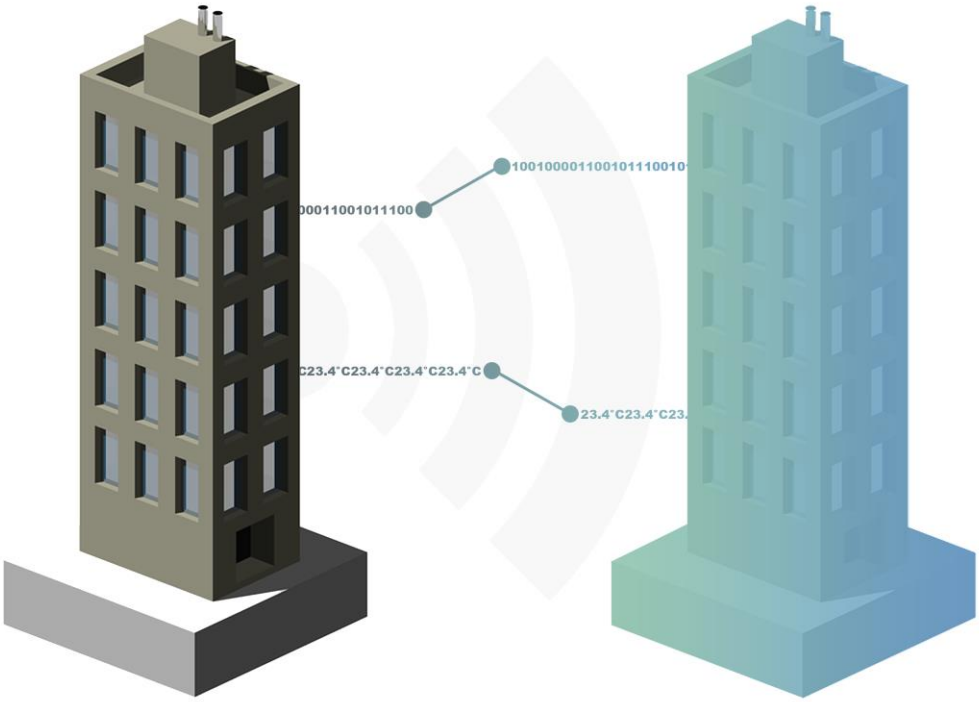
HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

SIEMENS

Cloud based BIM with real time visualisation



Bachelorarbeit

Abteilung Informatik
Hochschule für Technik Rapperswil

Frühlingssemester 2019

Autoren:	Sandro Blatter, Benjamin Daniels
Betreuer:	Prof. Dr. Daniel Patrick Politze
Projektpartner:	Siemens Zug
Experte:	Ramon Schildknecht

Inhaltsverzeichnis

1 Abstrakt

1.1 Einleitung

Den Impakt, den BIM auf die Baubranche haben wird, ist nur sehr schwierig abzuschätzen. Der bisherige Prozess wie ein Gebäude geplant, gebaut, renoviert und gewartet wird, ist dem BIM Prozess anzupassen. Der Fokus dieser Arbeit besteht in der Visualisierung von BIM-Gebäuden und der Integration der Daten generierenden verbauten Sensoren. Die Sensoren sind in einem Demokoffer vorhanden. Der Demokoffer beinhaltet verschiedene Siemens eigene Sensoren und Controller, der die Daten über das Protokoll BACnet zur Verfügung stellt. Die Sensordaten werden mittels MSIB, das auf der Nanobox installiert wurde, weitergeleitet in die Cloud von Amazon. Sind die Daten über AWS verfügbar, kommen die in dieser Arbeit entwickelten Anwendungen zum Zuge.

1.2 Vorgehen

Das Vorgehen in dieser Bachelorarbeit ist von einer Agilität kaum zu übertreffen. Die erhaltene Freiheit seitens Siemens ist beachtlich und führte dazu, dass die Arbeit kein Müssen, sondern ein Dürfen war. Das wöchentliche Review und die Planung waren essentiell für das Verständnis und für die Zusammenarbeit mit Siemens. Während einem Sprint wurde ein Kanbanboard geführt, damit jeder Stakeholder im Projekt über den aktuellen Status Kenntnis hat. Ein Hindernis manifestierte sich in der anderen parallel entwickelten Software MSIB. Die Software hat den Zweck Sensordaten aus dem Gebäude auszulesen und an die AWS Cloud weiterzuleiten. Durch mehrere Abwesenheiten einer Ansprechperson und den sehr knappen Deadlines wurde daher für einen ersten Schritt auf die Live-Daten des Siemens HQ in Zug gewechselt. Dank der sehr unterstützenden Hilfe seitens Siemens und einem weiteren HSR Studenten konnte die Software so konfiguriert werden, dass nun Live-Daten aus dem Demokoffer in unseren Anwendungen anzeigt werden können.

1.3 Ergebnis

Die hochgesteckten Ziele dieser Arbeit wurden seitens des Projektteams und Siemens als durchwegs erfolgreich bezeichnet. Die drei Anwendungen, die in dieser Bachelorarbeit geplant und konstruiert wurden, sind so geplant, dass sie in der nahen oder fernen Zukunft gut erweitert werden können. Obwohl der Treolizer die Anwendung ist, die als erste das Augenmerk auf sich zieht, sind die anderen zwei Anwendungen nicht zu unterschätzen. Gerade der Duolizer bietet eine sehr gute Basis, auf der weitere Ideen umgesetzt werden können. Die leichtgewichtige Realisierung lässt zu, dass die Anwendung von jedem Gerät, ob gross oder klein gleichauf benutzt werden kann. Hinsichtlich den aus dieser Bachelorarbeit hervorgegangenen Anwendungen ist ein sehr grosses Spektrum von möglichen Endgeräten abgedeckt. Die Vielfalt der Anwendungen ist einer der Aspekte, auf den das Projektteam sehr stolz ist. Eine andere Facette ist die sehr tolle Zusammenarbeit mit Siemens und unseren Experten der HSR. Das Fazit dieser Bachelorarbeit "Cloud based BIM with real time visualization" lässt sich folgendermassen zusammenfassen: Dem Projektteam hat die Zusammenarbeit und das Resultat aus dieser Arbeit sehr grosse Freude bereitet und es wird gehofft, dass die Anwendungen in Zukunft weiterentwickelt werden.

2 Management Summary

Die Bandbreite an Themen, die in dieser Bachelorarbeit tangiert werden, ist erstaunlich. Sei es in der Hardwareebene mit BACnet und die verwendeten TCP / UDP Protokolle oder in der Softwareebene mit Unity und der Datenbesorgung mittels REST API. Ziel der Bachelorarbeit besteht darin, die Technologien in ein grosses System einzubinden, um Gebäudedaten möglichst realitätsnah in drei verschiedenen Applikationen zu visualisieren.

Die erste Applikation hat die Aufgabe, den Demokoffer in einer schematischen Ansicht innerhalb einer Webseite anzuzeigen. Die im Vorhinein definierten Sensoren: Temperatur, Präsenz, Luftqualität und Luftfeuchtigkeit werden als Icons über einem Bild des Demokoffers angezeigt. Der Benutzer kann auf die Icons klicken und die Daten des Sensors werden danach in einem Diagramm angezeigt. Es besteht die Möglichkeit der Veränderung der Herkunft der Sensordaten. Die erste Option ist, die Informationen aus dem Siemens HQ in Zug zu verwenden. Die zweite Option besteht darin, dass die Daten vom Demokoffer generiert werden. Die Daten werden vom Demokoffer über die Docker Container Version von MSIB, die auf der Nanobox läuft, gesammelt und an Amazon Web Services weitergeleitet. Um die eine oder andere Option auszuwählen, müssen die Links zu den Diagrammen innerhalb der Applikation angepasst werden.

Die zweite Applikation ähnelt der ersten Anwendung stark. Der einzige Unterschied ist der Hintergrund. Anstatt des Demokoffers wird der Raumplan des sechsten Stocks des Siemens HQ in Zug verwendet. Im Raumplan ist es möglich, ein Zimmer auszuwählen und deren Sensordaten in einem Diagramm anzuzeigen.

Die dritte und letzte Applikation funktioniert unterschiedlich zu den ersten beiden Anwendungen. Die Visualisierung verwendet vier verschiedenen BIM Modelle vom Siemens HQ in Zug:

- Heizung
- Architektur (Grundstrukturen wie Wände, Böden, Verankerungen, usw.)
- Installationen (Lüftung, Strominstallationen)
- Fassade

Die BIM Modelle werden als FBX Datei in Unity importiert. Leider ist es momentan nicht möglich anhand der eindeutigen Sensor-Id, direkt auf die Sensordaten in der REST API zu schliessen. Deshalb sind die einzelnen Sensoren direkt den dazugehörenden Diagrammen zugewiesen. Die Navigation im dreidimensionalen Raum wird mittels Xbox Controller gewährleistet. Zusätzlich besteht die Option, falls kein Xbox Controller vorhanden ist, sich mittels Maus und Tastatur vorzubewegen.
Um den Benutzer die Zurechtfindung zu erleichtern, wird für den sechsten Stock vom Siemens HQ in Zug der Raumplan als kleine Karte im Heads-up-Display (HUD) angezeigt.

Mit den drei Lösungen ist etwas für jedermann dabei. Sei es vom kleinen Mobile Device, das die Sensordaten über eine Webseite abgreifen kann oder ein anspruchsvolleres Device, welches die Sensordaten im Gebäudeplan einsehen kann.

3 Danksagung

An dieser Stelle sind einige Danksagungen angebracht.

Daniel Politze
An erster Stelle wird ein Dank an den Experten Daniel Politze ausgesprochen. Die Zusammenarbeit verlief sehr positiv, da auf das bestehende Vertrauen beider Parteien gebaut werden konnte. Das Projektteam genoss sehr grosse Freiheiten in Bezug auf die Zusammenarbeit mit dem Industriepartner.

Markus Winterholer
Seitens Siemens und gerade der Ansprechperson namentlich Markus Winterholer ist ein grosses Dankeschön auszusprechen für die unkomplizierten Terminvereinbarungen und das Vertrauen in die vorgeschlagenen Lösungen der jeweiligen Anwendungen.

Jan Forlin
Einem Mitstudenten ist ebenfalls ein Dank auszusprechen. Jan Forlin gab dem Projektteam wertvolle Inputs und Lösungen in den Bereichen Linux, Docker und Docker Networking. Die Inputs haben stark zum Erfolg der Bachelorarbeit beigetragen.

4 Einleitung

4.1 Problembeschreibung

Versetzt man sich in die Situation eines Installateurs oder eines Technikers, der den Auftrag hat einen bestimmten Sensor in einem Haus zu finden, merk man schnell, dass die Suche sehr schwierig werden kann. Die Anzahl der Sensorinstallationen in einem Gebäude nimmt stetig zu. Die Frage stellt sich nach der Installation: “Wie kann ich den verbauten Sensor wiederfinden?”. Das Problem soll mit einem neuen Standard mitigiert werden. Der Standard hört auf die Bezeichnung BIM (Building Information Model). BIM erlaubt die Erstellung eines digitalen Zwillings eines Gebäudes. Der digitale Zwilling hat das Ziel möglichst identisch zum physischen Gebäude zu sein. Sensoren und andere Installationen wie Wände, Fenster, Ventilation, Heizung, Storen usw. werden im digitalen Zwilling als Objekte an gleicher Position im Modell platziert. Daraus entsteht der Vorteil, dass Sensoren leichter im digitalen Zwilling aufgefunden werden können, als sich physisch im Haus bewegen zu müssen.
Der BIM Standard befindet sich in Entwicklung. Eine generelle Lösung für Punkte wie Gebäudeinstandhaltung, Live Sensordatenanbindung und Navigation in einem 3D digitalen Zwilling befindet sich noch im Entwicklungsstatus. Das Ziel dieser Bachelorarbeit besteht darin, neue Ansätze und mögliche Lösungen für die genannten Baustellen in BIM zu finden.

4.2 Motivation

Das Ergebnis dieser Arbeit soll Siemens helfen, den bestehenden Prozess der Anschliessung eines Gebäudes an die DigitalTwin Plattform zu optimieren. Die Visualisierung der empfangen Daten wird erneut evaluiert und in verschiedenen Arten und Dimensionen dargestellt. Siemens besitzt so die Möglichkeit, ihre Plattform live mit dem Demokoffer vor dem Kunden zu präsentieren.

4.3 Komponente

In dem folgenden Kapitel werden jegliche Komponenten beschrieben, die in dieser Bachelorarbeit verwendet wurden.

4.3.1 Demokoffer

Der “Demokoffer” verhält sich wie ein kompaktes Gebäude mit Sensoren. Der Koffer ist ausgestattet mit verschiedenen Sensoren, wie zum Beispiel ein Bewegungssensor, Temperatursensor oder ein Luftqualitätssensor. Jegliche Komponenten sind an einem PCX3 Siemens Kontroller angeschlossen.

Abbildung 1 - Demokoffer

4.3.2 Nanobox

Die Simatic Nanobox ist ein Industrie-Computer der Siemens. Ein Industrie-Computer hat zusätzliche Eigenschaften zu einem üblichen Computer. Die Kühlung der Nanobox ist von Ventilatoren befreit, das hat den Vorteil, dass die Nanobox weniger Staubanfällig ist.

Abbildung 2 - Simatic Nanobox

Das Betriebssystem der Nanobox ist eine Linux Distribution. Zusätzlich ist Docker auf der Nanobox installiert.

4.3.3 BACnet

Das Building Automation Control Network (BACnet) ist ein Protokoll für die Übermittlung von Daten und Befehlen vom und zu einem Gebäudecontroller.

4.3.4 MSIB

MSIB ist ein von Siemens entwickelter BACnet Leser, der es ermöglicht, mehrere BACnet Devices auszulesen. MSIB verwendet die Polling Art um Daten des Gebäudekontrollers auszulesen. Polling bedeutet, dass die Daten nach einem variablen Zeitintervall von MSIB aktiv abgefragt werden.

4.3.5 OpenSSH / DropbearSSH

DropbearSSH ist eine Alternative zu OpenSSH. Die Kommunikation über das SSH Protokoll kann damit umgesetzt werden. Verwendung findet diese Applikation um Dateien von den Entwicklungscomputern auf die Nanobox zu transferieren. DropbearSSH ist bereits auf der Nanobox installiert. Deshalb wird mit dieser Applikation gearbeitet. Der Vorteil zu OpenSSH ist der geringe Arbeitsspeicher und CPU Verbrauch.

4.3.6 Amazon Web Services

Im Projekt werden folgende Services von Amazon tangiert:

Service	**Beschreibung**
AWS IoT Core	Übersicht aller IoT Geräten und Ort für die Registrierung des Demokoffer. AWS IoT Core bildet die Kommunikationsschnittstelle zwischen dem Demokoffer und der Nanobox. IoT steht für “Internet of Things” und ist als Oberbegriff für die Vernetzung von Gegenständen zu verstehen. Ein registriertes Objekt in AWS IoT Core wird als “Thing” bezeichnet. Ein “Topic” ist ein eindeutiger Pfad, der für ein bestimmtes “Thing” gedacht ist. In einer “Regel” ist es möglich auf gesendete Daten von ein “Topic” zuzugreifen. Eine “Regel” kann dazu verwendet werden, Daten die zu einem “Topic” gesendet werden an einen anderen Amazon Web Service weiterzuleiten.
Amazon Kinesis Data Firehose	Ermöglicht die dezentralisierte Datenübermittlung an einen Amazon Elasticsearch Service.
Amazon Elasticsearch	Service um die erhaltenen Daten zu indexieren und für eine Weiterverarbeitung vorzubereiten.
Kibana	Visualisiert die von Amazon Elasticsearch vorbereiteten Daten.

Tabelle 1 - Amazon Web Services (AWS)

4.3.7 BIM

Das Ziel vom Building Information Modeling (BIM) besteht darin, den gesamten Bau und die folgende Instandhaltung von Gebäuden zu optimieren. Als erstes wird der Bauplan als ein virtuelles Modell abgebildet. Je nach Detailgrad können selbst kleinere Komponenten wie zum Beispiel Steckdosen geplant und im Modell exakt platziert werden. Die Kosten eines Gebäudes sind so einfacher zu kalkulieren. Jegliche Personen, die mit dem Gebäude zu tun haben, sollen auf die gleichen Daten im Modell zugreifen. Der Informationsaustausch zwischen den verschiedenen Parteien wird dadurch optimiert. Ist ein Modell erstellt, besteht das Ziel darin, dass das Modell während dem gesamten Lebenszyklus des Gebäudes aktuell gehalten wird.

4.3.8 DigitalTwin

DigitalTwin ist eine webbasierte Plattform, die sämtliche Dokumente für ein reales Gebäude abspeichert. Zum Beispiel ist es möglich, die Gebäudepläne 2D / 3D dort zu speichern. Jede Änderung an den Plänen ist für alle Stakeholder sofort ersichtlich. Je nach Integration des Gebäudes mit der Plattform lässt diese zu, die installierten Komponente aufzulisten oder Applikationen auszuführen, die aktuelle Gebäudedaten analysieren.

4.3.9 Schemalizer

Der Schemalizer hat die Hauptaufgabe, den Demokoffer und dessen Daten im 2D Format in einer Webapplikation anzuzeigen. Der abgebildete Demokoffer soll interaktiv sein und so dem Benutzer eine einfache und klare Übersicht der generierten Sensordaten bieten. Zusätzlich werden im Schemalizer verschiedene Darstellungsmöglichkeiten der Daten analysiert, die auch im Duolizer und Treolizer wiederverwendet und eingesetzt werden können.

4.3.10 Duolizer

Der Duolizer ist auf dem Schemalizer aufgebaut. Hier wird wiederum eine Webapplikation aufgesetzt, um 2D Daten zu visualisieren. Nur werden anstatt Daten vom Demokoffer, Live-Daten der Siemens Gebäude in Zug verwendet. In einem ersten Schritt wird eine Etagenübersicht angezeigt, wo ein beliebiges Zimmer ausgewählt werden kann. Anschliessend werden die Live-Daten des ausgewählten Zimmers angezeigt. Möglichst viele Komponente des Schemalizers sollen übernommen werden.

4.3.11 Treolizer

Der Treolizer soll die 3D Modelle eines Gebäudes anzeigen können. Dazu wird eine Computerspiel Engine als Basis verwendet. Welche Engine verwendet wird, ist in einer Evaluation herauszufinden. Die Navigation im 3D Raum soll mit einem Computerspiel Controller möglich sein und so eine neue Art der Orientierung in einem grösseren Gebäude veranschaulichen.

4.3.12 Grafana

Grafana ist ein weiteres Tool für Datenvisualisierung und Datenüberwachung. Die Anbindung zu Elasticsearch wird unterstützt und geschieht mit wenigen Klicks.

4.3.13 Chart.js

Chart.js ist eine Open Source Javascript Bibliothek. Sie ermöglicht eine einfache und optisch angenehme Darstellung der Daten.

4.3.14 Siemens API für Live-Daten

Wie bereits im Management Summary beschrieben, werden zwei verschiedene Datenquellen verwendet. Einerseits die Daten vom Demokoffer, anderseits Live-Daten des Siemens Gebäude in Zug. Zwei verschiedene API's für die Daten des Siemens Gebäude in Zug werden zur Verfügung gestellt.

API	Beschreibung
API offen (search-nrm3-cf-elastic)	Diese API benötigt keine Autorisation und ist für alle offen. So können Daten schneller getestet werden und die API ist nicht von einem Key abhängig.
API geschlossen (https://v2.api.demo.digitaltwin.bt.siemens.cloud)	Diese API benötigt eine Autorisation. Mit den Siemens Login Daten ist es möglich eine API-Key zu generieren.

Tabelle 2 - Verwendete Siemens API's

4.4 Verwendete Software

In diesem Projekt wurden verschiedene Applikationen für verschiedene Zwecke benutzt.

Software	Einsatzzweck
Trello	Kanban Tool für Projekt Management und Sprint Planung.
WebStorm	Entwicklung der Web Applikation.
Office	Erstellung der Dokumente für das Projektes.
DropbearSSH	Geschützter Zugriff auf Nanobox.
Postman	Testen von REST APIs.
Unity	3D Engine zur Visualisierung von BIM Plänen.
Revit	Ansicht und Exportierung BIM Plänen.
OneDrive	Speichern und teilen von Dokumenten.
GitHub	Speichern und teilen von Code.

Tabelle 3 - Verwendete Software

5 Projekt Initialisierung

Die Projekt Initialisierung ist dazu verwendet worden, einen Überblick über die Komponenten und das technische Umfeld zu erarbeiten.

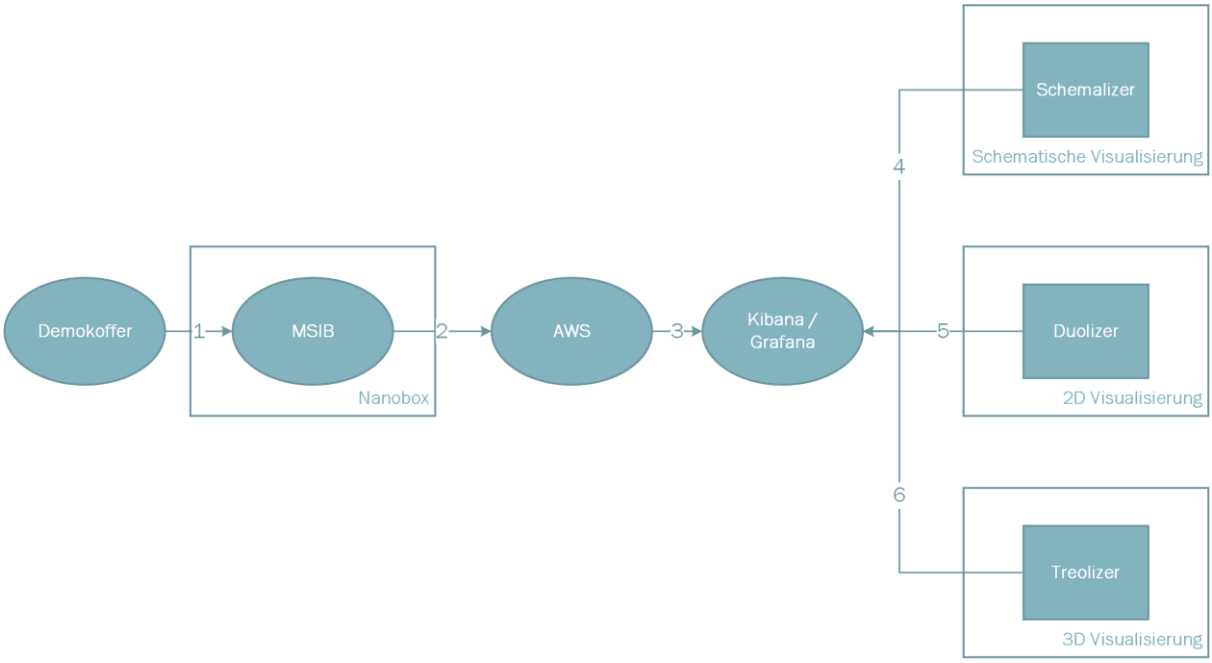


Abbildung 3 - Gesamtübersicht

Als erstes wird der Rahmen für das Projekt abgesteckt. Die Visualisierung des Rahmens ist in der Folge in Form eines Diagrammes erstellt worden. Im Diagramm sind die einzelnen Komponenten ersichtlich und wie sie zusammen im Gesamtsystem stehen. Das Diagramm wird für die Sprintplanung verwendet, um den Gesamtfortschritt zu visualisieren. Die einzelnen Zahlen stehen für die Prioritäten und die Reihenfolge, wann welche Verbindung erstellt werden soll.

Die Einarbeitung in die einzelne Komponente des Projekts erfolgte durch “Onboardings”. “Onboardings” sind Sitzungen, in denen ein Experte das Thema mit den anderen Sitzungsteilnehmern erklärt und offene Fragen beantwortet.

Onboarding Thema
MSIB Installation auf Nanobox
DigitalTwin Plattform
Revit & BIM Viewer

Tabelle 4 - Siemens Onboardings

6 Projekt Elaboration

Die Elaboration beinhaltet die Analyse der Anforderungen, Konzeptionierung der Lösung und Planung einer möglichen Umsetzung.

6.1 Anforderungsanalyse

Die Anforderungsanalyse ist in einem Use Case Diagramm persistiert. Für jede Anwendung wird ein Diagramm erstellt.

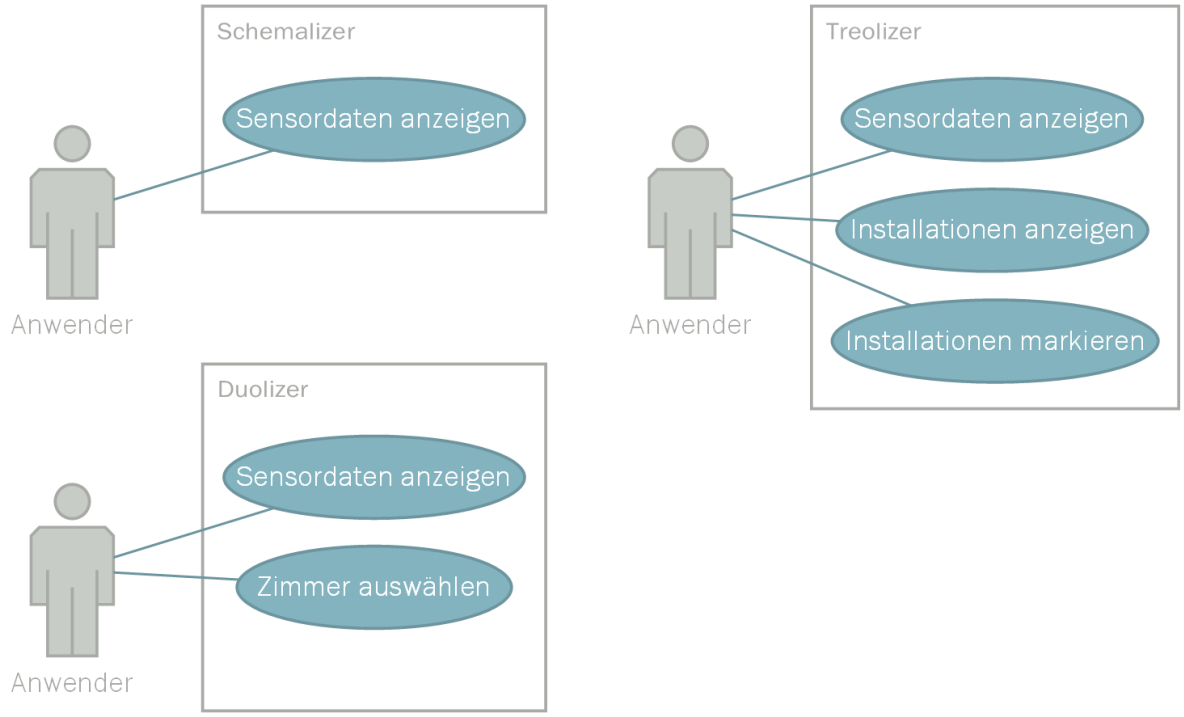


Abbildung 4 - Use Case Diagramm

Use Case	Beschreibung
Sensordaten anzeigen	Der Use Case “Sensordaten anzeigen” ist für jede Anwendung ähnlich. Jedoch ist die Umsetzung von System zu System unterschiedlich. Dennoch ist das Ziel des Use Cases das Gleiche, dem Anwender die Sensordaten anzuzeigen.
Installationen anzeigen	Die Auflistung aller Installationen in Kategorien ist das Ziel des Use Cases “Installationen anzeigen”.
Installationen markieren	Eine ganze Kategorie von Installationen sollen im Treolizer markiert werden, sobald der Anwender den Use Case “Installationen markieren” durchführt.
Zimmer auswählen	Im Duolizer wird ein Raumplan ersichtlich sein. Der Anwender soll dort einen gewissen Raum auswählen können, um deren Sensordaten anzuzeigen.

Tabelle 5 - Use Case Beschreibung

6.1.1 Nicht funktionale Anforderungen

Die nicht funktionalen Anforderungen werden gemäss den FURPS Kriterien notiert.

Functionality	Die Visualisierungen sollten im MVC Prinzip aufgebaut werden um die Wiederverwendbarkeit zu erhöhen.
Usability	Die Visualisierungen, egal Demokoffer oder Stockwerk, sollen hauptsächlich für Tastatur und Maus optimiert gebaut werden.
	Die Navigation im dreidimensionalen Raum wird mittels Xbox Controller realisiert werden.
	Die Benutzerführung im Userinterface der Visualisierungen hat das Ziel, die Daten mit weniger als zwei Klicks anzuzeigen.
Reliability	Die Diagramme sollen von den Userinterfaces der Visualisierungen unabhängig geladen werden.
	Die Visualisierungen sind so zu planen, dass sie auf die AWS Cloud migriert werden können.
Performance	Die Visualisierungen stehen im Zeichen der Einfachheit und werden daher mit möglichst wenigen zusätzlichen Ressourcen belastet.
Supportability	Die Visualisierungen werden mit dem MVC Prinzip umgesetzt und sind durch die Trennung von Daten und Businesslogik einfacher zu warten.

Tabelle 6 - Nicht funktionale Anforderungen

6.2 Pflichtenheft

Das Pflichtenheft beschreibt, wie die Anforderungen vom Auftraggeber umgesetzt werden können. Das Vorgehen für die Erfüllung der Anforderungen wird anhand einer Checkliste beschrieben.

Nr.	Schritte	Beendigung
1	Initialisierung	☑
1.1	Einarbeitung in vorgegebene Themen	☑
2	Elaboration	☑
2.1	Anforderungsanalyse erstellen	☑
2.2	IST und SOLL Zustand analysieren	☑
2.3	Software Architektur Pattern evaluieren	☑
3	Konstruktion	☑
3.1	Gemäss Priorisierung Sprint Planung planen - Für jeden Sprint ist es das Ziel ein Minimum Viable Produkt (MVP) als Resultat zu erhalten	☑
3.2	Manuelles Testen der einzelnen Komponenten	☑
4	Transition	☑
4.1	Stand des Gesamtsystems dokumentieren	☑

Tabelle 7 - Pflichtenheft

6.3 IST Zustand Analyse

Die einzelnen Hardware-Komponente Nanobox und Demokoffer sind vorhanden. Seitens der Software muss der Docker Container von MSIB noch besorgt werden. Der Docker Container für MSIB wurde noch nie im Zusammenhang mit der Nanobox und der installierten Linux Distribution eingesetzt.

Sensordaten können von MSIB ausgelesen und an AWS weitergeleitet werden. Die genaue Konfiguration der MSIB Komponente wurde bereits von einigen Siemens Mitarbeiter durchgeführt, jedoch noch nie mit dem Docker Container.

Der MSIB Docker Container ist als File hinterlegt und muss über das SCP Protokoll auf die Nanobox übertragen werden. Auf der Nanobox ist ein SSH Anwendung DropBearSSH installiert um die Authentifikation beim Zugriff auf die Nanobox sicherzustellen.

Der Demokoffer und deren Steuerung ist programmiert. Hinsichtlich des Demokoffers muss in dieser Arbeit nichts erledigt werden. Eine Liste mit allen BACnet bereiten Sensoren und Komponenten ist bereits in der Studienarbeit “Integration von Siemens Sensoren mit Amazon Web Services” erarbeitet worden. Die Liste soll nun auf Sensoren und Komponenten getrimmt werden, die für dieses Projekt benötigt werden.

Ein DigitalTwin ist für den Demokoffer auf der DigitalTwin Plattform erstellt. Die DigitalTwin Plattform ermöglicht es zusätzlich auf Gebäudedaten zuzugreifen, die bereits mit MSIB verknüpft worden sind. Um die Applikationen "Schemalizer" und "Duolizer" umzusetzen, kann auf eine bestehende Codebasis zugegriffen werden. Für den “Treolizer” besteht keine Codebasis und muss daher von Grund auf entwickelt werden.

Für die Darstellung eines Gebäudes in einem dreidimensionalen Raum bestehen vier Modelle vom HQ von Siemens in Zug. Die Modelle wurden mit Revit erstellt.
Es gibt vier Kategorien für die BIM Modelle: Heizung, Architektur, Installationen und Fassade.

6.4 Architektur Pattern

In dem Projekt werden zwei Architektur Patterns verwendet.

6.4.1 Pipes and Filters

In der AWS Umgebung wird mit verschiedenen Services gearbeitet, die Daten von einem zum nächsten übertragen und allenfalls Änderungen vollziehen. Die Datenquelle ist hier der Demokoffer. Die Datensenken sind die Visualizer.

Abbildung 5 - Pipes and Filters

6.4.1.1 Vorteile

Einzelne Services können ausgetauscht und allenfalls kann die Pipeline erweitert werden. Die Struktur der Services wird von AWS so bereitgestellt, dass die Weiterleitung zum nächsten Service getestet werden kann. Die Simplizität von Pipes and Filters ist ebenfalls ein Vorteil.

6.4.1.2 Nachteile

Als Nachteil kann die Performance Einbusse bei aufwändigen Konversationen der Daten betrachtet werden. Das Error Handling, falls ein Service fehlschlägt, ist aufwendig und kann eine lange Fehlersuche verursachen.

6.4.1.3 Mitigation

Der Nachteil der Performance Einbussen kann mittels AWS gelöst werden, in dem mehr Ressourcen für die einzelnen Services gegeben werden. Das Error Handling der einzelnen AWS Services ist sehr ausgeprägt entwickelt. Beim Debugging der Lösung muss der Grundsatz immer eingehalten werden, dass Fehler zu einer Überprüfung aller Servicekonfiguration führen.

6.4.2 MVC

Applikationen werden mit dem Architektur Pattern MVC (Model, View, Controller) entwickelt.

Das Model ist für die Daten und für die Datenaufbereitung zuständig. Der Controller besorgt sich die aktuellen Daten beim Model und leitet diese zur View. Die Darstellung und jegliche Benutzerinteraktionen erfolgen in der View. Nachfolgend werden die Benutzerinteraktionen an den Controller weitergeleitet. Der Controller erledigt die Anfrage und leitet die verlangten Daten wieder zur View.

Im Pattern besteht die Möglichkeit, dass das Model direkt seine aktualisierten Daten an die View übergibt.

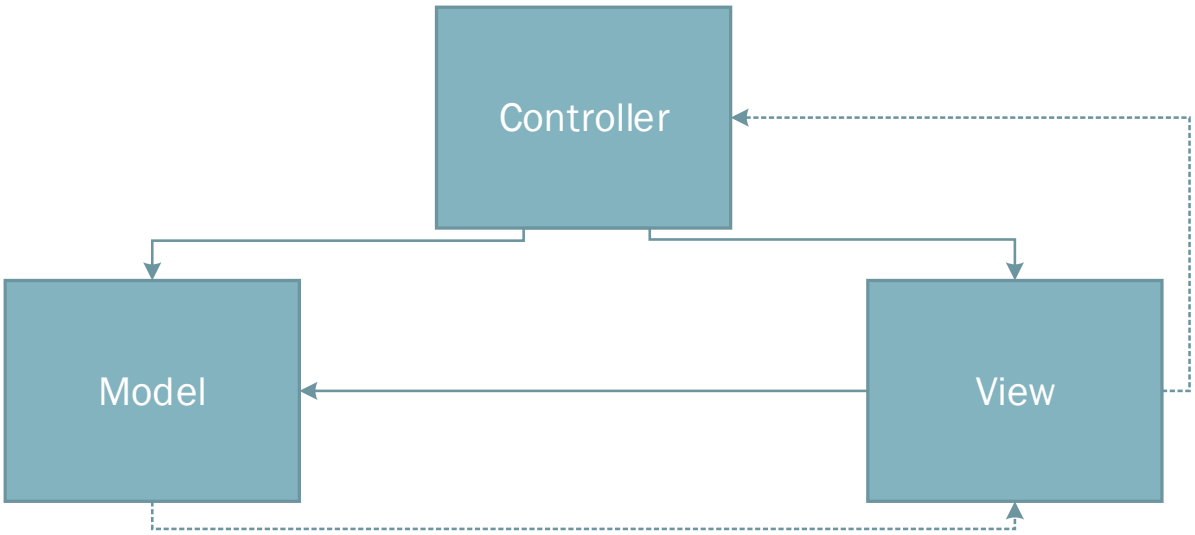


Abbildung 6 - MVC Pattern

6.4.2.1 Vorteile

Ein Vorteil besteht in der Austauschbarkeit der View. Das User Interface kann so leicht für verschiedene Zwecke angepasst werden. Zusätzlich ist es möglich, mehrere Views für das gleiche Model zu verwenden. Übersichtlichkeit und die Trennung von Programmlogik sind weitere wichtige Vorteile von MVC.

6.4.2.2 Nachteile

Bei grösserer Komplexität kann das Pattern an die Grenze der Übersichtlichkeit stossen. Ebenfalls ein Nachteil besteht in der starken Verknüpfung zwischen View und Controller.

6.4.2.3 Mitigation

In diesem Projekt sind die Applikationen von geringer Komplexität und Grösse. Der Nachteil der grösseren Komplexität kann so gelindert werden. Die Kopplung von View und Controller ist schwierig zu beheben, da diese ein essentieller Bestandteil des MVC Patterns ist. Bei der Entwicklung muss darauf geachtet werden, dass die Austauschbarkeit der View Priorität hat. Zusätzlich wird bei der Entwicklung ein Augenmerk auf möglichst simple Controller Funktionssignaturen gelegt.

6.5 Lösungskonzepte Komponente

Einzelne Komponente werden einer genaueren Analyse der Lösungskonzepte unterzogen.

6.5.1 Nanobox

Auf der Nanobox soll der MSIB Reader installiert werden. Das hat den Vorteil, dass der Prozess der Installierung und Konfigurierung des MSIB Lesers genauer analysiert wird. Ein anderer BACnet Reader wäre jener aus der Studienarbeit: "Integration von Siemens Sensoren mit Amazon Web Services". Dieser ist jedoch von seinen Möglichkeiten her weniger ausgebaut als MSIB. Besonders die Funktionalität, dass Daten von mehreren Sensoren effizient ausgelesen und weitergegeben werden, ist für diese Arbeit wichtig. Aus diesem Grund wird MSIB verwendet.

6.5.2 Schemalizer

Der Schemalizer wird auf einer bereits erstellten Software der Siemens "api-usage-mvp" aufgebaut. Das hat den Vorteil, dass wichtige Elemente wie die Verlinkung der Visualisierungen (bei Siemens noch Kibana) realisiert sind. Die Basis wird allenfalls soweit umgebaut gemäss dem MVC Prinzip. Dadurch erhält der Schemalizer grössere Übersicht und bessere Wartbarkeit.

"api-usage-mvp" ist eine Node.JS Applikation, die Express.JS verwendet. Die beiden Technologien sind im regen Gebrauch und für den geplanten Scope vom Schemalizer und Duolizer optimal, da nur ein geringer Overhead generiert wird. Die Skalierung der Applikation und dessen Abhängigkeiten sind ebenfalls durch Node.Js gewährleistet. Andere Alternativen wären zum Beispiel Node.JS im Zusammenhang mit Vue.js oder React.js. In diesem Szenario müsste die gesamte Codebasis der bestehenden Software "api-usage-mvp" angepasst werden. Die Vorteile einer solchen Kombination werden jedoch dem grösseren Aufwand nicht gerecht.

Die Möglichkeiten von SVG-Objekten wurden analysiert um die Applikation interaktiv zu gestalten. Alternativen wären normale .png Bilder. Diese können jedoch nicht so einfach mit JavaScript manipuliert werden.

Kriterium	**Begründung**
Interaktivität	Der Benutzer soll die gewünschten Daten anzeigen lassen und schnell begreifen was auf der Webseite passiert.
Skalierbarkeit	Weitere Sensoren oder Auswechslung der Visualisierungen sollen mühelos erfolgen.
Wiederverwendbarkeit	Gewisse Elemente sollen für weitere Applikationen verfügbar sein bzw. verwendet werden.

Tabelle 8 - Kriterien Schemalizer

Jedes Kriterium spricht für das Verwenden von SVG-Objekten, deshalb werden SVG-Objekten benutzt.

6.5.3 Duolizer

Es wird geplant, die Architektur und Funktionsweise vom Schemalizer äquivalent auf den Duolizer umzusetzen. Des Weiteren ist eine Etagenübersicht geplant, wo beliebige Zimmer angeklickt und angezeigt werden. Dadurch kann sich der Benutzer schnell zurechtfinden. SVG-Objekte sind ebenfalls für diese Verwendung perfekt geeignet, da Zimmer direkt mit der jeweiligen Zimmernummer im SVG gespeichert und angezeigt werden können.

6.5.4 Treolizer

Für den Treolizer sind die beiden Computerspiel Engines "Unity" und "Unreal Engine" analysiert worden. Es wird eine Computerspiel Engine verwendet, da diese sehr performant 3D Modelle darstellen kann. Die Kriterien für die Engines sind wie folgend definiert:

Kriterium	**Begründung**
Embedded Webbrowser	Ein Webbrowser wird benötigt um die Visualisierungen seitens Grafana oder Kibana anzuzeigen.
Programmiersprache	Je nach Programmiersprachen sind allenfalls schon Kenntnisse vorhanden, die sich positiv auf die Entwicklung des Treolizers auswirken.
Build Destinationen	Treolizer soll in einem ersten Schritt als Windows PC Anwendung entwickelt werden. Weitere Plattformen sind im Falle von restlichen Ressourcen zu analysieren.
Import Dateien im Format OBJ und FBX	Die BIM Modelle vom Siemens HQ in Zug können als OBJ oder als FBX exportiert werden. Deshalb muss die Engine solche Modelle importieren können.
Xbox Controller Support	Die Navigation im dreidimensionalen Raum soll mit einem Xbox Controller erfolgen.

Tabelle 9 - Kriterien Treolizer

Die beiden Optionen werden gemäss den Kriterien verglichen und eine Entscheidung betreffend der zu benutzenden Engine gefällt.

Kriterium	**Unity**	**Unreal Engine**
Embedded Webbrowser	Ja (Plug-In)	Vorhanden
Programmiersprache	C#, JavaScript	Blueprint, C++
Build Destinationen	Windows PC, Mac, Linux, iOS, WebGL, Android	Windows PC, Linux, HTML5 / JavaScript, Android, iOS
Import Dateien im Format OBJ und FBX	Ja	Ja
Xbox Controller Support	Ja	Ja

Tabelle 10 - Vergleich Unity / Unreal Engine

Ein gewichtiger Punkt für Unity sind die verwendeten Programmiersprachen. Das Wissen und die Erfahrungen mit den beiden Programmiersprachen C# und JavaScript sind im Projektteam grösser als bei Blueprint und C++.Es wird ein FBX Export aus Revit bezogen, da ein FBX Export ohne Plug-In leicht in Revit generiert werden kann.

Die Benutzeroberfläche und die Handhabung haben sich ebenfalls auf die Produktentscheidung ausgewirkt. Unity ist dort eher spartanisch jedoch übersichtlicher als jene von der Unreal Engine.

Neben der Programmiersprache und der Benutzeroberfläche ist ein weiterer Faktor die bereits vorhandene Erfahrung in Unity. Aus diesen Gründen wird Unity der Unreal Engine

vorgezogen. Sind nach der Umsetzung mit Unity Projektressourcen noch vorhanden, kann eine Umsetzung in der Unreal Engine analysiert werden.

6.6 Entscheid des Lösungskonzepts (Visualisierung)

Das Hauptziel der Visualisierung ist eine möglichst vielfältige und saubere Darstellung der Datensätze. Präsenzdaten sollten nicht gleich wie Temperaturdaten dargestellt werden, um auf den ersten Blick einen klaren Unterschied der Sensoren darzustellen. Ein weiteres Ziel ist die Wiederverwendbarkeit in allen drei Applikationen. Folgende vier Varianten wurden in Betracht gezogen und genauer analysiert:

Variante	**Vorteile**	**Nachteile**
API Daten mit JavaScript / HTML / CSS manipulieren	Eigene Ideen können umgesetzt werden. Keine externen Abhängigkeiten werden benötigt.	Umfangreicher, da von Grund auf alles neu zu implementieren ist.
Grafana	Schnittstelle zu Elasticsearch ist vorhanden. Mehrere Sensoren können gleichzeitig dargestellt werden. Wiederverwendbarkeit in allen drei Applikationen.	Grafana muss auf einem Server oder Lokal Installiert werden, um die Live-Daten korrekt darzustellen.
Kibana	Einfachste und schnellste Variante, da Siemens bereits Kibana Visualisier-ungen implementiert hat. Kenntnisse aus Studienarbeit. Wiederverwendbarkeit in allen drei Applikationen.	Keine geeignete Möglichkeit zur Darstellung von mehreren Sensoren auf einen Blick.
Chart.js	Schnell einsetzbar durch eine kleine Lernkurve. Wiederverwendbarkeit in zwei Applikationen.	Keine Möglichkeit für eine Verwendung im Treolizer.

Tabelle 11 - Visualisierungsmöglichkeiten

Die Entscheidungskriterien waren hauptsächlich eine übersichtliche Darstellung der Daten und die Wiederverwendbarkeit in den Applikationen zu erhalten. Die Zuweisung, wo welche Darstellung verwendet wird, ist von Applikation zu Applikation unabhängig auszuwählen.

6.6.1 Schemalizer & Duolizer

Die beiden Applikationen sind webbasierend und arbeiten mit JavaScript, HTML und CSS. Die Nähe zu den Varianten “API Daten” manipulieren und Chart.js sprechen stark für deren Verwendung. Die Visualisierungen müssen nur einmal implementiert werden und können anschliessend für beide Applikationen übernommen werden. Diese Varianten wurden verwendet, um Trenddaten (z.B. Aktivität letzter Stunde) oder Live Daten (Zimmer Frei oder Besetzt) darzustellen.

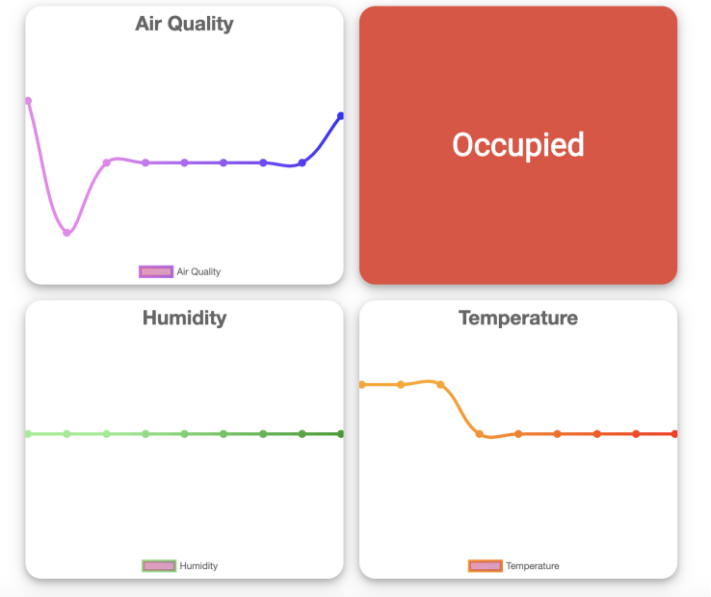


Abbildung 7 - Trenddaten

Um längere Zeitperioden und Dashboards darzustellen wurde Grafana statt Kibana gewählt. Die unterschiedliche Darstellung der Daten und die Möglichkeit Dashboards zu erstellen sprachen für Grafana.

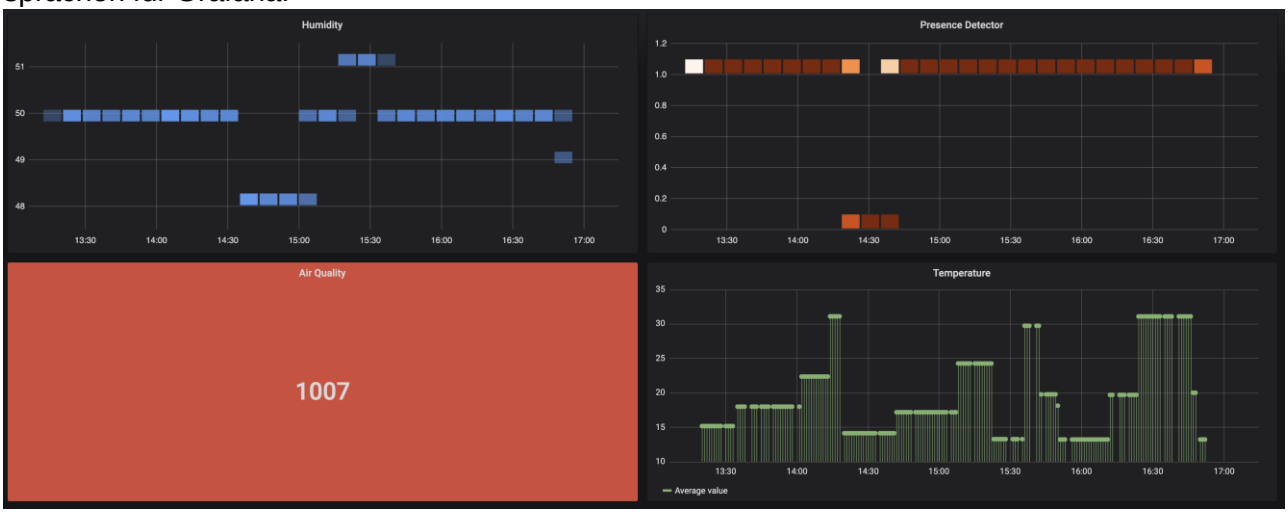


Abbildung 8 – Grafana

6.6.2 Treolizer

Der Treolizer manipuliert ebenfalls die Daten direkt aus der API. Somit können die Daten beliebig dargestellt werden. Die aktuellen Daten der Sensoren werden beim Starten der Applikation geladen und können wie Button neu geladen werden. Die Darstellung erfolgt über Kacheln.

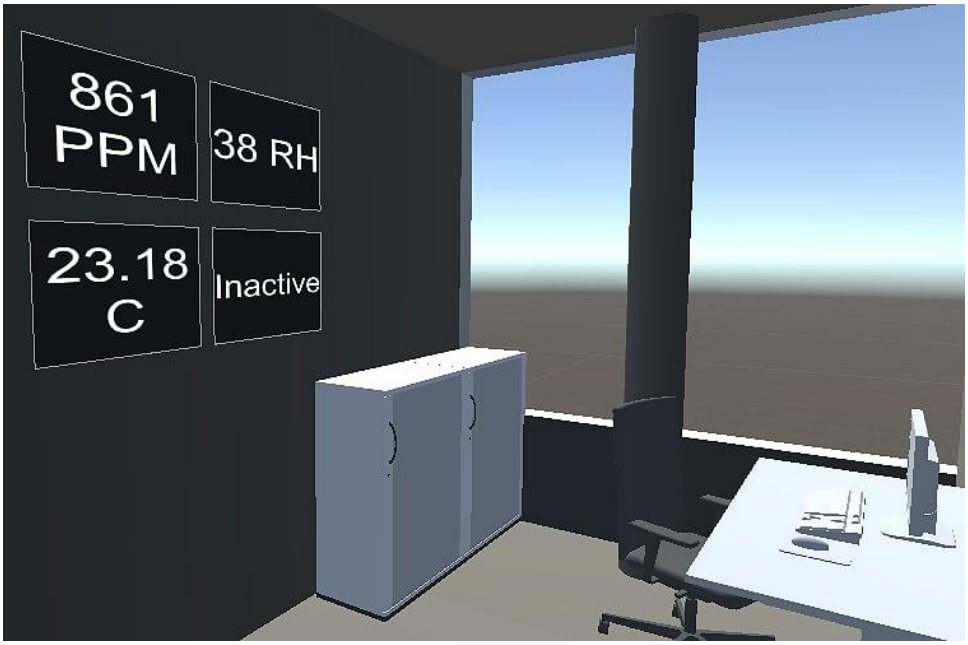


Abbildung 9 - Treolizer

Auch beim Treolizer wurde Grafana verwendet, um längere Zeitperioden und Dashboards darzustellen. Dieselben Grafana Dashboards können bei allen drei Applikationen eingesetzt werden. Die Dashboards werden bei einer Kollision des dazugehörenden Sensors angezeigt.

6.7 Visualisierungen

6.7.1 Grafana

Grafana hat den grossen Vorteil, dass Dashboards ohne grossen zusätzlichen Aufwand erstellt und bearbeitet werden können. Sensoren mit multiplen Sensordaten wie zum Beispiel ein Touch Panel wird mittels eins Dashboards dargestellt.

Die Anbindung zu Elasticsearch erfolgt unkompliziert über die Datenquellenkonfiguration von Grafana.

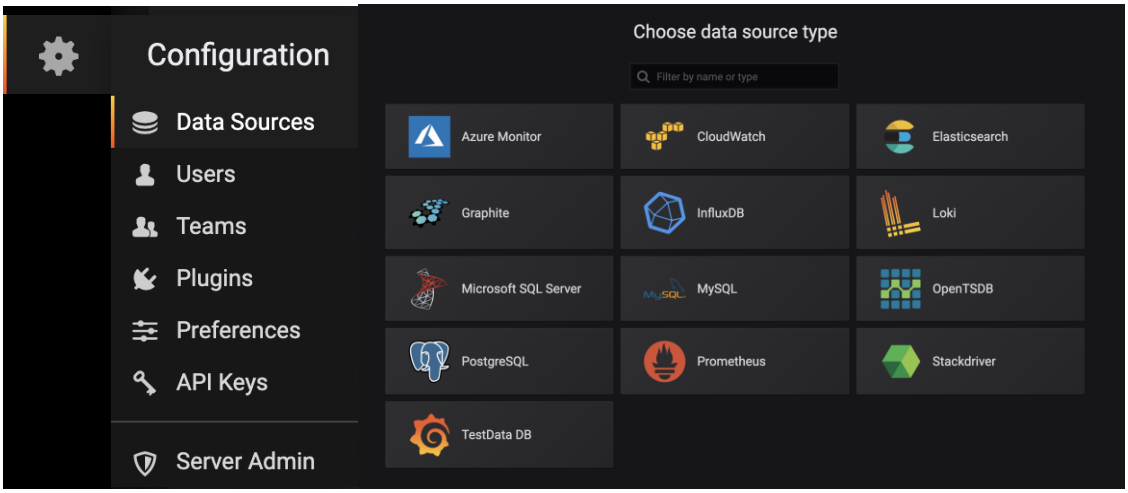


Abbildung 10 - Grafana Data Source

Eine genaue Anleitung wie die Konfiguration zu erledigen ist, kann im Anhang gefunden werden.

6.7.1.1 Heatmap

Eine Sonderheit von Grafana ist die Heatmap Ansicht. Im folgenden Bild sind Daten von einem Präsenzmelder dargestellt.

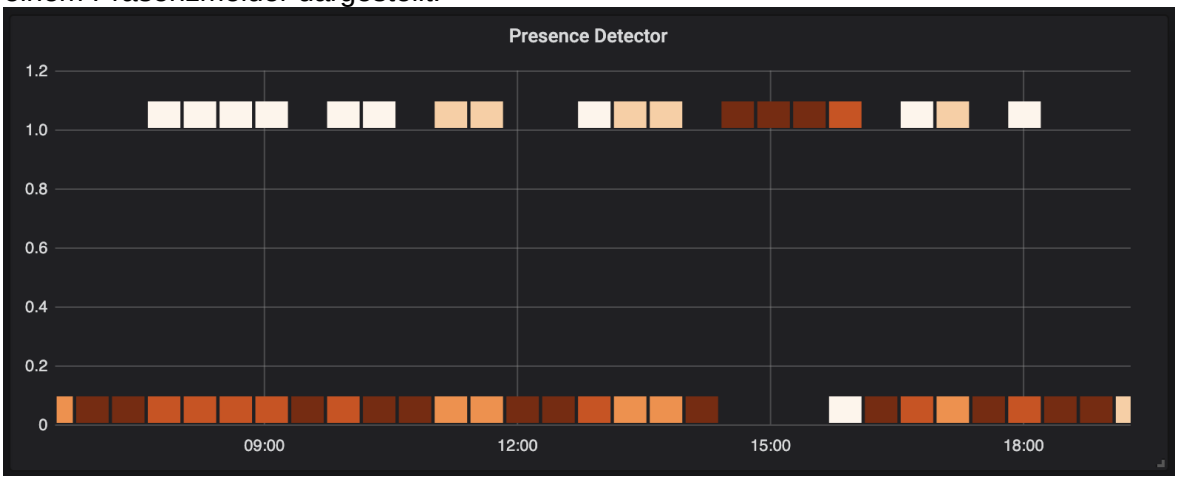


Abbildung 11 - Grafana Heatmap

Hier ist die Besonderheit, wie die einzelnen Hits dargestellt werden. Je dunkler die Farbe desto mehr Anschläge wurden in diesem Zeitraum erhalten. Gerade beim Präsenzmelder ist diese Ansicht interessant, da sie eine gute Übersicht bietet. Im oberen Bild (Daten vom 16.04.2019, Siemens Zug) können verschiedene Muster erkannt werden – Arbeitsbeginn ca. 07:00 Uhr bis 08:00 Uhr, Mittagspause um 12:00 Uhr und Feierabend um ca. 18:00 Uhr.

6.7.1.2 Singlestat

Weiterhin können auch einzelne Datenwerte für eine schnelle Übersicht dargestellt werden mit der “Singlestat” Visualisierung. Hier ein Beispiel für die Luftqualität im Zimmer.

Abbildung 12 - Singlestat (Luftqualität gut)

Je nach Wert wird auch die Farbe im Hintergrund angepasst. Die dargestellte Zahl muss nicht auf Anhieb verstanden bzw. interpretiert werden. Es wird schnell erkannt ob die Luftqualität gut, zufrieden oder schlecht ist.

Abbildung 13 - Singlestat (Luftqualität zufrieden)

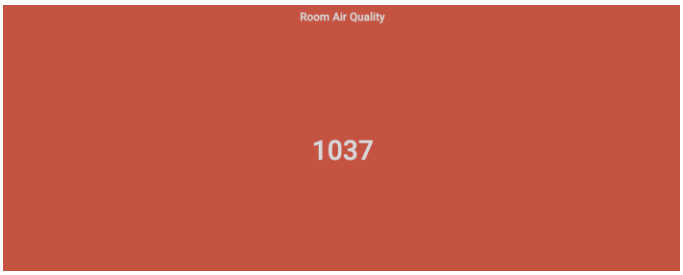


Abbildung 14 - Singlestat (Luftqualität schlecht)

6.7.1.3 Graph (Linie)

Änderungen über eine längere Zeitperiode können gut mit der Graph-Visualisierung dargestellt werden. Unten im Bild ist ein Beispiel der Temperatur über 24 Stunden zu sehen.

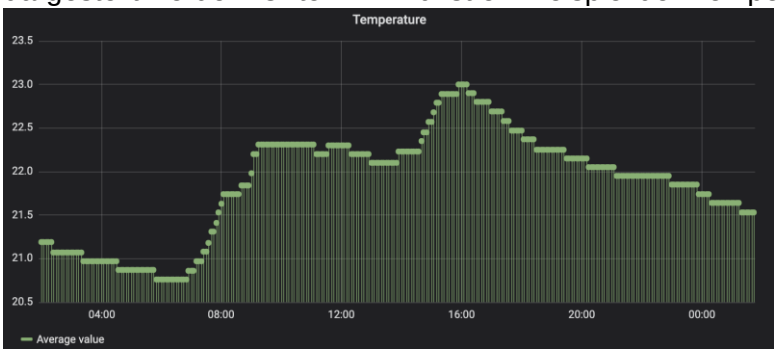


Abbildung 15 - Graph (Linie)

6.7.2 Demokoffer Sensorauswahl

Der Demokoffer besitzt eine grosse Anzahl von Sensoren, die Daten erzeugen. Für die Auswertung relevant sind die Sensoren in der folgenden Liste.

BACnet Id	Beschreibung	Datentyp	Sensor ID
AV:56	Relative humidity for room	Fliesskommazahl	R'RHVACCOO'RHUREL
AV:57	Room air quality	Fliesskommazahl	R'RHVACCOO'RAQUAL
AV:58	Room temperature	Fliesskommazahl	R'RHVACCOO'RTEMP
BV:0	Room presence detection	Boolean	R'RHVACCOO'RPSCDET

Tabelle 12 - Demokoffer Sensoren

6.7.3 Verknüpfung Sensor und Grafana Visualisierung

Die einzelnen Sensoren und welche Diagrammtypen zur Visualisierung verwenden werden, ist in der folgenden Tabelle beschrieben.

Sensor	Diagrammtyp	Begründung
Relative humidity for room	Heatmap	Hier wurde die Heatmap-Darstellung gewählt, da Werte und deren Häufigkeit in bestimmten Zeitabschnitten dargestellt werden können. Veränderungen werden schnell erkannt.
Room air quality	Singlestat	Die Veränderung der Luftqualität lässt sich am Besten in einem Singlestat ablesen. Kombiniert mit verschiedenen Hintergrundfarben, je nach Wert, lässt sich die Luftqualität schnell ablesen.
Room temperature	Graph (Linie)	Die Veränderungen an der Raumtemperatur ist am einfachsten durch ein Liniendiagramm auszulesen.
Room presence detection	Heatmap	Der Präsenzmelder sendet entweder eine von zwei Meldungen - aktiv oder inaktiv. Diese Werte sollen sich visuell unterscheiden. Deshalb wurde Heatmap verwendet.

Tabelle 13 - Verknüpfung Sensor und Grafana Visualisierung

7 Projekt Konstruktion

Im Kapitel Projekt Konstruktion wird die Umsetzung der Komponenten genauer beschrieben.

7.1 Schemalizer

Der Schemalizer soll die Daten der Sensoren vom Demokoffer schematisch darstellen. Dabei soll die Applikation interaktive Elemente beinhalten.

7.1.1 Ausgangslage

Die Ausgangslage ist eine bereits erstelle Node.js mit Express.js Software der Siemens. In dieser Software zeigt Kibana Visualisierungen zu den jeweiligen Sensoren an.

7.1.2 Technische Umsetzung

7.1.3 Schwierigkeiten

Die Interaktivität und visuelle Integrierung des Demokoffers ist eine Schwierigkeit dieser Anwendung. Die einzelnen Sensoren sollen anklickbar sein. SVG-Objekte lösen dieses Problem, da sie beliebig platziert werden können und auf gewisse Benutzerinteraktionen reagieren.

Eine weitere Schwierigkeit ist der Aufbau der Applikation. Ein Hauptziel des Schemalizers ist die Wiederverwendbarkeit der einzelnen Komponenten im Duolizer. Die klare Trennung der Visualisierungen löst dieses Problem.

7.1.4 Anpassungen

Durch das Verwenden von SVG Elementen ist die Applikation interaktiv. Solche SVG-Objekte lassen sich über die Webseite Editor Method [1] oder mit Inkscape erstellen. Um die Interaktivität zu sichern, werden die erstellten SVG's mittels XHR Requests in das bestehende Projekt geladen und anschliessend mit JavaScript manipuliert.

Weiterhin werden im Schemalizer Daten direkt aus der offenen API geholt und angezeigt. Zugriff auf die geschützte API wurde erst in den letzten Wochen der Bachelorarbeit gegeben, weshalb beim Schemalizer bis dahin auf die offene API zurückgegriffen werden musste. Um die Live-Daten der API zu holen werden wiederum XHR Requests benötigt. Die POST Requests werden wie nachfolgend beschrieben ausgeführt:

Request URL:

1. Base URL (search-nrm3-cf-elastic)
2. Index von Elasticsearch (AWS)
3. /_search

Hier ein Beispiel:

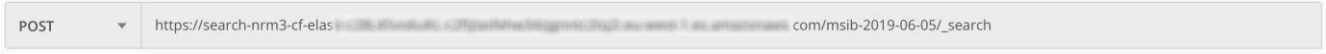


Abbildung 16 - Request URL

Body:

1. Sortieren
2. Nach SensorID filtern (Die SensorID kann aus dem generierten File von MSIB gelesen werden -> Siehe Kapitel MSIB on Nanobox.)
3. Zeitperiode definieren

Hier ein Beispiel:

```
let dataTemperatur = JSON.stringify({
  sort: [{ timestamp: { order: "desc" } }],
  query: {
    bool: {
      must: [
        { range: { timestamp: { lte: "now" } } },
        { match: { sensorId: "R'RHVACCOO'RTEMP" } }
      ]
    }
  }
});
```

Abbildung 17 - Request Body

Response:


```
{
    "took": 1,
    "timed_out": false,
    "_shards": {
        "total": 5,
        "successful": 5,
        "skipped": 0,
        "failed": 0
    },
    "hits": {
        "total": 210,
        "max_score": null,
        "hits": [
            {
                "_index": "msib-2019-06-05",
                "_type": "MsibType",
                "_id": "4959353811037301773154497708413881021581699455823065907 4.0",
                "_score": null,
                "_source": {
                    "building_id": "BUILDING-1539349692670-14142727",
                    "nodename": "AS_1",
                    "description": "",
                    "guid": "F125D520B6B498A1CF16EA33559AE29B",
                    "sensorId": "R'RHVACCOO'RTEMP",
                    "quality": "normal",
                    "timestamp": "2019-06-05T12:53:00+00:00",
                    "value": "26.760",
                    "insertTimestamp": "2019-06-05T12:53:21.472Z"
                },
                "sort": [
                    1559739180000
                ]
            },
            {
                "_index": "msib-2019-06-05",
                "_type": "MsibType",
                "_id": "4959353811037301773154497699459004797950256357705148006 6.0",
```


Abbildung 18 - Response

Der aktuelle Wert (value) wird über "*response.hits.hits[i]._source.value*" geholt.
Anschliessend werden die Daten mittels Chart.js oder durch einfaches JavaScript graphisch dargestellt.

Die Grafana Visualisierungen werden mit Hilfe von IFrames eingebettet. Dadurch können die Dashboards beliebig und schnell ausgetauscht werden. Updates oder Anpassungen werden ohne jegliche Änderungen im Code ersichtlich.

7.1.5 Testing und Debugging Möglichkeiten

Tests werden in erster Linie manuell durchgeführt. Die Begründung dafür ist die Flexibilität und Einfachheit der Applikation. Zu den manuellen Tests kommen noch automatisierte Tests mit Cypress dazu. Die Cypress Tests überprüft ob der Koffer Daten liefert und ob alle Visualisierungen korrekt zugeteilt und angezeigt werden. Das Debuggen erfolgt in den Chrome Developer Tools.

7.2 Duolizer

Hier werden Live-Daten der Siemens Gebäude in Zug angezeigt. Dazu kommt eine Etagenübersicht der einzelnen Zimmer.

7.2.1 Ausgangslage

Der Duolizer ist eine Erweiterung des Schemalizers. Gewisse Elemente werden kopiert und wiederverwendet.

7.2.2 Schwierigkeiten

Die Navigation in dieser Applikation war eine Herausforderung. Der Benutzer soll das gewünschte Zimmer mit Sicherheit finden und unmittelbar realisieren ob Sensordaten vorhanden sind. Eine Anzeige der Zimmernummer und sofortiges Einfärben des Zimmers löst dieses Problem und gibt dem Benutzer klare Informationen.

7.2.3 Anpassungen

Die Etagenübersicht wird bereits als SVG von der Siemens zu Verfügung gestellt. Dieses SVG-Objekt wird von der Datenquelle etwas schief und gespiegelt gesendet. Somit müssen zuerst diese Werte in Inkscape angepasst werden.

Damit möglichst einfach ersichtlich ist, ob ein Zimmer Daten aufweist und angeklickt werden kann, wird eine farbliche Unterscheidung verwendet.

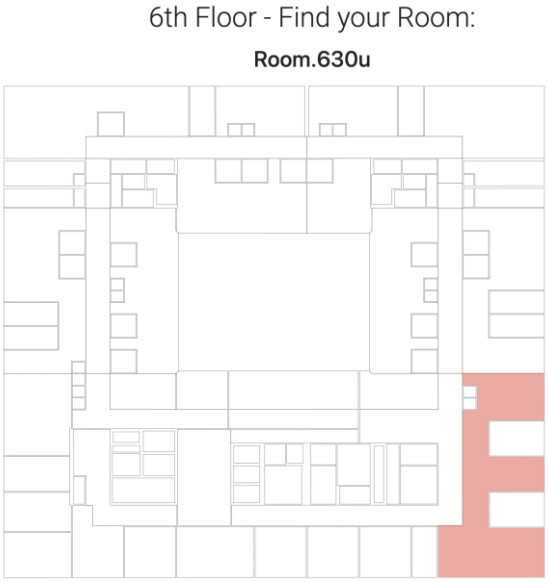


Abbildung 19 - Etagenübersicht (Sensoren nicht vorhanden)

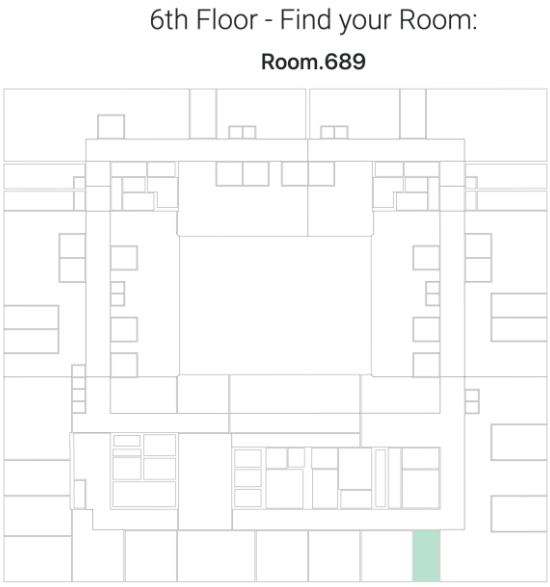


Abbildung 20 - Etagenübersicht (Sensoren vorhanden)

Die ID's der einzelnen Zimmer, die Visualisierungen enthalten, werden in einer Map gespeichert. Beim Bewegen des Mauszeigers über ein Zimmer wird die ID überprüft. Falls dieses Zimmer in der Map vorhanden ist, wird das Zimmer grün eingefärbt, sonst rot.

Weitere Anpassungen mussten gemacht werden, damit die ID's der einzelnen Zimmer richtig angezeigt werden. Ein Ausschnitt des SVG-Objekts direkt von der Siemens API:

```
<g
   id="Room.689"
   transform="matrix(0.20995935,-0.0143587,-0.01578539,-0.19098317,6173.1192,3397.5345)">
  <polygon
     points="8221.605,2729.953 8195.678,2731.893 8191.352,2674.105 8217.28,2672.164 "
     style="fill:#ffffff; stroke:#000000;stroke-width:0.2"
     id="polygon4315"/>
</g>
```

Abbildung 21 - SVG direkt von API

Hier ist das Problem, dass das Zimmer 689 zwei ID's aufweist. Einmal im <g> Tag und einmal im <polygon>. Die untere ID im <polygon> Tag ist für dieses Projekt irrelevant. Das Styling ist aber in diesem Tag definiert und überschreibt das Styling im <g> Tag. Damit die Zimmer mit den richtigen ID's angezeigt und korrekt eingefärbt werden, muss das Fill Property beim <polygon> gelöscht und beim <g> Tag eingefügt werden:

```
<g style="fill:#ffffff;"
   id="Room.689"
   transform="matrix(0.20995935,-0.0143587,-0.01578539,-0.19098317,6173.1192,3397.5345)">
  <polygon
     points="8221.605,2729.953 8195.678,2731.893 8191.352,2674.105 8217.28,2672.164 "
     style="stroke:#000000;stroke-width:0.2"
     id="polygon4315"/>
</g>
```

Abbildung 22 - SVG angepasst

Der Rest der Applikation ist aufgebaut wie der Schemalizer.

7.2.4 Testing und Debugging Möglichkeiten

Das Testing und Debugging wird analog dem Schemalizer durchgeführt.

7.3 Treolizer

Der Treolizer soll BIM Gebäudepläne in einer 3D Umgebung darstellen. Über einen Kontroller ist die Navigation in der Umgebung geplant.

7.3.1 Ausgangslage

Die Ausgangslage ist ein leeres 3D Projekt in Unity. Dort sind lediglich eine Kamera und eine Lichtquelle vorhanden. Eine Kamera bestimmt was der Benutzer schlussendlich sehen kann.

Unity besitzt einen Asset Store. Dort gibt es eine sehr grosse Anzahl von vorgefertigten Unity Komponenten, die leicht in das Projekt implementiert werden können. Komponente, die von einer dritten Partei verwendet wurden, sind in der folgenden Tabelle erläutert.

Name	Verwendung
FlyCam_Extended [2]	Die Steuerung des Spielers wird mittels diesem C# Skript umgesetzt.
SimpleUnityBrowser [3]	Die Anzeige von Webseiten für die einzelnen Bauteile wie zum Beispiel Kibana wird ein Browser benötigt.
Outline Effect [4]	Kollidiert der zentrierte Punkt mit einem anderen Objekt sollte dieses einen Rahmen erhalten. Die gewünschte Funktionalität wird mittels "Outline Effect" umgesetzt.
Modelle für Möblierung (STRABAG/ZÜBLIN)	Realistische Möbel für die genauere Darstellung des Zimmers 689 wird von der Firma STRABAG / ZÜBLIN bezogen. Das Zimmer sieht so realitätsnäher aus.
Suitcase 3D [5]	Für die Visualisierung des Demokoffers wird ein Koffermodell benötigt.

Tabelle 14 - Verwendete Elemente von Drittparteien

Eine weitere wichtige Ressource ist eine Übersicht [6], wie die einzelnen Bedienelemente eines Xbox One Kontrollers in Unity zugewiesen werden können.

Mit der Einbindung des FlyCam_Extended Skriptes ist die Kontroller Unterstützung umgesetzt worden.

7.3.2 Technische Umsetzung

7.3.3 Schwierigkeiten

Die Performance bei sehr grossen FBX Modellen war eine grosse Schwierigkeit, denn jegliche Inputs werden verzögert. Eine Lösung dazu bestand darin, die Sichtweite der Kamera einstellbar zu gestalten. Durch das Betätigen der Plus oder Minus Taste lässt sich die Sichtweite anpassen. Desto geringer die Sichtweite desto performanter läuft die Applikation. Hier ist eine Balance für das verwendete Gerät zu finden.

Eine weitere Schwierigkeit bestand darin, die Anforderungen des Browsers einzuhalten. Eine konkrete Anforderung war das Verwenden von der .NET Version 3.5. Dort sind Features wie zum Beispiel Thread Pools nicht vorhanden. Deshalb wurde das Feature der interaktiven Sensoren angepasst. Der Umbau hatte jedoch nur geringe Nebenwirkungen auf die Performance der Applikation.

7.3.4 Anpassungen

Um einem Model in Unity Leben einzuhauchen wird eine Kollision benötigt. Unity definiert eine Kollision als ein Zusammentreffen zwischen zwei Collider. Ein Collider kann unterschiedlichen Formen haben. Unity unterstützt folgende Arten von Collidern:

- Box Collider für Quader.
- Capsule Collider für Zylinder
- Sphere Collider für Kugeln

Eine Kollision zwischen zwei Collider in Unity kann über die bereits vorhandenen Funktionen (OnTriggerEnter, OnTriggerLeave) erkannt werden. Wichtig ist hier der Unterschied zu den Funktionen (OnCollisionEnter und OnCollisionLeave), da diese beide kollidierende Objekte benachrichtigt. Die Trigger-Funktionen benachrichtigen nur jenes Objekt, das die Option "isTrigger" gesetzt hat.

Es werden die Funktionen (OnTriggerEnter und OnTriggerLeave) verwendet um eine bessere Performance der Applikation zu erreichen.

Die Steuerung über den Kontroller wird anhand zweier Grafiken genauer beschrieben. Die erste Grafik beschreibt die Funktionen des Kontrollers im Standardmodus. Das ist der Modus in welchem sich der Benutzer frei im Model bewegen kann. Die zweite Grafik beschreibt die Möglichkeiten, wenn alle Installationen als Liste angezeigt werden.

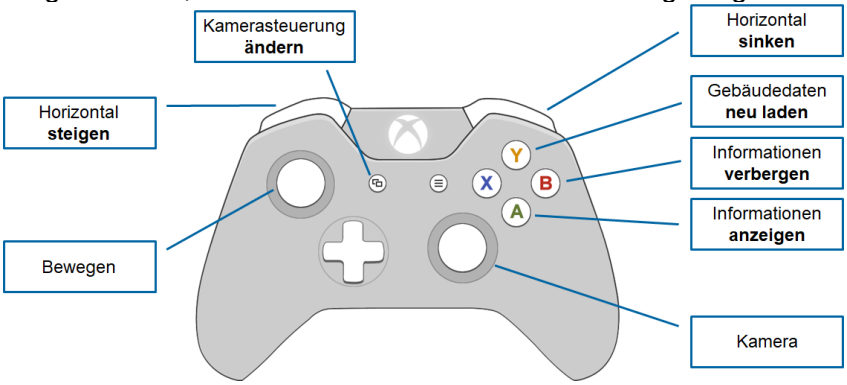


Abbildung 23 – Standardmodus [7]

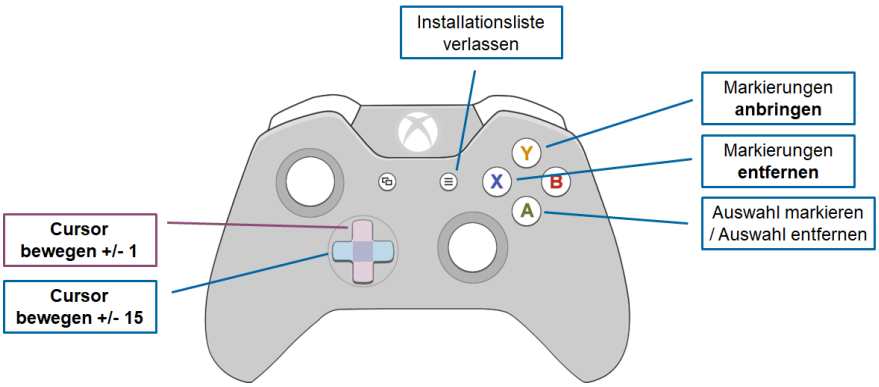


Abbildung 24 – Installationsliste [7]

7.3.5 Testing und Debugging Möglichkeiten

Für den Treolizer wird nach jedem Sprint ein Systemtest getätigt um sicherzustellen, dass keine unverhofften Nebenwirkungen auftreten. Der Systemtest wird von Hand durchgeführt, da sich der Input vom Kontroller und das Feeling der Applikation stark von Sprint zu Sprint verändern kann.

7.3.6 Prozessbeschreibung

In der Prozessbeschreibung wird aufgezeigt welche Änderungen vollzogen werden müssen, um den Treolizer mit unterschiedlichen Gebäudedaten zu füttern. Dazu wird der Vorgang anhand des Siemens HQ beschrieben.

1	**Exportierung von FBX File aus Revit Model.**
	Die Exportierung eines FBX Files benötigt eine 3D Ansicht im Revit Model. Das Siemens HQ ist in vier verschiedenen Revit Modellen beschreiben. Die unterschiedlichen Modelle werden mit F (Fassade), A (Architekture), H-SG (Heizung) und I (Installationen) abgekürzt. Revit benötigt unterschiedliche Zeiten bis ein FBX File exportiert ist. Die Zeiten werden unten genauer aufgelistet. • F (9min 27s) • A (1min 43s) • H-SG (0min 57s) • I (20min 54s)
2	**Importieren der FBX Files in Unity.**
	Der Import der FBX Files in Unity ist einfach gestaltet. Es kann entweder über die Option "Asset => Import New Asset…" oder das File kann direkt in die Filestruktur gezogen werden. Die Zeiten für den Import sind von File zu File unterschiedlich. Die einzelnen Zeiten werden unten aufgelistet. • F (3min 32s) • A (4min 02s) • H-SG (0min 12s) • I (2h 33min 45s)
3	**Allfälliges Debugging von fehlerhaften Installationen.**
	Es kann vorkommen, dass irgendwo bei der Exportierung oder gar im Revit Model Installationen nicht korrekt platziert sind. Der Fehler äussert sich in der Exception: "transform.local(Position, Rotation, Scale) assign attempt for 'Installation' is not valid. Input local local(Position, Rotation, Scale) is {NaN, NaN , NaN}". Als vorübergehende Lösung können die Installationen deaktiviert werden. Die Fehler sollen jedoch an das zuständige Team weitergeleitet werden.
4	**Konfiguration der importierten Files.**
	Jedes FBX File beinhaltet eine Kamera. Diese Kamera kann Unity verwirren. Daher wird in jedem Model die "3D View" deaktiviert und mit einem Tag "ModelCamera" versehen. An der Ausrichtung der importierten Modelle muss nichts verändert werden. Das C# Skript "StartUp" muss jedem Modell hinzugefügt werden, welches eine Interaktionsmöglichkeit erhalten sollte.

Tabelle 15 - Prozessbeschreibung Treolizer

7.4 MSIB on Nanobox

Bevor die Konfiguration von MSIB beginnen konnte, mussten neue SSH Zertifikate erstellt werden. Die Zertifikate müssen auf die Nanobox transferiert werden. Das genaue Verfahren ist in einer Anleitung im Anhang beschrieben. Zusätzlich muss die Firewall Konfiguration angepasst werden. Die Anpassung erfolgt über den Befehl *"iptables -I INPUT –p [[Protokoll]] -m [[Protokoll]] –dport [[Portnummer]] –j ACCEPT"*, dieser Befehl muss für die Optionen ([tcp;tcp;8080], [udp;udp;47808], [tcp;tcp;47808]) ausgeführt werden. Somit ist der Zugriff innerhalb des Netzwerkes auf die Nanobox ermöglicht. Der Port 47808 wird für den BACnet Datenverkehr verwendet. Der Port 8080 ermöglicht den Zugriff auf die Web-Konsole von MSIB.

Die Optionen des Firewall Befehls werden in der untenstehenden Tabelle beschrieben.

Option	Beschreibung
-I INPUT	Firewall Regel wird an der Kette "INPUT" angehängt.
-p	Pakete werden nur geprüft, wenn sie vom Typ [[Protokoll]] sind.
-m	Erweitert die Regel beim [[Protokoll]] mit der Option –dport.
-dport	Die Regel wird nur für diesen Port durchgeführt.
-j	Bestimmt welche Aktion ausgeführt werden soll, sobald ein Paket die Regel erfüllt, hier "ACCEPT".

Tabelle 16 - Firewall Optionen

Die Konfiguration von MSIB wird in zwei Teile geteilt. Erstens die Optionen, die beim Starten des Docker Containers von MSIB auf der Nanobox berücksichtig werden müssen. Zweitens die wichtigsten Einstellungen, die in der Konfiguration von MSIB erwartet werden.

7.4.1 MSIB Docker Container

Es gibt zwei Hürden, die bezwungen werden müssen um den MSIB Docker Container zu starten. Die beiden Docker Features Docker Network und Docker Volume werden folgend kurz erklärt und die ausgewählte Option genauer beschrieben.

Docker Network beschreibt die Möglichkeit wie sich ein Docker Container im lokalen Netzwerk verhält. Die Optionen für Docker Network sind:

Network Option	Beschreibung
none	Keine Netzwerkmöglichkeiten vorhanden innerhalb des Containers.
bridge	Erstellt eine Brücke um die angegebenen Ports mit jenen des Host-Gerätes zu verknüpfen.
host	Übernimmt den gesamten Network Stack vom Host-Gerät.
NETWORK	In Docker kann ein eigenes Network definiert werden. Mit dem erstellten Network ist es möglich, dass der Docker Container im Network läuft.

Tabelle 17 - Docker Network

In einem ersten Schritt wird die Option "host" ausgewählt. Die Option ermöglicht es dem Docker Container auf die Netzwerkschnittstelle des Hosts zuzugreifen.

Docker Volume beschreibt die Möglichkeit, wie ein Ordner im Filesystem des Hosts mit einem Ordner im Docker Container synchronisiert wird. MSIB erstellt beim Discovery Service CSV Dateien, die jegliche BACnet Devices im Netzwerk beinhalten. Die Dateien werden nachher gefiltert um nur jene Devices zu abonnieren, die auch benötigt werden.
Nebst den CSV Dateien müssen ebenfalls die Zertifikate für die Anbindung zu AWS im MSIB Docker Container hinterlegt werden. Dazu kann das gleiche Volume verwendet werden. Um die Zertifikate am korrekten Ort zu hinterlegen, kann auf die sich im Anhang befindende Anleitung zurückgegriffen werden.

Der verwendete Docker Run Befehl sieht folgendermassen aus:

docker run --network host -d --rm -v /tmp/msib_data:/msib/lib/msib-2.5.0/priv/msib_discovery msib

Docker Run Befehl Option	**Bedeutung**
--network host	Die Netzwerkeigenschaften vom Host sollen übernommen werden.
-d	Der Docker Container soll im Hintergrund ausgeführt werden.
--rm	Falls ein Docker Container zu MSIB besteht soll dieser entfernt werden bevor der Neue gestartet wird.
-v	Ermöglicht die Angabe eines Docker Volumes.

Tabelle 18 - Docker Run Befehl

Der Befehl muss jedoch nur einmal ausgeführt werden. Danach kann über den Befehl "*docker container ps –a*" alle vorhandenen Docker Container ausgegeben werden. Ist der gewünschte Container vorhanden, kann dieser mit dem Befehl "*docker start [[docker-ID]]*" gestartet werden.

7.4.2 MSIB Docker Container Konfiguration

Die Konfiguration vom MSIB Docker Container setzt voraus, dass der Discovery Service erfolgreich durchgeführt wurde und die generierten CSV Dateien von der Nanobox auf den lokalen Computer kopiert wurden. Die Optionen und die wichtigsten Punkte für das Befüllen für den AWS IOT Configuration Service sind:

MSIB Option	Beschreibung
AWS Client ID	Der Name des AWS IoT Things, das in AWS vorhanden ist.
AWS IoT Endpoint	Endpunkt für das Thing beschreibt die Region in welcher sich das Thing befindet.
AWS Certificate File Path	Beschreibt den Pfad für das öffentliche Zertifikat des Things.
AWS Key File Path	Beschreibt den Pfad für den privaten Schlüssel für das Zertifikat des Things.
AWS CACertFile File Path	Beschreibt den Pfad für das CA Zertifikat von Amazon.
AWS Update Interval	Beschreibt wie oft Daten an AWS gesendet werden.
AWS SSL Port	Beschreibt den Port für die AWS SSL Verbindung.
Proxy Enabled	Ist für den Fall einer Verwendung eines Proxies.
Import Type	Beschreibt welche BACnet Objekte verwendet werden sollten und wie diese Liste importiert wird. In diesem Fall wird die Option “Rule Mapping Tool Export File” verwendet.
Import File	Ist die Lokalität, in die das gefilterte CSV hochgeladen werden muss.
Source Service	Beschreibt den Service von welchem die BACnet Daten besorgt werden. Hier ist “BACnet Polling” auszuwählen.
Polling Interval	Beschreibt wie oft die Daten von den BACnet Objekten bezogen werden.
Separation	Beschreibt in welchen Zeitunterschieden die einzelnen BACnet Objekte angefragt werden.

Tabelle 19 - AWS Optionen bei MSIB

Sind diese Werte korrekt gesetzt funktioniert die Verbindung MSIB zu AWS.

8 Projekt Transition

8.1 Lösungs Review

8.1.1 Schemalizer

Die erste Applikation demonstriert, wie grosse Mengen von Daten übersichtlich dargestellt werden können. Die Interaktivität mit dem Koffer hilft auch dem Benutzer, die gewünschten Daten mit wenigen Klicks anzuzeigen. Die Erfahrung mit Grafana Dashboards und deren Wiederverwendbarkeit in allen drei Applikationen war eine grosse Hilfe für das gesamte Projekt.

Ein Verbesserungspunkt der Applikation wäre in naher Zukunft auf die geschlossene API umzustellen, da die offene API nicht unbegrenzt verfügbar ist.

8.1.2 Duolizer

Das Anwenden des MVC Patterns hat einen klaren Vorteil beim Skalieren des Projekts. Weitere Zimmer und deren Sensoren können mit geringem Aufwand hinzugefügt werden. Die Komponente des Schemalizers konnten mühelos übertragen und spezifisch für diese Applikation angepasst werden.

Momentan werden die Trenddaten (dargestellt mit Chart.js) nur beim Nachladen der Webseite aktualisiert. Eine automatisierte Lösung wäre hier etwas eleganter und ein Verbesserungspunkt hinsichtlich der Benutzerfreundlichkeit.

8.1.3 Treolizer

Die Anwendung Treolizer zeigt auf, dass sehr komplexe und aufwendige BIM Modelle performant dargestellt werden können. Gerade die Interaktivität mit den einzelnen Komponenten in den Modellen wird als grosser Pluspunkt betrachtet. Die Implementierung mit den Live Daten von Grafana oder direkt von der gesicherten API ist sehr gelungen. Die Navigation mit dem Xbox Controller ist nach einer kurzen Einarbeitung schnell und trotzdem präzise. Treolizer hat zusätzlich die Möglichkeit die Sensibilität des Inputs anzupassen.

Treolizer ist so aufgebaut, dass dieser leicht verbessert oder erweitert werden kann. Eine dem Projektteam sinnvolle Erweiterung wäre die Funktion der Kollisionsentdeckung zu überarbeiten. Es kann öfters passieren, dass mehrere Installationen markiert werden. Um nur eine auszuwählen, ist ein Dialog zu gestalten, der jegliche Objekte, die eine Kollision melden, anzeigt und der Benutzer ein gewünschtes Objekt auswählen kann.

Ein Verbesserungspunkt ist die Kommunikation über die geschützte API. Das Access Token soll automatisch von der API generiert werden. Im aktuellen Zustand muss das Access Token manuell in die Applikation kopiert werden. Ein Access Token ist nach Erfahrungen vom Projektteam nur 2h gültig.

8.1.4 Datenanbindung mittels MSIB

Die Datenanbindung vom Demokoffer über die Nanobox und MSIB ist nach einer ereignisreichen Konstruktionsphase erstellt worden. Es wurde aufgezeigt, dass es der Nanobox mit den sehr restriktiven Einstellungen möglich ist, MSIB so laufen zu lassen, dass Sensordaten innert ungefähr drei Minuten in Elasticsearch ersichtlich sind. Die Firewall Konfiguration muss jedoch noch verfeinert werden um einen möglichst kleinen Zugang zur Nanobox zu ermöglichen. Bei der Docker Container Konfiguration wird bis jetzt der Network Modus “host” verwendet. Die Verwendung dieser Option ist von Docker mit einem Hinweis versehen, dass es so dem Container möglich ist, die Netzwerk Konfiguration des Hosts zu verändern. Diese Tatsache muss bei einer Weiterverwendung von MSIB Docker bedacht werden.

8.1.5 Komponente

8.2 Projektablauf Review

Im Projektablauf wurde Agilität grossgeschrieben. Gerade bei Ausfällen von Personen die benötigt wurden um MSIB zu erhalten, zu konfigurieren und zu testen, fiel die Agilität sehr positiv auf. Die drei Applikationen konnten eine andere Datenquelle verwenden, um die Funktionalität zu gewährleisten.

Die endgültig verwendete Zeit beider Projektmitglieder betrug 684 h. Die im Vorhinein zugewiesene Anzahl Stunden ist mit 360 h pro Person vorgegeben. Die Diskrepanz der Zahlen lässt sich auf die verkürzte Einarbeitungszeit in der Phase Projekt Initialisierung zurückführen. Die verkürzte Einarbeitungszeit ist durch eine Ungewissheit des Projektrahmen entstanden.

Gerade das Ausweichen bei solchen Vorkommnissen ist ein Zeichen von Agilität im Projekt. Nebst der Agilität ist einer der Grundpfeiler des Projektes eine enge Zusammenarbeit mit Siemens. Eine wöchentliche Absprache über Stand und weiteres Vorgehen des Sprints war sehr wertvoll für beide Seiten. Das Projektteam genoss sehr grosses Vertrauen seitens Siemens und dem HSR Experten. Das Vertrauen manifestierte sich in die Ausrichtung und den Fokus der einzelnen Anwendungen und deren Funktionalitäten. Diese Freiheit wurde vom Projektteam geschätzt, und die fertigen Produkte können als gelungen bezeichnet werden.

9 Quellenverzeichnis

[1] E. Method, «Editor Method,» [Online]. Available: https://editor.method.ac/.

[2] Unity, «Wiki Unity3D,» 18 August 2012. [Online]. Available: https://wiki.unity3d.com/index.php/FlyCam_Extended.

[3] V. Chashin, «Simple Unity Browser,» 28 Februar 2017. [Online]. Available: https://bitbucket.org/vitaly_chashin/simpleunitybrowser/src/default/.

[4] Cakeslice, «Unity Asset Store,» [Online]. Available: https://assetstore.unity.com/packages/vfx/shaders/fullscreen-camera-effects/outline-effect-78608.

[5] LazyCreativity, «Turbo Squid - Suitcase 3D,» 10 Juni 2017. [Online]. Available: https://www.turbosquid.com/3d-models/suitcase-3d-1166896.

[6] tag104367, «Xbox one controller mapping [SOLVED],» 07 May 2017. [Online]. Available: https://answers.unity.com/questions/1350081/xbox-one-controller-mapping-solved.html.

[7] A. Debenham, «Game Console Browsers,» 27 Februar 2016. [Online]. Available: http://console.maban.co.uk/device/xboxone/.

10 Abkürzungsverzeichnis

Begriff	Beschreibung
API	Application Programming Interface
AWS	Amazon Web Services
BACnet	Building Automation and Control Networks
BIM	Building Information Modeling
CA	Certificate authority
CSS	Cascading Stye Sheets
CSV	Comma-separated Values
FBX	Filmbox
HTML	Hypertext Markup Language
HUD	Heads-up-Display
HQ	Headquarters
IoT	Internet of Things
JSON	JavaScript Object Notation
MSIB	Siemens Middleware Service for Integrated Buildings
MVC	Model View Controller
MVP	Minimum Viable Product
OBJ	OBJ (Dateiformat für 3D Modelle)
REST	Representational State Transfer
SCP	Secure Copy
SSH	Secure Shell
SSL	Secure Sockets Layer
SVG	Scalable Vector Graphics
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
XHR	XMLHttpRequest

Tabelle 20 – Abkürzungsverzeichnis

11 Abbildungsverzeichnis

Alle Abbildungen ausser die Abbildungen 23 und 24, sind von Sandro Blatter und Benjamin Daniels erstellt worden.

12 Tabellenverzeichnis

13 Anhänge

13.1 Inhalt Archiv

- Software
  - Schemalizer
  - Duolizer
  - Treolizer
- Dokumentation
  - Bericht.pdf
  - Bericht_mit_Anhang.pdf

13.2 Dokumente des Projekts

- Persönliches_Fazit_Sandro_Blatter.pdf
- Persönliches_Fazit_Benjamin_Daniels.pdf
- Projektplan.pdf
- Poster.pdf
- Journal.pdf
- Projektantrag.pdf
- Anleitung_Anbindung_MSIB_an_AWS.pdf
- Anleitung_Generate_SSH_Access_to_Nanobox.pdf
- Anleitung_Inbetriebnahme_Grafana.pdf
- Eigenständigkeitserklärung.pdf
- Einverständniserklärung_Publikation_auf_eprints.pdf
- Vereinbarungen_Urheber- und_Nutzungsrechte.pdf