffenen Maschine und einer beliebigen Anzahl an Messungen
RecordParser	Konvertierung des Payloads aus der Message Queue in ein Record-Objekt gemäss dem Schnittstellenkonzept
DataCollector	- Subscriber der Message Queue, der die ankommenden Messungen parst und in der InfluxDB persistiert - Publisher der Message Queue, der in einem gegebenen Zeitintervall die aktuelle Uhrzeit zur Verfügung stellt

3.1.5.1 Verwaltung Client-Zertifikate

Wie dem Schnittstellenkonzept zu entnehmen ist, werden neue Maschinen mit einem vom Hersteller SUNCAR HK AG generierten und signierten Client-Zertifikat ausgestattet, um die Integrität der übertragenen Messwerte sicherzustellen. Aufbau und Betrieb der dazu nötigen Certificate Authority (CA) ist in Absprache mit dem Auftraggeber nicht Teil dieser Arbeit.

Um trotzdem die Funktionalität der Client-Zertifikate zu testen, wurde vom Projektteam eine Test CA aufgesetzt und Test-Client-Zertifikate erstellt. Die dabei verwendeten Batch-Skripts zur Automatisierung sind im Source-Code-Repository zu finden.

3.1.6 Collector.Legacy

In diesem Projekt ist der **Legacy Data Collector** enthalten, welcher den alten, bestehenden Data Collector von SUNCAR HK AG ersetzt und daher nicht in der Cloud, sondern on-premise auf dem Server des Auftraggebers betrieben wird. Der Legacy Data Collector sorgt dafür, dass bestehende, bereits ausgelieferte Maschinen auch ohne Anpassung gemäss dem Schnittstellenkonzept weiterhin mit dem Gesamtsystem verbunden bleiben. Dazu werden die Messdaten gemäss der alten Schnittstellendefinition empfangen und an die Message Queue weitergeleitet.

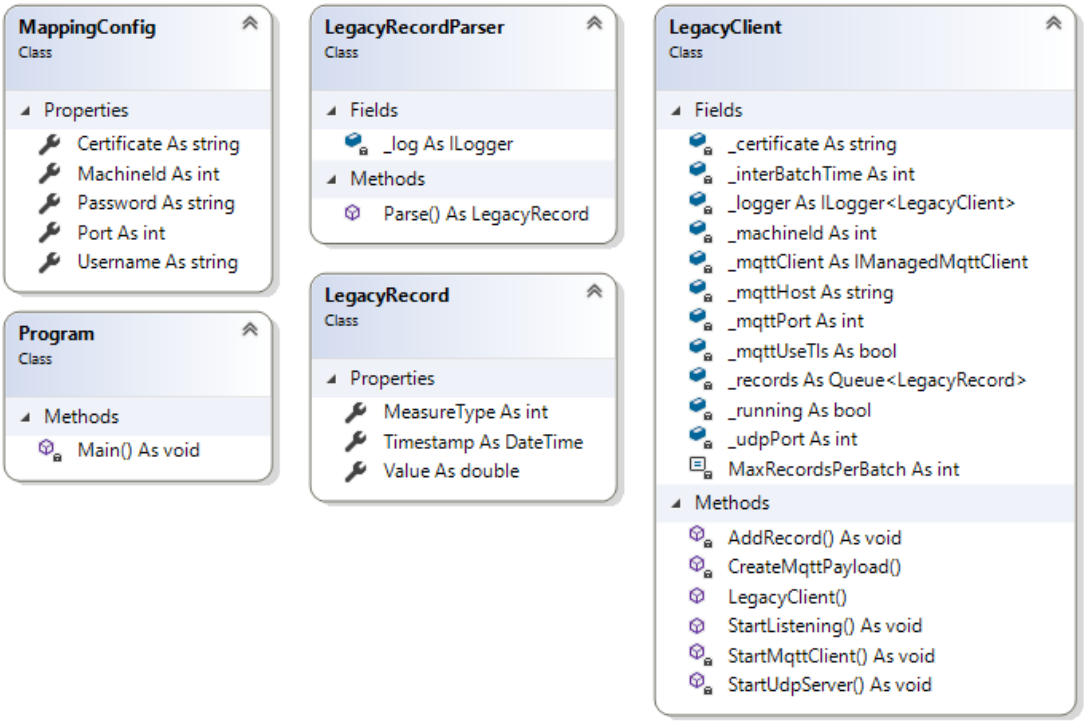


Abbildung 12: Klassendiagramm «Collector.Legacy»

Die im obigen Diagramm ersichtlichen Klassen haben dabei die folgenden Einsatzzwecke:

Klasse	Beschreibung
LegacyRecord	Datentransferobjekt einer alten Messübertragung mit Messtyp, Messwert und Zeitstempel
LegacyRecord-Parser	Konvertierung des Payloads aus der UDP-Verbindung in ein LegacyRecord-Objekt gemäss dem Schnittstellenkonzept
Mapping-Config	Konfigurationsobjekt eines Port-Maschinen-Mappings inklusive Zertifikat, Benutzername und Passwort (Konfiguration via appsettings.json)
LegacyClient	Empfang der Messdaten via UDP inklusive Parsing und Weiterleitung der gesammelten Messdaten an die Message Queue
Program	Start eines eigenen LegacyClients für jede konfigurierte MappingConfig

Da die neue Schnittstelle zwingend ein Client-Zertifikat vorsieht, kann in der Konfiguration des Legacy Collectors für jede Legacy Maschine das zu verwendende Zertifikat hinterlegt werden. Während der Entwicklung dieses Projektes wurden dabei Client-Zertifikate gemäss Abschnitt 3.1.5.1 verwendet.

3.1.7 Collector.FakeClient

Das Projekt «Collector.FakeClient» bietet die Möglichkeit, eine in Betrieb stehende **Legacy Machine** zu simulieren und dem **Legacy Data Collector** via UDP beliebige Messungen aus einer CSV-Datei zu übermitteln.

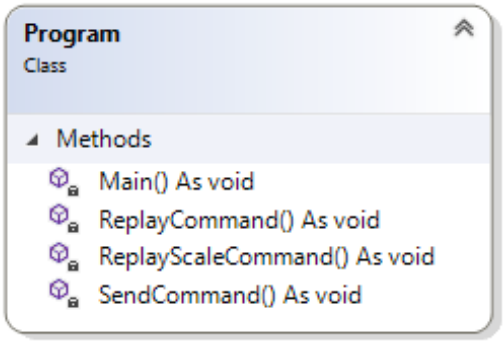


Abbildung 13: Klassendiagramm «Collector.FakeClient»

Da es sich beim FakeClient lediglich um ein Tool zu Entwicklungs- und Testzwecken handelt, wurde dieser als sehr einfach gehaltenes Konsolenprogramm implementiert. Dabei stehen folgende Konsolenbefehle zur Verfügung:

Command	Beschreibung
h	Hilfe anzeigen
send <ASCII>	UDP-Paket (ASCII) an Legacy Collector senden
replay <path_to_csv>	Alle UDP-Pakete aus der CSV-Datei an Legacy Collector senden (Format der CSV-Datei: `'<text:string><TAB><port:integer><TAB><sleep_before_seconds:double><CRLF>'`)
replay_scale <scale>	Sleep-Multiplikator für das Replay festlegen (Default: 1)

3.1.8 Collector.BackupImporter

Wie der FakeClient dient auch das Projekt «Collector.BackupImporter» nicht dem produktiven System, sondern dem Importieren der Archivdaten aus dem «alten» System in die InfluxDB des neuen Systems. Zudem konnte das Projekt erfolgreich als Stresstest zum Simulieren vieler Messwerte für den neuen DataCollector verwendet werden.

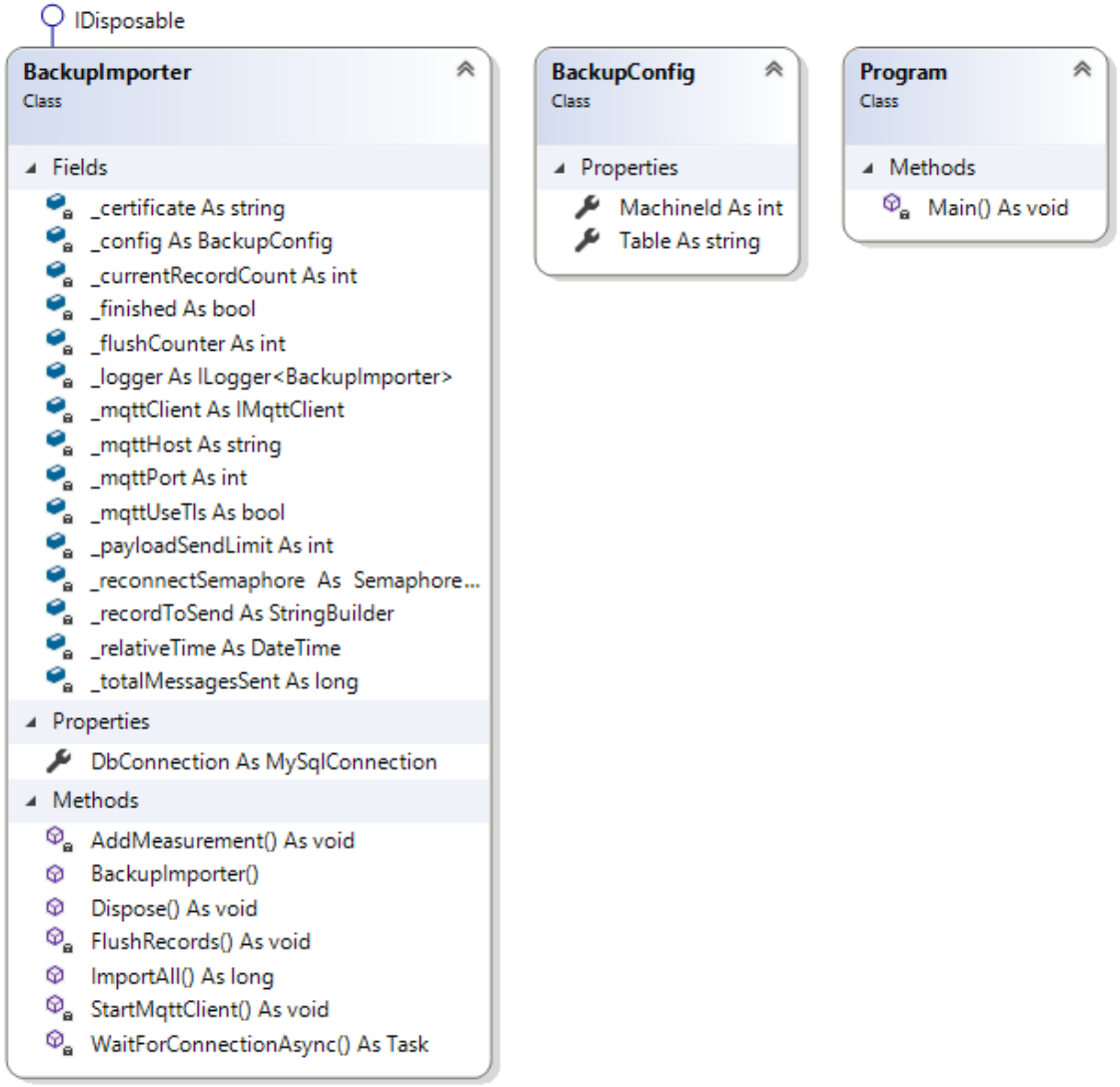


Abbildung 14: Klassendiagramm «Collector.BackupImporter»

Die im obigen Diagramm ersichtlichen Klassen haben dabei die folgenden Einsatzzwecke:

Klasse	Beschreibung
BackupConfig	Backup-Konfigurationsobjekt mit Maschinen-ID und Name der Archivtabelle aus der «alten» Datenbank (Konfiguration via appsettings.json)
BackupImporter	Auslesen der Messdaten aus einer Archivtabelle der «alten» Datenbank und Weiterleitung der (gemäss Schnittstellenkonzept geparsten) Messdaten an die Message Queue
Program	Aufteilung aller BackupConfigs auf alle Worker-Threads und Start des BackupImporters für jede konfigurierte BackupConfig

3.2 Frontend

Nebst dem Backend und dem Data Collector bildet das Frontend eine weitere wichtige Stütze im Gesamtsystem. Dieses Teilsystem widmet sich vollumfänglich der Client-Seite und ist zuständig für Aufbereitung und Präsentation der über die Backend-API abgefragten Daten.

Für die Implementierung des Frontends wurde primär auf «React» als UI-Library, «Redux» als State-Verwaltung und «TypeScript» als Programmiersprache gesetzt.

React ist eine deklarative JavaScript Library von Facebook zur Erstellung von User Interfaces und basiert auf dem Konzept von gekapselten, verschachtelten und modular aufgebauten Komponenten.[6] Redux ist eine JavaScript Library für die Verwaltung des Applikationszustands (State). React-Komponenten können ihren eigenen Status grundsätzlich selbst verwalten und die Daten im Bedarfsfall an untergeordnete Komponenten weiterreichen. Ab einer gewissen Grösse der Applikation bringt der Einsatz eines zentralen States als «single source of truth» jedoch viele Vorteile. TypeScript ist eine Programmiersprache von Microsoft, welche JavaScript um die Funktionalität der Typisierung, Klassen und Module erweitert.

Für eine Übersicht zu allen verwendeten Modulen sei an dieser Stelle auf die Tabelle im Anhang (Kapitel 9.2) verwiesen.

3.2.1 Struktur

Die logische Struktur des Frontends zeigt sich anhand der nachfolgenden Abbildung, welche die Gliederung des Projektes in die verschiedenen Unterverzeichnisse und deren Abhängigkeiten untereinander aufzeigt.

[6] (React, 2018)

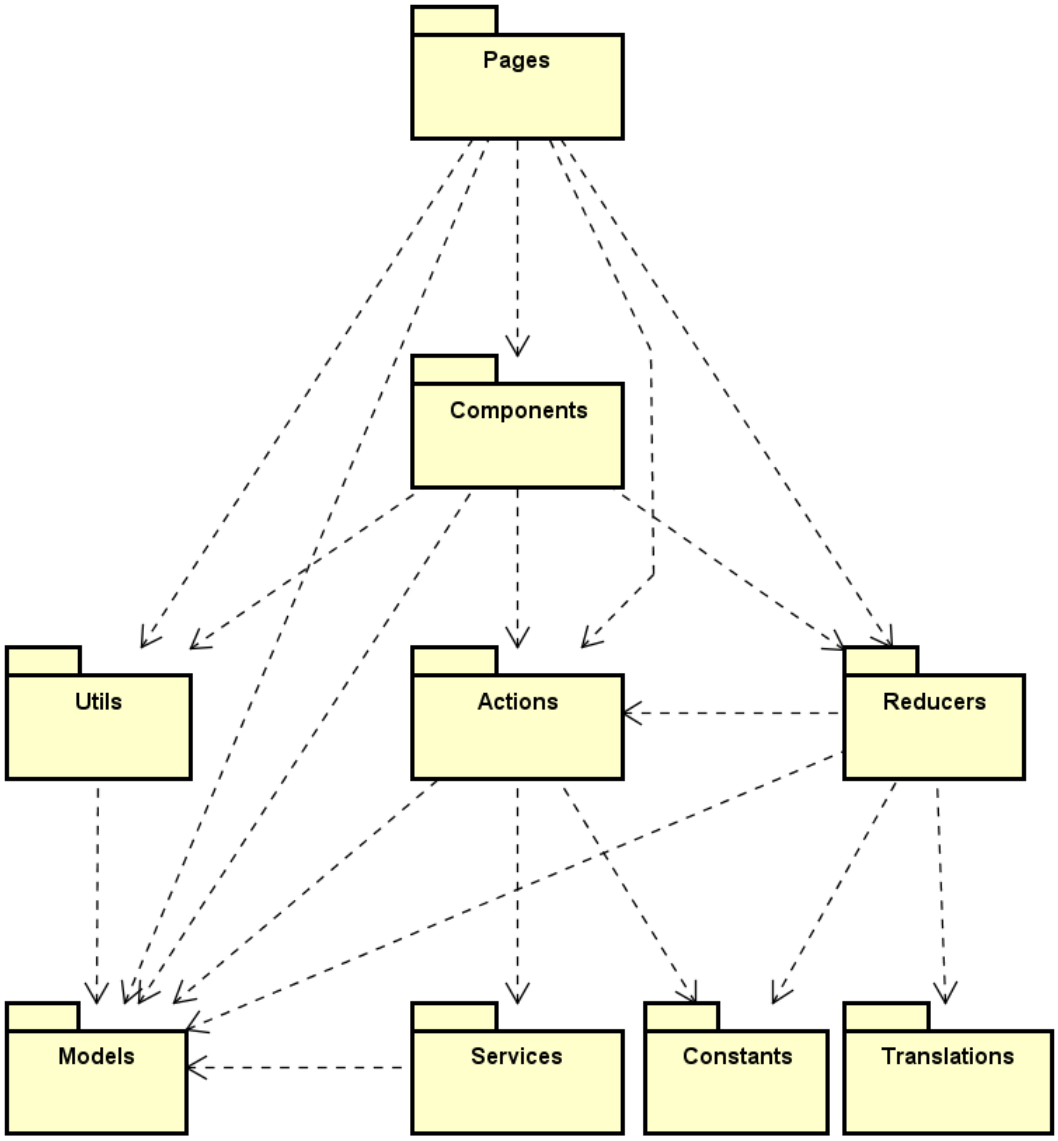


Abbildung 15: Struktur und Abhängigkeiten des Frontends

3.2.2 Sektionen

Für eine klare Struktur des Frontends wurden verschiedene Sektionen definiert, welche vor allem in den Teilbereichen «Pages», «Actions», «Reducers» und «Services» vorzufinden sind. Bei der Mehrheit dieser Sektionen handelt es sich um CRUD-Sektionen, welche in der nachfolgenden Tabelle aufgelistet sind.

CRUD-Sektionen	
Access	Machine
ApiToken	MachineType
Company	MeasureType
Event	Ownership
EventTemplate	User

Alle CRUD-Sektionen entsprechen grundsätzlich einem Entity aus dem Backend und sind über alle Ebenen des Frontends für das Hinzufügen, Bearbeiten und Löschen sowie für die Detail- und Listenansicht zuständig. Dabei muss eine CRUD-Sektionen nicht alle CRUD-Operationen abdecken, wenn die entsprechende Funktionalität nicht benötigt wird.

Das Frontend besteht jedoch nicht nur aus CRUD-Sektionen, sondern verfügt zusätzlich über die nachfolgenden Sektionen.

Sektion	Beschreibung
Authentication	Login, Logout, Zurücksetzen des Passworts, etc.
Dashboard	Dashboard des Frontends
Measurements	Messdaten für Diagramme, GPS-Karten, etc.
Settings	Anzeigesprache, Seitentitel
Support	Supportanfrage
SystemConfig	Konfiguration der Systemeinstellungen
ThemeSelection	Auswahl des Themes (Development/Production)

In den nachfolgenden Kapiteln erfolgt die Beschreibung aller Teilbereiche des Frontends und des Zusammenspiels untereinander. Dabei wird stets auf die soeben definierten Sektionen verwiesen.

3.2.3 Models

Wie bereits in Kapitel 3.1.4 (Codegen) beschrieben, werden basierend auf den Datentransferobjekten des Backends entsprechende Models generiert, damit die gleichen Objekte auch im Frontend eingesetzt werden können. Die dadurch entstandenen Models sind in diesem Verzeichnis untergebracht.

3.2.4 Translations

Im Verzeichnis «Translations» sind die Übersetzungen für alle angezeigten Texte im Frontend in Form eines JSON-Files pro Sprache enthalten. Für die Implementierung der Internationalisierung wurde auf die Library «react-intl» von Yahoo gesetzt, welche unter anderem die sprachspezifische Übersetzung und Formatierung von Datumswerten, Zahlen, etc. zur Verfügung stellt.

3.2.5 Constants, Actions, Services, Reducers

Die Teilbereiche «Constants», «Actions», «Services» und «Reducers» sind durch ein enges Zusammenspiel mit Hilfe von Redux geprägt und werden deshalb gemeinsam erläutert. Durch alle Teilbereiche hindurch wird die Unterteilung der Sektionen berücksichtigt. Die nachfolgende Tabelle beinhaltet eine kurze Beschreibung der Redux-Konzepte.

Redux-Konzept	Beschreibung
State (Store)	Der State ist ein globales JavaScript-Objekt für den zentralen Applikationszustand, welcher nach Belieben strukturiert werden kann.[7] Der State ist unterteilt in mehrere Sub-States, welche den Sektionen entsprechen.
Action	Der State ist schreibgeschützt und so müssen alle State-Änderungen über Actions mitgeteilt werden (Dispatching).[8] Eine Action ist wiederum ein einfaches JavaScript-Objekt, welches einen Action-Typ und optional weitere Daten für die Änderung des States beinhaltet.[9]
Constant	Um die Typisierung gewährleisten zu können, müssen die verfügbaren Typen einer Action als Konstanten und TypeScript-Typen definiert werden.
Reducer	Nachdem eine Action ausgelöst wurde, wird diese durch einen Reducer behandelt. Der Reducer erstellt basierend auf dem bisherigen State-Inhalt und der Action ein neues State-Objekt.[10] Dabei existiert pro Sektion ein Sub-Reducer, welcher für den entsprechenden Sub-State verantwortlich ist. Alle Sub-Reducers werden im Root-Reducer zusammengeführt.

Die nachfolgenden Unterkapitel beinhalten eine Beschreibung des Zusammenspiels zwischen «Constants», «Actions», «Services» und «Reducers» innherhalb von CRUD-Sektionen und den weiteren Sektionen.

3.2.5.1 Zusammenspiel: CRUD-Sektion

Wie das Zusammenspiel aller Teilbereich innerhalb einer CRUD-Sektion abläuft, kann am Beispiel der CRUD-Sektion **«MachineType»** aufgezeigt werden. Diese Sektion ist für die folgende Funktionalität zuständig:

- Alle Maschinentypen abrufen
- Einzelner Maschinentyp abrufen
- Maschinentyp hinzufügen
- Maschinentyp bearbeiten
- Maschinentyp löschen

Um diese Funktionalität anbieten zu können, werden «Actions» benötigt, welche wiederum auf die «Constants» und den «Service» zurückgreifen müssen, wie im nachfolgenden Klassendiagramm ersichtlich ist. In den folgenden Abschnitten werden die Beziehungen und Abläufe zwischen den betroffenen Teilbereichen genauer behandelt.

[7] (Store, 2018)

[8] (Three Principles, 2018)

[9] (Actions, 2018)

[10] (Reducers, 2018)

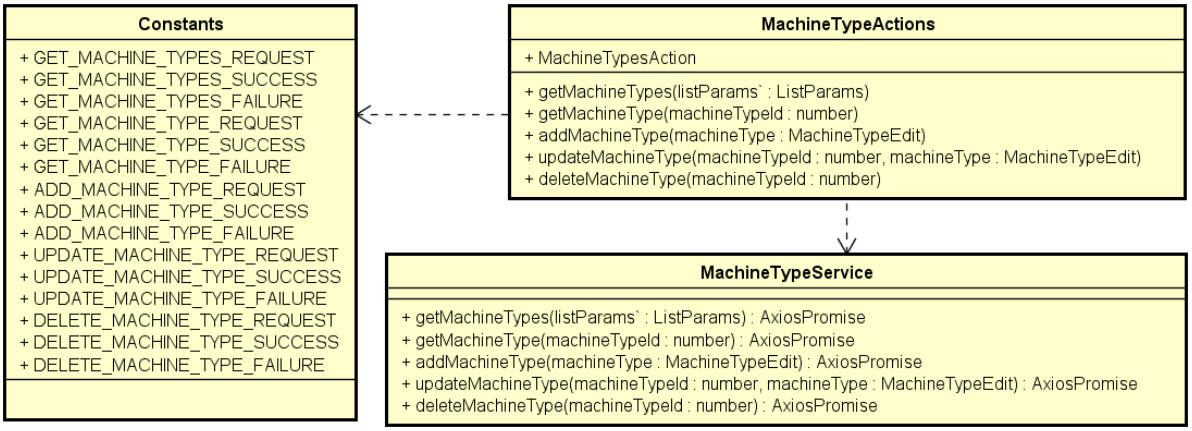


Abbildung 16: Klassendiagramm «Constants», «Actions» und «Service» bezüglich Sektion «MachineType»

Constants

Bei allen zuvor genannten Funktionen handelt es sich um asynchrone API-Aufrufe, wobei die API entweder die gewünschten Daten oder aber einen Fehlercode zurückliefert. Damit dieser asynchrone Vorgang im State abgebildet werden kann, werden pro Funktionalität drei Action-Typen (Constants) definiert, wie das nachfolgende Beispiel für die Funktion «Maschinentyp hinzufügen» zeigt.

Action-Typ	Bedeutung
ADD_MACHINE_TYPE_REQUEST	Es wird ein API-Aufruf zum Hinzufügen eines Maschinentyps gestartet.
ADD_MACHINE_TYPE_SUCCESS	Die Antwort der API ist eingetroffen: Der Maschinentyp wurde erfolgreich hinzugefügt.
ADD_MACHINE_TYPE_FAILURE	Die Antwort der API ist eingetroffen: Es ist ein Fehler aufgetreten.

Analog dazu verfügen die weiteren Funktionen aller CRUD-Sektionen über die drei Action-Typen «*%Funktion%*_REQUEST», «*%Funktion%*_SUCCESS» und «*%Funktion%*_FAILURE», welche im Teilbereich «Constants» definiert sind.

Actions

Im Teilbereich «Actions» werden nun einerseits die eigentlichen Actions mit den genannten Action-Typen und andererseits sogenannte Action Creators zur Verfügung gestellt. Ein Action Creator bildet einen asynchrone API-Aufruf inklusive Antwort-Behandlung ab, wie wiederum das Beispiel der Funktion «Maschinentyp hinzufügen» zeigt:

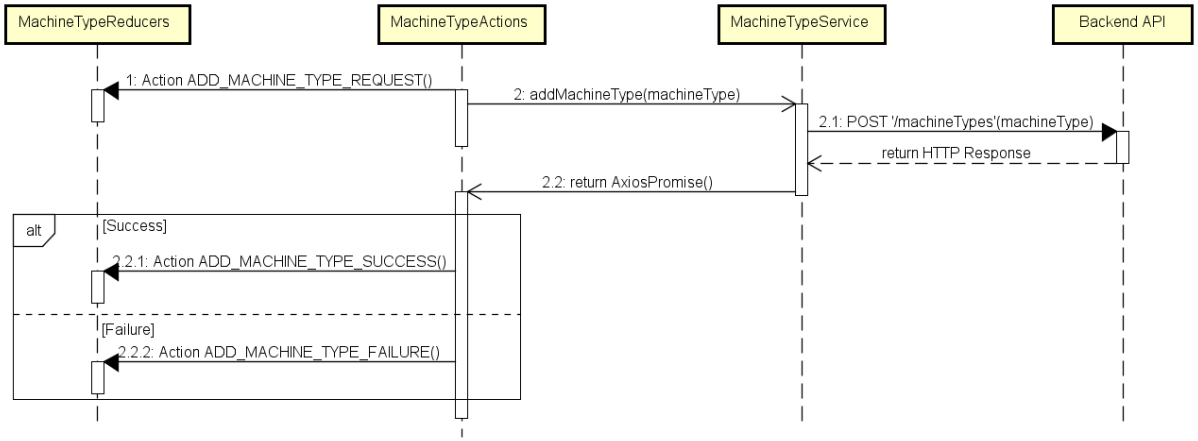


Abbildung 17: Sequenzdiagramm beim Aufruf des Action Creators «addMachineType»

Wie dem Sequenzdiagramm entnommen werden kann, teilt der Action Creator dem Reducer zuerst mit, dass er einen API-Aufruf startet (Action _REQUEST), was er anschliessend auch tut. Wenn die Antwort der Backend API via Services eintrifft, wird je nach Antwort-Status entweder die Action _SUCCESS oder _FAILURE ausgelöst. Bei dieser asynchronen Aktion[11] werden folglich in jedem Fall zwei Actions ausgelöst. Alle weiteren API-Aufrufen der CRUD-Sektionen aber auch aller anderen Sektionen verfolgen das selbe Vorgehen. Die Action Creators werden als Funktionen exportiert und den anderen Teilbereichen zur Verfügung gestellt.

Services

Dieser Teilbereich kümmert sich voll und ganz um die Kommunikation mit dem Backend, wozu der Promise-basierte HTTP-Client «axios» eingesetzt wird. Für jeden API-Endpunkt des Backends wurde in diesem Verzeichnis eine entsprechende Funktion implementiert, welche die Parameter von den Action Creators entgegennimmt und damit einen API-Aufruf an das Backend erzeugt. Die darauf erhaltene Antwort wird dann wieder an den aufrufenden Action Creator zurückgegeben. Dazu steht für jede Sektion ein eigener Service zur Verfügung.

Reducers

Unabhängig von allen anderen Teilbereichen sind die Reducers nur dafür zuständig, eine vom Teilbereich «Actions» ausgelöste Action zu behandeln und den State entsprechend neu aufzubauen. Jeder Sub-Reducer ist für seinen eigenen Sub-State verantwortlich, welcher bei allen CRUD-Sektionen gleich aufgebaut ist. Die nachfolgende Tabelle zeigt beispielhaft den Sub-State der Sektion «MachineType» auf.

Sub-State «MachineType»	Bedeutung
items: MachineTypeOverview[ ]	Von der Funktion «getMachineTypes» erhaltene Liste von Maschinentypen
count: number	Von der Funktion «getMachineTypes» erhaltene totale Anzahl der vorhandenen Maschinentypen (Pagination)
isLoading: boolean	Wird die Liste der Maschinentypen gerade geladen?

[11] (Async Actions, 2018)

error: string	Allfällige Fehlermeldung der Funktion «getMachineTypes»
add:	Sub-Bereich für die Funktion «addMachineType»
item: MachineTypeOverview	Hinzugefügter Maschinentyp (API Antwort)
isLoading: boolean	Wird gerade ein Maschinentyp hinzugefügt?
error: string	Allfällige Fehlermeldung
edit:	Sub-Bereich für die Funktion «updateMachineType»
item: MachineTypeOverview	Bearbeiteter Maschinentyp (API Antwort)
isLoading: boolean	Wird gerade ein Maschinentyp bearbeitet?
error: string	Allfällige Fehlermeldung
delete:	Sub-Bereich für die Funktion «deleteMachineType»
isLoading: boolean	Wird gerade ein Maschinentyp gelöscht?
error: string	Allfällige Fehlermeldung
detail:	Sub-Bereich für die Funktion «getMachineType»
item: MachineTypeDetail	Erhaltener Maschinentyp (API Antwort)
isLoading: boolean	Wird gerade ein Maschinentyp geladen?
error: string	Allfällige Fehlermeldung

Wie dem Aufbau entnommen werden kann, wird im Sub-State für jede Funktion grundsätzlich der Ladestatus (isLoading) und die API-Antwort (items & count, item oder error) abgespeichert. So können die Daten im Frontend mit allfälligen Loading-Animationen und Fehlerbenachrichtigungen angezeigt werden.

Analog zum Sub-State «MachineType» verfügen alle weiteren CRUD-Sektionen über einen ähnlichen Sub-State, wodurch die verschiedene generische BaseStates extrahiert werden konnten. Die BaseStates «CUDBaseState» (Sub-Bereiche «add», «edit» und «delete») und «CRUDBaseState» (zusätzlich Sub-Bereich «detail») decken so alle CRUD-Sektionen ab. Aufgrund der ähnlichen Struktur der Sub-States und Actions aller CRUD-Sektionen konnte auch die Funktionalität der CRUD-Reducer in einen BaseReducer extrahiert werden.

3.2.5.2 Zusammenspiel: Weitere Sektionen

Die restlichen Sektionen sind ebenfalls sehr ähnlich wie die zuvor beschriebenen CRUD-Sektionen aufgebaut. Jede asynchrone API-Abfrage wird durch einen Action Creator gestartet, vom entsprechenden Service ausgeführt und die Änderung vom zuständigen Reducer in den State übertragen. Im Sub-State werden wie bei den CRUD-Reducer jeweils die Informationen «isLoading» und «error» sowie der Inhalt der API-Antwort abgelegt.

3.2.6 Utils

Im Bereich «Utils» sind alle Hilfsfunktionen enthalten, welche von mehreren Komponenten aus den Bereichen «Pages» und «Components» benötigt werden. Wie die nachfolgende Abbildung zeigt, handelt es sich dabei um die Funktionalität für das clientseitige, generische Tabellen-Handling (Filterung, Sortierung, Pagination), für die Überprüfung der Status-Schwellenwerte der Cockpit-Messwerte und für die clientseitige Validierung der Passwortrichtlinien.

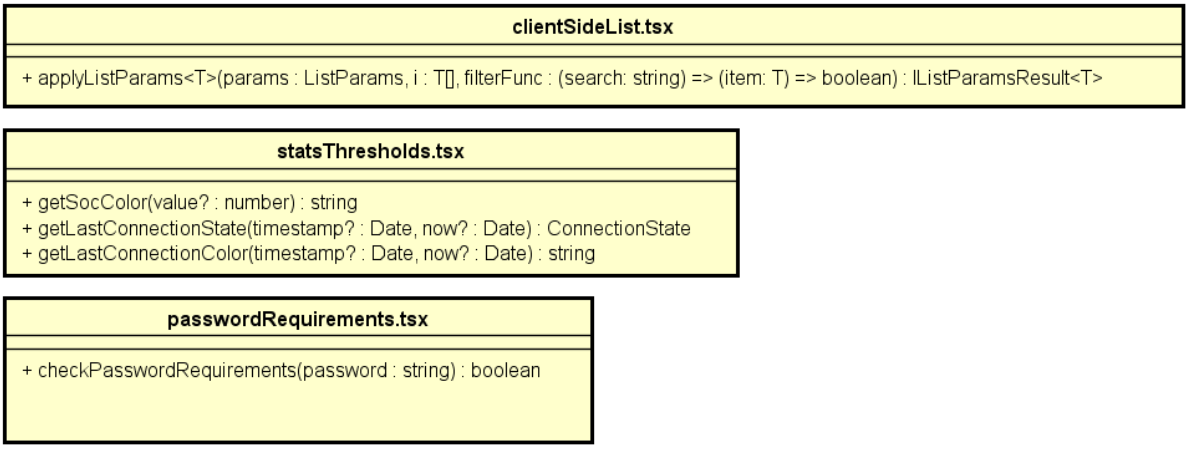


Abbildung 18: Klassendiagramm «Utils»

3.2.7 Pages

Der Bereich «Pages» beinhaltet alle React-Komponente, welche für die Darstellung und den Inhalt einer Seite des Frontends zuständig sind. Auch in diesem Bereich wird die Unterteilung in die verschiedenen Sektionen fortgesetzt. In allen CRUD-Sektionen sind dabei grundsätzlich die folgenden Komponenten enthalten.

CRUD-Komponente	Beschreibung
*%Sektion%*AddEdit	Formular zum Hinzufügen/Bearbeiten eines Elementes
*%Sektion%*DetailAdmin	Detailansicht des Elementes für die Administration
*%Sektion%*List	Auflistung aller Elemente
*%Sektion%*Row	Einzelne Element-Zeile der Auflistung

Nebst den oben aufgeführten CRUD-Komponenten existieren noch weitere Komponenten. In der nachfolgenden Tabelle sind alle zusätzlichen Page-Komponenten pro Verzeichnis aufgeführt und mit einer kurzen Beschreibung versehen.

Komponente	Beschreibung
Layout:	
AppBar	Titelleiste des Frontends
Drawer	Seiten-Menü des Frontends
Layout	Grundgerüst einer Frontend-Seite
Dashboard:	
SuncarDashboard	Dashboard des Frontends
DashboardPanel	Einzelne Kachel auf dem Dashboard

Login:	
Login	Login-Formular
Machine:	
MachineCards	Bebilderte Auflistung von Maschinen in Form von «Cards»
MachineInfo	Detailansicht einer Maschine
Cockpit	Register in «MachineInfo» mit der Cockpit-Ansicht
GpsMap	Register in «MachineInfo» mit der GPS-Ansicht
GpsMapContent	Inhalt der Karte von Google Maps (GPS-Spur als Polyline)
Map:	
MachinesMap	Kartenansicht aller Maschinen
MachinesMapContent	Inhalt der Karte von Google Maps (Maschinen als Marker)
Measurements:	
MeasurementChartContainer	Messdaten-Diagrammansicht einer beliebigen Maschine
Reset-Password:	
PasswordReset	Formular zum Zurücksetzen des Passworts
RequestPasswordReset	Formular zum Beantragen eines Reset-E-Mails
Support:	
Support	Support-Seite mit Kontaktinformationen
SupportRequestForm	Formular zum Absenden einer Supportanfrage
System-Config:	
SystemConfig	Konfiguration der Systemparameter
User:	
UserChangePassword	Formular zum Ändern des Passworts
UserInformation	Informationen zu einem Benutzer
UserProfile	Profilansicht des eigenen Benutzers

3.2.8 Components

Im Bereich «Components» sind alle React-Komponenten enthalten, welche in mehreren «Pages»-Komponenten benötigt werden und somit sinnvoll ausgelagert werden können. Die verschiedenen Komponenten und ihr Einsatzzweck sind in der nachfolgenden Tabelle kurz beschrieben.

Komponente	Beschreibung
Chart:	
MaterialLegend	Diagramm-Legende in Material Design
MaterialTooltip	Diagramm-Tooltip in Material Design
MeasurementChart	Messdatendiagramm mit dem NPM-Modul «Recharts»
Cockpit:	
RpmGauge	Drehzahl-Diagramm mit dem NPM-Modul «@syncfusion»
TemperatureGauge	Thermometer-Diagramm mit dem NPM-Modul «@syncfusion»

Style:	
AddEditStyle	CSS-Design für Page-Komponenten «_AddEdit»
DetailStyle	CSS-Design für Page-Komponenten «_DetailAdmin»
ListStyle	CSS-Design für Page-Komponenten «_List»
MachineFilterStyle	CSS-Design für Page-Komponenten mit einem Machine-Filter
MapStyle	CSS-Design für Page-Komponenten mit einer Karte
PublicPageStyle	CSS-Design für öffentliche Page-Komponenten (z.B. Login)
RowStyle	CSS-Design für Page-Komponenten «_Row»
Table:	
TableFooter	Fusszeile einer Tabelle mit Pagination
TableHeader	Kopfzeile einer Tabelle mit Spaltennamen und Sortierung
TableToolbar	Toolbar einer Tabelle mit Suche
AccessControlCtx	React-Context für die clientseitige Überprüfung der Zugriffsberechtigung eines Benutzers
AvatarIcon	Bild-Avatar mit Icon als Ersatz bei fehlendem Bild
DeleteConfirmation	Dialogfeld zur Bestätigung beim Löschen eines Elementes
DialogWrapper	Dialog-Grundgerüst (Fullscreen in der Mobileansicht)
I18NEditValue	Bearbeitungsfelder für mehrsprachige Felder
I18NValue	Anzeige der richtigen Ausgabe eines mehrsprachigen Feldes
ImageSelector	Bearbeitungsfeld für den Bild-Upload
LoadingWrapper	Grundgerüst mit «Loading»-Animation und anschliessender Inhaltsanzeige
MachinePrint	Maschineninformationen für die Print-Ansicht
MachineSelector	Maschinen-Auswahl, um von einer zur anderen Maschine zu wechseln
MediaCard	Grundgerüst mit Bild und Text nebeneinander
MessageSnackbar	Benachrichtigungsanzeige am unteren Bildrand
PasswordRequirements	Passwortrichtlinien
PercentageBar	Anzeige der aktuellen Batterieladung
PublicPage	Grundgerüst für öffentliche Seiten (z.B. Login)
SuncarSelect	Generisches Auswahlfeld für beliebige Elemente
withTracker	Integration von Google Analytics
PrivateRoute	Container für Seiten, welche nur für angemeldete Benutzer angezeigt werden sollen

Da sich viele Page-Komponenten in ihrer Tätigkeit kaum unterscheiden, konnten deren Funktionalität in verschiedene Base-Komponenten extrahiert werden. Nebst den oben aufgeführten Komponenten sind diese Base-Komponenten ebenfalls in diesem Bereich angesiedelt und werden in den folgenden Unterkapiteln beschrieben.

3.2.8.1 BaseAddEdit

Die BaseAddEdit-Komponente fasst die Funktionalität eines Formulars zum Hinzufügen oder Bearbeiten eines Elementes zusammen. Die Base-Komponente gibt das Design und die Struktur des Formulars vor und enthält beispielsweise die Funktionalität für das Ändern des Formularinhalts und das Speichern der Änderungen.

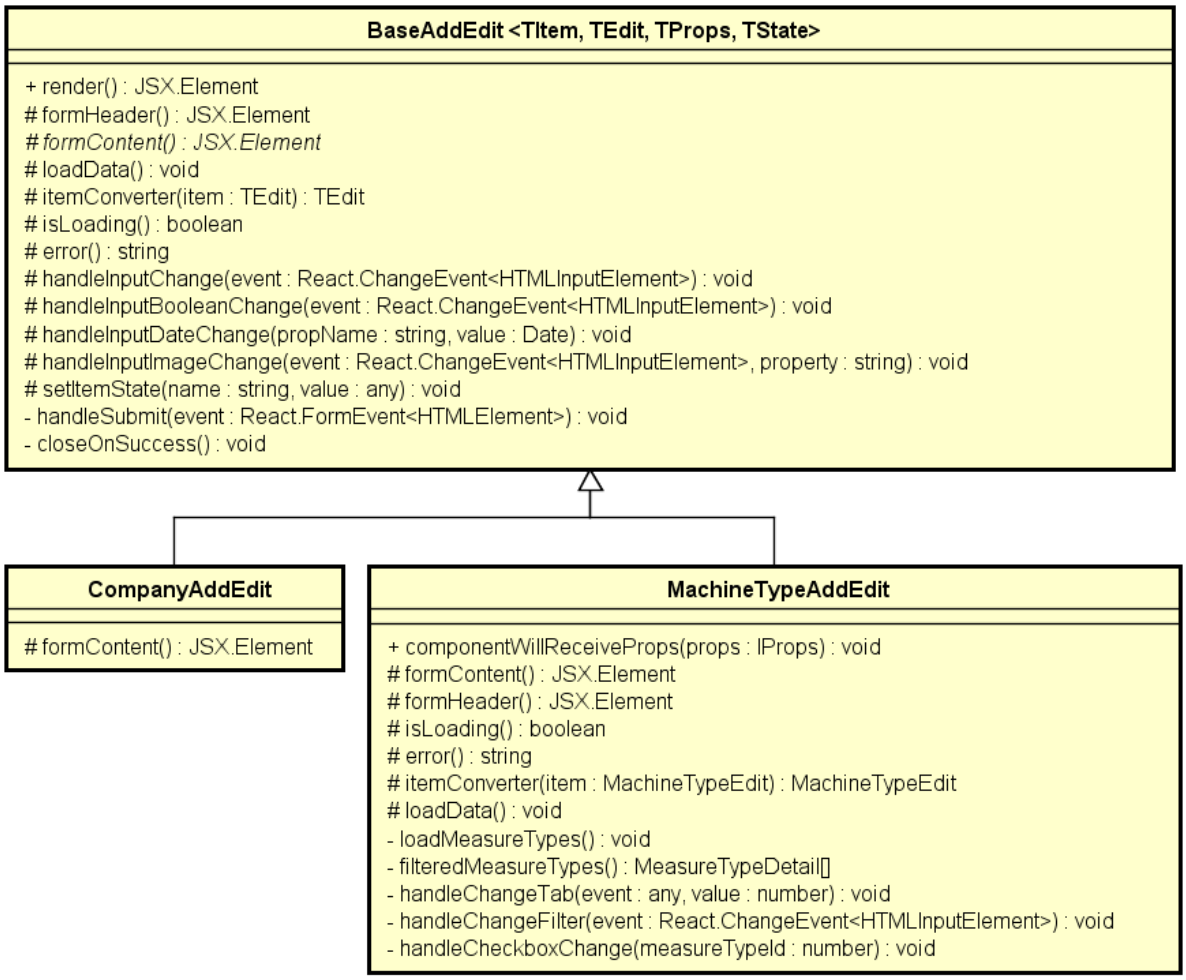


Abbildung 19: Klassendiagramm «BaseAddEdit»

Das Klassendiagramm zeigt beispielhaft anhand von zwei Sub-Komponenten, wie die Base-Komponente geerbt wird. Insgesamt wird die BaseAddEdit-Komponente von allen «*%Sektion%*AddEdit»-Komponenten und zusätzlich von den Komponenten «SupportRequestForm» und «UserChangePassword» geerbt.

Viele Sub-Komponenten wie beispielsweise «CompanyAddEdit» überschreiben lediglich die Methode «formContent», um dadurch den Formularinhalt zu bestimmen. Andere Sub-Komponenten (z.B. «MachineTypeAddEdit») überschreiben mehrere Methoden und besitzen zusätzliche Methoden, um einen Kopfbereich mit Tabs einzufügen, Detailinformationen nachzuladen oder Elemente zu konvertieren.

3.2.8.2 BaseCrudList

Die BaseCrudList-Komponente beinhaltet eine generische Auflistung von Elementen und fasst die Funktionalität zum Anzeigen der Komponenten für das Hinzufügen, Bearbeiten und Löschen zusammen.

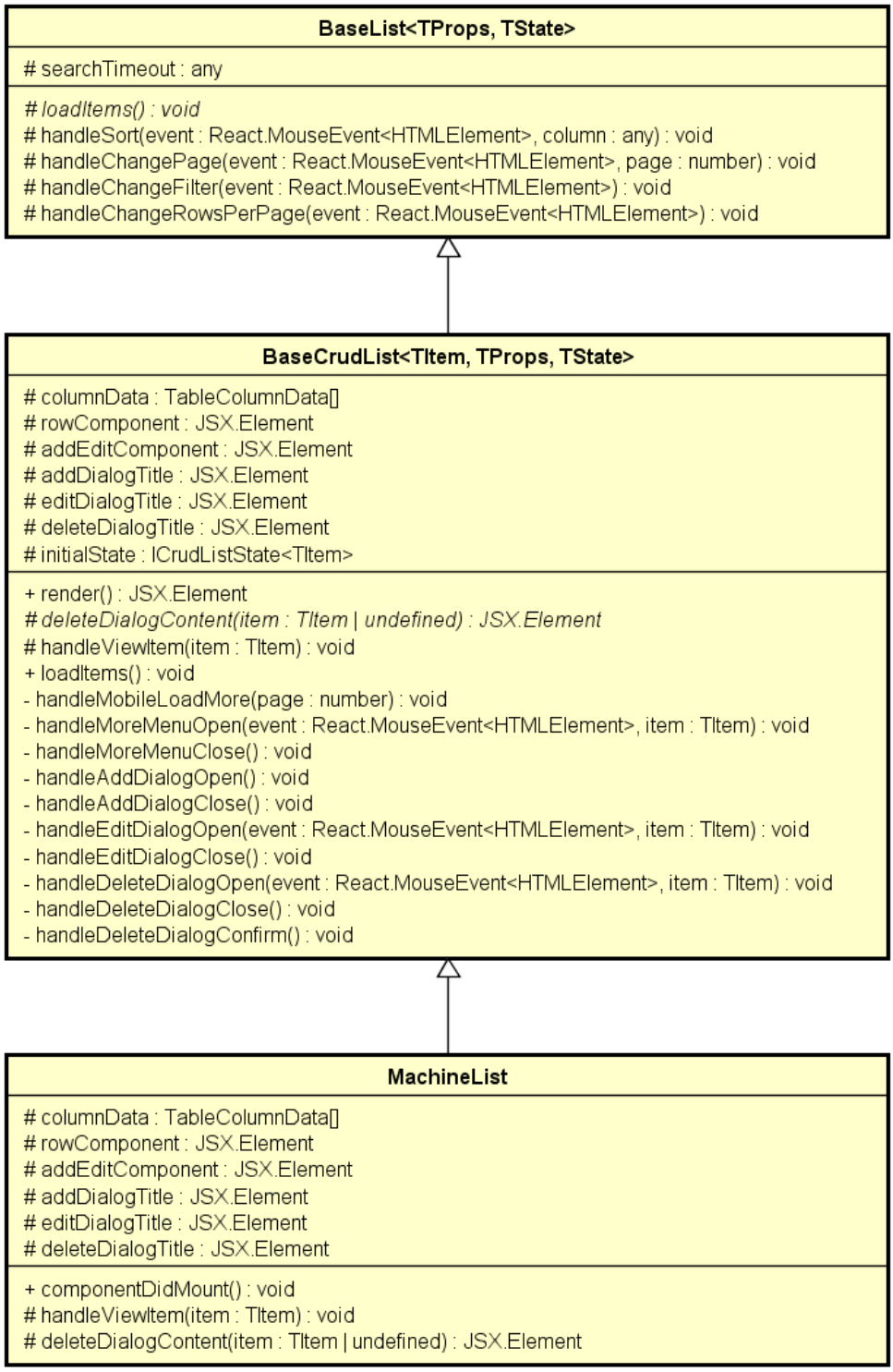


Abbildung 20: Klassendiagramm «BaseCrudList»

Wie dem Klassendiagramm entnommen werden kann, erbt die BaseCrudList-Komponente von der zusätzlichen Komponente «BaseList», welche die grundlegende Listen-Funktionalität (Sortierung, Filterung, Pagination) beinhaltet. Die BaseCrudList-Komponente gibt dann wiederum das Design und die Struktur vor. Die Individualisierung der verschiedenen Sub-Komponenten entsteht anschliessend durch die Überschreibung der geerbten Properties, wie am Beispiel der «MachineList» zu sehen ist. Des Weiteren wird die BaseList-Komponente auch von allen anderen «*%Sektion%*List»-Komponenten geerbt.

3.2.8.3 BaseDetail

Die BaseDetail-Komponente beinhaltet die Struktur, Darstellung sowie die grundlegende Funktionalität für die Detailansicht eines generischen Elementes.

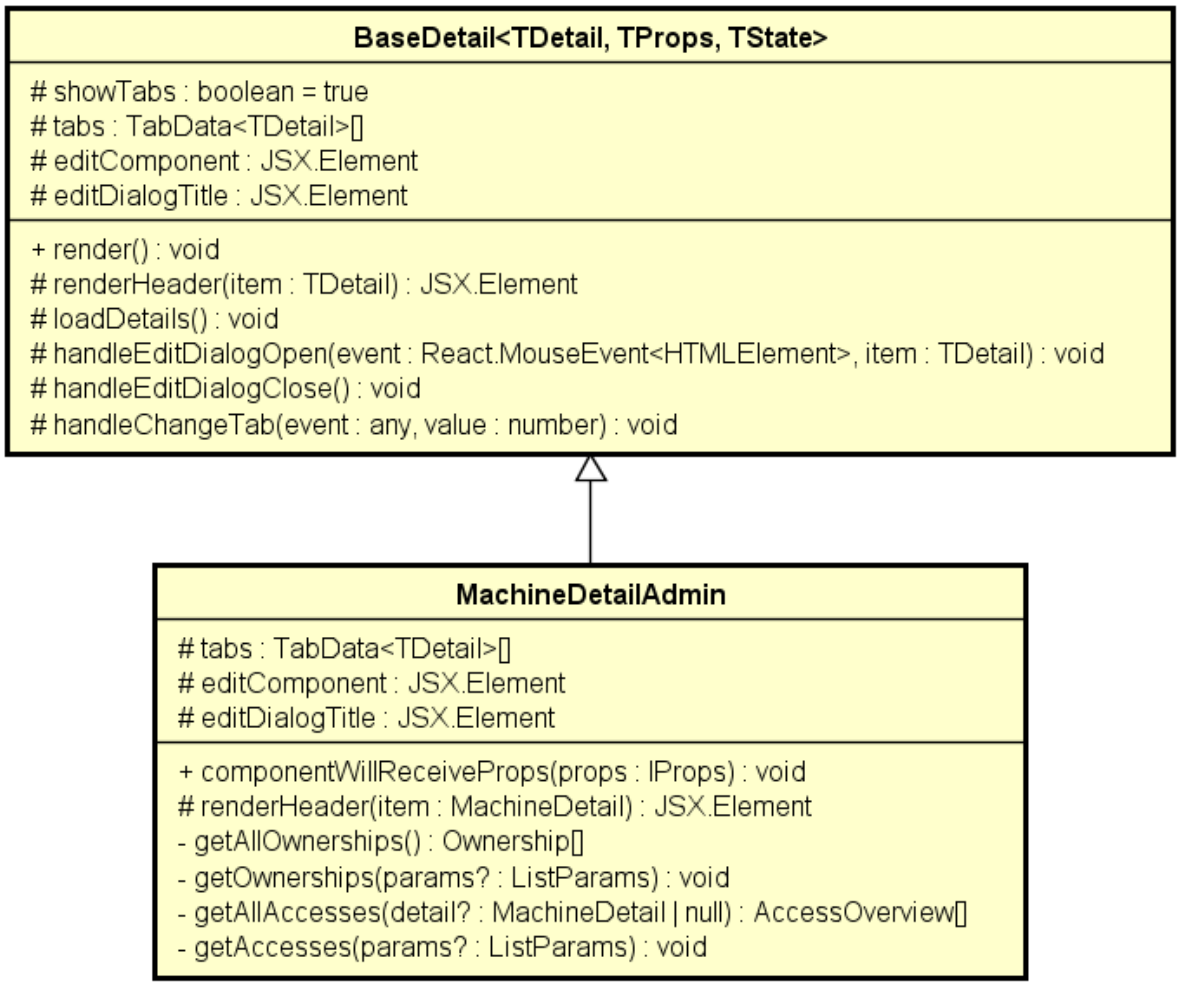


Abbildung 21: Klassendiagramm «BaseDetail»

Wie im obigen Klassendiagramm ersichtlich ist, können durch das Überschreiben der Properties und Methoden die Tabs, deren Inhalte sowie der Kopfbereich bestimmt werden. Des Weiteren zeigt sich am Beispiel der Komponente «MachineDetailAdmin», dass zusätzliche Methoden benötigt werden, um beispielsweise weitere Informationen nachzuladen. Die BaseDetail-Komponente wird von allen «*%Sektion%*DetailAdmin»-Komponenten und zusätzlich von den Komponenten «MachineInfo» und «UserProfile» geerbt.

3.2.8.4 BaseRow

Die BaseRow-Komponente entspricht einer generischen Zeile in einer Auflistung der BaseCrud-List-Komponente.

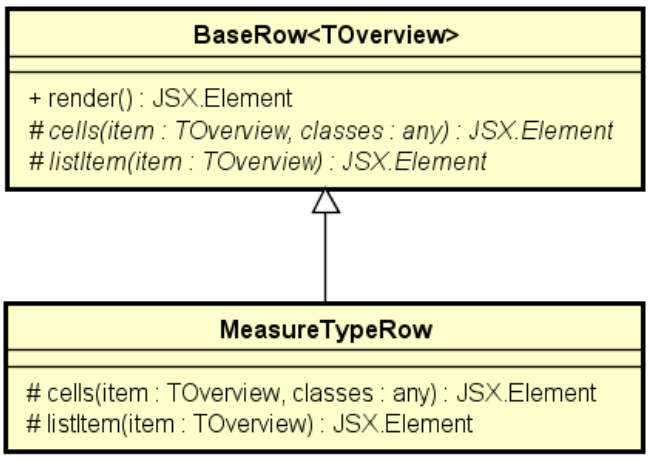


Abbildung 22: Klassendiagramm «BaseRow»

Wie im Klassendiagramm ersichtlich ist, stehen in der BaseRow-Komponente zwei abstrakte Methoden bereit, welche durch die Sub-Komponente überschrieben werden müssen. Dabei wird mit der Methode «cells» die Zellen einer Tabellen-Zeile und mit der Methode «listItem» einen Listeneintrag für die Mobile-Ansicht generiert. Nebst der als Beispiel verwendeten MeasureTypeRow-Komponente erben auch alle anderen «*%Sektion%*Row»-Komponenten die BaseRow-Komponente.

3.2.8.5 Weitere Base-Komponenten

Nebst den zuvor beschriebenen Base-Komponenten, welche über sehr viele Sub-Komponenten verfügen, gibt es noch weitere Base-Komponente, die in der nachfolgenden Tabelle aufgeführt und beschrieben sind.

Base-Komponente	Beschreibung
BaseDateRangePicker	Base-Komponente für alle Page-Komponenten, welche die Auswahl eines Zeitfensters beinhalten: MeasurementChart, EventList, GpsMap
BaseMachineFilter	Base-Komponente für alle Page-Komponente, welche die Filterung der Maschinen-Auflistung nach Maschine, Maschinentyp und Live-Status beinhalten: MachineCards, MachinesMap
BaseMap	Base-Komponente für alle Page-Komponente, welche eine Karte von Google Maps integriert haben: GpsMapContent, MachinesMapContent

3.3 Authentifizierung / Berechtigungen

An verschieden Orten im System werden Berechtigungen vergeben, geprüft und entzogen. Dieses Kapitel beschreibt die unterschiedlichen Subsysteme sowie die dabei verwendeten Verfahren.

3.3.1 Frontend

Das Frontend nimmt die Zugangsdaten vom Benutzer beim Login entgegen und gibt diese zur Validierung an die interne API weiter. Dabei erfolgt die Übertragung des Benutzernamens und Passworts über einen mit HTTPS gesicherten Kanal und ist somit vor unerwünschten Abhörversuchen sicher.

Bei erfolgreicher Validierung der Benutzerdaten erhält das Frontend von der internen API ein JWT-codiertes Access Token, welches vom Backend signiert, verschlüsselt und mit der aktuellen UserId sowie der aktuellen Benutzerrolle versehen ist. Dieses Access Token besitzt eine maximale Gültigkeit von 15 Minuten. Damit der Benutzer nicht alle 15 Minuten seine Zugangsdaten eingeben muss, wird nebst dem Access Token ein länger gültiges Refresh Token ausgestellt, welches ebenfalls signiert und verschlüsselt ist. In diesem Refresh Token ist aber lediglich die UserId und keine Rollen im JWT-Payload enthalten. Es wird im Hintergrund automatisch verwendet, um nach Ablauf eines Access Token ein neues Token bei der internen API anzufordern. Falls sich in der Zwischenzeit die Rolle des Benutzers geändert hat, dann berücksichtigt das neue Access Token diese Änderung.

Das geschilderte Verfahren hat den Vorteil, dass es keinen State, also beispielsweise eine SessionId auf der Serverseite, benötigt. Durch das Verschlüsseln und Signieren des JWT Token wird der Session-State auf den Client ausgelagert, gleichzeitig ist aber sichergestellt, dass der Client seine eigenen Access Token nicht manipulieren kann. Dazu würde er das Secret des Backend WebAPI Servers benötigen, um die Signatur des Access bzw. Refresh Tokens zu erstellen. Dieses Secret verlässt aber nie den Backend WebAPI Server und somit sind die JWT Tokens vor Manipulation geschützt. Das Access Token und das Refresh Token werden auf Seiten des Clients im Local Storage abgelegt.

3.3.2 Internal API

Die Interne API wird mittels eines JWT Access Tokens abgesichert. Dabei wird das Access Token in einem HTTP-Header «Authentification: Bearer <token>» übertragen. Falls das übertragene Access Token fehlt oder nicht mehr gültig ist, nicht oder ungültig signiert wurde oder nicht dem erwarteten Format entspricht, wird der Request mit dem HTTP-Status-Code 401 (Unauthorized) abgewiesen.

Falls der Zugriff auf eine nicht sichtbare Ressource versucht wird, wird der HTTP-Status-Code 403 (Forbidden) zurückgegeben. Im Falle, dass es sich bei dem versuchten Zugriff um eine besonders schützenswerte Ressource handelt, bei der auch deren Existenz vor unerlaubten Zugriffen geschützt werden soll (z.B. existiert ein User mit einer bestimmten Id oder E-Mailadresse), dann wird bei unerlaubtem Zugriff statt eines 403 (Forbidden) ein 404 (Not Found) zurückgegeben. Damit kann ein potenzieller Angreifer keine Rückschlüsse über das Existieren einer Ressource anstellen.

3.3.3 ISO API

Die ISO 15143-3 schreibt die Absicherung der API mittels OAuth1.0A oder OAuth2.0 gemäss RFC 6749[12] vor. Für die vorliegende Problemstellung wurde OAuth2.0 umgesetzt. Speziell an der Implementation von OAuth2.0 im Rahmen dieses Projektes ist, dass die Rollen des «Resource Owner», «Authorization Server» als auch die des «Resource Servers» alle durch die gleiche Partei, in diesem Fall SUNCAR HK AG, übernommen werden.

Der Resource Owner, in diesem Falle vertreten durch das Frontend, erzeugt ein API Token, welches ein Satz «client_credentials» im Sinne von RFC 6749 Abschnitt 4.4 darstellt. Diese client_credentials werden im API-Consumer hinterlegt und sind standardmässig 360 Tage gültig (Dauer kann in der Systemkonfiguration definiert werden). Ein API Token wird mit der UserId des erstellenden Benutzers versehen. Es handelt sich um ein signiertes und verschlüsseltes JWT Token, dessen Funktion dem Refresh Token der internen API sehr ähnlich ist. In diesem Token ist nur die UserId gespeichert. Will der API-Consumer auf die ISO 15143-3 API zugreifen, dann muss er gemäss RFC 6749 Abschnitt 4.4 zuerst beim Authorization Server mit seinem API Token (client_credentials) ein temporär gültiges Access Token anfordern. Dieses Access Token kann er anschliessend für den Zugriff auf die geschützte API (Resource Server) benutzen. Das ausgestellte Access Token ist wie das Access Token im Frontend nur für eine kurze Zeitspanne gültig und muss nach Ablauf erneuert werden.

Die Speicherung des API Tokens hat an einem sicheren Ort zu erfolgen und muss durch den API-Consumer sichergestellt werden. Beim Verlust des API Tokens muss dieses durch den API-Consumer im Frontend als «Revoked» gekennzeichnet werden. Wenn dies geschehen ist, kann mit dem entsprechenden API Token kein gültiges Access Token mehr beim Authorization Server erzeugt werden, da dieser eine Gültigkeitsprüfung vollzieht.

3.3.4 Kapacitor API

Die Data Processing Engine hat die Möglichkeit, erkannte Alerts im Datenstrom an eine JSON-API zu melden. Diese Funktionalität wird in diesem System genutzt, um beispielsweise E-Mails bei erkannten Unstimmigkeiten in den Daten zu versenden. Da diese API jedoch vom Backend zur Verfügung gestellt wird, muss sie entsprechend abgesichert werden.

Im Idealfall befinden sich Backend und Kapacitor im gleichen Subnetz, sodass Netzwerktechnisch sichergestellt werden kann, dass nur interne und damit vertrauenswürdige Clients die entsprechende API aufrufen können. Im aktuellen Deployment ist dies der Fall und damit sind die im folgenden beschriebenen Massnahmen theoretisch nicht nötig

Da allerdings je nach Deployment die Kapacitor Event API auch über das Internet verfügbar sein muss, weil beispielsweise der Kapacitor nicht im selben Subnetz oder bei einem anderen Cloud Anbieter gehostet wird, muss diese API auch geschützt werden.

[12] (RFC 6749: The OAuth 2.0 Authorization Framework, 2012)

Die Möglichkeiten sind hier allerdings durch technische Limitationen auf Seiten von Kapacitor eingeschränkt. In der aktuellen Version von Kapacitor ist es lediglich möglich, eine URL anzugeben, auf welcher ein Event gemeldet werden soll. Somit muss auch die Authentifizierung URL-basiert sein.

Daher wird sowohl in der Kapacitor-Konfiguration als auch im Backend ein gemeinsames Token hinterlegt. Die Backend WebAPI prüft bei einem Aufruf auf den entsprechenden Endpoint, ob das korrekte Token als GET-Parameter übergeben wurde und erlaubt den Zugriff nur, wenn das Token mit dem eigenen Token übereinstimmt. Da die Übertragung in jedem Fall per HTTPS stattfindet, kann das Token direkt in der URL übertragen werden.

3.3.5 Validierung mittels automatisierten End-to-End Tests

Um die Konformität der internen API mit den im Dokument «Domainanalyse» definierten Berechtigungstabelle zu gewährleisten und auch vor zukünftigen unberücksichtigten Code-Änderungen zu schützen, wurden automatisierte API-Tests mittels Cypress implementiert. Basierend auf einem Daten-Szenario wird versucht, auf geschützte Ressourcen zuzugreifen. Anschliessend wird der HTTP-Status-Code sowie der erhaltene Response Body überprüft. Diese Tests sind zudem als hartes Release Gate im VSTS eingerichtet. So ist ein Deployment auf das Live-System nur möglich, wenn alle API-Zugriffstests erfolgreich waren, d.h. keine unbeabsichtigte Sicherheitslücke im Sinne von Abschnitt 4.5 möglich ist.

3.3.1 Data Collector

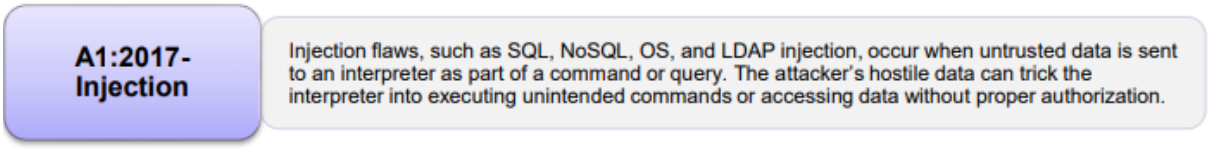
Die Verifizierung zugelassener Maschinen, welche Messdaten an den Data Collector liefern dürfen, findet mittels eines Client-Zertifikats statt, welches von einer durch SUNCAR HK AG betriebene Certificate Authority signiert sein muss. Für detailliertere Informationen sei an dieser Stelle auf das Dokument «Schnittstellenkonzept» verwiesen.

4 Sicherheitsmassnahmen

Dieses Kapitel beleuchtet einige gängige Angriffsvektoren auf eine Webapplikation und die im Rahmen dieses Projektes getroffenen Massnahmen, um einen entsprechenden Angriff zu verhindern. Dabei wurden die OWASP Top 10 Security Risks[13] wo sinnvoll als Vorlage genommen.

Zudem wird beschrieben, welche Massnahmen und Abklärungen vor bzw. während einer produktiven Inbetriebnahme und bei einer allfälligen Weiterentwicklung des Projektes beachtet werden müssen.

4.1 A1 – SQL/Query Injection



Injection-Angriffsvektoren können in der vorliegenden Applikation in der Theorie sowohl auf die MariaDB und die InfluxDB ausgeführt werden. Im Falle der MariaDB setzen wir auf das Entity Framework Core und schreiben somit keine eigenen SQL-Queries. Da das Entity Framework Datenbank-Abfragen grundsätzlich als Prepared Statement ausführt, sind Standard-Abfragen mittels LINQ-To-SQL bereits geschützt.[14] Falls jedoch eigene SQL-Fragmente oder Filter erstellt werden, dann ist immernoch ein Angriff möglich. Gegen diese Art von Vulnerability haben wir uns mit regelmässigen Code Reviews bzw. dem 4-Augen-Prinzip für einen Pull Request auf den Master-Branch abgesichert.

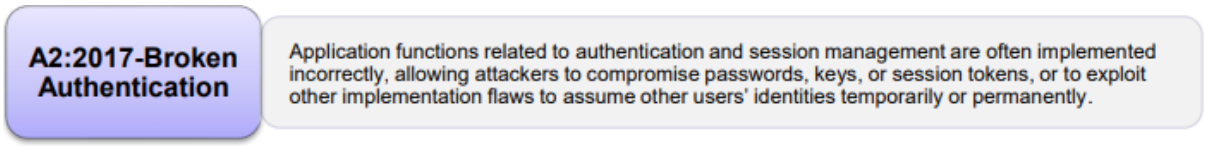
Im Falle von InfluxDB existiert keine ähnliche Technologie wie Entity Framework. Trotzdem kann dieser Angriffspunkt als weitgehend entschärft bewertet werden, da in den Queries auf Influx prinzipiell nur Zahlen als variable Teile enthalten sind. Wo Zeitpunkte als Argumente verwendet werden müssen, sind diese stets in eine entsprechende DateTime-Struktur verpackt und niemals direkt der rohe String. Zudem galt hier die Devise, dass Pull Requests mit Influx Queries besonders genau unter die Lupe zu nehmen waren.

Dem ganzen Projektteam ist während der Entwicklung stets bewusst gewesen, dass sämtliche Informationen, welche von der API/als Web Request hereinkommen, als prinzipiell nicht vertrauenswürdig behandelt werden müssen. Kritische Funktionen wurden stets im Team diskutiert und auf mögliche Angriffsvektoren hin untersucht.

[13] (OWASP, 2017)

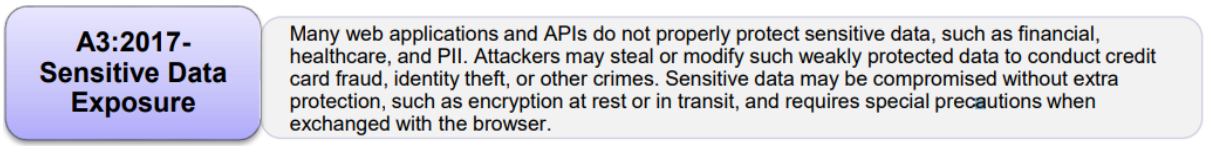
[14] (Overview of ASP.NET Core Security, 2017)

4.2 A2 – Broken Authentication



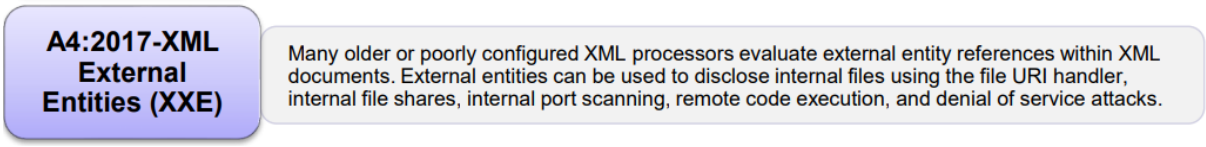
Die Verfahren zur Authentifizierung bzw. der Sicherstellung, dass nur autorisierte Benutzer auf schützenswerte Ressourcen Zugriff erhalten, wurden bereits in Kapitel 3.3 (Authentifizierung / Berechtigungen) erläutert.

4.3 A3 – Sensitive Data Exposure



In diesem Projekt wurde grossen Wert auf den Schutz der Daten vor unzulässigem Zugriff gelegt. Die für diesen Punkt relevanten Massnahmen sind im Detail im Abschnitt 4.5 erläutert.

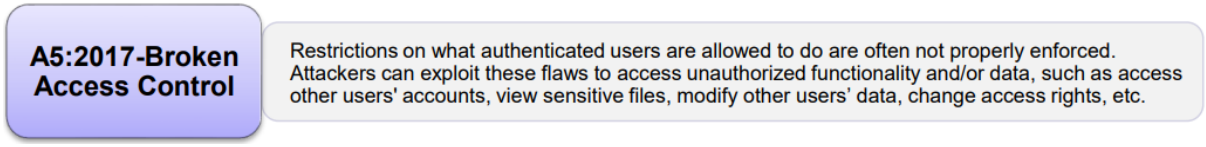
4.4 A4 – XML External Entities XXE



Das vorliegende Projekt konsumiert an keinem Punkt XML als Input. Die internen WebAPIs werden ausschliesslich als JSON angesprochen und sind daher vom Backend auch nur auf JSON verfügbar. Die externe ISO 15143-3 API gibt zwar sowohl JSON als auch XML als Response zurück, es handelt sich jedoch um eine reine Query API, bei welcher allfällige Request-Parameter in der Request URL übergeben werden.

Sollten in Zukunft auch XML-Requests benötigt werden, dann muss mithilfe des (XML External Entity (XXE) Prevention Cheat Sheet, 2018) abgeklärt werden, welcher XML-Parser verwendet wird und ob dieser korrekt konfiguriert ist.

4.5 A5 – Broken Access Control



Der Schutz vor Zugriffen auf Informationen, welche eigentlich nicht abrufbar sein sollten, hat einen grossen Stellenwert in diesem Projekt erhalten. So wurde eine ausführliche Tabelle mit Zugriffsregeln erstellt, welche während der Entwicklung immer wieder als Nachschlagewerk benutzt werden konnte. Um die Applikation auch zukünftig und nachhaltig vor diesem Angriffsvektor zu schützen, wurden umfassende und automatisierte End-to-End Tests der API implementiert. Damit wird automatisch im Hintergrund bei jeder Code-Änderung geprüft, ob die Zugriffskontrolle immer noch wie definiert zutrifft.

4.6 A6 – Security Misconfiguration

A6:2017-Security Misconfiguration	Security misconfiguration is the most commonly seen issue. This is commonly a result of insecure default configurations, incomplete or ad hoc configurations, open cloud storage, misconfigured HTTP headers, and verbose error messages containing sensitive information. Not only must all operating systems, frameworks, libraries, and applications be securely configured, but they must be patched and upgraded in a timely fashion.

Um dieses Risiko zu minimieren, wurde die Applikation auf Docker aufgebaut. Damit entfällt die ganze Thematik bezüglich Patching/Updating des Betriebssystems, da dies dem Docker Cloud Provider überlassen werden kann. Für den produktiven Betrieb muss daher zwingend von der aktuellen selber gepflegten VM auf einen Docker Anbieter gewechselt werden oder die Thematik der Misconfiguration der VM inklusive einer Update/Patching-Strategie festgelegt werden.

Die Standardkonfigurationen der verwendeten Docker-Container sowie Bibliotheken wurden bei der Inbetriebnahme vom Projektteam jeweils aus Security-Sicht kritisch hinterfragt und wo nötig entsprechend angepasst.

Leider gibt es keine Garantie, dass alle möglichen Angriffsvektoren ausgeschlossen werden. Es bleiben nur das regelmässige Review der verwendeten Komponenten und bei Bedarf das Einspielen eines Security-Patches. Um dies zu erleichtern, wurden die einzelnen Komponenten so weit als möglich getrennt und können somit separat ausgetauscht oder aktualisiert werden.

Mit dem Einsatz von Docker können zudem auf standardisierte und durch Experten geprüfte Container und Konfigurationen verwendet werden.

4.7 A7 – Cross-Site Scripting (XSS)

A7:2017-Cross-Site Scripting (XSS)	XSS flaws occur whenever an application includes untrusted data in a new web page without proper validation or escaping, or updates an existing web page with user-supplied data using a browser API that can create HTML or JavaScript. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.

Die entwickelte Applikation ist hauptsächlich zum Darstellen von Daten konzipiert und bietet daher recht wenig Angriffsfläche für eine XSS. So kann ein normaler Benutzer nur sehr wenig Daten eingeben, welche in der Datenbank persistiert werden und potenziell von anderen Benutzern angesehen werden können (Stored XSS-Angriff). Die verwendete UI-Bibliothek React bietet von Haus aus bereits Schutz vor XSS Angriffen. So muss ein String, welcher HTML-Inhalt hat (und damit potenziell eine XSS ermöglicht), falls es denn gefordert ist, explizit als «unsafe» gekennzeichnet werden[15]. Die dabei verwendete Funktion trägt den passenden Namen «dangerouslySetInnerHTML». Das ermöglicht es, in Code Reviews gezielt nach heiklen Code-Passagen zu suchen und diese besonders kritisch zu hinterfragen.

Um einen erfolgreichen XSS-Angriff weiter zu verhindern, sollte für den Live-Betrieb der Einsatz einer Content Security Policy (CSP) erwogen werden. Diese verhindert im Falle einer erfolgreichen XSS die Exfiltration allfälliger Daten durch den Angreifer, da Ressourcen der Website nur noch von bekannten Orten geladen werden.

[15] (JSX Prevents Injection Attacks, 2018) & (DOM Elements, 2017)

4.8 A8 – Insecure Deserialization

A8:2017-Insecure Deserialization	Insecure deserialization often leads to remote code execution. Even if deserialization flaws do not result in remote code execution, they can be used to perform attacks, including replay attacks, injection attacks, and privilege escalation attacks.

Die Deserialisierung erfolgt immer in DTOs und mit den in zahlreichen andere Projekten auch verwendeten ASP.NET Core Serialisierungs-Funktionalitäten. Nach der Deserialisierung in die DTOs, werden diese je nach Rolle des Benutzers an einen verarbeitenden BackgroundService weitergegeben. Bei Änderungen an der Datenbank, bei denen der User nur ein Subset der Attribute eines Entitys modifizieren darf, werden die erlaubten Attribute per Whitelisting an den Service mitgegeben, welcher in der Folge nur Updates auf erlaubte Attribute zulässt.

4.9 A9 – Using Components with Known Vulnerabilities

A9:2017-Using Components with Known Vulnerabilities	Components, such as libraries, frameworks, and other software modules, run with the same privileges as the application. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications and APIs using components with known vulnerabilities may undermine application defenses and enable various attacks and impacts.

Um dem Betreiber der Applikation das Verfolgen der einzelnen Komponenten zu vereinfachen, bzw. allfällige Sicherheitslücken in den Komponenten schnellstmöglich zu erkennen, wurden im Anhang alle verwendeten externen Module aufgelistet. Es ist jedoch unumgänglich, dass auch während dem Betrieb stets ein Auge auf sicherheitskritische Updates dieser Komponenten geworfen wird.

Durch die Aufteilung der Applikation in mehrere, separat «hostbare» Docker-Container kann der mögliche Schaden durch die Kompromittierung eines Docker-Containers zumindest auf einen Teilbereich eingegrenzt werden. Zudem wurden die Rechte zwischen den einzelnen Komponenten des Systems so restriktiv als möglich eingesetzt. Die Staging- und die produktive Umgebung sind komplett getrennt und haben keine geteilten Ressourcen (ausser eines eventuell gemeinsamen Docker-Hosts, welcher aber bei Bedarf auch separiert werden kann).

4.10 A10 – Insufficient Logging & Monitoring

A10:2017-Insufficient Logging & Monitoring	Insufficient logging and monitoring, coupled with missing or ineffective integration with incident response, allows attackers to further attack systems, maintain persistence, pivot to more systems, and tamper, extract, or destroy data. Most breach studies show time to detect a breach is over 200 days, typically detected by external parties rather than internal processes or monitoring.

Zugriffe und Aktionen werden in allen Komponenten bereits mittels des Logging Frameworks NLog erfasst. Im aktuellen Deployment werden diese Log-Daten allerdings noch nicht an einem zentralen Ort zusammengefasst und archiviert. Für den produktiv-Betrieb muss zwingend noch ein solches zentrales Logging und Monitoring System konfiguriert werden. Hier gibt es diverse SaaS Anbieter (z.B. Logz.io, Splunk, Microsoft Azure Log Analytics), welche beim Einsatz eines solchen Log Collectors helfen.

5 Cloud Ready – Die zwölf Faktoren

Das Zwölf Faktor-Modell[16] wurde mit dem Ziel entwickelt, die wichtigsten Hinweise an Entwicklung, Deployment und Monitoring von Applikationen, welche «As-a-service» angeboten werden, festzuhalten. Dieses Kapitel vergleicht die vorliegende Architektur mit den zwölf Faktoren und ermöglicht so eine grobe Einschätzung, wie einfach es sein wird, die Applikation in Zukunft zu skalieren.

5.1 Codebase

«Eine im Versionsmanagementsystem verwaltete Codebase, viele Deployments»

Der gesamte Sourcecode der Applikation ist in einem einzigen GIT-Repository abgelegt und verwaltet. Mittels der unterschiedlichen Branches «master» und «production» werden aus derselben Codebase mehrere Deployments abgeleitet.

➔ Dieser Faktor ist erfüllt.

5.2 Abhängigkeiten

«Abhängigkeiten explizit deklarieren und isolieren»

Die externen Abhängigkeiten werden im Falle der .NET Applikationen im Project-File deklariert und durch Nuget automatisch heruntergeladen/installiert. Wenn .NET Core auf dem Deploy-Target installiert ist, können somit auch das Backend und der Data Collector betrieben werden.

Im Frontend-Projekt werden externe Abhängigkeiten über den Package Manager npm verwaltet und installiert. Dabei wird zwischen «Dev-Dependencies» (Tools/Bibliotheken zur Entwicklung) und «Dependencies» (Bibliotheken für den Betrieb) unterschieden. Die Abhängigkeiten werden mittels «npm install» installiert und aktualisiert.

Für den Betrieb der Applikation sind mehrere Docker-Container notwendig. Diese sind in einem docker-compose.yml File festgehalten und können mittels «docker-compose build && docker-compose up» gestartet werden.

➔ Dieser Faktor ist erfüllt.

5.3 Konfiguration

«Die Konfiguration in Umgebungsvariablen ablegen»

Die Konfiguration einer Instanz der Applikation kann, wie von diesem Faktor gefordert, ausschliesslich über Umgebungsvariablen erfolgen. Sowohl der Konfigurations-Mechanismus von .NET Core (Backend und Data Collector) als auch der der React-App (Frontend) unterstützen Umgebungsvariablen als Konfigurations-Quelle.

[16] (Wiggins, 2017)

Wie im Kapitel «Deployment» des Dokuments «Continuous Integration / Continuous Delivery (CI/CD)» ersichtlich ist, wird diese Art der Konfiguration auch für die produktive und die Staging-Umgebung angewendet.

➔ Dieser Faktor ist erfüllt.

5.4 Unterstützdende Dienste

«Unterstützende Dienste als angehängte Ressourcen behandeln»

Unterstützende Dienste wie beispielsweise die MariaDB-Datenbank oder Azure Blob Storage werden über Umgebungsvariablen konfiguriert und können so sehr einfach ausgetauscht werden. Die Message Queue zur Zwischenspeicherung der eigehenden Messdaten wird ausschliesslich über das MQTT Protokoll angesprochen, wodurch die Koppelung an die tatsächlich verwendete Message Queue Implementation so gering als möglich gehalten wird. Tatsächlich wurde während der Entwicklung die verwendete Message Queue mehrfach ausgetauscht bzw. die Implementation wurde gewechselt.

➔ Dieser Faktor ist erfüllt.

5.5 Build, release, run

«Build- und Run-Phase strikt trennen»

Das Builden, Testen und Releasen dieses Projektes wurden mit Hilfe von Visual Studio Team Services weitgehend automatisiert. So wird beim Mergen eines Pull Requests in den Master-Branch automatisch die Staging-Umgebung mit der neusten Version aktualisiert. Gleiches geschieht beim Mergen in den Production-Branch für die produktive Version der Applikation.

Dank VSTS erhält jeder Build eine eindeutige Identifikation und kann erneut deployt werden. Zudem ist die VM, welche das produktive System hostet, so konfiguriert, dass sie jederzeit neu gestartet werden kann und die aktuell laufenden Deployments automatisch neu gestartet werden. Da das Projekt mit Docker-Containern implementiert wurde, handelt es sich hierbei lediglich um das Konfigurieren eines Cron-Jobs, welcher @reboot den Befehl «docker-compose up» ausführt.

Allerdings ist die Build- und Release-Phase nicht strikt getrennt. So gibt es unterschiedliche Build-Definitionen für Staging und Production für das Frontend. Dies hat den Grund darin, dass gewisse Konfigurationen (URL der API, Google Maps API Token und dergleichen) direkt in den Frontend-Code hereinkompiliert werden. Dies ist notwendig, da das Frontend aus Performance-Gründen als statische Ressourcen gehosted werden und somit keine Konfigurationen mehr zur Laufzeit geändert werden können.

➔ Dieser Faktor ist aufgrund obiger Restriktion nur teilweise erfüllt.

5.6 Prozesse

«Die App als einen oder mehrere Prozesse ausführen»

Ein grosses Thema in diesem Faktor ist die Implementation der einzelnen Komponenten als Stateless Services. Das heisst, dass alle Daten, welche längere Zeit überleben müssen, ausserhalt der Prozesse in den unterstützenden Diensten gespeichert werden sollen. Dies ermöglicht das problemlose Neustarten eines einzelnen Prozesses ohne Datenverlust.

Alle Komponenten der Applikation sind als eigener Docker-Container implementiert und haben im Falle der Datenbank-Container ein Shared Directory eingebunden. Es wird ansonsten kein Zustand innerhalb der Docker-Container gehalten, sodass diese jederzeit neu gestartet werden können, ohne dass Daten verloren gehen.

Da die Website als Single Page Application (SPA) vor allem im Browser läuft und das Backend über eine stateless API angesprochen wird, muss anders als bei einer klassischen Webapplikation kein State innerhalb des Backends gespeichert werden. Die für die Authentifizierung notwendigen JWT Refresh Token werden in der MariaDB-Datenbank gespeichert. Die Access Token, welche den Zugriff auf die API gewähren, werden mit einem nur dem Backend bekannten Key verschlüsselt, mit einem Ablaufdatum versehen und signiert. Damit können diese notwendigen Session-Daten auf dem Client gespeichert werden. Durch das Verschlüsseln und Signieren sind die Daten vor Manipulation und Einsicht geschützt.

➔ Dieser Faktor ist erfüllt.

5.7 Bindung an Ports

«Dienste durch das Binden von Ports exportieren»

Die verschiedenen Komponenten kommunizieren ausschliesslich über Ports miteinander. Diese sind in einem zentralen File (docker-compose.yml) sehr einfach sichtbar. Durch die Verwendung von internen Docker-Netzwerken und einem vorgeschalteten HAProxy Server, werden auch nur diejenigen Ports nach aussen veröffentlicht, welche tatsächlich auch sichtbar sein müssen.

➔ Dieser Faktor ist erfüllt.

5.8 Nebenläufigkeit

«Mit dem Prozess-Modell skalieren»

Da jede Komponente als eigener Docker-Container läuft, kann jederzeit eine zusätzliche Instanz gestartet werden. Dies bedingt natürlich einen vorgestellter Load Balancer. Aktuell ist dieser mangels Cloud Anbieter, der alle unsere Anforderungen abdeckt, noch manuell mittels eines HAProxys realisiert. Der manuelle Aufwand zum Hochfahren einer weiteren Instanz ist daher nicht vernachlässigbar. Allerdings wurde die Applikation so entworfen, dass sie beispielsweise in einem Kubernetes-Cluster deployt werden kann, wo das Load Balancing von Kubernetes übernommen wird. An den eigentlichen Containern müssen für ein Deployment in ein Kubernetes-Cluster keine Änderungen mehr vorgenommen werden.

Um die bestehenden Ressourcen bestmöglich auszunutzen, verwenden das Backend und der Data Collector wo möglich asynchrone I/O-Operationen. Damit kann der Durchsatz im Vergleich

zum Ansatz «One-Thread-Per-Request» markant verbessert werden, weil die WepAPI Requests vor allem I/O-lastig sind.

➔ Dieser Faktor ist prinzipiell erfüllt. Bei einer allfälligen horizontalen Skalierung ist jedoch noch mit kleineren Anpassungen zu rechnen.

5.9 Einweggebrauch

«Robuster mit schnellem Start und problemlosen Stopp»

Dank dem Einsatz von Docker, ist das schnelle Starten und Stoppen von beliebigen Containern jederzeit möglich. Einzig im Falle, dass eine neue Version zum ersten Mal deployt wird, dauert der Startup eventuell etwas länger, da allfällige Datenbank-Migrationen noch durchlaufen müssen. Sämtliche Komponenten wurde fehlertolerant entworfen, sodass ein fehlender Docker-Container oder unterstützender Dienst nicht gleich das gesamte System in die Knie zwingt.

Es lässt sich allerdings nicht vermeiden, dass die Backend-API beim Fehlen der MariaDB nicht mehr einsetzbar ist, da jeder Request in irgendeiner Form auf die Datenbank zugreifen muss.

➔ Dieser Faktor ist erfüllt.

5.10 Dev-Prod-Vergleichbarkeit

«Entwicklung, Staging und Produktion so ähnlich wie möglich halten»

Dank dem Einsatz von Docker können sämtliche Container auch auf der Entwicklungsumgebung lokal gestartet werden. Je nachdem, welcher Teil gerade bearbeitet wird, kann die Applikation auch direkt aus der Entwicklungsumgebung gestartet werden, sodass ein Debugger schnell angehängt werden kann. Externe unterstützende Dienste (z.B. E-Mail oder Blob Storage) wurden während der Entwicklung mit einem geeigneten lokalen Mock ersetzt. Damit wird verhindert, dass für jeden Entwickler ein eigener Blob Storage eingerichtet werden muss und die Latenzzeit wird verringert. So wird beispielsweise ein E-Mail nicht tatsächlich versendet, sondern das zugehörige HTML in ein temporäres File geschrieben und direkt im Browser geöffnet. Selbstverständlich können diese lokalen Mocks aber auch vom Entwickler deaktiviert und auf konkrete unterstützende Dienste umgeleitet werden.

Die produktive und die Staging-Umgebung unterscheiden sich lediglich in den Konfigurationen der unterstützenden Dienste.

➔ Dieser Faktor ist erfüllt.

5.11 Logs

«Logs als Strom von Ereignissen behandeln»

Das Logging ist mit NLog realisiert. Dieses Framework ermöglicht die Konfiguration der Log-Targets in einem externen File oder per Umgebungsvariablen. Es ist somit kein neues Erstellen des Projektes beim Anpassen des Loggings nötig.

Aktuell werden die Logs jedoch noch nicht an einem zentralen Ort gesammelt. Dies vor allem aus dem Grund, dass die Hosting-Umgebung aktuell noch nicht final ist. Momentan läuft die Applikation in einer selber vom Projektteam aufgesetzten virtuellen Maschine mit einer manuell installierten und konfigurierten Docker-Umgebung. Langfristig soll das Projekt in einer Docker-

SaaS Cloud (z.B. gehostetes Kubernetes) betrieben werden. Bei diesem Umstieg kann dann auch das Logging entsprechend konfiguriert werden.

➔ Dieser Faktor ist grundsätzlich erfüllt, im aktuellen Deployment jedoch noch nicht vollständig umgesetzt.

5.12 Admin-Prozesse

«Admin/Management-Aufgaben als einmalige Vorgänge behandeln»

Die administrativen Tasks zur Pflege der Applikation wurden weitgehend automatisiert. So wird die Datenbank beim Deployment einer neuen Version automatisch auf das neuste Datenbank-Schema migriert. Dies wird mit Hilfe der Migrationen des Entity Frameworks umgesetzt. Da jede Version vor dem Deployment in die Produktion mindestens einmal auf der Staging-Umgebung deployt wurde, werden fehlerhafte Migrationen, welche allenfalls zu inkonsistenten Daten führen könnten, bereits sehr früh erkannt. Zudem werden Pull Requests mit Datenbank-Änderungen besonders vorsichtig gereviewt und anschliessend auf der Staging Umgebung getestet. Dadurch konnten im gesamten Projektverlauf die produktiven Daten immer konsistent gehalten werden.

➔ Dieser Faktor ist erfüllt.

5.13 Fazit

Wie obigen Ausführungen entnommen werden kann, ist das während dieser Bachelorarbeit entstandene Projekt gemäss dem «Twelve Factor»-Modell bereits sehr gut gerüstet für ein Deployment in eine SaaS-Cloud mit High-Availability und Scalability-Anforderungen. Die Faktoren, welche nicht ganzheitlich erfüllt sind, wurden mit Absicht noch offengelassen, da die nötigen Informationen ohne konkreten SaaS-Cloud Provider noch nicht entschieden werden können. Zudem sind Entscheide, die erst bei einer stärkeren Auslastung relevant werden und je nach Art der Skalierung anders ausfallen, auch noch nicht getroffen, sodass sie zu gegebenem Zeitpunkt basierend auf den hoffentlich dann vorhandenen Informationen getroffen werden können.

Während den Stresstests und Systemtests hat sich gezeigt, dass die ganze Applikation schon sehr Cloud-Ready ist. So konnten wir innerhalb von 20 Minuten die ganze Applikation auf einem anderen Cloud Provider (Google Cloud) in Betrieb nehmen und dort testen, ob die Performance ähnlich ist wie in der Azure Cloud. Ebenfalls konnten wir die verwendete Message Queue bzw. den entsprechenden Docker Container sehr rasch austauschen und so herausfinden, welcher Teil der Applikation als erstes an seine Grenzen stösst und bei wie vielen Messwerten pro Sekunde dies der Fall ist.

6 Prozesse und Threads

Dieses Kapitel beschäftigt sich mit den unterschiedlichen Arten, wie mit nebenläufigen und parallelen Prozessen in dieser Arbeit umgegangen wird. Dabei werden die Bereiche Data Collector, Backend und Frontend unterschieden.

6.1 Data Collector

Der Data Collector wurde mit .NET Core entwickelt. Um einen möglichst hohen I/O-Durchsatz zu erreichen, wurde wo immer möglich eine Async API[17] bevorzugt. Um das Debugging zu erleichtern, wurde statt Task.Result jeweils Task.GetAwaiter().GetResult() verwendet. [18]

Um die Implementation des Legacy Collectors zu vereinfachen, wird hier pro konfigurierter Legacy Maschine ein Thread für den UDP-Listener sowie ein Thread für den zugehörigen MQTT-Client erstellt. Damit werden die einzelnen Legacy Clients sauber getrennt. Da die Anzahl der Legacy Clients ab dem Zeitpunkt, wo neue Maschinen direkt an die neue Schnittstelle angebunden werden (siehe Schnittstellenkonzept), nur noch sinken wird, muss der Legacy Client nicht auf Durchsatz optimiert werden, sondern eher auf Robustheit.

6.2 Backend

Das Backend ist mit ASP.NET Core implementiert, welches über die ganze Request/Response-Pipeline das Verwenden von Async-Methoden erlaubt und ermutigt. Alle API Requests, welche auf irgendeine Art I/O-Zugriffe machen, sind mittels Async/Await implementiert.

Da keine langlaufenden Jobs im Backend anfallen, kann auf das Verwenden eines Background Workers wie von (Cleary, 2014) beschrieben, verzichtet werden.

6.3 Frontend

JavaScript ist grundsätzlich eine Single Threaded Umgebung, d.h. nur ein Event Loop behandelt Ereignisse. Über sogenannte Web Worker[19] besteht zwar grundsätzlich die Möglichkeit, CPU-intensive Tasks im Hintergrund und parallel zum Event Loop zu verarbeiten. Allerdings sind solche CPU-Intensiven Tasks im vorliegenden Projekt nicht vorhanden und daher werden Web Workers nicht genutzt.

Trotzdem kommt es bei JavaScript-Applikationen immer wieder vor, dass Probleme durch Nichtdeterminismus insbesondere im Zusammenhang mit Rendering und Event Handling auftreten. So kann es beispielsweise vorkommen, dass das DOM und die interne Repräsentation nicht mehr synchron sind. Die verwendete UI-Bibliothek React bzw. die zugehörige State-Management-Bibliothek Redux erzwingen, dass der Datenfluss nur in eine Richtung geht und im DOM endet. Sämtliche Änderungen an den Daten werden mittels Redux Actions ausgelöst und

[17] (Asynchronous programming, 2016)

[18] (Marbach, 2018)

[19] (Web Workers API, 2018)

anschliessend von sogenannten Reducern durch den State propagiert. Aus der Sicht einer Komponente ist der State immer «immutable» und kann somit nicht direkt geändert werden. Mehr Informationen zu React/Redux ist im Abschnitt 3.2.5 (Constants, Actions, Services, Reducers) zu finden.

7 Abbildungsverzeichnis

8 Literaturverzeichnis

Actions. (6. März 2018). Abgerufen am 3. Juni 2018 von Redux: https://redux.js.org/basics/actions

Async Actions. (2018). Abgerufen am 8. Mai 2018 von Redux: https://redux.js.org/advanced/async-actions

Asynchronous programming. (20. Juni 2016). Abgerufen am 12. Juni 2018 von https://docs.microsoft.com/en-us/dotnet/csharp/async

AutoMapper. (2018). Abgerufen am 7. Mai 2018 von AutoMapper: https://automapper.org/

Cleary, S. (7. Juli 2014). *Fire and Forget on ASP.NET*. Abgerufen am 13. Juni 2018 von http://blog.stephencleary.com/2014/06/fire-and-forget-on-asp-net.html

DOM Elements. (10. Dezember 2017). Abgerufen am 12. Juni 2018 von React Docs: https://reactjs.org/docs/dom-elements.html

Entity Framework Core: Soft Delete using Query Filters. (7. Juli 2017). Abgerufen am 7. Mai 2018 von Meziantou's Blog: https://www.meziantou.net/2017/07/10/entity-framework-core-soft-delete-using-query-filters

Global Query Filters. (3. November 2017). Abgerufen am 7. Mai 2018 von Microsoft Docs: https://docs.microsoft.com/en-us/ef/core/querying/filters

JSX Prevents Injection Attacks. (9. Mai 2018). Abgerufen am 12. Juni 2018 von React Docs: https://reactjs.org/docs/introducing-jsx.html#jsx-prevents-injection-attacks

Marbach, D. (29. Mai 2018). *Refactor your codebase to async/await.* .NET Day 2018, Zürich.

NSwag. (Juni 2018). Abgerufen am 13. Juni 2018 von GitHub: https://github.com/RSuter/NSwag

Overview of ASP.NET Core Security. (1. November 2017). Abgerufen am 11. Juni 2018 von https://docs.microsoft.com/en-us/aspnet/core/security/?view=aspnetcore-2.0

OWASP. (2017). *OWASP Top 10.* Abgerufen am 12. Juni 2018 von https://www.owasp.org/index.php/top10

React. (2018). Abgerufen am 8. Mai 2018 von React: https://reactjs.org/

Reducers. (12. Mai 2018). Abgerufen am 3. Juni 2018 von Redux: https://redux.js.org/basics/reducers

RFC 6749: The OAuth 2.0 Authorization Framework. (Oktober 2012). Abgerufen am 22. April 2018 von IETF: https://www.ietf.org/rfc/rfc6749.txt

Store. (28. März 2018). Abgerufen am 3. Juni 2018 von Redux: https://redux.js.org/basics/store

Three Principles. (2018). Abgerufen am 8. Mai 2018 von Redux: https://redux.js.org/introduction/three-principles

Web Workers API. (4. März 2018). Abgerufen am 11. Juni 2018 von Mozilla: https://developer.mozilla.org/en-US/docs/Web/API/Web_Workers_API

What is Code-First? (2018). Abgerufen am 7. Mai 2018 von Entity Framework Tutorial: http://www.entityframeworktutorial.net/code-first/what-is-code-first.aspx

Wiggins, A. (2017). *The Twelve-Factor App*. Abgerufen am 11. Juni 2018 von https://12factor.net

XML External Entity (XXE) Prevention Cheat Sheet. (12. März 2018). Abgerufen am 11. Juni 2018 von OWASP: https://www.owasp.org/index.php/XML_External_Entity_(XXE)_Prevention_Cheat_Sheet#.NET

9 Anhang

9.1 Backend: Externe Module / Nuget Packages

Im Backend und DataCollector wurde, wie in grösseren Projekten üblich, nicht alles selber implementiert, sondern wo möglich auf existierende Bibliotheken und Frameworks zurückgegriffen. Die verwendeten NuGet-Packages werden an dieser Stelle kurz erläutert.

Modul	Einsatzzweck
AutoMapper	Automatisches Mapping der Entity-Klassen auf die DTOs
AutoMapper.Extensions.Microsoft.DependencyInjection	Extension von AutoMapper, welcher die Integration von AutoMapper mit der.NET Core Dependency Injection implementiert
InfluxData.Net	.NET Wrapper der InfluxDb/Kapacitor REST-API Da das Projekt nicht die allerneusten Funktionalitäten der Kapacitor-API abbildete, musste dieser Projekt geforkt, erweitert und als Submodul eingebunden werden.
Microsoft.AspNetCore.All	Zusammenfassendes Modul, welches die gängigen ASP .NET Core Module einbindet
Microsoft.AspNetCore.Mvc.Formatters.Xml	XML-Formatter Support für die ISO-15143-3 Schnittstelle
Microsoft.AspNetCore.TestHost	Ermöglicht In-Memory Tests der API-Controller
Microsoft.EntityFrameworkCore	Entity Framework für.NET Core Applikationen
Microsoft.EntityFrameworkCore.InMemory	In-Memory SQL Datenbank für schnelles Unit Testing
Microsoft.EntityFrameworkCore.Relational	Modul für den Zugriff auf relationale Datenbanken aus Entity Framework Core
Microsoft.Extensions.Configuration.Binder	Ermöglicht direktes Mapping der Konfigurationen auf Objekte
Microsoft.Extensions.Configuration.EnvironmentVariables	Konfiguration von .NET Core Applikationen mittels Umgebungsvariablen
Microsoft.Extensions.Configuration.Json	Konfiguration von .NET Core Applikationen mittels File «appsettings.json»
Microsoft.Extensions.DependencyInjection	Dependency Injection Framework von Microsoft
Microsoft.IdentityModel.Logging	Erweiterte Logging-Funktionalitäten für Azure Services
Microsoft.IdentityModel.Tokens	Modul, das Funktionalitäten zur Erstellung, Signierung und Überprüfung von Security Tokens zur Verfügung stellt
Microsoft.NET.Test.Sdk	Unterstützende Funktionalitäten für Testprojekte

Microsoft.NETCore.App	APIs, welche jede .NET Core Applikation verwendet
Moq	Mocking und Stubbing Framework für Unit Tests
MQTTnet	Client-Implementation des MQTT Protokoll als Schnittstelle für die verwendete Message Queue
MySqlConnector	MySQL Connector für den Zugriff auf MariaDB MariaDB ist ein Fork von MySQL und daher protokollkompatibel zu MySQL.
NETStandard.Library	Standard Library von .NET Core Applikationen
Newtonsoft.Json	Modul zur einfachen De-/Serialisierung von JSON-encodierten Daten
NLog	Verwendetes Logging Framework
NLog.Extensions.Logging	Extension Methoden zur einfachen Verwendung von NLog in .NET Core Konsolenapplikationen
NLog.Web.AspNetCore	ASP.NET Core-spezifische Extension Methods, um NLog einfach einbinden zu können
Pomelo.EntityFrameworkCore.MySql	Verwendeter MySQL-Entity Framework Core Adapter
Sendgrid	Client-Bibliothek für den Mailversand mittels Sendgrid
Swashbuckle.AspNetCore	Tool, um automatisiert eine Swagger-Dokumentation der internen und externen APIs zu generieren
System.IdentityModel.Tokens.Jwt	Funktionalitäten zum Erstellen und Validieren von JWT-Tokens
System.Net.Http	Standard-Bibliothek, welche den Zugriff auf HTTP-Funktionen ermöglicht
WindowsAzure.Storage	Client-Bibliothek für den Zugriff auf Azure Blob Storage
xunit	Verwendetes Testing-Framework
xunit.runner.visualstudio	Integration von xunit mit Visual Studio

Für weitere Informationen zu den verwendeten Modulen sei an dieser Stelle direkt auf die Website des NuGet Package Managers (https://www.nuget.org) verwiesen, welche als Nachschlagewerk für alle Module dient.

9.2 Frontend: Externe Module / NPM Packages

Für die Realisierung des Frontends wurden – wie bei der Entwicklung von Webapplikationen üblich – auf viele Module für verschiedene Einsatzzwecke zurückgegriffen. Die nachfolgende Tabelle enthält alle externen Abhängigkeiten und dem entsprechenden Einsatzzweck.

Modul	Einsatzzweck
@syncfusion	Drehzahl-Diagramm und Thermometer im Cockpit einer Maschine
axios	Promise-basierter HTTP-Client für die Kommunikation mit der Backend-API
date-fns	Zum Rendering von Datum und Zeit benötigte Library von material-ui-pickers
material-ui	Implementierung des gesamten Material Designs
material-ui-pickers	Datum-/Zeitauswahlfelder gemäss Material Design
rc-slider	Zeit-Slider bei der GPS-Spur in der Detailansicht einer Maschine
react	React Library
react-copy-to-clipboard	Kopieren des API-Tokens in die Zwischenablage
react-dom	React-Zusatz für DOM-Rendering
react-ga	Google Analytics zur Auswertung der Benutzeraktivität
react-google-maps	Google Maps zur Anzeige der aktuellen Maschinenstandorte
react-infinite-scroller	Automatischen Nachladens von Listenelementen in der Mobile-Ansicht
react-intl	Internationalisierung
react-redux	Redux Library
react-router-dom	Routing für React
react-router-redux	Integration React Router mit Redux
react-scripts-ts	Skripte zur Erstellung von React-Applikationen in TypeScript
react-select	Auswahlfelder in Formularen
recharts	Diagramm für Messdaten
redux	Redux Library
redux-thunk	Redux-Middleware für Action Creators

Für weitere Informationen zu den verwendeten Modulen sei an dieser Stelle direkt auf die Website des NPM Package Managers (https://www.npmjs.com) verwiesen, welche als Nachschlagewerk für alle Module dient.

Umsetzung Schnittstelle nach ISO 15143-3

Projektmitglieder
Stähli Tobias
Vincenz Dumeni
Wälter Jonas

Änderungsgeschichte

Datum	Version	Änderung	Autor
22.04.2018	1.0	Erstellung des Dokuments	Tobias Stähli
12.06.2018	1.1	Review des Dokuments	Tobias Stähli

Inhaltsverzeichnis

1 Einführung

1.1 Beschreibung

Dieses Dokument beschreibt und spezifiziert die Unterstützung der Web-Schnittstelle gemäss ISO 15143-3[1]. Es definiert und dokumentiert die Abbildung der Daten aus der von SUNCAR HK erfassten Datensätze auf die in ISO 15143-3 Norm spezifizierten Datenelemente.

1.2 Gültigkeitsbereich

Die Gültigkeit dieses Dokuments beschränkt sich auf die gesamte Projektdauer der Bachelorarbeit im Frühjahrssemester 2018. Allfällige spätere Anpassungen oder Ergänzungen im Dokument werden in der Änderungsgeschichte festgehalten.

1.3 Referenzen

Als Nachschlagewerk für unklare Begriffe sowie Abkürzungen sei an dieser Stelle auf das Glossar verwiesen, welches fortlaufend nachgeführt wird. In der nachfolgenden Tabelle sind alle weiteren referenzierten Dokumente und Links aufgelistet.

Thema	Referenz
Dokumentvorlage	OneDrive: \00_Administration\Template.docx
Glossar	OneDrive: \00_Administration\Glossar.docx
Projektplan	OneDrive: \02_Projektplan\Projektplan.docx
Anforderungsspezifikation	OneDrive: \03_Anforderungsanalyse\ Anforderungsspezifikation.docx
Logo Suncar HK	http://www.suncar-hk.com/

[1] (ISO/TS 15143-3:2016, 2016)

2 Datenelemente

Die nachfolgende Tabelle beschreibt die von der ISO 15143-3 Norm aufgezählten Datenelemente sowie deren Abbildung auf die von SUNCAR HK erfassten und von den Maschinen gelieferten Datenelemente.

In der letzten Spalte wird Auskunft darüber gegeben, ob die aktuelle Implementation das Datenelement unterstützt (✅), nicht unterstützt (❌) oder eine Implementation aktuell nicht möglich ist, da dafür auf den Maschinen mehr Daten erfasst werden müssten (⛔).

Elementname	Inhalt	Mapping SUNCAR HK	
EquipmentHeader. UnitInstallDateTime	Installationsdatum der Telematik Unit	DB: Maschine. InstallDate	✅
EquipmentHeader. OEMName	Marke des Herstellers (z.B. SUNCAR HK AG)	DB: MachineType. OEMName	✅
EquipmentHeader.Name	Modell des Gerätes	DB: MachineType.Name	✅
EquipmentHeader. EquipmentID	Vom Betreiber vergebene ID	DB: Machine. ConfiguredMachineId	✅
EquipmentHeader. SerialNumber	Nur nötig, wenn keine PIN oder VIN	DB: Machine. ConfiguredMachineId	✅
EquipmentHeader.PIN	PIN (ISO 6165) oder VIN (ISO 22242)	DB: Machine.Pin	✅
Location.datetime	Zeitpunkt der letzten Position	InfluxDB: Letzter Positionsdatenpunkt	✅
Location.Latitude	Latitude (GPS)	InfluxDB: Messtyp 80	✅
Location.Longitude	Longitude (GPS)	InfluxDB: Messtyp 81	✅
Location.Altitude, Location.AltotudeUnits	Höhe (optional); wird aktuell nicht geliefert von der Maschine	Aktuell vom GPS-Modul nicht unterstützt	⛔
CumulativeOperating-Hours.datetime, CumulativeOperating-Hours.Hour	Laufendes Total der Stunden, während Motor eingeschaltet war	InfluxDB: Messtyp 74 (WorkHours)	✅
FuelUsed.datetime, FuelUsed.FuelUnits, FuelUsed.FuelConsumed	Verbrauchter Treibstoff in Litern	Aktuell nicht unterstützt; müsste berechnet werden: Motoren-/Batterieleistung	❌
FuelUsedLast24.*	Analog zu FuelUsed.*, aber nur für die letzten 24 Stunden	Analog zu FuelUsed.*	❌

Distance.OdometerUnits, Distance.Odometer	Totale Distanz, die die Maschine seit Inbetriebnahme zurückgelegt hat.	Wird aktuell nicht erfasst und daher nicht unterstützt	⊖
CautionMessages. Description CautionMessages. Identifier	Fehlercode/Zustände, welche die Maschine meldet; Identifier gemäss ISO 6405-1 oder 6405-2.	Wird aktuell nicht unterstützt	⊖
CumulativeIdleHours. Datetime, CumulativeIdleHours.Hour	Zeit, während welcher die Maschine zwar lief, aber nicht in Gebrauch war; Bedeutung von Idle wird vom Hersteller	InfluxDB: TotalOperatingHours (76) – WorkHours (74)	✓
FuelRemaining.Datetime, FuelRemaining.Percent, FuelRemaining. FuelTankCapacity, FuelRemaining. FuelTankCapacityUnits	Wie viel Treibstoff die Maschine noch hat; Hier wird der prozentuale Wert (BatSOCUser) zurückgeliefert, weil Capacity nicht unterstützt wird	InfluxDB: Messtyp 2 (BatSOCUser)	✓
DEFRemaining	Wie viel «Diesel Exhaust Fluid» (DEF) noch vorhanden ist	Nicht relevant für Maschinen ohne Diesel Motor	✗
EngineStatus. EngineNumber	Alphanummerische Bezeichnung des Motors; fortlaufende Nummer	DB: Maschinen-Nummer, da Motor keine separate Nummer hat	✓
EngineStatus.Running	Status, ob Motor läuft oder nicht	InfluxDB: MotorSpeed (20) != 0, falls letzter Wert mehr als 15 Minuten her	✓
SwitchStatus.*	Digitale Inputs des Geräts, also wahr oder falsch mit Nummer	Wird nicht unterstützt	✗
CumulativePower- TakeOffHours.*	Totale Anzahl Power Take-off Stunden	Aktuell nicht verfügbar *Kommentar SUNCAR HK: Stunden-Nebenantrieb müssten neu summiert werden*	⊖
AverageLoadFactorLast24	Definiert als « Anteil (%) tatsächlicher Spritverbrauch von theoretisch möglichem/ rated Spritverbrauch»	Aktuell nicht verfügbar *Kommentar SUNCAR HK: Müsste neu berechnet/erfasst werden*	⊖
MaximumSpeedLast24	Maximal-Speed der letzten 24h in km/h oder m/min	Wird nicht unterstützt	✗

CumulativeLoadCount	Anzahl Ladungen, welche die Maschine insgesamt gemacht hat; Was genau eine Ladung ist, ist nicht näher spezifiziert	Wird nicht unterstützt	⊖
CumulativePayloadTotals	Total Masse in kg, die von der Maschine bewegt wurden	Wird nicht unterstützt	⊖
CumulativeActive-RegenerationHours	Zeit, während der die Maschine in der aktiven Regenerierung war; Diesel-Maschinen müssen regelmässig mit speziellen Prozeduren/Verfahren ihren Verbrennungsmotor reinigen	Wird nicht unterstützt, da für Elektrofahrzeuge nicht relevant	✖
CumulativeNonproductive-IdleHours	Zeit, während der die Maschine läuft, aber nicht operiert wird (kein Pedal-Input)	Wird nicht unterstützt; Aktuell wird nur CumulativeIdleHours erfasst	⊖
DiagnosticTroubleCode.*	Vom Hersteller festgelegte Fehlercodes mit Schweregrad und Temperaturen, zu welchen der Fehler aufgetreten ist	Aktuell nicht unterstützt; Fehlercodes werden von SUNCAR HK nicht auf die von der Norm geforderte Art erfasst; Benötigt Anpassung auf Maschine, um ISO-Norm gerechte Fehlermeldungen zu liefern	⊖

3 Authentifizierung API

Die ISO schreibt vor, dass die Authentifizierung der API mittels OAuth in der Version 1.0A oder 2.0 erfolgen soll. Welche genaue Unterart der Authentifizierung verwendet wird, ist nicht spezifiziert.

Die vorliegende Implementation im Rahmen dieser Arbeit verwendet OAuth 2.0 gemäss RFC 6749[2]. Ferner findet die Authentifizierung der Drittapplikation gegenüber der API mittels des im Abschnitt 4.4 von RFC 6749 beschriebenen Verfahrens «Client Credentials Grant» statt.

Das «Client Credentials Token» kann in der User-Detailansicht der Webapplikation generiert werden und ist mit denselben Rechten wie der generierende Benutzer ausgestattet. An gleicher Stelle können bestehende Tokens auch widerrufen oder gelöscht werden.

4 Response

4.1 Datenformat

Gemäss Abschnitt 5 der ISO 15143-3 Norm sind Antworten bevorzugt in XML zu liefern, die Norm gibt dafür auch XSD-Schemas vor. Zudem kann optional eine Antwort als JSON-Dokument geliefert werden. Dabei sollen die Namen der JSON-Attribute an das XSD-Schema angelehnt werden. Es fehlen jedoch Beispiele, wie eine mögliche JSON-Antwort konkret auszusehen hat. Die vorliegende Implementation unterstützt sowohl XML als auch JSON Responses. Die zurückgesendeten Datentransferobjekte wurden direkt aus den XSD-Schemas[3] generiert.

4.2 Zeitstempel

Alle Zeitstempel sollten gemäss dem ISO 15143-3 Standard nach ISO 8601[4] formatiert werden. Ein solcher Zeitstempel hat das Format «YYYY-MM-DDThh:mm:ssZ», also beispielsweise «2018-06-12T14:13:12Z». Das Z am Ende zeigt dabei an, dass es sich um einen Zeitpunkt in der Coordinated Universal Time (UTC) bezieht. Das Text-Encoding der Responses sollte UTF-8 sein.

[2] (RFC 6749: The OAuth 2.0 Authorization Framework, 2012)

[3] (ISO 15143-3 XSD, 2018)

[4] (Date and time format - ISO 8601, 2012)

4.3 Pagination

Für Responses, welche viele Elemente beinhalten, ist vorgesehen, dass die API die Resultate in mehreren Seiten zurückliefert. Vorgeschlagen von der ISO-Norm und in der vorliegenden Arbeit implementiert ist eine Pagination auf jeweils 100 Elemente. Um die Navigation zwischen den einzelnen Seiten zu erleichtern und insbesondere um diese für automatisierte API-Consumer zu ermöglichen, sehen die XSD-Schemas die Verwendung von Links vor. So wird beispielsweise bei Paginated Responses mittels eines Links mit «rel=next» auf die nächste Seite verwiesen bzw. der Link «rel=last» gibt die letzte Seite an.

5 Unterstützte Endpoints

Detaillierte Informationen zu den unterstützten Endpoints dieser Implementation und deren unterstützten Request-URIs können der automatisch generierten Swagger-Dokumentation[5] Abschnitt «Suncar Remote External API V1» entnommen werden.

6 Literaturverzeichnis

Date and time format - ISO 8601. (27. September 2012). Abgerufen am 12. Juni 2018 von ISO: https://www.iso.org/iso-8601-date-and-time-format.html

ISO 15143-3 XSD. (2018). Abgerufen am 22. April 2018 von ISO Standards Maintenance Portal: http://standards.iso.org/iso/15143/-3

ISO/TS 15143-3:2016. (Dezember 2016). Abgerufen am 22. April 2018 von ISO: https://www.iso.org/standard/67556.html

RFC 6749: The OAuth 2.0 Authorization Framework. (Oktober 2012). Abgerufen am 22. April 2018 von IETF: https://www.ietf.org/rfc/rfc6749.txt

Suncar ISO15134-3 compatible API. (2018). Abgerufen am 12. Juni 2018 von SUNCAR Remote: https://remote.suncarhk.com/swagger/

[5] (Suncar ISO15134-3 compatible API, 2018)

Systemtest:
Spezifikation und Protokoll

Projektmitglieder
Stähli Tobias
Vincenz Dumeni
Wälter Jonas

Änderungsgeschichte

Datum	Version	Änderung	Autor
10.06.2018	1.0	Erstellung des Dokuments	Jonas Wälter
10.06.2018	1.1	Systemtest: User Stories	Jonas Wälter
12.06.2018	1.2	Systemtest: Nichtfunktionale Anforderungen	Jonas Wälter
12.06.2018	1.3	Ergänzungen und orthografische Korrekturen	Tobias Stähli

Inhaltsverzeichnis

1 Einführung

1.1 Beschreibung

Dieses Dokument enthält die Spezifikationen der durchzuführenden Tests des Systems und das dazugehörige Testprotokoll.

1.2 Gültigkeitsbereich

Die Gültigkeit dieses Dokuments beschränkt sich auf die gesamte Projektdauer der Bachelorarbeit im Frühjahrssemester 2018. Allfällige spätere Anpassungen oder Ergänzungen im Dokument werden in der Änderungsgeschichte festgehalten.

1.3 Referenzen

Als Nachschlagewerk für unklare Begriffe sowie Abkürzungen sei an dieser Stelle auf das Glossar verwiesen, welches fortlaufend nachgeführt wird. In der nachfolgenden Tabelle sind alle weiteren referenzierten Dokumente und Links aufgelistet.

Thema	Referenz
Dokumentvorlage	OneDrive: \00_Administration\Template.docx
Glossar	OneDrive: \00_Administration\Glossar.docx
Anforderungsspezifikation	OneDrive: \03_Anforderungsanalyse\ Anforderungsspezifikation.docx
Logo Suncar HK	http://www.suncar-hk.com/

2 Systemtest

Der Systemtest wurde entsprechend den funktionalen Anforderungen (User Stories) und den nichtfunktionalen Anforderungen aufgebaut, welche in der Anforderungsspezifikation definiert sind. Für detailliertere Informationen sei daher an dieser Stelle auf die Anforderungsspezifikation verwiesen.

2.1 Funktionale Anforderungen

Wie der Anforderungsspezifikation entnommen werden kann, wurden alle User Stories mit entsprechenden Akzeptanzkriterien ergänzt. Im Zuge des Systemtests wurden diese Akzeptanzkriterien nun bezüglich Erfüllungsgrad überprüft.

Die nachfolgenden Kapitel enthalten die Akzeptanzkriterien zu allen User Stories für jedes Feature. Die Akzeptanzkriterien sind in Form von Stichworten verfasst und müssen daher nicht den höchsten schriftlichen Anforderungen genügen. Nicht erfüllte User Stories werden gekennzeichnet und mit einer Bemerkung versehen.

2.1.1 Feature «Dashboard»

Dashboard ✅

Als User (oder höher):

- Wartungsmeldung/Release-Information falls durch Admin erfasst
- Anzeige der Anzahl vorhandener sichtbarer Maschinen
- Anzeige der letzten fünf Events (max. der letzten 24h) aller sichtbaren Maschinen

Als Superuser (oder höher):

- Anzeige der Anzahl vorhandener sichtbaren Benutzer inklusive Online-Status

Als Distributor (oder höher):

- Anzeige der Anzahl vorhandener Firmen

Als Administrator:

- Anzeige der Anzahl vorhandener Maschinen- und Messtypen
- Anzeige noch nicht erfasster Maschinen und Messtypen, welche Messdaten in der InfluxDB haben, aber noch nicht in der MariaDB erfasst sind
- Anzeige fehlender Messtyp-Maschinentyp-Zuweisungen gemäss InfluxDB

Statuspage ✅

- Integration in Dashboard
- Anzeige der totalen Anzahl von Messwerten
- Messwerte-Diagramm der letzten Stunde
- Wichtigste Statistiken zur SQL-Datenbank
- Wichtigste Statistiken zur InfluxDB
- Links zu Grafana, HAProxy, MQTT-Interface, Google Analytics

Kontaktdaten ansehen ✅

- Kontaktadressen von SUNCAR HK AG sind ersichtlich
- Über Formular kann direkt eine Supportanfrage oder ein Fehler gemeldet werden

- Support-Mailadresse kann durch Administrator in der Systemkonfiguration verwaltet werden

2.1.2 Feature «User-Verwaltung»

Login ✅

- Felder: E-Mailadresse, Passwort
- Formular-Validierung
- Error-Handling für falsches Password/unbekannte E-Mail nicht unterscheidbar
- Session-Handling (JWT-Token, Refresh-Token)
- Automatischer Login bei gültigem JWT-Token im Local Storage

Logout ✅

- Local Storage & Redux Store leeren
- Refresh-Token invalidieren
- Weiterleitung zu Login-Page inkl. Logout-Message

Account aktivieren ✅

- Empfang E-Mail
- Formular zum Setzen des Passworts
- Passwortrichtlinien müssen eingehalten werden
- Automatischer Login nachdem Passwort gesetzt wurde

Account löschen ✅

- Der Benutzer kann in seinem Profil seinen Account löschen
- Dialog zur Bestätigung
- Automatischer Logout

Passwort zurücksetzen ✅

- Link bei Login
- Eingabe E-Mailadresse
- Empfang E-Mail
- Formular zum Zurücksetzen des Passworts
- Passwortrichtlinien werden durchgesetzt
- Automatischer Login nachdem Passwort zurückgesetzt wurde

User erfassen (als Admin) ✅

- Add-Button in User-Liste
- Pflichtfelder: E-Mail, Vorname, Nachname, Rolle, Sprache, Firma
- Formular-Validierung inkl. E-Mail-Check auf bereits verwendete E-Mailadressen
- Automatische Aktualisierung der Liste
- Aktivierungs-Mail versenden
- Zugriff nur für Distributor/Administrator (Frontend & Backend)
- Distributor darf keinen Administrator erfassen können

Superuser/User erfassen ✅

zusätzlich zu User Story «User erfassen (als Admin)»:

- Firma nicht wählbar
- Kein Zugriff für User (Frontend & Backend)
- Superuser darf keinen Distributor/Administrator erfassen können

Mein Profil ansehen ✅

- Informationen, welche im System zum eigenen Account erfasst sind: Nachname, Vorname, E-Mail, gewählte Sprache, Rolle, zugewiesene Firma

Andere Profile ansehen ✅

Zusätzlich zu User Story «Mein Profil ansehen»:

- Für Administrator/Distributor: alle Benutzer ansehen
- Für Superuser: Benutzer der eigenen Firma ansehen

User-Liste ansehen ✅

- Spalten: Nachname, Vorname, E-Mail, Rolle, Firma (nur für Distributor/Administrator), letzter Login
- Sortierung/Suche nach allen angezeigten Spalten
- Pagination/Infinite Scroll (Mobile)
- Zugriff nur für Superuser/Distributor/Administrator (Frontend & Backend)
- Keine «isDeleted» (User & Company)

Mein Profil bearbeiten ✅

- Eigenes Passwort zurücksetzen
- Eigene Informationen anpassen
- Anzeigesprache für Frontend anpassen
- Benutzer kann eigene Rolle nicht höher setzen, als er aktuell ist
- Aktivieren/Deaktivieren Sammeln von nutzerbezogenen Google Analytics Daten
- Aktivieren/Deaktivieren von E-Mail-Benachrichtigungen

Andere Profile bearbeiten ✅

Zusätzlich zu User Story «Mein Profil bearbeiten»:

- Edit-Button in User-Liste
- Administratoren/Distributoren: alle Benutzer bearbeiten
- Superuser: Benutzer der eigenen Firma bearbeiten
- Benutzer kann E-Mailadresse, Rolle, Passwort eines Benutzers mit höherer Rolle nicht ändern

Andere Accounts löschen ✅

- Delete-Button in User-Liste
- Dialog zur Bestätigung
- isDeleted-Flag setzen
- Automatische Aktualisierung der Liste
- Zugriff nur für Superuser/Distributor/Administrator (Frontend & Backend)

Auswertung User-Aktivität ✓

- Google Analytics mit User ID
- Google Analytics Token werden in Build/Release-Konfiguration von VSTS abgelegt
- Google Analytics Token sind für DEV-/PROD-Umgebung unterschiedlich
- Benutzer kann Teilen der nutzerbezogenen Daten deaktivieren (anonymisierte Daten werden aber weiterhin gesammelt)

2.1.3 Feature «Company-Verwaltung»

Company erfassen ✓

- Add-Button in Liste
- Pflichtfelder: Name, Logo
- Formular-Validierung
- Automatische Aktualisierung der Liste
- Zugriff nur für Distributor/Administrator (Frontend & Backend)

Company ansehen ✓

- Superuser können eigene Firma ansehen (inklusive zugehöriger Benutzer und Maschinen-Zuweisungen)
- Normale User sehen die Administrationsseite der eigenen Firma nicht

Andere Company ansehen ✓

Zusätzlich zu User Story «Company ansehen»:

- Zugriff nur für Distributor/Administrator (Frontend & Backend)

Company-Liste ansehen ✓

- Spalten: Logo, Name,
- Sortierung/Suche nach Name
- Pagination/Infinite Scroll (Mobile)
- Zugriff nur für Distributor/Administrator (Frontend & Backend)
- Keine «isDeleted» Company

Company bearbeiten ✓

- Edit-Button in Detailansicht (für Superuser/Distributor/Administrator)
- Pflichtfelder: Name, Logo
- Formular-Validierung
- Automatische Aktualisierung der Liste
- Zugriff nur für Superuser/Distributor/Administrator (Frontend & Backend)

Andere Company bearbeiten ✓

Zusätzlich zu User Story «Company bearbeiten»:

- Edit-Button in Company-Liste
- Automatische Aktualisierung der Liste
- Zugriff nur für Distributor/Administrator (Frontend & Backend)

Company löschen

- Delete-Button in Company-Liste
- Dialog zur Bestätigung
- isDeleted-Flag setzen
- Automatische Aktualisierung der Liste
- Zugriff nur für Distributor/Administrator (Frontend & Backend)

2.1.4 Feature «Maschinen-Verwaltung»

Maschine erfassen

- Add-Button in Maschine-Liste
- Pflichtfelder: Konfigurierte ID, Name, isBlacklisted, Maschinentyp
- Weitere Felder: PIN/VIN, Hersteller, Installationsdatum, Event Tasks
- Formular-Validierung
- Automatische Aktualisierung der Liste
- Zugriff nur für Administrator (Frontend & Backend)

Details von Maschine ansehen

- Detailansicht für User (Cockpit & Map)
- Detail-Adminansicht für Superuser/Distributor/Administrator (Accesses, Ownerships)

Maschinen-Liste ansehen

- Keine «isBlacklisted» für User/Superuser/Distributor
- Keine «isDeleted» (Maschine, Maschinentyp)

Maschinen-Cards für alle User:

- Cards mit Bild, Name, Name & Beschreibung von Maschinentyp

Maschinen-Liste für Superuser/Distributor/Administrator:

- Spalten: Bild, Konfigurierte ID, Name, Maschinentyp, isBlacklisted
- Sortierung/Suche nach Konfigurierte ID, Name, Maschinentyp, isBlacklisted
- Pagination/Infinite Scroll (Mobile)
- Zugriff nur für Distributor/Administrator (Frontend & Backend)

Maschine bearbeiten

- Edit-Button in Detail-Adminansicht (für Superuser/Distributor/Administrator)
- Edit-Button in Liste (für Superuser/Distributor/Administrator)
- Pflichtfelder: Konfigurierte ID, Name, isBlacklisted, Maschinentyp
- Für Superuser/Distributor nur Feld «Name» und «Event Tasks» sichtbar/veränderbar
- Formular-Validierung
- Automatische Aktualisierung der Liste
- Zugriff nur für Superuser/Distributor/Administrator (Frontend & Backend)

Maschine löschen

- Delete-Button in Maschine-Liste
- Dialog zur Bestätigung
- isDeleted-Flag setzen

- Automatische Aktualisierung der Liste
- Zugriff nur für Administrator (Frontend & Backend)

Aktueller Maschinen-Status ansehen ✓

- Register «Cockpit» in Detailansicht von Maschine
- Werte: SOC, Last Connection, Outdoor Temp, Total Charged Power, Hour Counter Working, Hour Counter Battery, Trip1, Trip2, Trip3
- regelmässiges Nachladen der Daten

Aktueller Maschinen-Standort ansehen ✓

- Register «Karte» in Detailansicht von Maschine
- Maschinen-Standort mit Marker markiert

Virtuelles Cockpit ansehen ✓

- Charts für Öl-Temperatur, RPM, Wasser-Temperatur, Batteriestrom
- regelmässiges Nachladen der Daten

Maschinenzustand PDF ✓

- In Absprache mit dem Auftraggeber reicht eine Druckansicht, da alle modernen Browser Drucken nach PDF unterstützen
- Print-Button in Register «Cockpit» und «Karte»
- Ansicht ist für Druck optimiert
- Zusätzlicher «Machine Header» als Print Header
- UI-Controls in Print-View ausgeblendet

Maschinen-Blacklist ✓

- Schalter in «Add/Edit Machine»-Dialog
- Blacklisted Maschinen sind für User/Superuser/Distributor nicht mehr sichtbar (Maschine-Cards, Maschine-Liste, Karte, zugewiesene Maschinen (Firma), Detailansicht, Messdaten-Diagramm)

Maschinen-Cards: Filterung ✓

- Filterung nach angezeigten Maschinen
- Filterung nach enthaltenen Maschinentypen
- Filterung nach Status (Farbe)

Cockpit: farbiger Status ✓

- Status der letzten Verbindung
- Farbe: grün [=live] (<=1min), gelb (<=7d), rot (>7d)

Detailansicht: Maschine auswählen ✓

- Für schnellere Navigation/Vergleichbarkeit kann Maschine direkt auf der Detailansicht einer Maschine ausgewählt werden
- Machine-Selector oberhalb der Register in der Detailansicht
- Machine-Selector standardmässig ausgeklappte (Desktop) und eingeklappt (Mobile)
- Vorhandene Messdaten und Zeiträume werden gespeichert bzw. übernommen

2.1.5 Feature «Maschinentype-Verwaltung»

Maschinentyp erfassen ✅

- Add-Button in Maschinentyp-Liste
- Pflichtfelder: Name, Beschreibung, Bild
- Weitere Felder: Name_i18n, Beschreibung_i18n, Zuweisung Messtypen
- Formular-Validierung
- Automatische Aktualisierung der Liste
- Zugriff nur für Administrator (Frontend & Backend)

Details von Maschinentyp ansehen ✅

- Detailansicht mit Eventvorlagen
- Zugriff nur für Administrator (Frontend & Backend)

Maschinentypen-Liste ansehen ✅

- Spalten: Bild, Name, Beschreibung
- Sortierung/Suche nach Name, Beschreibung
- Pagination/Infinite Scroll (Mobile)
- Zugriff nur für Administrator (Frontend & Backend)
- Keine «isDeleted» (Maschinentyp)

Maschinentyp bearbeiten ✅

- Edit-Button in Maschinentyp-Liste
- Pflichtfelder: Name, Beschreibung, Bild
- Weitere Felder: Name_i18n, Beschreibung_i18n, Zuweisung Messtypen
- Formular-Validierung
- Automatische Aktualisierung der Liste
- Zugriff nur für Administrator (Frontend & Backend)

Maschinentyp löschen ✅

- Delete-Button in Maschinentyp-Liste
- Dialog zur Bestätigung
- isDeleted-Flag setzen
- Automatische Aktualisierung der Liste
- Zugriff nur für Administrator (Frontend & Backend)

2.1.6 Feature «Messtyp-Verwaltung»

Messtyp erfassen ✅

- Add-Button in Messtyp-Liste
- Pflichtfelder: Kurzname, Beschreibung, Einheit, Messtyp Nr, erforderliche Rolle
- Optionale Felder: Kurzname_i18n, Beschreibung_i18n, Einheit_i18n
- Formular-Validierung
- Automatische Aktualisierung der Liste
- Zugriff nur für Administrator (Frontend & Backend)

Details von Messtyp ansehen

Diese User Story wurde nach Absprache mit dem Auftraggeber nicht implementiert. Eine Detailansicht für einen Messtyp macht keinen Sinn, weil alle vorhandenen Daten bereits in der Übersichtsliste aller Messtypen angezeigt werden.

Messtypen-Liste ansehen

- Spalten: Messtyp Nr, Kurzname, Beschreibung, Einheit, erforderliche Rolle
- Sortierung/Suche nach allen angezeigten Spalten
- Pagination/Infinite Scroll (Mobile)
- Zugriff nur für Administrator (Frontend & Backend)
- Keine «isDeleted» (Messtyp)

Messtyp bearbeiten

- Edit-Button in Messtyp-Liste
- Pflichtfelder: Kurzname, Beschreibung, Einheit, Messtyp Nr, erforderliche Rolle
- Optionale Felder: Kurzname_i18n, Beschreibung_i18n, Einheit_i18n
- Formular-Validierung
- Automatische Aktualisierung der Liste
- Zugriff nur für Administrator (Frontend & Backend)

Messtyp löschen

- Delete-Button in Messtyp-Liste
- Dialog zur Bestätigung
- isDeleted-Flag setzen
- Automatische Aktualisierung der Liste
- Zugriff nur für Administrator (Frontend & Backend)

Messtyp-Sichtbarkeit verwalten

- Messtypen und zugehörige Messwerte sind nur für Benutzer mit «Required Role» oder höher sichtbar bzw. im Messdiagramm auswählbar
- Es gibt die Rollen Administrator > Distributor > Superuser > User
- Auswahlmöglichkeit im «Add/Edit Measuretype»-Formular

Messtyp an Maschinentyp zuweisen

- Multi-Selector im «Add/Edit Machinetype»-Formular
- Konzept «MeasureTypeUsage»

2.1.7 Feature «Ownership-Verwaltung»

Maschine an Company zuweisen

- Add-Button in Ownership-Liste
- Pflichtfelder: Firma, Maschine, Startdatum
- Weitere Felder: Enddatum
- Formular-Validierung
- Automatische Aktualisierung der Liste
- Zugriff nur für Distributor/Administrator (Frontend/Backend)

- Unbefristete Zuweisungen sind möglich
- Überlappende Zuweisungen sind möglich
- Eine Maschine kann gleichzeitig mehreren Firmen zugewiesen sein

Zuweisungen von Maschinen zu Company ansehen ✅

- Spalten: Firma/Maschine, Startdatum, Enddatum
- Sortierung/Suche/Pagination/Infinite Scroll (Mobile)
- Zugriff nur für Distributor/Administrator (Frontend/Backend)
- Zugriff via Detail-Adminansicht von Maschine sowie von Firma

Maschine-Company-Zuweisung bearbeiten ✅

- Edit-Button in Ownership-Liste
- Zugriff nur für Distributor/Administrator (Frontend/Backend)
- Nur Startdatum und Enddatum änderbar
- Unbefristete Zuweisungen sind möglich
- Automatische Aktualisierung der Liste

Maschine-Company-Zuweisung löschen ✅

- Delete-Button in Ownership-Liste
- Dialog zur Bestätigung
- Automatische Aktualisierung der Liste
- Zugriff nur für Distributor/Administrator (Frontend/Backend)

2.1.8 Feature «Access-Verwaltung»

Maschinen-Access für User erfassen ✅

- Add-Button in Access-Liste
- Pflichtfelder: Zuweisung (Ownership), Benutzer, Startdatum
- Weitere Felder: Enddatum
- Formular-Validierung
- Automatische Aktualisierung der Liste
- Zugriff nur für Superuser/Distributor/Administrator (Frontend/Backend)
- Access kann nur in einen Zeitraum sein, in welchem die Firma die entsprechende Maschine zugewiesen hat (Ownership)
- Superuser kann Maschinen-Access nur für Benutzer der eigenen Firma erfassen
- Unbefristete Zuweisungen sind möglich
- Überlappende Zuweisungen sind möglich

Eigene Maschinen-Accesses anzeigen ✅

- Benutzer sieht nur Maschinen, Maschinentypen und Messdaten mit gültigem Access
- Superuser sieht nur Maschinen, Maschinentypen und Messdaten mit gültigem Ownership
- Distributor/Administrator sieht alles

Maschinen-Access-Liste ansehen ✅

- Spalten: Benutzer/Maschine, Startdatum, Enddatum
- Sortierung/Suche/Pagination/Infinite Scroll (Mobile)

- Zugriff nur für Superuser/Distributor/Administrator (Frontend/Backend)
- Zugriff via Detail-Adminansicht von Maschine sowie von Benutzer
- Superuser sieht nur Zuweisungen für Maschinen, welche seiner Firma auch zugewiesen sind (Ownership)
- Distributor/Administrator sieht alle Zuweisungen von Maschinen an Benutzer (vergangene und aktuelle)

Maschinen-Access für User bearbeiten ✅

- Edit-Button in Access-Liste
- Nur Startdatum und Enddatum veränderbar
- Automatische Aktualisierung der Liste
- Superuser kann Maschinen-Zuweisungen für die Benutzer seiner Firma ändern
- Zuweisung kann nur in einen Zeitraum sein, für welche die Firma die entsprechende Maschine zugewiesen hat (Ownership)

Maschinen-Access für User löschen ✅

- Delete-Button in Access-Liste
- Dialog zur Bestätigung
- Automatische Aktualisierung der Liste
- Superuser kann Maschinen-Access nur für Benutzer der eigenen Firma löschen

Beliebiger Maschinen-Access für User verwalten ✅

Zusätzlich zu User Stories «Maschinen-Access für User erfassen / bearbeiten / löschen»:

- Zugriff nur für Distributor/Administrator auf beliebige Maschinen-Accesses

2.1.9 Feature «Karte»

Maschinen-Karte ansehen ✅

- Menüpunkt «Karte»
- Integration von Google Maps mit eingeschränktem API Token
- Maschinen-Standort mit Marker markiert
- Keine 0/0-Koordinaten oder ausserhalb des Wertebereichs für Lat/Long

Maschinen-Details auf Karte ansehen ✅

- Aktueller/Letzter Standort pro Maschine markiert
- Beim Klicken auf Marker erscheint ein «InfoWindow»
- Informationen im InfoWindow: Maschinen-Name, aktueller Batterieladestand, aktuelle Drehzahl, letzter Kontakt (Zeitstempel)

Karte zentrieren ✅

- Button auf der Karte
- Zentrierung/Zooming, damit alle Maschinen im Kartenausschnitt sichtbar sind

Von Maschinen-Karte zu Detailansicht navigieren ✅

- Link im InfoWindow
- Link führt zur Maschinen-Detailansicht (Cockpit)

GPS-Spur anzeigen

- Register «Karte» in der Maschinen-Detailansicht
- GPS-Spur als Pfad auf der Karte
- Auswahl von Zeitfenster (Startdatum/Enddatum) inkl. History
- Slider mit allen GPS-Punkten innerhalb des gewählten Zeitfensters
- Anzeige der Position zu einem bestimmten Zeitpunkt als Marker

Maschinen-Karte: farbiger Status

- Marker-Farbe gemäss dem letzten Kontakt: grün [=live] (<=1min), gelb (<=7d), rot (>7d)

Maschinen-Karte: Filterung

- Filterung nach angezeigten Maschinen
- Filterung nach enthaltenen Maschinentypen
- Filterung nach Status (Farbe)

2.1.10 Feature «Messdatenauswertung»

Messdaten-Diagramm ansehen

- Auswahl von Zeitfenster (Startdatum/Enddatum) inkl. History
- Auswahl eines Messtyps
- Linien-Diagramm mit Datum/Zeit (X-Achse), Einheit (Y-Achse) und Legende für ausgewählten Messtyp
- Anzeige des genauen Wertes beim Mouseover
- Zooming-Möglichkeit

Diagramm mit mehreren Messdaten ansehen

Zusätzlich zu User Story «Messdaten-Diagramm ansehen»:

- Multi-Selector für Messtypen (Mehrfachauswahl)
- Separate Y-Achse für alle vorkommenden Einheiten (Gruppierung gleicher Einheiten)
- Y-Achsen gleichmässig auf linke/rechte Seite verteilt

Alte Messdaten ansehen

- Mit Zeitfenster-Auswahl kann beliebig in die Vergangenheit navigiert werden
- Möglichkeit zum Import der Archivdaten aus dem alten System
- Konfiguration, welche Tabellen auf welche Maschinen abgebildet werden sollen, erfolgt in einem separaten File
- Import-Tool kann Stand-Alone betrieben werden und benötigt die Verbindungsparameter der MQTT-Anbindung

Messdaten PDF

- In Absprache mit dem Auftraggeber reicht eine Druckansicht, da alle modernen Browser Drucken nach PDF unterstützen
- Print-Button auf Seite «Measurements» und in Register «Messdaten» (Maschinen-Detailansicht)
- Ansicht ist für Druck optimiert
- Zusätzlicher «Machine Header» als Print Header

- UI-Controls in Print-View ausgeblendet

Messdaten-Diagramm: Maschine auswählen ✓

- Menüpunkt «Measurements»
- Auswahl einer Maschine mittels Machine-Selector
- Machine-Selector standardmässig ausgeklappte (Desktop) und eingeklappt (Mobile)
- Beibehaltung der Messtypen/Zeitrahmen

CSV-Export Messdaten ✓

- Export-Button bei Messdiagramm
- Export der aktuell auf dem Messdiagramm sichtbaren Daten als CSV
- Zeitstempel als UTC
- Zeitliche Auflösung einstellbar
- Administrator kann Rohdaten exportieren
- Mehrere Messwerte im gleichen Export ersichtlich
- In Zukunft andere Export-Formate (ausser CSV) denkbar

2.1.11 Feature «Zugriff»

Datenschutz ✓

- Kommunikation über öffentliche Netzwerke ist überall mittels HTTPS abgesichert
- Frontend ist gegen gängige Angriffsmöglichkeiten gemäss OWASP Top 10 abgesichert
- Kommunikation zwischen neuen Maschinen und Data Collector ist mittels HTTPS und Client Zertifikaten abgesichert und gegen Man-in-the-middle geschützt

2.1.12 Feature «Data Collector»

Messdaten senden (neu) ✓

- Neue Maschine kann Messdaten über die im Dokument «Schnittstellenkonzept» spezifizierte Schnittstelle versenden
- Weitere Anforderungen siehe Schnittstellenkonzept

Messdaten senden (Legacy) ✓

- Bestehende Maschine kann Messdaten wie gewohnt über eine VPN-Verbindung an den Data Collector senden
- Weitere Anforderungen siehe Schnittstellenkonzept

Data Collector ✓

- Eingehende Messdaten (aus der Message Queue) werden gemäss Schnittstellenkonzept geparst und an die InfluxDB weitergeleitet
- Zeitstempel für die Synchronisierung der Maschinen wird periodisch übertragen, das tiefst-mögliche QoS-Level wird verwendet
- Periodisches Logging von Statistiken (über Standard Logging), damit Zustand des Data Collectors im Störungsfall schnell ermittelt werden kann

Legacy Data Collector

- Eingehende Messdaten (als UDP-Pakete via VPN-Verbindung) werden gemäss Schnittstellenkonzept geparst und an den Data Collector weitergeleitet
- Annahme: Zeitstempel = Eingangszeitpunkt des UDP-Pakets
- Gebündelte Übermittlung der Messdaten, um Netzwerklast zu reduzieren
- Legacy-Maschinen werden in einer externen Konfigurationsdatei definiert und verwaltet
- Jeder Legacy-Maschine ist ein Zertifikat hinterlegt, sodass für den Data Collector alle Maschinen genau gleich daherkommen
- Messdaten werden genau einmal erfolgreich an Data Collector (Message Queue) zugestellt
- Falls Data Collector (Message Queue) nicht erreichbar ist, werden die empfangenen Messdaten zwischengespeichert, bis er wieder erreichbar ist

2.1.13 Feature «API für Drittanbieter»

API: Maschinenstatus lesen

- Abfrage des Maschinenstatus via externe API für Drittanbieter
- Unterstützung der TimeSeries- und Fleet-Endpoints gemäss Dokument «Schnittstelle nach ISO 15143-3»

API: Messdaten lesen

- Abfrage der Messdaten einer Maschine via externe API für Drittanbieter
- Unterstützung der TimeSeries- und Fleet-Endpoints gemäss Dokument «Schnittstelle nach ISO 15143-3»

API: Authentifizierung

- Authentifizierung für die API für Drittanbieter
- Authentifizierung erfolgt über OAuth 2.0
- Details und weitere Anforderungen siehe «Schnittstelle nach ISO 15143-3»

Schnittstelle ISO 15143-3

- Authentifizierung mittels OAuth 2.0 (Token für den erstellenden Benutzer)
- Unterstützung API Endpoints gemäss Dokument «Schnittstelle nach ISO 15143-3»
- Notwendige Zusatzinformationen werden im Frontend erfasst
- Zugriff erfolgt autorisiert: API gibt nur heraus, wofür der Benutzer hinter dem Token auch Zugriff hat
- Pagination mit Standard-Werten aus ISO-Standard

2.1.14 Feature «Datenanalyse»

Datenanalysen durchführen

- In Absprache mit dem Auftraggeber wird dies nur beispielhaft implementiert. Der Auftraggeber hat bereits bestehende Auswertungsmöglichkeiten in MATLAB, daher ist der Daten-Export wichtiger (User Story «CSV-Export Messdaten»)
- Rohdaten können als CSV exportiert werden für die Datenanalyse mit MATLAB

- Mittels Kapacitor können einfache Analysen als TICK Skript definiert/verwaltet werden
- Möglichkeit A): Ergebnisse werden als Pseudo-Messwert in der InfluxDB gespeichert und können im Frontend wie normale Messwerte angezeigt werden
- Möglichkeit B): Ergebnisse werden als Event in der InfluxDB gespeichert und werden im Frontend auf dem Dashboard und der Maschinen-Detailansicht aufgeführt
- Möglichkeit C): Ergebnisse werden an das Backend zurückgegeben, welches E-Mail-Benachrichtigungen versendet
- Möglichkeiten sind kombinierbar

Benachrichtigung bei Schwellwert ✅

- Administrator kann eine Eventvorlage (TICK Skript) für einen Maschinentyp erstellen
- Superuser/Distributor/Administrator kann die erstellte Eventvorlage für eine Maschine des entsprechenden Maschinentyps aktivieren (Task erstellen)
- Wenn ein in der Eventvorlage definierter Schwellwert überschritten wird, wird eine Benachrichtigung für die betroffene Maschine ausgelöst
- Benachrichtigung kommen in Form eines Events im Frontend oder per E-Mail
- E-Mails werden nur an Benutzer mit einem gültigem Access für die betroffene Maschine gesendet
- Benutzer kann E-Mail-Benachrichtigung im Profil aktivieren/deaktivieren

Fehlermeldungen ansehen ⛔

Fehlermeldung per Mail erhalten ⛔

Die Übermittlung von Fehlercodes der Maschine an den Data Collector ist aktuell ein Experiment des Auftraggebers, welches möglichen Änderungen unterworfen ist. Da sich ein Ende des Experiments während der Projektphase nicht mehr abzeichnete, wurden diese beiden User Stories in Absprache mit dem Auftraggeber zurückgestellt und nicht implementiert.

2.2 Nichtfunktionale Anforderungen

Neben den funktionalen Anforderungen wurden auch die nichtfunktionalen Anforderungen überprüft. In der nachfolgenden Tabelle sind alle nichtfunktionalen Anforderungen aus dem Dokument «Anforderungsspezifikation» aufgeführt und auf den jeweiligen Erfüllungsgrad hin überprüft.

NFR-01: Skalierbarkeit

Um die Performance und Skalierbarkeit der Gesamtlösung zu testen, wurde ein Stresstest durchgeführt. Als Messdaten dienten dabei die Archivdaten von vier Maschinen mit insgesamt rund 3.25 Millionen Messwerten, welche von der bestehenden Datenbank in die InfluxDB migriert werden sollten.

Innerhalb von 17 Minuten (von 14:25 bis 14:42) konnten dabei 3'206'218 Messwerte in der Measurements-DB gespeichert werden. Bei 10 Messwerten pro Sekunde und pro Maschine, wie im Schnittstellenkonzept als Obergrenze definiert ist, entspricht dies mindestens 314 Maschinen. Somit kann die geforderte Auslegung auf 200 Maschinen erfüllt werden.

Die Differenz zwischen den gesendeten und den tatsächlich in der Datenbank eingelesen Messwerten entstand dadurch, dass einige der Messwerte im Backup mit dem gleichen Zeitstempel/Messtyp erfasst waren. Da die zeitliche Auflösung der Messwerte im alten System bei einer Sekunde lag, ist dieser Umstand nur durch das Verwenden des Durchschnitts der Werte mit gleichem Zeitstempel aufzulösen. Zudem sind einige Messwerte nicht als gültige Gleitkommazahl in der alten Datenbank erfasst, da das alte Datenschema den Messwert als String abspeichert.

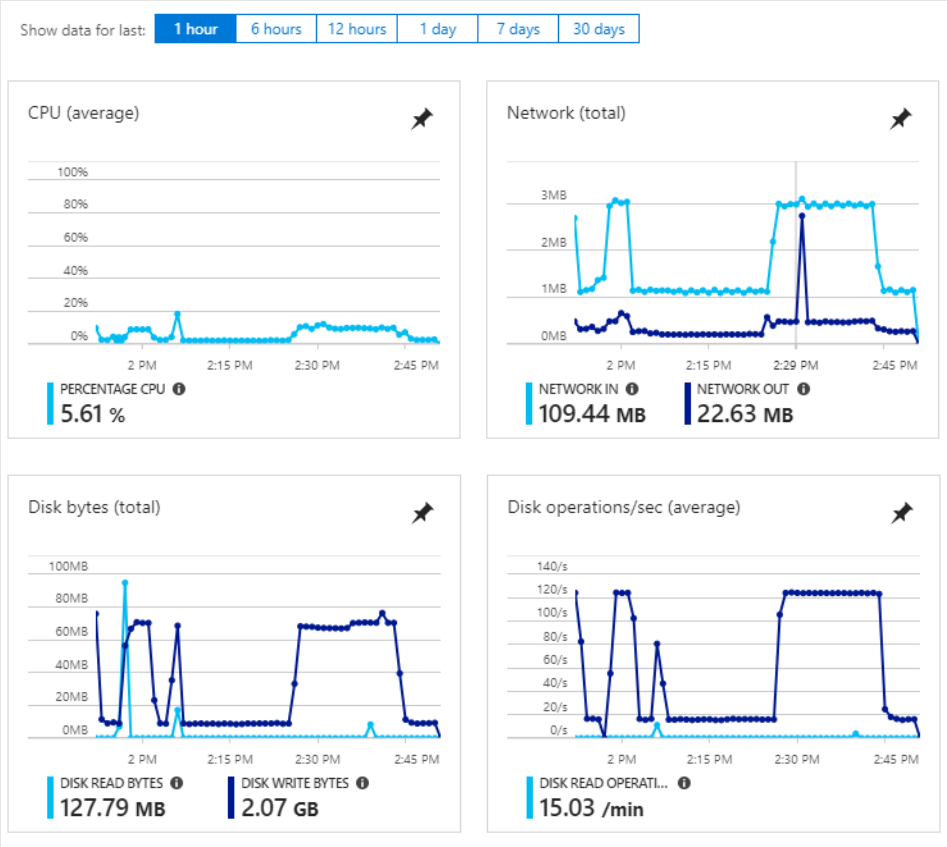


Abbildung 1: Auslastung gemäss Microsoft Azure Portal

Wie im Microsoft Azure Portal ersichtlich ist, hat dieser Test das System nicht an seine Grenzen gebracht und wir konnten leider nicht wie erwartet einen Bottleneck im System finden. Nach einigen Experimenten hat sich die Vermutung erhärtet, dass Azure den «Incoming Network-Traffic» künstlich drosselt. Um diesen Verdacht zu bestätigen, wurde die Applikation testweise auf eine bei Google Cloud gehosteten VM installiert. Dank der Verwendung von Docker war das neue System innert 20 Minuten installiert. Wir konnten auf diesem System nachweisen, dass das Gesamtsystem noch wesentlich mehr Messdaten verarbeiten kann, denn nun lag der Bottleneck auf der Seite des Import-Tools, also dem Messdaten-Erzeuger.

NFR-02: Internationalisierung

Das gesamte Frontend wurde von Grund auf mit Englisch und Deutsch als verfügbare Anzeigesprachen ausgestattet. Die dadurch geschaffene Grundlage ermöglicht das Hinzufügen von weiteren Sprachen, ohne dass abgesehen von den Übersetzungen weitere aufwändige Codeänderungen vorgenommen werden müssen.

NFR-03: Responsive Design für Mobile Devices

Das gesamte Frontend-Design wurde stets mit Rücksicht auf verschiedene Bildschirmgrössen entwickelt und fortlaufend für die Mobile-Ansicht optimiert. Durch den Einsatz von End-to-End-Tests wird zudem sichergestellt, dass das Frontend auch auf kleinen Displays bedient werden kann.

NFR-04: Browsersupport

Die Funktionalitäten und das Design des Frontends wurde für die aktuellen Browser Google Chrome, Mozilla Firefox, Safari und Microsoft Edge erfolgreich getestet.

Nicht Bestandteil der unterstützten Browser gemäss Anforderungsspezifikation ist der veraltete Browser Internet Explorer. Um die Nutzung dieses nicht unterstützten Browsers zu verhindern, wird im Frontend anstelle des Logins eine entsprechende Meldung angezeigt und auf die empfohlenen Browser hingewiesen.

NFR-05: Datensicherheit

Für den Zugriff auf das Frontend und das Backend ist eine vorgängige Authentifizierung mit einem registrierten Benutzer erforderlich. Es ist nicht möglich, sich eigenhändig als Benutzer zu registrieren. Ein entsprechender Benutzer kann nur durch einen Superuser, einen Distributor oder einen Administrator erfasst werden.

Mit Hilfe eines umfassenden Berechtigungskonzeptes wird sichergestellt, dass jeder Benutzer nur auf die für ihn freigegebenen Ressourcen (Maschinen, Maschinentypen, Messtypen, Firmen, Benutzer, Access, Ownership, etc.) zugreifen kann. Für eine Übersicht zum Berechtigungskonzept sei an dieser Stelle auf das Dokument «Domainanalyse» verwiesen.

Des Weiteren ist sowohl die InfluxDB als auch die MariaDB nicht direkt aus dem Internet erreichbar. Jede Kommunikation über öffentliche Netzwerke erfolgt verschlüsselt über TLS oder HTTPS. Wie bei heutigen Applikationen üblich, werden auch Passwörter nur als «salted» Hash in der Datenbank abgelegt.

NFR-06: Modulare Architektur

Das Gesamtsystem besteht aus verschiedenen modularen Teilsystemen, wobei viele Komponenten bei Bedarf sehr einfach ausgetauscht werden können. Die Kommunikation zwischen den einzelnen Komponenten findet wo immer möglich über standardisierte Protokolle statt, um die Koppelung möglichst gering zu halten. So wurde beispielsweise inmitten des Projektes die Message Queue-Implementation ausgetauscht (von Mosquitto zu ActiveMQ), um eine höhere Performance zu erzielen und Datenverlust zu verhindern. Dank dem Einsatz von Docker konnte die neue Message Queue in wenigen Minuten in Betrieb genommen werden. Mehr Details zur Skalierung des Systems sind dem Dokument «Continuous Integration / Continuous Delivery & Deployment (CI/CD)» zu entnehmen.

NFR-07: Aufbewahrung Messdaten

Durch den Einsatz von InfluxDB besteht kein Kapazitätsproblem mehr bei der Datenspeicherung. So können die Messdaten ohne Probleme auch über eine Zeitdauer von fünf Jahren hinaus aufbewahrt werden. Aktuell befinden sich in der Messwerte-Datenbank rund 28 Millionen Messwerte und damit die kompletten Messdaten von über 20 Maschinen welche seit knapp zwei Monaten live Daten liefern. Diese Daten nehmen dank der effizienten Speicherung und gezielten Redundanz-Eliminierung nur noch gut 170 MB Festplattenplatz ein.

NFR-08: Redundante Softwarearchitektur

Durch die gewählte Softwarearchitektur und den Einsatz von Docker als Containerlösung, ist eine redundante Auslegung bei Bedarf möglich. Durch das Deployment in einer Docker Cloud oder auf einem Docker Swarm wie beispielsweise Kubernetes Cloud könnten bei Bedarf mehrere Instanzen der jeweilig ausgelasteten Komponente gestartet werden. Mehr Details zur Skalierung und einem redundanten Deployen des Systems sind dem Dokument «Continuous Integration / Continuous Delivery & Deployment (CI/CD)» zu entnehmen.

NFR-09: Benutzerfreundliches GUI

Beim Frontend-Design wurde auf die Material Design Guidelines zurückgegriffen, um eine intuitive, moderne und vertraute Umgebung für den Benutzer zu erschaffen. Alle Formulare sind clientseitig mit entsprechenden Validierungsmeldungen ausgestattet, wobei die Validierung auch serverseitig implementiert wurde.

NFR-10: Branchenspezifische Normen

Bei der Implementierung der API für Drittanbieter wurde die ISO-Norm ISO 15143-3 für «Earthmoving machinery and mobile road construction machinery» eingehalten und detailgetreu umgesetzt.

2.3 Testing

Neben dem durchgeführten Systemtest wurde ein verteiltes Testing implementiert, um die korrekte Funktionalität aller Teilsysteme fortlaufend zu überprüfen und sicherzustellen.

So wurden für alle ASP.NET Core Projekte (Backend, Data Collector, Common, etc.) entsprechende Unit Tests erstellt, welche die wichtigsten Funktionen testen. Ergänzt werden die Unit Tests durch separate End-to-End Tests mit Hilfe von Cypress. Dabei werden einerseits die Bedienung des Frontends mit verschiedenen Bildschirmgrössen und andererseits direkt die API-Endpunkte des Backends auf die Einhaltung des Berechtigungskonzeptes getestet.

Continuous Integration / Continuous Delivery & Deployment (CI/CD)

Projektmitglieder
Stähli Tobias
Vincenz Dumeni
Wälter Jonas

Änderungsgeschichte

Datum	Version	Änderung	Autor
10.05.2018	1.0	Erstellung des Dokuments	Dumeni Vincenz
12.06.2018	1.1	Überarbeiten des Dokuments	Dumeni Vincenz
13.06.2018	1.2	Review des Dokuments	Tobias Stähli
13.06.2018	1.3	Review des Dokuments	Jonas Wälter

Inhaltsverzeichnis

1 Einführung

1.1 Beschreibung

Dieses Dokument beschreibt den Prozess der Integration und Deployment der Applikation in der Testumgebung und der Produktion – auch bekannt als «Continuous Integration (CI)» und «Continuous Delivery (CD)» Prozess. Zudem ist in diesem Dokument der Zusammenhang des Continuos Delivery Prozesses mit dem Deployment beschrieben.

Das Ziel von CI/CD ist es, die Applikation automatisch und ohne manuelle Arbeit aktualisieren zu können. Weiter soll es auch möglich sein, ad hoc weitere Umgebungen zu erstellen.

Das Deployment soll modern, sicher und stabil gestaltet werden. Weitere Details zu den Deployment-Varianten werden im Dokument «Cloud-Evaluation» beschrieben.

1.2 Gültigkeitsbereich

Die Gültigkeit dieses Dokuments beschränkt sich auf die gesamte Projektdauer der Bachelorarbeit im Frühjahrssemester 2018. Allfällige spätere Anpassungen oder Ergänzungen im Dokument werden in der Änderungsgeschichte festgehalten.

1.3 Referenzen

Als Nachschlagewerk für unklare Begriffe sowie Abkürzungen sei an dieser Stelle auf das Glossar verwiesen, welches fortlaufend nachgeführt wird. In der nachfolgenden Tabelle sind alle weiteren referenzierten Dokumente und Links aufgelistet.

Thema	Referenz
Aufgabenstellung	OneDrive: \01_Aufgabenstellung\Aufgabenstellung.docx
Dokumentvorlage	OneDrive: \00_Administration\Template.docx
Glossar	OneDrive: \00_Administration\Glossar.docx
Projektplan	OneDrive: \02_Projektplan\Projektplan.docx
Anforderungsspezifikation	OneDrive: \03_Anforderungsanalyse\ Anforderungsspezifikation.docx
Softwarearchitekturdokument	OneDrive: \04_Architektur_Design\ Softwarearchitekturdokument.docx
Cloud-Evaluation	OneDrive: \04_Architektur_Design\Cloud-Evaluation.docx
Logo Suncar HK	http://www.suncar-hk.com/

2 Continuous Integration

Als Basis für Continuous Integration wird das Source Code Management System Git[1] in Verbindung mit den Visual Studio Team Services (VSTS) eingesetzt. Für jede User Story oder teilweise für einzelne Tasks bzw. Bugs wird ein separater Branch eröffnet. Vom lokale Repository des Entwicklers wird der Branch ins Remote Repository auf VSTS gepusht. Dabei werden jeweils automatisch die Unit Tests sowie Codequalitäts-Checks durchgeführt.

Es wurde auf mehrere unabhängige Repositories verzichtet (z.B. Separation von Backend und Frontend). Stattdessen wird der gesamte Sourcecode in einem Monorepo[2] verwaltet. Dies vereinfach vieles, weil die unterschiedlichen Komponenten enge Abhängigkeiten haben können und die Umsetzung der User Stories von einem Entwickler für Backend und Frontend gleichzeitig erledigt wird.

Staging-Umgebung

Nach dem Abschluss eines Tasks oder einer User Story wird der erstellte Pull Request jeweils nur mit Zustimmung einer zweiten Person und nach dem Bestehen der Unit Tests sowie Codequalitäts-Checks in den Master-Branch gemerged. Bei jedem Merge in den Master wird automatisch das Deployment auf die Staging-Umgebung ausgelöst.

Production-Umgebung

Jeweils am Ende einer Iteration wird eine neue Version, ebenfalls per Pull Request, in den Production-Branch integriert. Für diesen Merge braucht es die Zustimmung zweier Entwicklers sowie das fehlerlose Durchlaufen der End-to-End Tests.

Für die End-to-End Tests wird beim Auslösen des Pull Requests eine isolierte Testumgebung mit dem aktuellen Stand des Systems hochgefahren, um die Tests durchführen zu können. Nach dem erfolgreichen Durchlaufen der Tests wird der Master-Branch automatisch in den Production-Branch gemerged und anschliessend autonom das Deployment in der produktiven Umgebung angestossen.

[1] (Git, 2018)

[2] (Monorepo, 2018)

In der nachfolgenden Abbildung wird der zuvor beschriebene Ablauf schematisch aufgezeigt.

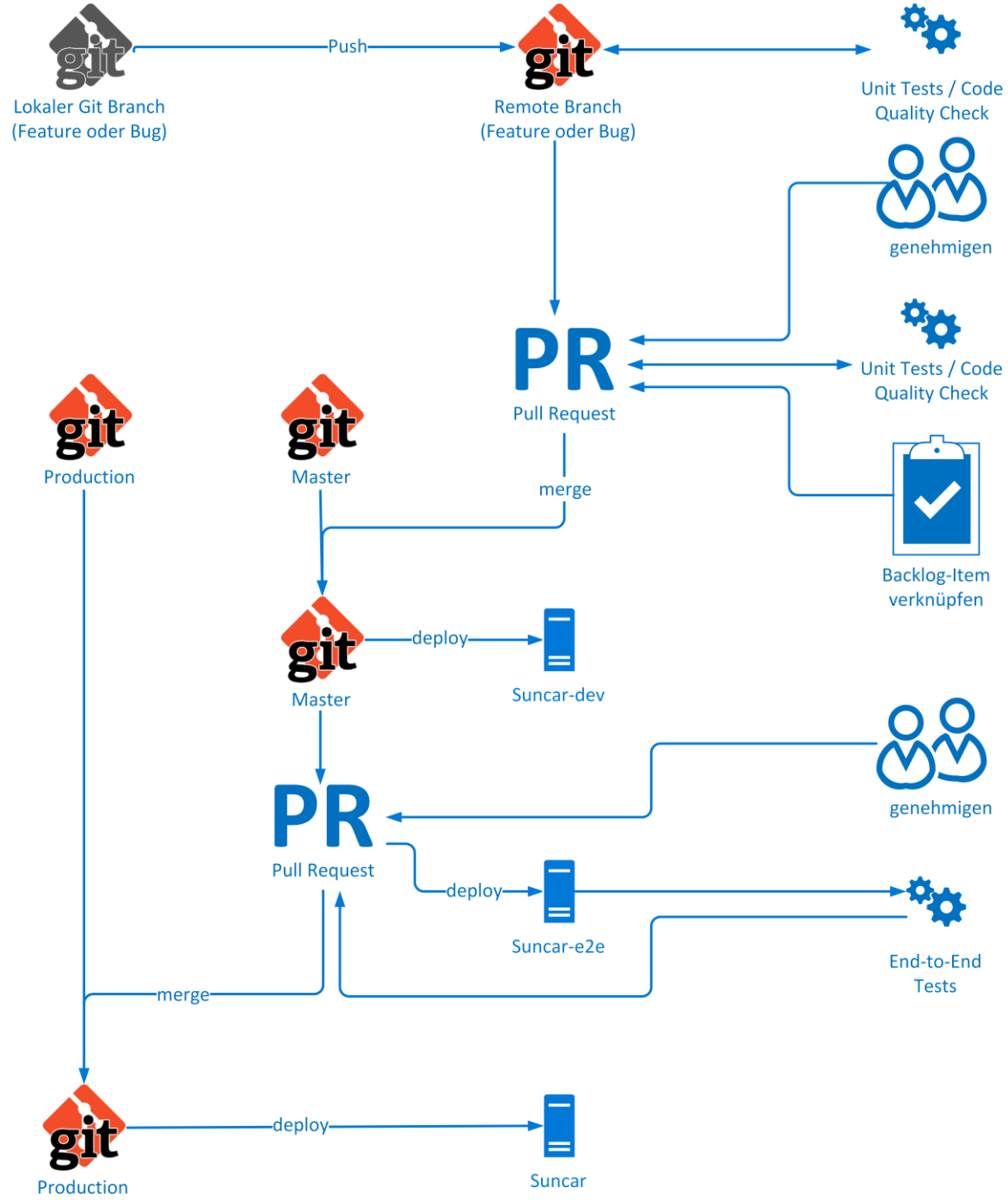


Abbildung 1: Continuous Integration-Prozess

3 Continuous Deployment

Wie bereits im Kapitel 2 (Continuous Integration) beschrieben, wird das Deployment jeweils automatisch angestossen. In diesem Kapitel ist das Deployment und der Aufbau der Umgebung dokumentiert.

3.1 Umgebung

Wie im Dokument «Cloud-Evaluation» ersichtlich ist, läuft die Umgebung komplett in Form von Docker Container auf einem einzelnen Host (VM). Für die kontrollierte Portweiterleitung sowie die Möglichkeit, mehrere Umgebungen auf einem Host zu deployen, wird HAProxy[3] eingesetzt. Mit HAProxy könnte theoretisch auch ein Throttling sowie Load Balancing gemacht werden, was jedoch zum jetzigen Zeitpunkt nicht erforderlich ist. Für ein einfaches, aber trotzdem reproduzierbares Deployment der Docker Container wird auf Docker-Compose[4] gesetzt. Das Deployment kann auf einem beliebigen Server, welche Docker und Docker-Compose installiert hat, durchgeführt werden. Es ist dafür nur eine Verbindung per SSH nötig. HAProxy ist grundsätzlich optional, vereinfacht jedoch das Handling von Port-Mappings und implementiert auch die SSL-Verschlüsselung gegen aussen inkl. Clientzertifikat – mehr dazu im Kapitel 4.2 (SSL Offloading und Zertifikate). Zusätzlich wurde auf dem aktuellen Server noch der LetsEncrypt Certbot[5] für die automatische Erneuerung der benötigten HTTPS-Zertifikate installiert, welcher mittels Cronjob periodisch ausgeführt wird.

3.2 Delivery

Nach dem erfolgreichen Mergen des Repositorys im Master-Branch (für die Staging-Umgebung) oder im Production-Branch (für die produktive Umgebung) wird das Deployment automatisch durch VSTS gestartet.

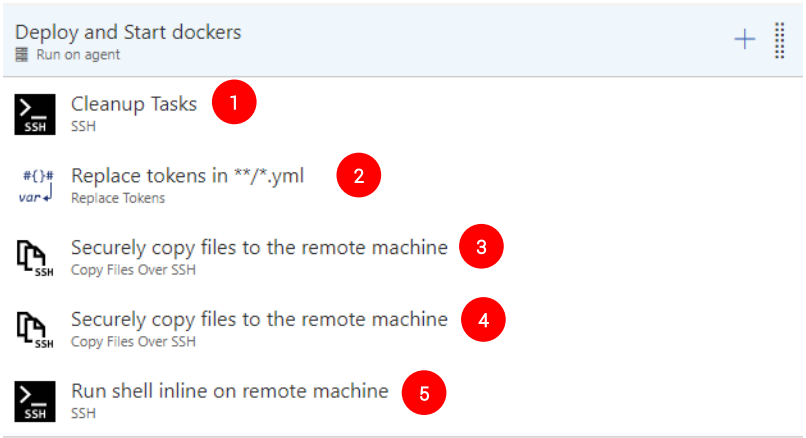


Abbildung 2: Deployment-Job

[3] (HAProxy, 2018)

[4] (Docker Compose, 2018)

[5] (Certbot, 2018)

Die vorherige Abbildung beinhaltet den in VSTS definierten Deployment-Job, welcher wie folgt abläuft:

1. Allenfalls noch vorhandene Daten von bereits vergangenen Deployments auf der Zielmaschine löschen («clean slate»)
2. Umgebungs-abhängige Docker-Compose Files generieren und die definierten Umgebungsvariablen in das File einfügen
3. Frontend-Artefakt des VSTS-Builds auf die Zielmaschine kopieren
4. Quelldaten der Teilsysteme auf die Zielmaschine kopieren und deployen
5. Docker-Container builden und starten

4 Deployment

In der nachfolgenden Abbildung wird das Deployment der produktiven Umgebung dargestellt. Dabei ist ersichtlich, wie die Container innerhalb des Docker-Netzwerkes miteinander kommunizieren. Aus Gründen der Übersichtlichkeit und Lesbarkeit wird im Diagramm auf die Darstellung von nicht relevante Informationen verzichtet.

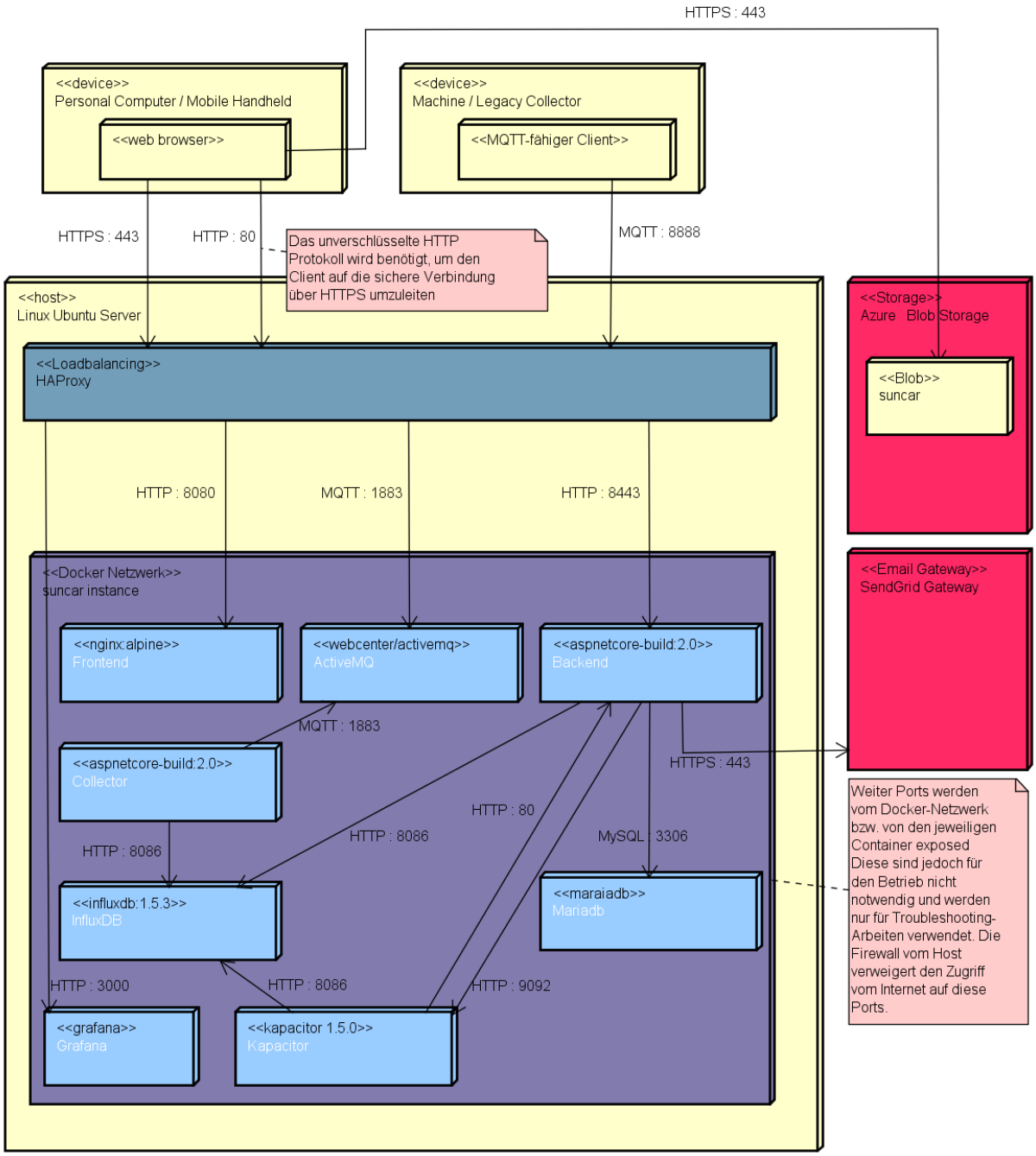


Abbildung 3: Deployment Diagramm

Mittels Docker-Compose wird automatisch ein separates Netzwerk pro Umgebung erstellt. Die Docker Container können innerhalb des Netzwerkes miteinander kommunizieren, ausserhalb des Netzwerkes ist dies nur über «Exposed Ports» möglich, welche dann an den Docker-Host gebunden sind.

Im Folgenden werden die einzelnen Docker Container aus dem Deployment Diagramm kurz beschrieben. Für detailliertere Informationen zu den Container bzw. den darin enthaltenen Applikationen, sei an dieser Stelle auf die Dokumente «Softwarearchitektur» sowie «Domainanalyse» verwiesen.

Docker Container	Beschreibung
Frontend	Statische Website basierend auf dem Image «nginx» (Docker Hub - nginx, 2018); Beim Build des Containers wird die kompilierte React-Applikation injiziert.
ActiveMQ	Message Queue, deren Konfiguration ausschliesslich mit Environment-Variablen erfolgt. Die ActiveMQ-Instanz stammt vom inoffiziellem Docker Image «webcenter/activemq» (Docker Hub - activemq, 2018)
Backend	.NET Core Applikation (Docker Hub - dotnet core, 2018), welche beim Build mit dem Quellcode generiert wird. Dafür ist kein persistenter Storage notwendig und die Konfiguration erfolgt ausschliesslich mittels Environment-Variablen.
Collector	.NET Core Applikation, welche beim Build mit dem Quellcode generiert wird. Dafür ist kein persistenter Storage notwendig und die Konfiguration erfolgt ausschliesslich mittels Environment-Variablen.
InfluxDB	Messdatenbank aus dem Image «InfluxDB» (Docker Hub - influxdb, 2018), deren Konfiguration mittels Environment-Variablen erfolgt. Der persistente Storage ist zwingend notwendig und enthält alle Messdaten.
Kapacitor	Addon der InfluxDB aus dem Open Source-Image «Kapacitor» (Docker Hub - kapacitor, 2018), deren Konfiguration mittels Environment-Variablen erfolgt.
MariaDB	Relationale Datenbank, welche vom offiziellen Image (Docker Hub - mariadb, 2018) stammt. Die Daten und Metadaten der Datenbank werden als persistenter Storage eingebunden und die Konfiguration des Docker-Container erfolgt mittels Environment-Variablen
Grafana	Grafana ist unter anderem ein Monitoring-Tool für die InfluxDB und wurde vor allem während der Entwicklung eingesetzt. Da Grafana jedoch auch als «Ad-Hoc»-Analysetool für SUNCAR HK AG einen Mehrwert bietet, bleibt es Teil des Deployments. Der Zugang ist allerdings nur für autorisierte Benutzer freigegeben. Grafana stammt aus dem offiziellen Docker Image (Docker Hub - grafana, 2018) und ist an die Influx Datenbank angebunden.

4.1 Multiinstance Deployment

Wenn das Deployment der verschiedenen Umgebungen betrachtet wird, präsentiert sich das System wie in der nachfolgenden Abbildung. Um eine weitere Umgebung auf dem gleichen Server zu deployen, muss lediglich der VSTS-Deployment-Job kopiert sowie die «Exposed Ports» und die Variable «*ENVIRONMENT_NAME*» angepasst werden. Dies ist notwendig, weil auf einem Server nicht zwei Mal der gleiche Port verwendet werden kann. Sollte die neue Instanz dann vom Internet aus erreichbar sein, müssen die Ports bzw. das DNS-Mapping noch im HAProxy angepasst werden. Sollte eine Umgebung auf einem anderen Server deployt werden, so reicht es, das Deployment zu kopieren, die Variablen entsprechend den Bedürfnissen anzupassen und die SSH-Verbindung mit den Parametern vom entsprechenden Server zu ergänzen. Danach kann eine weitere Instanz auf einem beliebigen Docker-fähigen Linux Server automatisch deployt werden.

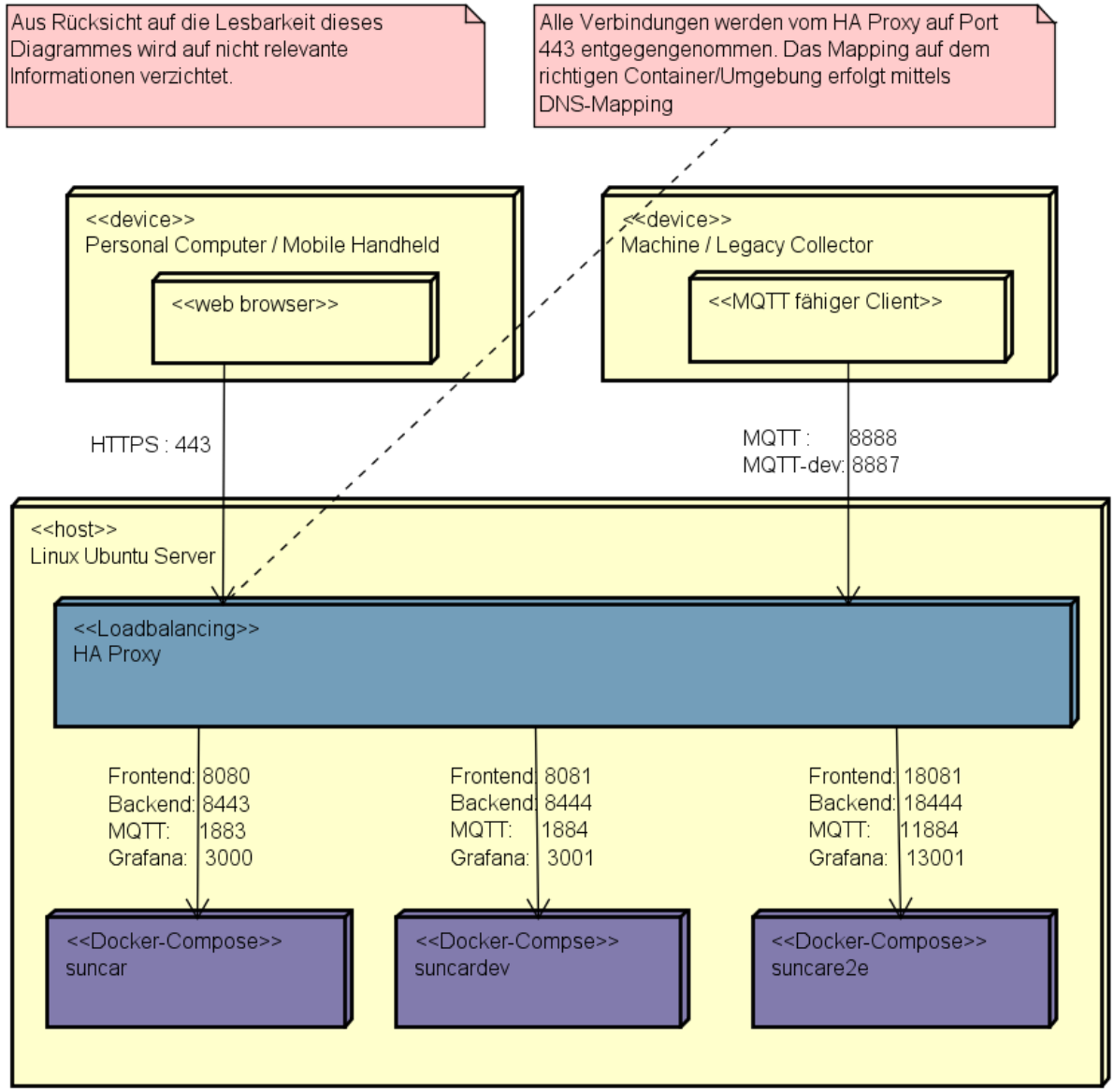


Abbildung 4 Multiinstance Deployment

4.2 SSL Offloading und Zertifikate

Damit keine Zertifikate in die einzelnen Docker-Container eingefügt werden müssen, wird ein sogenanntes SSL Offloading betrieben. Das heisst, die externen Verbindungen sind nur bis zum HAProxy verschlüsselt, danach wird intern unverschlüsselt weiter kommuniziert.

Die Zertifikate werden automatisch durch den LetsEncrypt-CertBot periodisch vor deren Ablauf erneuert. Der CertBot prüft mittels Crontab jede Nacht um 2:30 Uhr, ob die Zertifikate noch gültig sind. Mit einem pre-hook *service haproxy stop* und post-hook *service haproxy start* wird der HAProxy für den Vorgang kurz angehalten (einige wenige Sekunden). Dies ist notwendig, weil für die Zertifikatsprüfung der Port 443 notwendig ist, welcher jedoch durch den HAProxy bereits besetzt ist. Dabei hält der CertBot den HAProxy-Service nur an, wenn er tatsächlich auch ein neues Zertifikat erstellen muss. Damit beläuft sich die Downtime des HAProxy-Services auf einige wenige Sekunden alle paar Wochen, was mit Sicht auf die aktuellen Betriebsanforderungen absolut vertretbar ist.

Für den MQTT Message Broker ActiveMQ ist der Public-Key für die Client-Zertifizierung eingebunden. Im Abschnitt «Software Architektur» ist das Konzept der Clientzertifikate beschrieben. Wichtig ist dabei die HAProxy-Konfigurationsoption «*verify requried*» im MQTT Listener Backend. Dieser Eintrag verhindert den Zugriff eines Clients ohne gültiges Clientzertifikat. Im Moment ist die Option «*verify optional*» konfiguriert, da noch nicht bei allen Maschinen ein gültiges Clientzertifikat hinterleg ist. Somit ist es im Moment möglich, ohne Authentifizierung Daten an die Message Queue zu senden. Diese Option zu aktivieren bzw. die Clientzertifikate zu verteilen, liegt nicht im Scope dieser Arbeit und muss durch den Auftraggeber vor dem Go-Live durchgeführt werden. Die Konfiguration des HAProxys ist ebenfalls im Code Repository abgelegt.

4.3 Passwörter

Alle Passwörter, welche im Deployment hinterleget worden sind, wurden in einer dem Projektteam zugänglichen und verschlüsselten Passwortdatenbank abgelegt und werden bei Projektabschluss dem Auftraggeber übergeben. Die Passwortdatenbank ist aus Sicherheitsgründen nicht Bestandteil der Abgabe dieser Arbeit.

5 Skalierung

In den nichtfunktionalen Anforderungen wurde gefordert, dass die Applikation für bis zu 200 parallel aktive Maschinen auszulegen ist. Dies konnte erfolgreich getestet werden. Detaillierte Informationen dazu sind im Dokument «Systemtest» einsehbar. Weiter steht in den Anforderungen, dass die Applikation so zu bauen ist, dass sie theoretisch für bis zu 10'000 Maschinen inkl. sich daraus ergebenden Frontend-Zugriffe ausgelegt werden soll. Wie gross die Hardware bzw. Infrastruktur werden muss, kann erst mit der produktiven Erfahrung berechnet werden. Trotzdem wird in diesem Abschnitt darauf eingegangen, wie eine Skalierung bei Bedarf umgesetzt werden könnte.

5.1 Skalierungsmöglichkeiten

Die Applikation ist für verschiedene Skalierungsmöglichkeiten ausgelegt und unterstützt grundsätzlich die vertikale Skalierung (scale up). Weiter gibt es natürlich auch die Möglichkeit, horizontal zu skalieren (scale out). Die vertikale Skalierung sollte selbsterklärend sein und wird aus diesem Grund hier nicht weiter erläutert, besteht sie doch im Wesentlichen darin, dem System mehr Ressourcen (RAM, CPU, mehr I/O pro Sekunde, etc.) zuzuweisen.

5.2 Horizontale Skalierung

Eine horizontale Skalierung kann jeweils in jeder einzelnen Komponente erfolgen. So kann beispielsweise das Backend skaliert werden, sollte es sich als Flaschenhals der Applikation erweisen. Beim Anbinden von vielen Maschinen wird jedoch vor allem der Data Collector einen Engpass aufweisen. Der Data Collector kann im Moment nur gemeinsam mit der Message Queue skaliert werden. Zwischen Data Collector und Message Queue besteht mit der aktuellen Version eine 1:1-Abhängigkeit, welche auf die aktuelle Konfiguration der verwendeten Message Queue ActiveMQ zurückzuführen ist und im Bedarfsfall mit relativ geringem Aufwand angepasst werden kan.

Bei den Performance-Tests hat sich herausgestellt, dass die InfluxDB sehr viel Last verträgt und zuerst das Parsen der Messages im Collector zum Engpass wird. Bei einem solchen Szenario könnte eine Skalierung gemäss der nachfolgenden Abbildung Sinn machen.

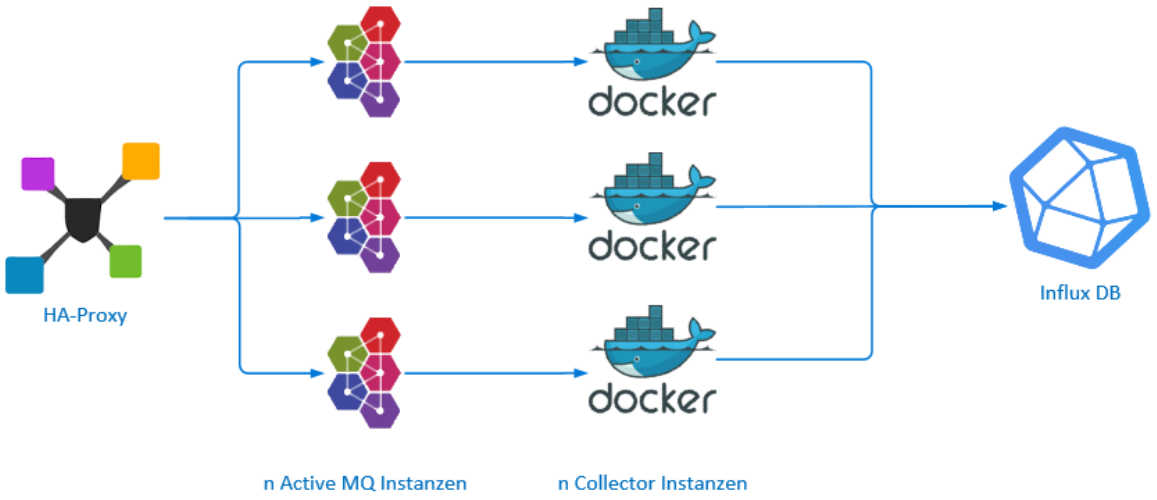


Abbildung 5: Skalierung Collector

Die Anzahl von ActiveMQ-Instanzen bzw. Collector-Instanzen ist nicht begrenzt und kann (theoretisch) unendlich weit skaliert werden. Sollte jedoch zu viele Daten geliefert werden, so kann auch eine Skalierung der InfluxDB zum Thema werden. Dies ist ebenfalls möglich, jedoch nur mit der lizenzpflichtigen Enterprise-Version. Eine schematische Darstellung der InfluxDB-Skalierung ist in der nachfolgenden Abbildung aufgezeigt.

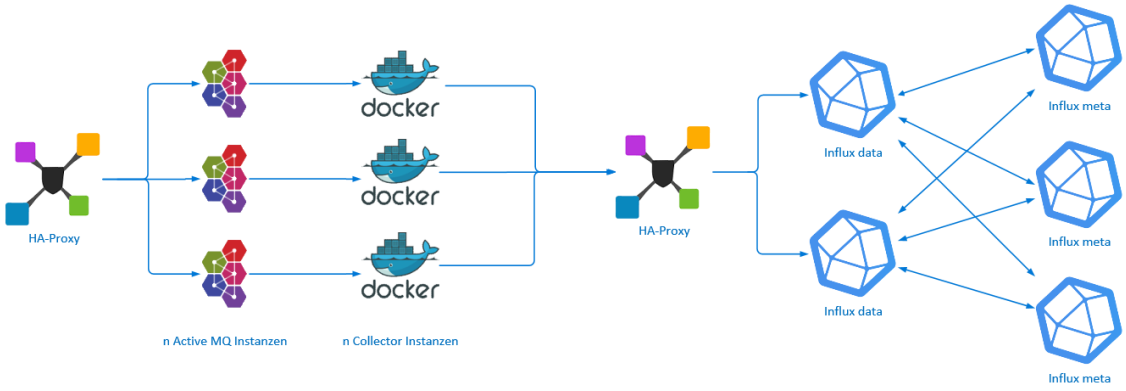


Abbildung 6: Skalierung InfluxDB

Als Loadbalancer kann auch eine andere Software eingesetzt werden, HAProxy ist jedoch sehr verbreitet und etabliert. Die InfluxDB kann die Anzahl Data Nodes weiter skalieren: Zwei Data Nodes und drei Meta Nodes ist die kleinste «Best Practice»-Skalierung der Datenbank. Auch der Kapacitor kann mit der InfluxDB skaliert werden.

Wenn das Backend skaliert wird, so macht es bestimmt auch Sinn, die MariaDB zu skalieren. Für die horizontale Skalierung gibt es diverse Lösungsansätze. Einer davon ist beispielsweise die Software-Komponente MaxScale[6]. Die detaillierte Funktionsweise wird an dieser Stelle nicht erläutert. Schematisch würde eine Skalierung jedoch wie in der nachfolgenden Grafik aussehen.

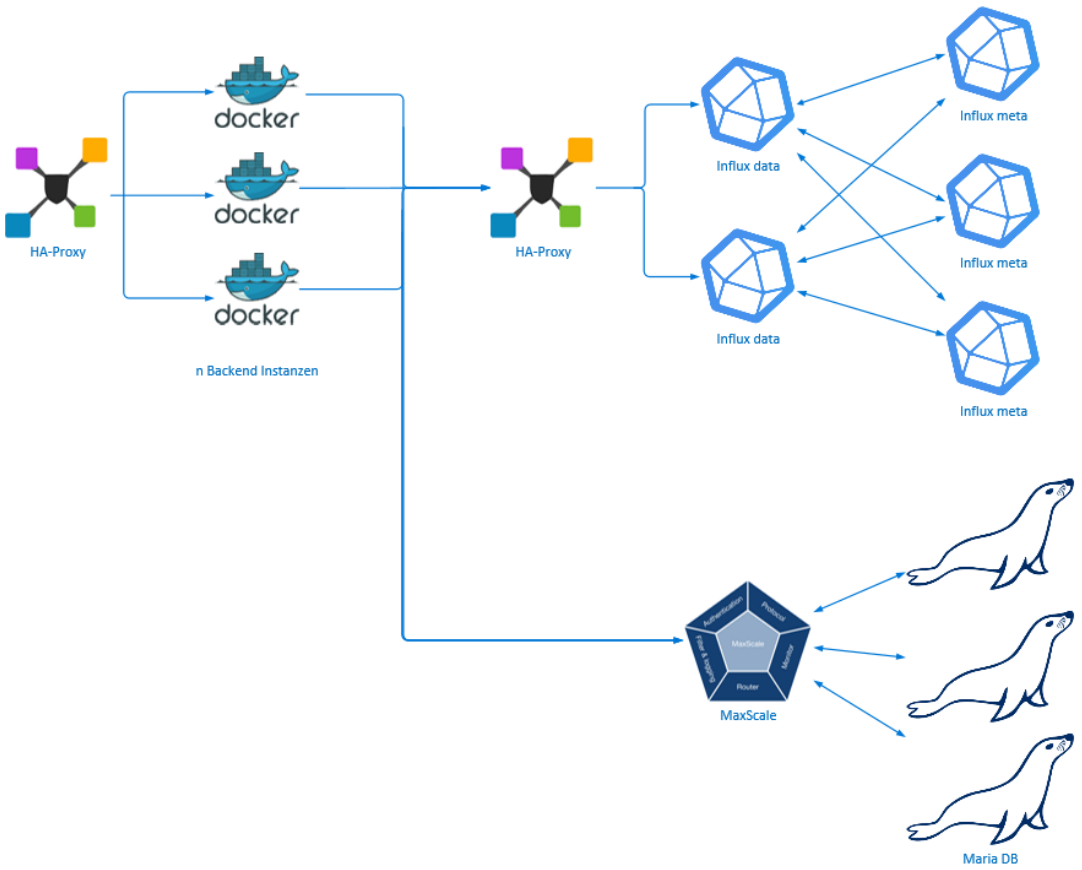


Abbildung 7: Skalierung Backend

Das Frontend kann beliebig und ohne Weiteres skaliert werden, da es sich um eine statische Webseite handelt.

5.2.1 Schlussfolgerung

Die Applikation lässt sich wie oben beschrieben sehr gut skalieren und könnte vermutlich für weit mehr als 10'000 Maschinen betrieben werden. Natürlich entstehen bei so hohe Skalierungen wiederum applikationsseitige Herausforderungen, welche zu diesem Zeitpunkt noch nicht in Betracht gezogen wurden. Beispielsweise ist das Rollenkonzept nicht für die Verwaltung von

[6] (Max Scale, 2018)

so vielen Geräten und die daraus resultierenden Benutzer ausgelegt. Diese Frage sollte in diesem Kapitel jedoch nicht beantwortet werden. Hier geht es lediglich darum aufzuzeigen, dass eine Skalierung mit dieser Softwarearchitektur möglich ist.

Es gibt auch andere Gründen, warum eine Applikation horizontal skaliert werden muss. Dies betrifft beispielsweise die Ausfallsicherheit. Werden zwei Instanzen der gleichen Softwarekomponente auf zwei unterschiedlichen Hosts betrieben, kann die Verfügbarkeit stark erhöht werden. Bei einem weltweiten Rollout der Applikation könnten mehrere Instanzen in unterschiedlichen Standorten zudem die Latenzzeiten für die einzelnen Benutzer verringern.

Sollte die Applikation skaliert oder hochverfügbar betrieben werden, so ist eine Docker- Orchestrierungssoftware wie Docker-Swarm oder Kubernetes unabdingbar.

6 Literaturverzeichnis

Certbot. (2018). Abgerufen am 13. Juni 2018 von Electronic Frontier Foundation: https://certbot.eff.org/

Docker Compose. (2018). Abgerufen am 13. Juni 2018 von Docker Docs: https://docs.docker.com/compose/

Docker Hub - activemq. (2018). Abgerufen am 13. Juni 2018 von Docker Hub: https://hub.docker.com/r/webcenter/activemq/

Docker Hub - dotnet core. (2018). Abgerufen am 13. Juni 2018 von Docker Hub: https://hub.docker.com/r/microsoft/aspnetcore/

Docker Hub - grafana. (2018). Abgerufen am 13. Juni 2018 von Docker Hub: https://hub.docker.com/r/grafana/grafana/

Docker Hub - influxdb. (2018). Abgerufen am 13. Juni 2018 von Docker Hub: https://hub.docker.com/_/influxdb/

Docker Hub - kapacitor. (2018). Abgerufen am 13. Juni 2018 von Docker Hub: https://hub.docker.com/_/kapacitor/

Docker Hub - mariadb. (2018). Abgerufen am 13. Juni 2018 von Docker Hub: https://hub.docker.com/_/mariadb/

Docker Hub - nginx. (2018). Abgerufen am 13. Juni 2018 von Docker Hub: https://hub.docker.com/_/nginx/

Git. (2018). Abgerufen am 13. Juni 2018 von Git: https://git-scm.com

HAProxy. (2018). Abgerufen am 13. Juni 2018 von HAProxy: http://www.haproxy.org/

Max Scale. (2018). Abgerufen am 13. Juni 2018 von Maria DB: https://mariadb.com/de/node/367

Monorepo. (21. Februar 2018). Abgerufen am 13. Juni 2018 von Microsoft Developer: https://blogs.msdn.microsoft.com/vsappcenter/how-mono-repo-and-one-infra-help-us-deliver-a-better-developer-experience

Betriebsdokumentation

Backup / Restore

Projektmitglieder
Stähli Tobias
Vincenz Dumeni
Wälter Jonas

Änderungsgeschichte

Datum	Version	Änderung	Autor
02.03.2018	1.0	Erstellung des Dokuments	Dumeni Vincenz
12.06.2018	1.1	Review Dokumentation	Tobias Stähli

Inhaltsverzeichnis

1 Einführung

1.1 Beschreibung

Das Dokument beinhaltet eine kurze Betriebsdokumentation bezüglich Backup- und Restore-Konzepte des Systems.

1.2 Gültigkeitsbereich

Die Gültigkeit dieses Dokuments beschränkt sich auf die gesamte Projektdauer der Bachelorarbeit im Frühjahrssemester 2018. Allfällige spätere Anpassungen oder Ergänzungen im Dokument werden in der Änderungsgeschichte festgehalten.

1.3 Referenzen

Als Nachschlagewerk für unklare Begriffe sowie Abkürzungen sei an dieser Stelle auf das Glossar verwiesen, welches fortlaufend nachgeführt wird. In der nachfolgenden Tabelle sind alle weiteren referenzierten Dokumente und Links aufgelistet.

Thema	Referenz
Aufgabenstellung	OneDrive: \01_Aufgabenstellung\Aufgabenstellung.docx
Dokumentvorlage	OneDrive: \00_Administration\Template.docx
Glossar	OneDrive: \00_Administration\Glossar.docx
Projektplan	OneDrive: \02_Projektplan\Projektplan.docx
Anforderungsspezifikation	OneDrive: \03_Anforderungsanalyse\ Anforderungsspezifikation.docx
Softwarearchitekturdokument	OneDrive: \04_Architektur_Design\ Softwarearchitekturdokument.docx
Logo Suncar HK	http://www.suncar-hk.com/

2 Backupanforderungen

Mit dem Backup soll ein Datenverlust vorgebeugt werden. Alle Messdaten sowie auch die Konfigurationsdaten der Applikation sind zu sichern, konkret handelt es sich dabei um folgende Daten:

- Influxdb (Messdaten und ausgelöste Events)
- Mariadb (Konfigurations- und Metadaten)
- Blob-Storage (Bilder)

Als nichtfunktionale Anforderung soll das Backup täglich durchgeführt und dabei eine sinnvolle Retention-Policy gewählt werden.

3 Backupkonzept

Recherchen haben ergeben, dass heutzutage als Backup von Datenbanken, welche in Docker Container laufen, oft nur der persistente Storage gesichert wird. Der grosse Vorteil dabei ist, dass das Backup und die Wiederherstellung von Datenbanken sehr einfach erfolgen kann. Dem gegenüber stehen die Nachteile, dass kein Live-Backup/Restore möglich ist sowie auch keine Transaction Logs geschrieben werden. So kann die Datenbank nicht zu jedem beliebigen Zeitpunkt wiederhergestellt werden, sondern nur zum Zeitpunkt eines Backups des Speichers.

Für die Anforderung dieser Applikation wird ein täglicher Restorepoint als ausreichend betrachtet, da die Applikation keine Transaktionen oder beispielsweise Kontostände oder ähnliches verwaltet und auch mit einer Datenlücke von bis zu 24 Stunden weiter funktionieren würde.

3.1 Backup

Der Backupspeicher ist sehr günstig, was dazu verleitet, auch einmal «zu viel» zu sichern, um den Backupprozess möglichst einfach und einheitlich für alle Applikationen zu halten. Dies hat auch zu dem Entscheid geführt, den von Azure angebotene Backupmechanismus für die ganze VM zu verwenden. Damit wird zwar «viel zu viel» gesichert, es werden jedoch die Azure-eigenen Tools verwendet und ein vielfach erprobtes Verfahren eingesetzt. Zudem ist auch der entsprechende Restore sehr einfach und komfortabel.

3.1.1 Backupintervall und Retention Policy

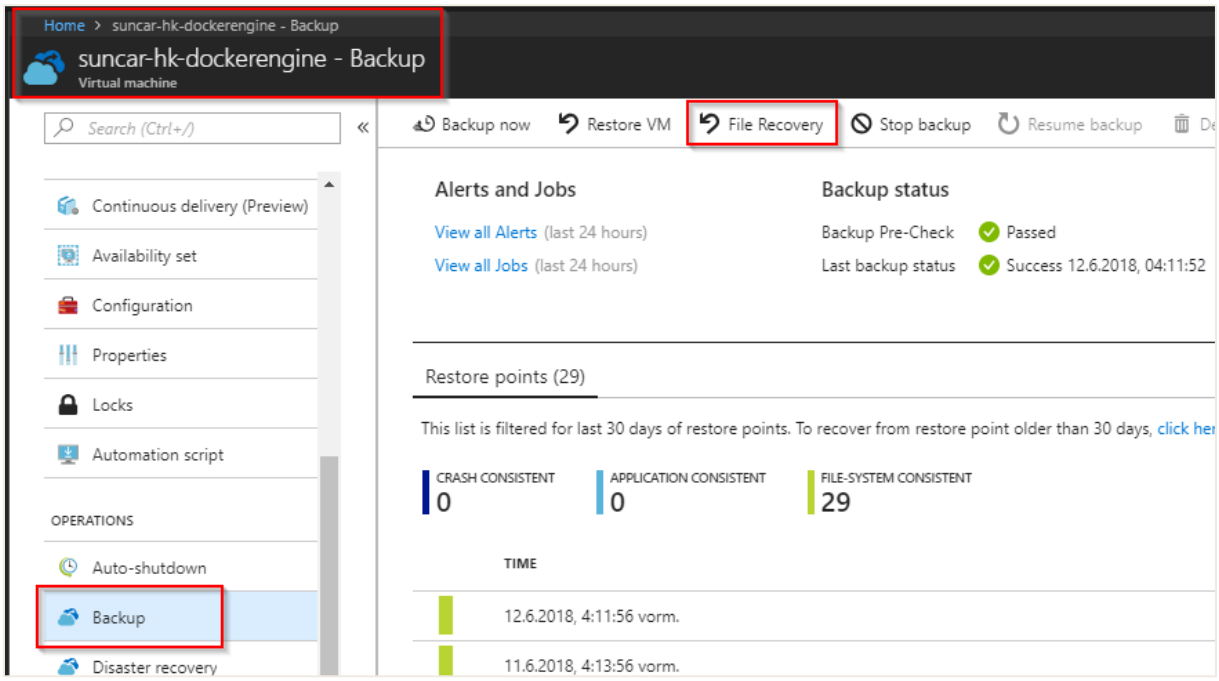
Das Backup wird jede Nacht durchgeführt und verringert so den theoretisch maximalen Datenverlust auf 24 Stunden. Die Daily Backups werden danach für 30 Tage und die Wochenbackups für 52 Wochen aufbewahrt. Damit ist die Wiederherstellung auf dem Tag genau einen Monat möglich und die maximale Wiederherstellungszeit ein Jahr. Ältere Backups sind für die Applikation kaum mehr von Bedeutung.

3.2 Restore

Als Restore kann die gesamte virtuelle Maschine wiederhergestellt werden. Dies wird wahrscheinlich nur dann verwendet werden, wenn die VM wirklich durch einen Konfigurationsfehler unbrauchbar gemacht wurde. Sollte beispielsweise ungewollt Daten gelöscht worden sein ist die Wiederherstellung von einzelnen Files oder Datenbanken nützlicher als die Wiederherstellung der ganze VM. Ein Restore der ganze VM kann im Portal von Microsoft Azure ausgeführt werden und sollte selbsterklärend sein. Der Restore von einzelne Files wird im nachfolgenden Abschnitt kurz erklärt, um in einem Fehlerfall den Stresslevel des für den Restore zuständigen Mitarbeiters nicht unnötig zu erhöhen.

3.2.1 Datenbank-Restore

Der Restore einer einzelnen Datenbank (in diesem Beispiel die InfluxDB) wird anhand des Restores vom persistenten Speicher durchgeführt.



Für den Restore der InfluxDB sind die folgenden Schritte notwendig:

1. Login in Microsoft Azure Portal und Navigation zur VM
2. Auswahl von **«Backup»** > **«File Recovery»**
3. Gewünschter Zeitpunkt des Restores auswählen und **«Download Script»** auswählen
4. Das generierte Bash Script wird im nächsten Schritt auf die VM geladen, auf welcher der Restore ausgeführt werden soll. Dies kann natürlich auch der Docker-Engine Server sein.
5. Danach kann das Skript mit **«sudo bash scriptname.sh»** gestartet werden. Bei der Anfrage muss das Passwort, welches im Portal angezeigt wird, eingegeben werden.

6. Anschliessend kann der gewünschte Pfad wiederhergestellt werden. Soll beispielsweise die produktive Datenbank wiederhergestellt werden, reicht ein der Befehl «cp -rp /path/to/mounted/restore/data1/suncar/influxdb/data /data1/suncar/influxdb/data». Es empfiehlt sich jedoch, den bestehenden Pfad auf dem Server separat zu sichern und die Daten zuerst zu löschen, um einen persistenten Bestand der Daten zu garantieren.
    a. Es kann natürlich auch ein Restore von «suncar» nach «suncar-dev» oder gar noch von einer weiteren Instanz der Applikation getätigt werden.
    b. Sollten nur Teile der Datenbank wiederhergestellt werden, empfiehlt es sich, den Restore der Datenbank in einer weiteren Instanz der Applikation zu tätigen. Anschliessend können mit Export- und Import-Methoden spezifische Teile der Daten wiederhergestellt werden.
7. Nach dem erfolgreichen Wiederherstellen der Daten sollte im Portal der Mount wieder aufgelöst werden.

Glossar

Projektmitglieder
Stähli Tobias
Vincenz Dumeni
Wälter Jonas

Änderungsgeschichte

Datum	Version	Änderung	Autor
22.02.2018	1.0	Erstellung des Dokuments	Tobias Stähli
13.06.2018	1.1	Finale Aktualisierung Einträge	Tobias Stähli

Inhaltsverzeichnis

1 Einführung

1.1 Beschreibung

Dieses Dokument dient als Glossar und enthält Definitionen von Begriffen und Abkürzungen, welche in anderen Projektdokumenten genutzt werden.

1.2 Gültigkeitsbereich

Die Gültigkeit dieses Dokuments beschränkt sich auf die gesamte Projektdauer der Bachelorarbeit im Frühjahrssemester 2018. Allfällige spätere Anpassungen oder Ergänzungen im Dokument werden in der Änderungsgeschichte festgehalten.

2 Glossar

Begriff	Definition
AMPQ	Advanced Message Queing Protocol – https://www.amqp.org
API	Eine Programmierschnittstelle eines Systems, worüber andere Kundensysteme auf die Funktionalität zugreifen können.
Big Data	Trendwort, welches Systeme bezeichnet, die eine grosse Menge an Daten verarbeiten und meist versuchen, mittels geeigneter Algorithmen Mehrwerte für ein Unternehmen zu schaffen.
CA	Certificate Authority: Teil einer PKI, der Zertifikate signiert und damit Vertrauen im Rahmen von TLS schafft.
CD	Continuous Delivery/Deployment: Bezeichnet einen Entwicklungsprozess, bei dem das laufende Produkt fortwährend und weitgehend automatisiert aktualisiert wird.
Code Coverage	Die Testabdeckung, welche sich aus dem Verhältnis der umgesetzten Unit Tests in Bezug auf die Anzahl möglicher Testfälle zusammensetzt.
CI	Continuous Integration: Bezeichnet einen Entwicklungsprozess, in dem jede Änderung am Source Code automatisiert definierten Tests unterzogen wird und anschliessend alle Artefakte des Projektes automatisiert erstellt werden.
Code Freeze	Meilenstein, nach dem der Programmiercode «eingefroren» wird und nicht mehr verändert werden darf.
DTO	Data Transfer Object: Eine Klasse, die in der Regel nur als Properties besteht und dazu dient, Daten von einem Layer in einen anderen zu transportieren.
End-to-End-Test	Art von automatisierten Integrationstest, die das gesamte System von Ende zu Ende testen und nicht wie beispielweise ein Unit Test nur einzelne Komponenten.
Feature Freeze	Meilenstein, nach dem der Funktionsumfang einer Software «eingefroren» wird und keine neuen Features implementiert werden dürfen.
Horizontale Skalierung	Auch «Scale out»-Skalierung: Ein System wird skaliert, indem die zu skalierende Komponente mehrfach instanziiert wird, d.h. parallel betrieben wird. Im Gegensatz zu vertikaler Skalierung ist der horizontalen Skalierung theoretisch keine Grenzen gesetzt.
IoT	Internet of Things: Ein Sammelbegriff, welches ein System beschreibt, bei der viele einzelne Dinge (Things) über das Internet verknüpft werden. Meist sammeln die «Things» in irgendeiner Form Daten und leiten diese an einen zentralen Daten-Verarbeiter weiter.
Message Broker	Modul in einem verteilten Softwaresystem, welches eingehende Nachrichten nach einem definierten Regelwerk an interessierte Empfänger weiterleitet und allenfalls die Zwischenspeicherung übernimmt.

MQTT	Message Queue Telemetry Transport: Bezeichnet ein für Devices mit geringen Ressourcen optimiertes Transportprotokoll für beliebigen Payload – http://mqtt.org
NFR	Non Functional Requirement: Eine nichtfunktionale Anforderung an ein System (z.B. zu verwendende Technologien oder Reaktionszeiten).
PIN	17-stellige Product Identification Number gemäss ISO 10261
PKI	Public Key Infrastructure: System, welches zur Verteilung von kryptologischen Zertifikaten zum Zwecke der sicheren Kommunikation zwischen mehreren Rechnern dient.
NPM	Node Package Manager: Ein bei der Webentwicklung weit verbreitetes Package Management System – http://npmjs.com
Predictive Maintenance	System, welches Daten sammelt und analysiert, mit dem Ziel, Fehlerzustände im überwachten System frühzeitig erkennen zu können und so proaktiv das System zu warten, um Fehlerfälle/Störungen zu beheben, bevor sie auftreten.
Public Cloud Lösung	Bezeichnet das Betreiben von Softwarelösungen auf einer von einer Drittpartei betriebenen Plattform. Die zugrundeliegende Hardware wird dabei mit anderen Kunden des Anbieters geteilt.
QoS	Quality of Service: Spezifiziert die nichtfunktionalen Anforderungen und Garantien, welche ein bestimmter Service erfüllt.
REST-API	Eine einheitliche Schnittstelle für die Kommunikation zwischen Clients und einem Webservice, wobei der Client nach der Einigung des Medientyps keine weiteren Informationen zur Schnittstelle kennen muss.
SaaS	Software as a Service: Eine Software, die von einem Anbieter betrieben wird und als Dienst ohne eigenen Aufwand bezogen werden kann.
SSL Offloading	Bezeichnet das Vorgehen, bei dem alle externen SSL-Verbindungen an einem zentralen Ort im internen Netzwerk terminiert (z.B. nach dem Load-Balancer) und anschliessend im internen Netzwerk unverschlüsselt an die jeweiligen Instanzen weitergeleitet werden. Damit können die eigentlichen Komponenten von der CPU-Laste einer SSL-Verbindung befreit werden.
TFVC	Team Foundation Version Control: Ein zentralisiertes Versionskontrollsystem von Microsoft mit ähnlicher Funktionalität wie Git.
TLS/SSL	Transport Layer Security/Secure Socket Layer: Auf TCP/IP aufbauendes Netzwerkprotokoll, das bei korrekter Anwendung eine sichere und vor Manipulationen geschützte Datenübertragung gewährleistet.
Toolchain	Sammlung von Software-Werkzeug-Programmen, welche oft in Form einer Kette nacheinander eingesetzt werden.
TypeScript	Pre-Compiler für JavaScript welche JavaScript um Typen erweitert und so gewisse Programmierfehler bereits beim Kompilieren des Quellcodes erkennen kann.
UML	Unified Modelling Language: Eine grafische Beschreibungssprache für die Darstellung von Softwaresystemen und Komponenten bzw. deren Interaktionen.

UTC	Coordinated Universal Time: die heute gültige Weltzeit.
Vertikale Skalierung	Skalierungsvariante, bei der die zugrundeliegende Hardware der zu skalierenden Komponente mit einer besseren Hardware ersetzt wird. Da Rechnerkapazitäten aufgrund von physikalischen Einschränkungen nicht beliebig aufgerüstet werden können, ist die vertikale Skalierung nur bis zu einem gewissen Punkt möglich. Diese Einschränkung ist bei horizontaler Skalierung nicht vorhanden.
VIN	17-sellige Vehicle Identification Number gemäss ISO 3779.
VPN	Virtual Private Network: Software-Technologie, um mehrere meist physikalisch entfernte Clients über ein unsicheres Netzwerk (i.d.R. das öffentliche Internet) in einem gemeinsamen und geschützten logischen Netzwerk miteinander zu verbinden.
VSTS	Visual Studio Team Services: Microsoft Plattform zur Verwaltung von Code-Versionierung, Arbeitspaketen, Builds, Tests, uvm.