|  |  |
| --- | --- |
| AVEC_LOGO_R_RGB.pdf | |
| IPA-Bericht 2020 | |
|  | |
| BBB Gebäude Management Pilot Versuch | |
|  | |
|  | |
| Nils Egger  Lernender | |
| 23.03.2020 |  |
|  | |

Änderungsindex

Tabelle 1 Änderungsindex

|  |  |  |  |
| --- | --- | --- | --- |
| Revision | Beschreibung | Erstellt | Datum |
| 0.1 | Erstellung des Dokumentes und dokumentieren der Phasen Informieren, Planen, Entscheiden und Realisieren | Nils Egger | 23.03.2020 |
| 0.2 | Korrektur des aktuellen Standes | Sibylle Rub und Mark Grabner | 02.04 |
| 0.3 | Fortführung Dokumentation der Phase Realisieren, Kontrollieren und Auswerten | Nils Egger | 03.04 |
| 0.4 | Finaliesierung | Nils Egger | 08.04 |
| 1.0 | Abgabe IPA-Bericht | Nils Egger | 08.04 |

Inhaltsverzeichnis

[1 Aufgabenstellung 10](#_Toc37241352)

[1.1 Titel der Arbeit 10](#_Toc37241353)

[1.2 Ausgangslage 10](#_Toc37241354)

[1.3 Detaillierte Aufgabenstellung 10](#_Toc37241355)

[1.3.1 Beschreibung 10](#_Toc37241356)

[1.3.2 Generelle Eigenschaften 10](#_Toc37241357)

[1.3.3 Anforderungen an die Webapplikation 10](#_Toc37241358)

[1.4 Mittel und Methoden 12](#_Toc37241359)

[1.5 Vorkenntnisse 12](#_Toc37241360)

[1.6 Vorarbeiten 12](#_Toc37241361)

[1.7 Neue Lerninhalte 12](#_Toc37241362)

[1.8 Arbeiten in den letzten 6 Monaten 12](#_Toc37241363)

[1.9 Individuelle Bewertungskriterien 12](#_Toc37241364)

[1.9.1 Responsive Design 12](#_Toc37241365)

[1.9.2 Browserkompatibilität 13](#_Toc37241366)

[1.9.3 Einhaltung Corporate Design 13](#_Toc37241367)

[1.9.4 Codingstyle – Dokumentation 14](#_Toc37241368)

[1.9.5 Validierung Eingaben 14](#_Toc37241369)

[1.9.6 Anzeige der Luftqualitäts-Messwerte 14](#_Toc37241370)

[1.9.7 Grafische Darstellung der Messwerte 15](#_Toc37241371)

[2 Projektaufbauorganisation 15](#_Toc37241372)

[2.1 Methode 15](#_Toc37241373)

[2.2 Auftraggeber 15](#_Toc37241374)

[2.3 Beteiligte Personen 15](#_Toc37241375)

[2.4 Durchführungsort 16](#_Toc37241376)

[3 Deklaration der Vorkenntnisse 16](#_Toc37241377)

[4 Deklaration der Vorarbeiten 16](#_Toc37241378)

[5 Deklaration der benützten Firmenstandards 17](#_Toc37241379)

[6 Zeitplan 17](#_Toc37241380)

[6.1 Bemerkungen 17](#_Toc37241381)

[6.2 Legende 17](#_Toc37241382)

[6.3 Meilensteine 17](#_Toc37241383)

[7 Arbeitsjournal 19](#_Toc37241384)

[7.1 Arbeitsjournal 23.03.2020 19](#_Toc37241385)

[7.1.1 Ausgeführte Arbeiten 19](#_Toc37241386)

[7.1.2 Erreichte Ziele / Erfolge 19](#_Toc37241387)

[7.1.3 Aufgetretene Probleme / Misserfolge 19](#_Toc37241388)

[7.1.4 Beanspruchte Hilfestellungen 19](#_Toc37241389)

[7.1.5 Pendenzenliste 19](#_Toc37241390)

[7.1.6 Reflexion 19](#_Toc37241391)

[7.1.7 Vergleich mit dem Zeitplan 20](#_Toc37241392)

[7.2 Arbeitsjournal 24.03.2020 20](#_Toc37241393)

[7.2.1 Ausgeführte Arbeiten 20](#_Toc37241394)

[7.2.2 Erreichte Ziele / Erfolge 20](#_Toc37241395)

[7.2.3 Aufgetretene Probleme / Misserfolge 20](#_Toc37241396)

[7.2.4 Beanspruchte Hilfestellungen 20](#_Toc37241397)

[7.2.5 Pendenzenliste 20](#_Toc37241398)

[7.2.6 Reflexion 21](#_Toc37241399)

[7.2.7 Vergleich mit dem Zeitplan 21](#_Toc37241400)

[7.3 Arbeitsjournal 25.03.2020 21](#_Toc37241401)

[7.3.1 Ausgeführte Arbeiten 21](#_Toc37241402)

[7.3.2 Erreichte Ziele / Erfolge 21](#_Toc37241403)

[7.3.3 Aufgetretene Probleme / Misserfolge 21](#_Toc37241404)

[7.3.4 Beanspruchte Hilfestellungen 21](#_Toc37241405)

[7.3.5 Pendenzenliste 21](#_Toc37241406)

[7.3.6 Reflexion 22](#_Toc37241407)

[7.3.7 Vergleich mit dem Zeitplan 22](#_Toc37241408)

[7.4 Arbeitsjournal 27.03.2020 22](#_Toc37241409)

[7.4.1 Ausgeführte Arbeiten 22](#_Toc37241410)

[7.4.2 Erreichte Ziele / Erfolge 22](#_Toc37241411)

[7.4.3 Aufgetretene Probleme / Misserfolge 22](#_Toc37241412)

[7.4.4 Beanspruchte Hilfestellungen 22](#_Toc37241413)

[7.4.5 Pendenzenliste 22](#_Toc37241414)

[7.4.6 Reflexion 22](#_Toc37241415)

[7.4.7 Vergleich mit dem Zeitplan 23](#_Toc37241416)

[7.5 Arbeitsjournal 30.03.2020 23](#_Toc37241417)

[7.5.1 Ausgeführte Arbeiten 23](#_Toc37241418)

[7.5.2 Erreichte Ziele / Erfolge 23](#_Toc37241419)

[7.5.3 Aufgetretene Probleme / Misserfolge 23](#_Toc37241420)

[7.5.4 Beanspruchte Hilfestellungen 23](#_Toc37241421)

[7.5.5 Pendenzenliste 23](#_Toc37241422)

[7.5.6 Reflexion 23](#_Toc37241423)

[7.5.7 Vergleich mit dem Zeitplan 23](#_Toc37241424)

[7.6 Arbeitsjournal 31.03.2020 24](#_Toc37241425)

[7.6.1 Ausgeführte Arbeiten 24](#_Toc37241426)

[7.6.2 Erreichte Ziele / Erfolge 24](#_Toc37241427)

[7.6.3 Aufgetretene Probleme / Misserfolge 24](#_Toc37241428)

[7.6.4 Beanspruchte Hilfestellungen 24](#_Toc37241429)

[7.6.5 Pendenzenliste 24](#_Toc37241430)

[7.6.6 Reflexion 24](#_Toc37241431)

[7.6.7 Vergleich mit dem Zeitplan 24](#_Toc37241432)

[7.7 Arbeitsjournal 01.04.2020 24](#_Toc37241433)

[7.7.1 Ausgeführte Arbeiten 24](#_Toc37241434)

[7.7.2 Erreichte Ziele / Erfolge 25](#_Toc37241435)

[7.7.3 Aufgetretene Probleme / Misserfolge 25](#_Toc37241436)

[7.7.4 Beanspruchte Hilfestellungen 25](#_Toc37241437)

[7.7.5 Pendenzenliste 25](#_Toc37241438)

[7.7.6 Reflexion 25](#_Toc37241439)

[7.7.7 Vergleich mit dem Zeitplan 25](#_Toc37241440)

[7.8 Arbeitsjournal 03.04.2020 25](#_Toc37241441)

[7.8.1 Ausgeführte Arbeiten 25](#_Toc37241442)

[7.8.2 Erreichte Ziele / Erfolge 25](#_Toc37241443)

[7.8.3 Aufgetretene Probleme / Misserfolge 26](#_Toc37241444)

[7.8.4 Beanspruchte Hilfestellungen 26](#_Toc37241445)

[7.8.5 Pendenzenliste 26](#_Toc37241446)

[7.8.6 Reflexion 26](#_Toc37241447)

[7.8.7 Vergleich mit dem Zeitplan 26](#_Toc37241448)

[7.9 Arbeitsjournal 06.04.2020 26](#_Toc37241449)

[7.9.1 Ausgeführte Arbeiten 26](#_Toc37241450)

[7.9.2 Erreichte Ziele / Erfolge 26](#_Toc37241451)

[7.9.3 Aufgetretene Probleme / Misserfolge 26](#_Toc37241452)

[7.9.4 Beanspruchte Hilfestellungen 27](#_Toc37241453)

[7.9.5 Pendenzenliste 27](#_Toc37241454)

[7.9.6 Reflexion 27](#_Toc37241455)

[7.9.7 Vergleich mit dem Zeitplan 27](#_Toc37241456)

[7.10 Arbeitsjournal 07.04.2020 27](#_Toc37241457)

[7.10.1 Ausgeführte Arbeiten 27](#_Toc37241458)

[7.10.2 Erreichte Ziele / Erfolge 27](#_Toc37241459)

[7.10.3 Aufgetretene Probleme / Misserfolge 27](#_Toc37241460)

[7.10.4 Beanspruchte Hilfestellungen 27](#_Toc37241461)

[7.10.5 Pendenzenliste 27](#_Toc37241462)

[7.10.6 Reflexion 27](#_Toc37241463)

[7.10.7 Vergleich mit dem Zeitplan 28](#_Toc37241464)

[7.11 Arbeitsjournal 08.04.2020 28](#_Toc37241465)

[7.11.1 Ausgeführte Arbeiten 28](#_Toc37241466)

[7.11.2 Erreichte Ziele / Erfolge 28](#_Toc37241467)

[7.11.3 Aufgetretene Probleme / Misserfolge 28](#_Toc37241468)

[7.11.4 Beanspruchte Hilfestellungen 28](#_Toc37241469)

[7.11.5 Pendenzenliste 28](#_Toc37241470)

[7.11.6 Reflexion 28](#_Toc37241471)

[7.11.7 Vergleich mit dem Zeitplan 28](#_Toc37241472)

[8 Organisation der Arbeitsergebnisse 28](#_Toc37241473)

[9 Management Summary 29](#_Toc37241474)

[9.1 Ausgangslage 29](#_Toc37241475)

[9.2 Vorgehen 29](#_Toc37241476)

[9.3 Ergebnis 29](#_Toc37241477)

[10 Informieren 29](#_Toc37241478)

[10.1 Ziel 29](#_Toc37241479)

[10.2 Absicht 30](#_Toc37241480)

[10.3 Anforderungen 30](#_Toc37241481)

[10.4 Ressourcen 30](#_Toc37241482)

[10.5 Deadline 30](#_Toc37241483)

[10.6 Herausforderungen 30](#_Toc37241484)

[10.6.1 JavaScript Diagramme 30](#_Toc37241485)

[10.6.2 Sensor Wert Beobachter 31](#_Toc37241486)

[10.6.3 Responsive Design 32](#_Toc37241487)

[10.6.4 Sphinx Dokumentation 33](#_Toc37241488)

[10.6.5 HTML Präprozessor 33](#_Toc37241489)

[10.6.6 Tür Schaltungssensor Wert 33](#_Toc37241490)

[11 Planen 34](#_Toc37241491)

[11.1 Use Case 34](#_Toc37241492)

[11.2 System 35](#_Toc37241493)

[11.2.1 Allgemein 35](#_Toc37241494)

[11.2.2 Uvicorn Server 35](#_Toc37241495)

[11.2.3 Websockets 35](#_Toc37241496)

[11.2.4 PostgreSQL Datenbank 35](#_Toc37241497)

[11.2.5 Visuell 35](#_Toc37241498)

[11.3 Sensoren 36](#_Toc37241499)

[11.4 Wert Beobachter 37](#_Toc37241500)

[11.5 Authentifizierung 37](#_Toc37241501)

[11.6 Frontend 38](#_Toc37241502)

[11.6.1 Admin Dashboard 38](#_Toc37241503)

[11.6.2 Personal Verwaltung 39](#_Toc37241504)

[11.6.3 Räume Verwaltung 39](#_Toc37241505)

[11.6.4 Raum Ansicht 41](#_Toc37241506)

[11.6.5 Sensoren Verwaltung 41](#_Toc37241507)

[11.7 Sensor Ansicht 42](#_Toc37241508)

[11.7.1 Reinigungspersonal Dashboard 43](#_Toc37241509)

[11.8 Konzeptionelles Modell 45](#_Toc37241510)

[11.9 Logisches Modell 46](#_Toc37241511)

[11.10 API Struktur 46](#_Toc37241512)

[11.11 Sicherheit 48](#_Toc37241513)

[11.11.1 Feld Validierung 48](#_Toc37241514)

[11.11.2 SQL Injection 49](#_Toc37241515)

[11.11.3 Cross Site Scripting 49](#_Toc37241516)

[11.12 Testen 49](#_Toc37241517)

[11.12.1 Testmethoden 49](#_Toc37241518)

[11.12.2 Testkonzept 49](#_Toc37241519)

[11.12.3 Testfallspezifikationen 49](#_Toc37241520)

[11.13 Vorgehen 54](#_Toc37241521)

[12 Entscheiden 54](#_Toc37241522)

[12.1 Allgemein 54](#_Toc37241523)

[12.2 Kriterien 54](#_Toc37241524)

[12.3 Nutzwertanalyse 55](#_Toc37241525)

[13 Realisieren 55](#_Toc37241526)

[13.1 Backend 55](#_Toc37241527)

[13.1.1 Backend Aufsetzen 55](#_Toc37241528)

[13.1.2 Konfiguration 56](#_Toc37241529)

[13.1.3 Tabs Sensor Dekodierung 58](#_Toc37241530)

[13.1.4 Loriot Listener 58](#_Toc37241531)

[13.1.5 Modelle 60](#_Toc37241532)

[13.1.6 Modell Controller 62](#_Toc37241533)

[13.1.7 Auflistungen 63](#_Toc37241534)

[13.1.8 Beobachter 64](#_Toc37241535)

[13.1.9 Resources 65](#_Toc37241536)

[13.1.10 Benutzer Eingaben Validierung 66](#_Toc37241537)

[13.2 Frontend 67](#_Toc37241538)

[13.2.1 Auth 67](#_Toc37241539)

[13.2.2 Verbindung zum Backend 67](#_Toc37241540)

[13.2.3 Listen 68](#_Toc37241541)

[13.2.4 Formen 69](#_Toc37241542)

[13.2.5 Reinigungspersonal Ansicht 71](#_Toc37241543)

[13.2.6 Darstellung der Sensor Daten 73](#_Toc37241544)

[13.2.7 Corporate Design 75](#_Toc37241545)

[14 Kontrollieren 76](#_Toc37241546)

[14.1 Testanlage 76](#_Toc37241547)

[14.2 Testmethoden 76](#_Toc37241548)

[14.3 Testprotokoll 76](#_Toc37241549)

[14.4 Testbericht 81](#_Toc37241550)

[15 Auswerten 81](#_Toc37241551)

[15.1 Reflexion 81](#_Toc37241552)

[15.2 Fazit 81](#_Toc37241553)

[15.3 Vergleich mit dem Zeitplan 82](#_Toc37241554)

[16 Glossar 82](#_Toc37241555)

[17 Literaturverzeichnis 83](#_Toc37241556)

[18 Tabellenverzeichnis 85](#_Toc37241557)

[19 Abbildungsverzeichnis 85](#_Toc37241558)

Teil 1: Umfeld und Ablauf

# Aufgabenstellung

## Titel der Arbeit

Auswertung und Visualisierung von IoT Sensor Daten

## Ausgangslage

Unsere Kundin die Berufsfachschule Baden (BBB) möchte in einem Pilotversuch erste Funktionen mittels IoT Sensoren für das Gebäudemanagement testen. Als erstes soll die Toiletten-Benutzung gemessen werden. Diese Informationen werden verwendet um den Reinigungsvorgang zu optimieren. Als zweite Messkategorie wird die Luftqualität in den Unterrichtszimmern überwacht. Die Pilotinstallation wird auf das Gebäude Bruggerstrasse der BBB begrenzt.

Für den Hausdienstleiter sollen die Daten der Sensoren in einem Dashboard angezeigt werden. Für das Reinigungspersonal sind Tablet-PCs vorgesehen, auf welchen die nötigen Informationen für die Reinigungstour angezeigt und Rückmeldungen erfasst werden können.

Die Applikation soll bezüglich Sicherheit den gängigen Standards entsprechen und eine unerlaubte Einsicht und Manipulation der Daten verhindern. Für die Umsetzung des GUI kommen die Corporate Design Vorgaben der BBB zum Einsatz.

## Detaillierte Aufgabenstellung

### Beschreibung

Die Aufgabe besteht im Groben in der Umsetzung der folgenden Funktionen:

* Interpretation und Auswertung der IoT Sensordaten aus der Datenbank.
* Programmierung eines Dashboards für das Gebäudemanagement (Reinigung, Luftqualität)
* Programmierung einer Detailansicht von Sensordaten
* Programmierung einer Übersicht für das Reinigungspersonal. Optimiert für die Benützung auf einem Tablet mit Touch-Bedienung.
* Programmierung von Notifikationen in der Anwendung bei der Erreichung von vordefinierten Grenzwerten der Toilettenbenutzung und bei der Luftqualität.

### Generelle Eigenschaften

* Die Anwendung muss als Webapplikation in der Programmiersprache Python (Backend) und HTML/Javascript/JQuery (Frontend) umgesetzt werden.
* Das GUI soll mit Responsive Design für Desktop, Tablet und Mobile umgesetzt werden. Für Mobile sollen die Standard-Browser der Android und iOS Geräte verwendet werden, respektive während der IPA können entsprechende Emulatoren verwendet werden.
* Die Anwendung soll flexibel und modular umgesetzt werden, so dass weitere durch Sensoren messbare Indikatoren in das Dashboard aufgenommen werden können.
* Die Anwendung soll die folgenden Browser in der aktuellen Version unterstützen: Chrome, Firefox und Edge.

### Anforderungen an die Webapplikation

#### Login Funktion

Der Zugang zur Applikation soll mit Benutzer/Passwort geschützt werden.

#### Dashboard Hausdienst

Das Dashboard dient dem Hausdienst als Übersicht über den aktuellen Zustand bezüglich Luftqualität und Reinigungsbedarf im Gebäude.

**Überblick aktuelle Luftqualität in den einzelnen Räumen**

In einer Übersicht sollen die aktuellen Messwerte (Temperatur, Luftfeuchtigkeit und CO2) angezeigt werden. Die Darstellung soll übersichtlich z.B. gemäss Aufteilung des Gebäudes angezeigt werden.

**Zeitlicher Verlauf der Luftqualität**

In einer Detailansicht eines Messpunktes sollen die Messwerte in grafischer Form angezeigt werden. Z.B. als Liniengrafik mit der Möglichkeit über einen wählbaren Zeitraum.

**Anzeige Reinigungsbedarf**

In einer separaten Seite sollen die Zählerdaten der Toilettenbenutzung angezeigt werden. Der Reinigungsbedarf soll grafisch sichtbar sein mit einem speziellen Symbol oder mit farblicher Unterscheidung.

**Administrationsseite**

Die Grenzwerte für die Luftqualität wie auch für die Toiletten-Reinigung sollen auf einer Administrationsseite angepasst werden können.  
Ebenfalls soll eine Reinigung unabhängig des Benutzungs-Zählers ausgelöst werden können.  
Alternativ kann diese Funktion auch in die Detailansicht der Sensoren eingebaut werden.  
Einfache Benutzerverwaltung (Benutzer erstellen, Rollen zuweisen, Passwort reset)

#### Übersicht für Reinigungspersonal

**Liste zu reinigender Stellen**

Die zu Reinigenden Toiletten sollen sortiert nach Stockwerk und Gebäude Seite angezeigt werden, so dass eine Reinigungstour optimiert durchgeführt werden kann.

**Rapportierung nach erfolgter Reinigung**

Pro Toilette soll eine Funktion zur Rapportierung der Reinigung mit Kommentarfunktion eingebaut werden. Zum Beispiel, wenn die Reinigungsperson einen Defekt feststellt.  
z.B. Checkbox «Reinigung erledigt», Textfeld «Möchten Sie etwas melden»  
Nach der Rapportierung startet der Zähler der Toilette automatisch einen neuen Zyklus.

**Materialliste zur Vorbereitung der Reinigungstour**

Aufgrund der Anzahl Benutzungen der zu reinigenden Toiletten soll eine Materialliste (WC-Papier, Handtücher, Seife etc.) erstellt werden, so dass das Reinigungspersonal die richtige Menge auf die Reinigungstour mitnehmen kann. Diese Funktion kann mit einer relativ einfachen Formel umgesetzt werden:

Anzahl Benutzung x Anzahl Toiletten x Erfahrungswert Material

Der Erfahrungswert des benötigten Materials im Bezug zur Benutzung muss noch definiert werden und ist nicht Bestandteil dieser IPA.

## Mittel und Methoden

* Python
* Javascript + JQuery
* IDE: Pycharm & Datalore
* Windows Server
* GIT

## Vorkenntnisse

* HTML/CSS/Javascript (3 Jahre)
* Python Programmierung (1 Jahr)
* JQuery

## Vorarbeiten

* Planen und Einrichten der Datenbank für die Sensordaten.
* Sensordaten erheben und/oder Testdaten erstellen damit genügend Daten für die Validierung der Funktionen vorhanden sind.
* Web Server mit Python einrichten.
* Wireframes für die verschiedenen Screens erstellen.
* HTML Templates vorbereiten.
* Token Basierte Authentifizierung
* GIT Repository
* Klassen Diagramm

## Neue Lerninhalte

* Web Grafiken
* Token Basierte Authentifizierung
* Sensor Daten auswerten und darstellen
* Darstellung von Echtzeitdaten in einem Web basierten Dashboard

## Arbeiten in den letzten 6 Monaten

* Python API mit Firebase
* Webseite mit PHP/Wordpress für BBBaden realisieren
* Wordpress Plugins programmieren
* HTML/CSS/Javascript Templates umsetzen

## Individuelle Bewertungskriterien

### Responsive Design

**Leitfrage**

Wird die Webanwendung dynamisch auf verschiedene Bildschirmgrössen für Desktop, Tablet und Mobile angepasst?

Tabelle 2 Responsive Design Bewertungskriterium

|  |  |
| --- | --- |
| Gütestufe | Beschreibung |
| 3 | Die Webanwendung wird auf allen drei Bildschirmgrössen angepasst angezeigt. |
| 2 | Zwei Bildschirmgrössen werden angepasst angezeigt. |
| 1 | Eine Bildschirmgrösse wird angepasst angezeigt. |
| 0 | Bei keiner Bildschirmgrösse wird die Anwendung angepasst angezeigt |

### Browserkompatibilität

**Leitfrage**

Unterstützt die Anwendung die drei in der Aufgabenstellung geforderten Browser Edge, Firefox und Chrome bezüglich Funktionen und Design.

Tabelle 3 Browserkompatibilität Bewertungskriterium

|  |  |
| --- | --- |
| Gütestufe | Beschreibung |
| 3 | Alle drei Browser werden bezüglich Funktionen und Design vollständig unterstützt. Es sind höchstens minimale Unterschiede im Design zu erkennen. |
| 2 | Zwei der drei in der Leitfrage erwähnten Browser werden vollumfänglich unterstützt. |
| 1 | Einer der drei in der Leitfrage erwähnten Browser wird vollumfänglich unterstützt. |
| 0 | Keiner der drei in der Leitfrage erwähnten Browser wird vollumfänglich unterstützt. |

### Einhaltung Corporate Design

**Leitfrage**

Ist bei der Webanwendung das Corporate Design der BBB gemäss BBB\_CD\_Manual.pdf eingehalten?

Tabelle 4 Corporate Design Bewertungskriterium

|  |  |
| --- | --- |
| Gütestufe | Beschreibung |
| 3 | Die Grafik-Elemente Schrift, Farben & Logo werden gemäss obenstehenden Vorgaben korrekt verwendet. |
| 2 | Zwei der Grafik-Elemente werden richtig verwendet. |
| 1 | Ein Grafik-Element wird korrekt eingesetzt. |
| 0 | Keines der Grafik Elemente wird korrekt eingesetzt. |

### Codingstyle – Dokumentation

**Leitfrage**

Ist der Quellcode dokumentiert? Hilft der Text, die Funktionalität zu verstehen und nachzuvollziehen?

Tabelle 5 Codingstyle Bewertungskriterium

|  |  |
| --- | --- |
| Gütestufe | Beschreibung |
| 3 | Der Text erklärt was notwendig ist, um die Funktion/Methode besser zu verstehen. Allfällige Richtlinien sind eingehalten. |
| 2 | Der Text hilft nur zum Teil weiter. Allfällige Richtlinien sind teilweise berücksichtigt. |
| 1 | Der Text hilft selten weiter. Allfällige Richtlinien sind offensichtlich verletzt. |
| 0 | Es ist wenig bis gar nichts dokumentiert. |

### Validierung Eingaben

**Leitfrage**

Werden die eingegebenen Daten aller Felder in allen Formularen validiert und allgemein verständliche Fehlermeldungen angezeigt?

Tabelle 6 Validierung Bewertungskriterium

|  |  |
| --- | --- |
| Gütestufe | Beschreibung |
| 3 | Alle drei Kriterien (Felder, Formulare, Fehlermeldungen) sind erfüllt |
| 2 | Zwei Kriterien sind erfüllt |
| 1 | Ein Kriterium ist erfüllt |
| 0 | Kein Kriterium ist erfüllt |

### Anzeige der Luftqualitäts-Messwerte

**Leitfrage**

Erfolgt eine korrekte Auswertung der Messwerte der Sensoren und werden diese richtig im Dashboard angezeigt? Beispielsweise soll die relative Luftfeuchtigkeit in Prozent angezeigt werden.

Tabelle 7 Anzeige Luftqualität Bewertungskriterium

|  |  |
| --- | --- |
| Gütestufe | Beschreibung |
| 3 | Alle drei Messwert-Kategorien werden richtig interpretiert und angezeigt. |
| 2 | Zwei Messwert-Kategorien werden richtig interpretiert und angezeigt. |
| 1 | Eine Messwert-Kategorie wird richtig interpretiert und angezeigt. |
| 0 | Keine Messwert-Kategorie wird richtig interpretiert und angezeigt. |

### Grafische Darstellung der Messwerte

**Leitfrage**

Kann der Verlauf der drei Luftqualitätsmesswerte grafisch und mit wählbarem Zeitraum angezeigt werden?

Tabelle 8 Grafische Darstellung der Messwerte Bewertungskriterium

|  |  |
| --- | --- |
| Gütestufe | Beschreibung |
| 3 | Alle drei Messwerte werden als Grafik und mit wählbarem Zeitraum angezeigt. |
| 2 | Alle drei Messwerte werden als Grafik angezeigt, jedoch ohne wählbaren Zeitraum. |
| 1 | Nur ein Teil der Messwerte werden als Grafik angezeigt. |
| 0 | Kein Messwert kann als Grafik angezeigt werden. |

# Projektaufbauorganisation

## Methode

Die gesamte IPA wird mit [IPERKA](https://www.tgabathuler.ch/Iperka/Einfuehrung.html) (Informieren, Planen, Entscheiden, Realisieren, Kontrollieren und Auswerten) durchgeführt. IPERKA ist für ein Projekt einfach anzuwenden und ich habe damit schon viele Erfahrungen bei verschiedenen Arbeiten gesammelt.

## Auftraggeber

Die Berufsbildung Baden ist der Auftraggeber für diesen Pilot Versuch.

## Beteiligte Personen

**Kandidat**

Nils Egger  
[nils.egger@avectris.ch](mailto:nils.egger@avectris.ch)

**Verantwortliche Fachkraft**

Daniel Säuberli  
[daniel.saeuberli@avectris.ch](mailto:daniel.saeuberli@avectris.ch)

**Berufsbildner**

Mark Grabner  
[mark.grabner@avectris.ch](mailto:mark.grabner@avectris.ch)

## Durchführungsort

Die IPA wird in der AVECTRIS AG an der unten aufgeführten Adresse durchgeführt.

Avectris AG  
Bruggerstrasse 68  
5401 Baden

# Deklaration der Vorkenntnisse

Tabelle 9 Deklaration der Vorkenntnisse mit Bemerkung

|  |  |
| --- | --- |
| Kenntnis | Bemerkung |
| Python | Mit Python automatisiere ich kurze wiederholende Tasks. Vor knapp einem Jahr habe ich damit mit einem Framework für Backend API’s begonnen. |
| HTML, CSS | HTML und CSS verwende ich seit Beginn meiner Lehre, dies ist definitiv kein Neuland mehr. |
| Bootstrap | Ich kann Bootstrap ohne Probleme anwenden, jedoch ist es für mich neu die SCSS Dateien anzupassen. |
| JavaScript, JQuery | Bei den meisten Webseiten, welche ich schreibe, verwende ich JavaScript und JQuery. Daher kenne ich diese gut. |
| PostgreSQL | Ich beherrsche gängige SQL-Befehle, habe aber noch keine vertiefte Routine in der Anwendung der Befehle. Ich arbeite erst seit Beginn des Jahres wieder mit SQL. |
| Jekyll | Jekyll ist ein HTML-Preprocessor, bei welchem ich die Basics verstehe. |

# Deklaration der Vorarbeiten

* Mockups der Webseite
* Sensor Wert Dekodierung vorbereitet für die Adeunis RF und Elsys CO2 Sensoren
* Sensor Wert Abhörung von Loriot
* Installation der Entwicklungsumgebung
* Vorbereitung des Servers
* Konzeptionelles und logisches Modell
* SQL Abfragen für den Aufbau der Datenbank
* Sphinx Dokumentation für das Backend eingerichtet
* Jekyll mit Bootstrap eingerichtet für das Frontend
* Token Authentifizierung
* Framework für die Erstellung von Model Controllers für Models
* Python Client für PostgreSQL

# Deklaration der benützten Firmenstandards

Für dieses Dokument wurde eine Word Vorlage für Projekt Berichte verwendet.

# Zeitplan

## Bemerkungen

Der Zeitplan ist in ein 2 Stunden Raster aufgeteilt. Somit besteht ein voller Tag aus 8 Stunden. Am 25. März und 01. April bin ich ein halber Tag in der Schule und daher bestehen diese 2 Tage nur aus 4 Stunden. Auch der Zeitplan ist nach IPERKA aufgeteilt, die Phasen sind mit einem grauen Balken aufgeteilt.

Nach Vorschlag des FArbeit\_2020.pdf Dokumentes sind mehr als 40% der Stunden für dokumentieren eingeteilt.

## Legende

Tabelle 10 Legende der gefärbten Felder im Zeitplan

|  |  |
| --- | --- |
| Zustand | Kennzeichnung |
| Ist | Grün |
| Soll | Orange |
| Ist-Meilenstein | Güner Richtungspfeil |
| Soll-Meilenstein | Oranger Richtungspfeil |

## Meilensteine

Die Meilensteine sind nach den IPERKA Phasen gesetzt. Zudem ist die Phase Realisierung in zwei Meilensteine aufgeteilt. Die Erste nach dem Ende der Realisierung des Backend und die Zweite nach der Fertigung des Frontend.

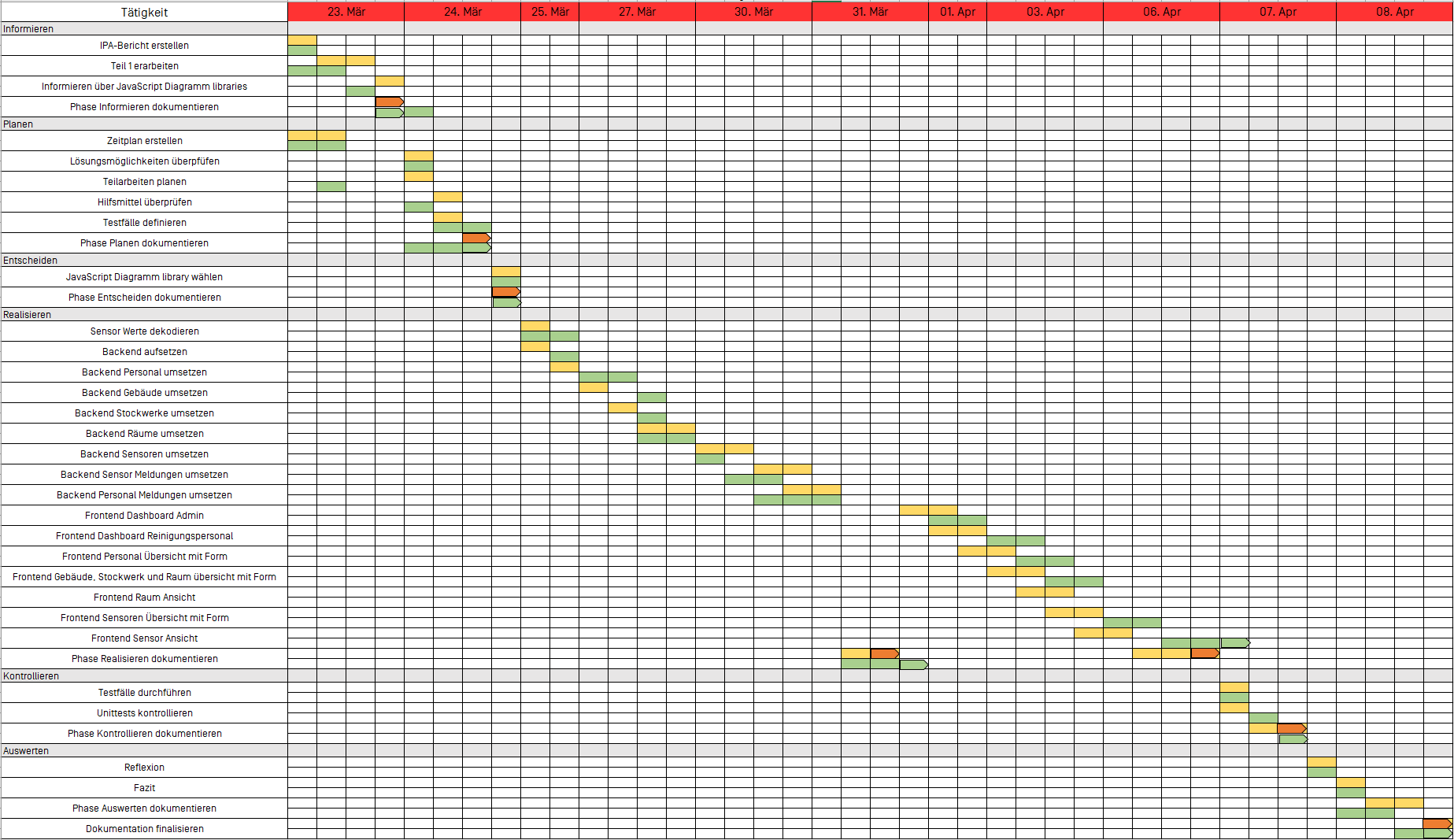


Abbildung 1 Zeitplan

# Arbeitsjournal

## Arbeitsjournal 23.03.2020

### Ausgeführte Arbeiten

Am Morgen las ich zuerst nochmals das Dokument FArbeit\_2020.pdf, ich wollte sichergehen, dass ich nichts Wichtiges im Dokument überlesen hatte. Dies dauerte nicht allzu lange und ich konnte schnell mit meinem Zeitplan beginnen. Aus eigener Erfahrung weiss ich, dass ich gerne Aufgaben kürzer einschätze als diese wirklich andauern. Ich habe daher immer ein bisschen mehr Zeit für die Arbeiten eingeplant.

Vom Morgen kann ich ein positives Fazit ziehen. Ich kam sehr gut voran und war um 11 Uhr bereits eine Stunde dem Zeitplan voraus. Danach konnte ich mit dem Informieren von IPERKA beginnen. Das einzige was mir noch ein bisschen fremd war ist das Darstellen der Sensordaten. Mit einer kurzen Google Suche konnte ich drei Varianten finden, welche alle vielversprechend aussahen. Chart.js, Chartist.js und d3.js. Nachdem ich für die ersten zwei ein kleines Linien Diagramm Beispiel erstellte, stellte sich heraus, dass die dritte Bibliothek, nämlich d3.js, eher für grössere und datenintensivere Projekte gedacht ist. Daher habe ich diese ausgelassen.

Beim Dokumentieren der Informieren Phase war ich mir oft unsicher ob ich genügend geschrieben habe und auch wirklich das Wichtigste auf den Punkt bringe. Schlussendlich bin ich jedoch zufrieden mit dem Resultat.

Um halb 4 war ich fertig mit der Informieren Phase und habe somit meinen ersten Meilenstein erreicht. Es folgt die Phase Planen.

### Erreichte Ziele / Erfolge

* Zeitplan erstellt
* Teil 1 der Dokumentation fertig
* Chart.js Grundkenntnisse erlernt
* Chartist.js Grundkenntnisse erlernt.
* Informieren Phase fertig

### Aufgetretene Probleme / Misserfolge

* Zeitplan Formatierung und Darstellung dauerte länger als erwartet
* Oft unsicher ob das Richtige dokumentiert wurde
* Anpassen der Diagramm Grösse von Chart.js

### Beanspruchte Hilfestellungen

Bisher habe ich keine Hilfestellungen beansprucht.

### Pendenzenliste

Der Zeitplan konnte ich gut befolgen und habe daher keine offenen Pendenzen.

### Reflexion

Für meinen nächsten Zeitplan werde ich nicht nur das Layout vorausplanen, sondern auch das Template bereits umsetzen, dies hätte mir sicher eine halbe Stunde Arbeit erspart.

Das Dokumentieren fällt mir noch schwer und ich hoffe diese Hürde lässt sich in den nächsten Tagen überwinden. Ich bin jedoch Stolz über meine UML Diagramme und habe weiterhin vor diese zu erstellen.

### Vergleich mit dem Zeitplan

Am Morgen war ich bereits fertig mit dem Teil 1 der Dokumentation, dies hatte ich eigentlich auch noch für den Anfang des Nachmittags eingeplant. Nach dem Mittag änderte dies sich ein bisschen. Ich bin zwar weiterhin dem Zeitplan voraus, aber nicht mehr mit dem gleichen Abstand wie am Morgen noch. Meine Überschätzung des Aufwandes zahlt sich hier bereits aus. Meinen ersten Meilenstein konnte ich daher ohne Zeitdruck einhalten.

## Arbeitsjournal 24.03.2020

### Ausgeführte Arbeiten

Am Morgen plante ich als erstes das System und erstellte eine Visuelle Grafik dazu. Danach überarbeitete ich kurz meinen Text von gestern und merkte schnell, dass ich vergessen habe mich über den Tabs Sensor zu informieren. Damit meine ich nur, dass ich wissen muss wie man die Bytes in Werte verwandelt. Zum Glück war dies gut im Datenblatt beschrieben und ich glaube ich werde dies einfach umsetzen können. Ich fing an die Planen Phase zu dokumentieren mit den Arbeiten welche ich bereits wie ich diese lösen möchte. Später konnte ich mir auch die ungeklärten Fragen erklären, wie ich zum Beispiel auf die verschiedenen Werte eines Sensors hören sollte und auch Meldungen dazu auslösen kann. Ich war der Meinung, dass dies für mich noch Probleme geben würde, jedoch bin ich Recht zufrieden mit meiner geplanten Lösung für die Wert Beobachter und bin zuversichtlich, dass diese funktionieren werden. Das Erstellen der Testfallspezifikation ging ein bisschen länger als ich mir erhoffte. Ich lag jedoch immer noch im Zeitplan, hatte jedoch gegen den Mittag das Gefühl, dass ich heute diesem hinterher falle. Später fügte ich das konzeptionelle und logische Modell in die Dokumentation ein. Leider fehlten ein paar Komponenten der Applikation und ich musste diese Verbessern. Nachdem ich die Planen Phase fertig hatte, investierte ich ein bisschen Zeit dafür, den aktuellen Stand der Dokumentation zu verbessern. Danach entschied ich mich für Chart.js und dokumentierte die Phase Entscheiden. Schlussendlich war ich eine halbe Stunde früher fertig als geplant und nutzte diese Zeit zum nochmals den Kriterienkatalog durchzugehen.

### Erreichte Ziele / Erfolge

* Ich weiss nun wie ich Sensor Wert Meldungen auslösen kann
* Dekodierung des Tabs Sensors geplant
* Testfallspezifikationen erstellt
* Planen Phase Meilenstein erreicht
* Entscheiden Phase Meilenstein erreicht

### Aufgetretene Probleme / Misserfolge

* Ich habe gestern vergessen mich über den Tabs Sensor zu informieren.
* Vorbereitetes konzeptionelles und logisches Modell mussten verbessert werden.
* Geplantes Testkonzept und Spezifikation entsprechen nicht ganz dem Kriterienkatalog.

### Beanspruchte Hilfestellungen

Ich habe keine Hilfestellungen beansprucht.

### Pendenzenliste

Testkonzept und Spezifikationen muss überarbeitet werden damit sie besser dem Kriterienkatalog entsprechen.

### Reflexion

Ich kam auch heute wieder gut voran. Es fällt mir nun einfacher zu dokumentieren und ich konnte auch den Text des letzten Tages verbessern. Ich werde jedoch oft unsicher ob jemand, welcher nicht gleich vertieft ist in meine Arbeit wie ich, alles verstehen würde. Es ist für mich schwierig einzuschätzen, ob ich auch wirklich alles Wichtige aufgeführt habe.

Zudem habe ich zurzeit die Befürchtung, dass ich den Kriterienkatalog nicht genügend genau befolgt habe, dies ist mir vor allem beim Testkonzept aufgefallen.

### Vergleich mit dem Zeitplan

Auch an diesem Tag zahlte es sich aus, dass ich für jede Arbeit ein bisschen mehr Zeit einplante. Die Phase Planen verlief nicht ganz nach Zeitplan, sondern war eher ein durcheinander was die Auftragschronologie anging. Die Entscheidungsphase verlief dagegen genau nach Zeitplan.

## Arbeitsjournal 25.03.2020

### Ausgeführte Arbeiten

Am Morgen musste ich noch kurz mein Testkonzept der Planen Phase verbessern. Dies ging nicht allzu lange. Danach startete ich die Realisierung Phase. Die Dekodierung des Tabs Sensors verlief eigentlich gut, jedoch schrieb ich zuerst einen Code für die Umwandlung von Hex in Dezimal, natürlich gibt es dies bereits in Python und ich verschwendete somit meine Zeit. Dazu kommt auch noch die Zeit welche ich nutze um diesen falschen Code zu dokumentieren. Danach setzte ich mich daran das Backend aufzubauen. Für die Datenbank hatte ich eigentlich bereits ein SQL File geschrieben, welches alle Tabellen aufsetzt. Jedoch musste ich gestern das logische Modell verändern und somit war diese Datei nicht mehr auf dem neusten Stand. Ich erweiterte und aktualisierte dieses. Letztendlich erstellte ich noch das Hauptfile für den Server und fügte die Login Route hinzu. Man könnte sich nun über die API einloggen.

### Erreichte Ziele / Erfolge

* Tabs Sensor korrekt dekodiert
* Datenbank Tabellen aktualisiert
* Login Route erstellt

### Aufgetretene Probleme / Misserfolge

* Zeit verschwendet mit dem Implementieren einer Funktion welche es natürlich bereits in Python gibt.
* Auch wenn ich den richtigen Wert aus dem Sensor auslese, weiss ich nicht wie ich diesen in eine Prozentuale Anzeige der Ladung umwandle.
* SQL Datenbank Aufbau Datei veraltet.

### Beanspruchte Hilfestellungen

Ich habe heute Daniel Säuberli gefragt ob er mir bei der Umwandlung der Batterien Werte helfen konnte, jedoch konnte er mir noch keine Hilfestellung bieten.

### Pendenzenliste

Es wäre von Vorteil, wenn ich noch herausfinden könnte wie ich die Spannung der Batterien der Sensoren in eine Prozentuale Anzeige umwandeln kann. Dies hat jedoch keine Priorität und ist auch kein muss.

Ich konnte nach Zeitplan das Personal im Backend noch nicht umsetzen, dies muss ich am Freitag realisieren.

### Reflexion

Ich hätte definitiv besser überlegen sollen, als ich meine eigene Funktion implementiert habe um Hex in Dezimal umzuwandeln. Es hätte klar sein sollen, dass es dies bereits gibt. Ich bin jedoch überrascht, dass meine Dekodierung beim ersten Versuch geklappt hat, dies war bei den Vorarbeiten noch nicht so. Darauf bin ich stolz.

Ich war heute nicht allzu konzentriert was sich auf meine Menge an geleisteter Arbeit ausgewirkt hat.

### Vergleich mit dem Zeitplan

Nachdem ich die Sensor Dekodierungen implementiert hatte, war ich eine Stunde dem Zeitplan hinterher. Jedoch habe ich gleichzeitig beim implementieren des Codes diesen dokumentiert. Das wird mir Zeit sparen für meine eigentliche eingeplante Zeit zur Dokumentation. Den Tag beendete ich zwei Stunden dem Zeitplan hinterher. Eine Aufgabe, nämlich das Personal im Backend umsetzen, muss ich verschieben.

## Arbeitsjournal 27.03.2020

### Ausgeführte Arbeiten

Heute Morgen musste ich noch meine offene Pendenz von gestern machen. Dies dachte ich sollte eigentlich nicht allzu lange gehen, jedoch hatte ich grosse Startprobleme. Beim Aufsetzen meiner Unit Tests funktionierte nichts und ich konnte dadurch erst um 10 Uhr startete. Die Personal und Gebäude API für das Erstellen, Aktualisieren und Löschen der Modelle konnte ich ohne grössere Probleme danach implementieren. Am Mittag war ich drei Stunden dem Zeitplan hinterher. Am Nachmittag konnte ich einen starken Endspurt hinlegen, ich schaffte es die drei Stunden beim Programmieren aufzuholen und dabei die Stockwerk und Raum API zu erstellen.

### Erreichte Ziele / Erfolge

* Unit Tests Problem gelöst
* Erste Modelle können erstellt werden

### Aufgetretene Probleme / Misserfolge

* Unit Tests aufsetzen funktionierte nicht so wie es sollte.
* Führe ich die Unit Tests einzeln aus funktionieren alle, führe ich jedoch alle nacheinander aus so wird bei einigen die Datenbank nicht korrekt verbunden. Dies ist nur bei Tests mit Starlette so.
* Es fehlte eine Spalte für die Stockwerke Tabelle

### Beanspruchte Hilfestellungen

Heute konnte mir Daniel Säuberli mit dem Batterien Problem helfen. Es stellte sich jedoch heraus, dass dies nicht so einfach wird. Ich bin mir noch unsicher, ob ich dies überhaupt umsetzen werde.

### Pendenzenliste

Ich habe keine offenen Pendenzen.

### Reflexion

Von allen Sachen welche Probleme mit sich bringen könnten war das Aufsetzen von Unit Tests das Letzte was mir in den Sinn kommen würde. Davon habe ich heute gelernt, dass ich auch nur die kleinsten Komponenten vorbereiten sollte.

Ich hatte heute sehr viele Flüchtigkeitsfehler, welche einfach verhindert werden hätten können. Ich hoffe diese kann ich am Montag mit neuer Kraft vermeiden.

Mit meinem Endspurt bin ich sehr zufrieden, ansonsten wäre ich nun sehr hinterher. Ich habe momentan das Gefühl, dass ich vielleicht doch ein wenig zu wenig Zeit für das Dokumentieren geplant habe, doch dann hätte ich zu wenig Zeit für das implementieren des Codes. Dies hätte vielleicht besser geplant werden können.

### Vergleich mit dem Zeitplan

Die Pendenz von gestern und das Problem mit den Unit Tests setzte mich am Morgen drei Stunden dem Zeitplan hinterher. Am Nachmittag konnte ich meinen Rückstand aufholen und bin jetzt wieder im Zeitplan.

## Arbeitsjournal 30.03.2020

### Ausgeführte Arbeiten

Heute Morgen startete ich mit den restlichen Modell Controllers. Diese konnte ich ohne grössere Probleme erstellen. Es lief sogar so gut, dass ich am Morgen mit der Erstellung von Meldungen beginnen konnte. Damit dies überhaupt funktioniert, müssen zuerst einmal die Daten von Loriot abgefangen werden. Als Vorarbeit hatte ich dies bereits ausprobiert und konnte es nun korrekt umsetzen. Am Nachmittag startete ich endlich damit die Sensor Beobachter zu implementieren. Dadurch ist es möglich eine Automatische Meldung erstellen zu lassen, wenn ein Sensor einen gewissen Wert überschreitet.

### Erreichte Ziele / Erfolge

* Loriot Listener
* Sensor Beobachter

### Aufgetretene Probleme / Misserfolge

* Meine SQL Abfragen waren nicht immer korrekt. Damit richtige Werte nicht überschrieben werden mit Null Werten sollte man Coalesce einsetzen, dies hatte ich vergessen.
* Beim Temperatur Sensor werden immer wieder Meldungen nacheinander ausgegeben, weil die Temperatur meistens über längere Zeit seinen Richtwert überschritten hat. Darum musste ich ein Timeout für die Beobachter implementieren.

### Beanspruchte Hilfestellungen

Ich habe heute keine Hilfestellungen beansprucht.

### Pendenzenliste

Es fehlen noch ein paar Auflistungen von Modellen in Backend.

### Reflexion

Dieser Tag verlief gut. Ich erwartete Schwierigkeiten beim Loriot Listener doch hatte keine.

Ich konnte mich heute sehr gut konzentrieren und bin stolz auf meine geleistete Arbeit. Ich hoffe das ich diese Arbeitsweise für die nächsten Tage beibehalten werde. Auch wenn ich heute viel geleistet habe bin ich immer noch unsicher ob die Zeit für die Dokumentation reichen wird. Dagegen kann ich jedoch noch nichts unternehmen.

### Vergleich mit dem Zeitplan

Der Morgen verlief sehr sauber und ich war am Morgen vom Realisieren Aspekt her dem Zeitplan voraus. Die Dokumentation ist aber noch nicht sehr gut, dafür ist aber morgen Zeit eingeplant. Am Nachmittag holte mich der Zeitplan wieder ein und ich bin wegen ein paar kleineren Aufträgen hinterher.

## Arbeitsjournal 31.03.2020

### Ausgeführte Arbeiten

Am Morgen musste ich noch das Modell und den Controller für die Materialien erstellen, danach ging ich an das Dokumentieren des Backend. Da ich vieles bereits während der Realisierung dokumentierte, sparte ich hier ein wenig Zeit ein. Trotzdem verbrachte ich fast den ganzen Tag damit. Am Ende des Tages konnte ich noch mit der Webseite anfangen. Ich implementierte hier die Corporate Design Vorlage in SCSS, somit stimmen nun die Farben und das Logo ist rechts oben in der Ecke.

### Erreichte Ziele / Erfolge

* Backend so gut wie fertig
* Erster Teil der Realisieren Phase fertig -> Meilenstein
* Corporate Design Vorlage umgesetzt

### Aufgetretene Probleme / Misserfolge

* Umsetzung der Grösse und entsprechenden Margins des Logos verlief eher lange.
* Materialien hatte ich gestern vergessen im Backend umzusetzen.

### Beanspruchte Hilfestellungen

Ich habe keine Hilfestellungen beansprucht.

### Pendenzenliste

* Code Dokumentation muss überarbeitet werden.
* Es fehlen stets noch einige Auflistungen im Backend, diese werden jedoch nach Gebrauch umgesetzt.
* Logo Grösse soll sich nach Grösse des Bildschirms verändern.

### Reflexion

Der heutige Tag verlief nicht sehr gut, er war für mich sehr anstrengend. Durch das viele Schreiben verlor ich oft meine Konzentration und musste mich durchkämpfen. Ich bin froh, dass das Backend so gut wie fertig ist. Es läuft auch ohne Probleme, dies habe ich zum Glück mit Tests sichergestellt. Somit muss ich keine Angst haben, dies finde ich von Vorteil und werde dies auch so in anderen Projekten beibehalten.

### Vergleich mit dem Zeitplan

Dadurch, dass ich die Materialien unerwartet noch umsetzen musste, bin ich nun zwei Stunden dem Zeitplan hinterher. Dies ist jedoch nicht so schlimm. Ich erwarte, dass ich das HTML schnell erarbeiten kann. Ich habe weiterhin Angst um die Zeit, es fühlt sich so an, als würden die letzten Tage eher stressig werden.

## Arbeitsjournal 01.04.2020

### Ausgeführte Arbeiten

Am Morgen implementierte ich das Login mit JavaScript, obwohl dies ohne grössere Probleme funktionierte dauerte dies lange. Man kann sich nun Einloggen und bekommt auch korrekte Meldungen falls dies nicht geklappt hat. Danach wird man auf die noch leere Dashboard Seite weitergeleitet. Dies wollte ich ändern und begann die Auflistung der Meldungen im Backend noch zu implementieren. Folgend machte ich das BBB Logo Responsiv, je nach Grösse des Bildschirmes verändert sich nun auch die Grösse des Logos. Die Grösse um welche sie sich ändert kann man einfach im SCSS verändern.

Am Schluss des Tages startete ich noch damit das HTML für die Dashboard Seite zu implementieren. Dabei merkte ich, dass die Mockups für einen eher grösseren Bildschirm bestimmt sind. Deswegen musste ich die Karten mit Tabellen austauschen, da man ansonsten zu wenig Platz hat. Ich konnte die Meldungen vom Backend laden, jedoch hatte ich keine Zeit mehr diese darzustellen.

### Erreichte Ziele / Erfolge

* Logo verändert sich je nach Grösse des Bildschirms
* Man kann sich nun einloggen

### Aufgetretene Probleme / Misserfolge

* Die Implementation des Logins dauerte sehr lange
* Access Token mit einem Refresh Token erneuern funktioniert noch nicht

### Beanspruchte Hilfestellungen

Ich habe keine Hilfestellungen beansprucht.

### Pendenzenliste

* Einen Access Token mit einem Refresh Token erneuern lassen
* Tabelle im Dashboard mit Meldungen füllen

### Reflexion

Der heutige Tag verlief nicht so wie erwartet, ich war in der Hoffnung zwei Seiten fertig stellen zu können. Ich bin bei der Ersten bislang in der Hälfe. Ich bin der Meinung, dass der Grund, wieso ich heute nicht so gut vorankam an kleineren Problemen liegt, welche nicht wieder passieren werden. Daher bin ich davon überzeugt, dass es sicherlich besser funktionieren wird am Freitag.

### Vergleich mit dem Zeitplan

Ich bin nun etwa vier Stunden hinterher. Zu diesem Zeitpunkt bin ich jedoch noch zuversichtlich, dass dies nicht zu gravierend sein wird.

## Arbeitsjournal 03.04.2020

### Ausgeführte Arbeiten

Ich startete den Tag mit dem fertigstellen der Admin Dashboard Seite, danach implementierte ich die Dashboard Seite für das Reinigungspersonal. Dies ging sehr lange, da ich die Räume und Stockwerke sortieren musste. Zudem ist die Ansicht für die Admins und das Reinigungspersonal sehr unterschiedlich und mussten daher beide von Grund auf selbst gebaut werden. Danach schrieb ich die Personal Seite. Auf dieser Seite ist es nun möglich, Personal zu erstellen, aktualisieren und löschen. Hier trat zuerst ein Problem auf, bei welchem es möglich war, Benutzer mit den gleichen Login Daten zu registrieren. Nachdem ich dieses Problem gelöst hatte, in dem ich im Backend eine SQL Abfrage erstellte, ging ich daran die Gebäude Verwaltung zu implementieren. Hierbei konnte ich zum Glück Code wiederverwenden und es ging ein bisschen schneller.

### Erreichte Ziele / Erfolge

* Dashboard für Admins und Reinigungspersonal fertig
* Personal Verwaltung fertig
* Gebäude Verwaltung fertig

### Aufgetretene Probleme / Misserfolge

* Meldungen konnten nicht als bearbeitet markiert werden, weil die SQL Abfrage, welche für das aktualisieren dieser zuständig ist, dieses Feld nicht enthielt.
* Es war möglich Benutzer mit den gleichen Daten zu registrieren.
* Wenn ein Benutzer gelöscht wurde, wurde sein Login nicht gelöscht.
* Wird ein Gebäude gelöscht, so werden Stockwerke und Räume in diesem auch gelöscht. Die Gebäude Verwaltung erkennt dies jedoch nicht und zeigt die Räume weiterhin an. Erst bei einem Refresh wird dies behoben.

### Beanspruchte Hilfestellungen

Ich habe keine Hilfestellung beansprucht.

### Pendenzenliste

* Raum Ansicht
* Sensoren Übersicht mit Form

### Reflexion

Heute realisierte ich, wie lange es geht HTML und JavaScript umzusetzen. Ich bin der Meinung, dass es vielleicht gut gewesen wäre, hätte ich ein Framework benutzt. Jedoch kenne ich keines und hätte dies als Vorbereitung lernen müssen. Ich bin sehr zufrieden mit meiner heutigen Leistung, da ich es geschafft habe, nicht noch weiter hinterher zu fallen. Langsam sinkt aber meine Motivation und ich habe Angst, ob ich auch wirklich noch fertig werde mit dem Frontend.

### Vergleich mit dem Zeitplan

Ich hinke weiterhin dem Zeitplan hinterher. Ich bin aktuell sechs Stunden hinter dem Plan. Dies liegt nur am HTML und ich habe nicht wirklich eine Chance dies zu verändern. Die einzige Hoffnung die ich noch habe ist, dass ich beim Dokumentieren nicht die volle eingeplante Zeit benötigen werden.

## Arbeitsjournal 06.04.2020

### Ausgeführte Arbeiten

Ich musste heute zuerst einmal das Frontend fertigbringen. Im Endspurt erstellte ich die Form für das Erstellen von Material, Beobachtern und Sensoren. Dies dauerte nicht allzu lange, jedoch verlor ich sehr viel Zeit beim Implementieren der Ansicht eines Sensors. Es fehlten noch ein paar Auflistung im Backend, welche ich alle auch noch erstellen musste. Danach fiel mir auf, dass die Webseite in der Mobilen Ansicht die Navigation nicht anpasst. Dies musste ich beheben. Danach fing ich an, die Docstrings für mein Backend Code zu verbessern und mit Sphinx eine Dokumentation aufzubauen.

### Erreichte Ziele / Erfolge

* Frontend so gut wie fertig.
* Sensor Daten visuell darstellen.

### Aufgetretene Probleme / Misserfolge

* Fehlende Auflistungen
* In JavaScript wurde manchmal Variablen wegen asynchronen Funktionen ungewollt überschrieben.

### Beanspruchte Hilfestellungen

Beim Darstellen der Sensoren Daten wusste ich nicht wie ich am besten viele Werte auf einmal darstellen. Daniel Säuberli brachte die Idee, dass pro Tag nur den kleinsten, grössten und auch noch den Durchschnittswert dargestellt werden soll.

### Pendenzenliste

Ich muss das Frontend noch dokumentieren. Dies war eigentlich für heute geplant.

### Reflexion

Ich bin zurzeit sehr gestresst, da ich eigentlich die Frontend Dokumentation heute fertig stellen sollte, jedoch hatte ich noch nicht mal Zeit diese anzufangen. Ich finde nicht, dass ich schlecht gearbeitet habe, denn ich war immer konzentriert und fleissig an der Arbeit. Das HTML ging viel länger als erwartet, dies hätte von mir als Vorarbeit besser abgeklärt werden sollen, wie lange es denn wirklich dauert, dass Frontend zu erstellen.

### Vergleich mit dem Zeitplan

Ich bin zurzeit dem Zeitplan sechs Stunden hinterher. Mit Hinsicht darauf, dass ich nur noch zwei Tage Zeit habe, muss ich wirklich Gas geben.

## Arbeitsjournal 07.04.2020

### Ausgeführte Arbeiten

Ich verbesserte am Morgen die Ansicht der Sensor Daten. Es ist nun möglich, zu definieren, in welchen Zeitlichen Abschnitten die Daten dargestellt werden sollen. Danach dokumentierte ich das komplette Frontend und schloss somit die Phase Realisieren ab. Anschliessend installierte ich das Endprodukt auf dem Server und testete die Webseite über das Intranet. Ich konnte die Phase Kontrollieren dokumentieren und auch gleich abschliessen.

### Erreichte Ziele / Erfolge

* Sensor Daten Zeitlicher Abschnitt
* Realisieren Meilenstein
* Kontrollieren Meilenstein

### Aufgetretene Probleme / Misserfolge

* Ich hatte am Morgen ein Problem mit Chrome, wobei Netzwerk Aufrufe nach Refresh des Browsers nicht komplett angezeigt werden. Die Lösung hierbei war den Browser nicht neuzuladen.

### Beanspruchte Hilfestellungen

Ich habe keine Hilfestellungen beansprucht.

### Pendenzenliste

Ich habe keine offenen Pendenzen.

### Reflexion

Der heutige Tag verlief sehr gut. Ich konnte gleich zwei Phasen ohne Probleme abschliessen. Die Phase Kontrollieren dauerte kürzer als erwartet. Ich hoffe dies liegt nicht an schlechten Tests. Meine Arbeitsweise war heute sehr gut, denn ich hatte einen Motivationsschub.

### Vergleich mit dem Zeitplan

Ich konnte den Zeitplan wieder aufholen und bin für Morgen nicht mehr hinterher. Dies liegt daran, dass ich sehr viel Zeit für die Phase Kontrollieren eingeplant hatte.

## Arbeitsjournal 08.04.2020

### Ausgeführte Arbeiten

Heute Morgen startete ich den Tag mit meinem eigenen Fazit und schloss danach die Phase Auswerten ab. Danach schrieb ich noch das Management Summary und war mit der Dokumentation eigentlich fertig. Es galt danach nur noch das Arbeitsjournal und den Anhang einzufügen. Nachdem dieses getätigt war, las ich die komplette Dokumentation nochmals durch und korrigierte meinen Text. Den Tag und die IPA schloss ich mit dem Hochladen der Dokumente auf pkorg ab.

### Erreichte Ziele / Erfolge

* Auswerten Phase fertig
* Dokumentation fertig
* IPA fertig

### Aufgetretene Probleme / Misserfolge

Ich hatte heute keine Probleme.

### Beanspruchte Hilfestellungen

Ich habe keine Hilfestellungen beansprucht.

### Pendenzenliste

Ich habe, da die IPA fertig ist, keine offenen Pendenzen mehr.

### Reflexion

Der heutige Tag verlief gut, auch wenn er stressig war, hatte ich nirgends Probleme. Sogar Word arbeitete nicht gegen mich. Ich konnte somit die IPA mit einem guten Gefühl abschliessen und bin sehr zufrieden mit der Arbeit.

### Vergleich mit dem Zeitplan

Ich konnte die IPA ein bisschen früher abschliessen als geplant und hatte somit viel Zeit für das Korrigieren und konnte nochmals die Kriterien durchgehen.

# Organisation der Arbeitsergebnisse

Die Arbeitsergebnisse werden mit dem mit GIT versioniert und auf github hochgeladen. So kann bei lokalem Datenverlust auf jede Version zurückgegriffen werden.

Jeder Commit ist mit einem einzelnen Kommentar versehen, damit ist es einfach jede Version wiederzufinden.

Alle Arbeitsergebnisse werden mindestens einmal pro Stunde hochgeladen.

Die Wiederstellung der Dokumente ist durch das Hochladen auf eine Cloud, in meinem Fall github, sichergestellt.

Die Arbeitsergebnisse werden die ganze IPA hindurch dieselbe Qualität haben. Dies wird durch wiederverwendung von Code sichergestellt.

Der IPA Bericht wird in Word verfasst und letztendlich zu einer PDF Datei konvertiert. Mein Python Code wird mit Inline Kommentaren und [Sphinx](https://www.sphinx-doc.org/en/master/) dokumentiert, und schlussendlich als Webseite im Anhang vorhanden sein.

Teil 2: Projekt-Dokumentation

# Management Summary

## Ausgangslage

Die BBB hat nach ihrem ersten erfolgreichen Projekt bei der AVECTRIS, nämlich ihre [Homepage](https://www.bbbaden.ch/), eine zweite Plattform angefragt. Sie werden ihr Schulhaus mit Sensoren ausstatten und möchten die Daten dieser gerne auf einer Webseite darstellen. Die Sensoren sollen das Reinigungspersonal modernisieren, es soll weniger unnötig geputzt werden. Als Beispiel werden Zählerstand Sensoren an die WC’s installiert, danach soll nach X Meldungen des Sensors eine Meldung auf der Webseite ausgehen. Das Reinigungspersonal soll diese Meldungen sortiert nach Raum, Stockwerk und Gebäude sehen, damit sie strukturiert das Gebäude putzen können. Zusätzlich soll die Webseite dazu fähig sein, dem Reinigungspersonal eine Materialliste zu generieren, je nachdem welche Meldungen existieren.

## Vorgehen

Das Projekt startete mit der Erstellung von sogenannten Mockups, Skizzen der Webseite. Dabei wurde klar, dass es zwei Rollen geben wird. Die Administratoren und das Reinigungspersonal. Diese sollen zwei unterschiedliche Ansichten haben. Die Administratoren haben im lediglich eine Tabelle mit allen Meldungen in chronologischer Ordnung mit der neusten zuoberst. Das Reinigungspersonal hat alle Meldungen sortiert nach Zimmer, Stockwerk und Gebäude in eigenen Tabelle, zuoberst soll dann auch noch die Materialliste sein. Danach wurde das komplette Backend erstellt. Dieses ist verantwortlich für das Erstellen, Aktualisieren, Löschen und Auflisten von Modellen. Nachdem dieses fertig war, wurde das Frontend geschrieben. Das Produkt musste danach kontrolliert werden mit den in der Planungsphase geschriebenen Tests. Diese wurden alle überstanden.

## Ergebnis

Die Applikation kann ohne Probleme verwendet werden und die Visualisierung der Sensordaten ist spannend für das Auge. Die Anwendung ist ein kleines Spielzeug. Ob Sie für das Reinigungspersonal nützlich ist, muss nach der Einführung ausgewertet und falls nötig verbessert werden. Allgemein erfüllt die Applikation die Anforderungen und ist für das Tablet des Reinigunspersonales optimiert.

Der Zeitplan konnte grösstenteils eingehalten werden. Die grösste Abweichung war während der Implementierung des Frontend.

# Informieren

## Ziel

Das Ziel der BBB ist, eine Webseite zu erstellen, welche dem Reinigungspersonal hilft zu wissen, wo und wann geputzt werden muss. Beim Pilotenversuch geht es primär darum Sensor Werte darzustellen und auf diese zu reagieren. Zum Beispiel soll eine Meldung im Dashboard erscheinen, wenn ein gewisser CO2 Wert überschritten wird. Diese Meldung soll anschliessend vom Reinigungspersonal abgearbeitet werden. Hierbei wäre damit Fenster öffnen gemeint.

## Absicht

Die Webapplikation soll dabei unterstützen, dass zielgerichteter geputzt werden kann. Somit kann im Idealfall erreicht werden, dass die Berufsbildung Baden weniger Reinigungspersonal anstellen muss.

## Anforderungen

* Responsive Design
* Browserkompatibilität
* Einhaltung Corporate Design
* Quellcode Dokumentation
* Validierung der Eingaben von Benutzern
* Anzeige der Luftqualitäts-Messwerte
* Grafische Darstellung der Messwerte

## Ressourcen

Die Entwicklung entsteht auf einem normalen Windows-10 PC und für die endgültige Applikation steht ein interner Windows Server zur Verfügung.

Zusätzlich sind für die Messung von Daten drei verschiedene Arten von Sensoren in Betrieb.

## Deadline

Die Applikation soll bis und mit dem 08.04.2020 fertig sein.

## Herausforderungen

### JavaScript Diagramme

Die Sensordaten müssen in einem Diagramm mit den Wert- und Datum Achsen dargestellt werden können. Ich stelle mir dies anhand einer JavaScript Bibliothek vor. Ich kenne noch keine und muss mir eine Liste mit Google erforschen.

Zwei Bibliotheken, welche im Netz immer wieder vorkommen sind Chart.js und Chartist.js. Beide Bibliotheken sind gratis und open source.

* [Chart.js](https://www.chartjs.org/)

Als ich ein einfaches Beispiel für Chart.js erstellte, fiel mir den Anfang sehr schwer, da es schwierig zu sein scheint, die Diagramm Grösse zu verändern. Die Lösung hierzu war es, das Diagramm in einem HTML Div zu verpacken und die Grösse des Div zu setzen. Somit lässt sich mit folgendem Code ein Diagramm für einen Sensor erstellen. Der Code wurde von der [offiziellen Dokumentation](https://www.chartjs.org/docs/latest/charts/line.html) entnommen.



Abbildung 2 Chart.js Resultat des Beispielcodes

Abbildung 3 Chart.js Code für die Darstellung eines Liniendiagramms

* [Chartist.js](https://gionkunz.github.io/chartist-js/)

Chartist konnte ich ohne weitere Probleme sehr schnell ein Beispiel mit den gleichen Daten wie oben implementieren. Es scheint minimiert auf die meist benutzen Diagramme zu sein.



Abbildung 4 Chartist.js Resultat des Beispielcodse

Abbildung 5 Chartis.js Code für die Darstellung eines Liniendiagramms

### Sensor Wert Beobachter

Ein Sensor kann mehrere Werte messen, auf eine Weise müssen Benutzer definieren können, auf welche Werte geachtet werden müssen und wann ein Richtwert oder Zählerstand überschritten wurde. Dies möchte ich möglichst erweiterbar lösen. Jedoch komme ich vermutlich nicht darum, dass bei jeder neuen Art von Sensor ein neuer Wert Dekodierung Algorithmus von einem Programmierer geschrieben werden muss. Die Werte eines Sensors werden meist in Hex übertragen und müssen dann Anhand der Datenblätter ausgelesen werden.

Folgende Lösung kann ich mir vorstellen:

Ein Sensor Wert wird erhalten und anhand des passenden Sensor Dekodierers in die verschiedenen Werte umgewandelt. Danach wird bei jedem der einzelnen Werte geprüft, ob dieser seinen Richtwert oder Zählerstand überschritten hat. Wenn ja, wird eine spezifische Meldung für jenen Wert ausgelöst.



Abbildung 6 Prototyp Algorithmus für die Dekodierung und Meldungserstellung eines Sensorwertes.

### Responsive Design

Auch mit Bootstrap ist ein Responsive Design schwierig korrekt umzusetzen. Zudem werde ich die SCSS Dateien von Bootstrap so ändern müssen, dass sie dem Corporate Design der BBB angeglichen sind. Das Reinigungspersonal wird sich mit Tablets im Gebäude herumgeben, daher ist es für mich aus sehr wichtig, dass das Frontend für Touch-Steuerung optimiert ist. Dieser Sachverhalt ist mir vorgängig nicht bekannt und werde vermutlich meine Browser Ansicht auf die Grösse eines Tablets ändern, damit ich ständig das Resultat sehe, welches schlussendlich von den Auftragsgebern benützt wird. In Chromium basierenden Browsers kann dies mit der F12 Taste erreicht werden. 

Abbildung 7 Chromium Entwickler Modus Tablet Ansicht aktivieren



Abbildung 8 Chromium Responsive Modus Menu

### Sphinx Dokumentation

Python lässt sich anhand von sogenannten Docstrings dokumentieren. Es ist möglich, die Docstrings mit Sphinx zu einer Webseite generieren zu lassen. Meine Herausforderung hier ist, dass ich diese gründlich nachführe und gleich nach der Erstellung einer Funktion diese auch wirklich schreibe.

Die Herausforderung ist, dass meine Docstrings dem Standard von Google folgen, und entsprechend aussehen sollten. Dies ist wichtig, damit der nächste Programmierer, welcher diese Projekt übernimmt, sich schnell in das Backend einarbeiten kann.



Abbildung 9 Beispiel eines Google Style Docstrings

### HTML Präprozessor

Damit ich nicht mehrmals das gleiche HTML, wie der Header, schreiben werde, muss ich mir einen HTML Präprozessor beibringen, welcher fähig ist, Templates in Seiten einzubinden und daraus eine statische Seite zu erstellen. Vor der IPA habe ich mir deswegen [Jekyll](https://jekyllrb.com/) evaluiert. Dieses Tool bietet die Features welche wichtig sind beim Aufbau einer Seite.

### Tür Schaltungssensor Wert

Die Werte des Adeunis RF und der Elsys CO2 Sensoren konnte ich bereits als Vorarbeit dekodieren.

Gemäss diesem [Datenblatt](https://iot-shop.de/wp-content/uploads/2020/03/RM_Door-_-Window-Sensor_20200205_v2.pdf) besteht der Wert eines Tabs Sensors aus 8 bytes.

Tabelle 11 Bedeutung eines Wertes eines Tabs Sensors

|  |  |  |  |
| --- | --- | --- | --- |
| Wert | Byte | Bits | Beschreibung |
| Status | 0 | [0] | 1 – offen, 0 – geschlossen |
| Batterie | 1 | [3:0] | Bereich 1 – 14  Batterie Spannung in V = (25 + ν) ÷ 10. |
| Temperatur (PCB) | 2 | [6:0] | Bereich 1-127  Für Grad noch -32 rechnen. |
| Zeit | 3, 4 | [15:0] | Bereich 0 - 65535 |
| Zählstand | 5, 6, 7 | [23:0] | Bereich 0 - 16777215 |

# Planen

## Use Case



Abbildung 10 Use Case für die Plattform

## System

### Allgemein

Das System ist fogend aufgebaut: Ein Benutzer lädt zuerst die statische Webseite von dem IIS Server und anschliessend alle Daten von der API, welche von [Uvicorn](https://www.uvicorn.org/) gehostet wird, holt. Beide Dienste werden auf dem internen Server SI1010023 gehostet. Ein solcher Vorgang erlaubt eine angenehmere Benutzung der Webseite, denn es muss nicht auf zum Beispiel PHP gewartet werden, welches alle Daten in das HTML zuerst renderet. Bei meiner Aufbau des Systems, wird zuerst eine Ladeanimation dargestellt während asynchronos die Daten von der API geladen werden.

Die wichtigsten Komponenten des Systems ist der Uvicorn Server, die PostgreSQL Datenbank und das Skript, welches von Loriot die Sensor Daten erhält. Der IIS könnte durch einen Apache oder Nginx Server ausgetauscht werden. IIS ist jedoch Standard auf Windows Servern.

### Uvicorn Server

Der Uvicorn Server erhält alle Anfragen eines Benutzers und leitet diese weiter an dass von mir erstellte Backend, dieses verarbeitet die Anfrage, lädt üblicherweise ein paar Daten von der Datenbank und gibt letztendlich eine Antwort zurück. Diese Antwort wird vom Uvicorn Server dem Benutzer zurückgeschickt.

### Websockets

Loriot ist eine Webseite, welche alle Daten der angehängten Sensoren sammelt. Es kann durch Websockets mit diesem Verbinden und mithören, wenn neue Sensordaten eingefangen werden. Diese Daten werden dann gleich in die Datenbank geschrieben.

### PostgreSQL Datenbank

PostgreSQL ist ein leistungsstarkes, objektrelationales Open-Source-Datenbanksystem. Diese Datenbank ist Zuverlässig und Leistungsstark.

### Visuell

Das komplette System ist unten visuell dargestellt.



Abbildung 11 System Aufbau

## Sensoren

Für den Pilot Versuch können drei verschiedene Sensoren benutzt werden.

* [Adeunis RF](https://www.adeunis.com/en/produit/ftd-network-tester/)
* [Elsys ERS CO2](https://www.elsys.se/shop/product/ers-co2/?v=1ee0bf89c5d1)
* [Tabs Tür Sensor](https://iot-shop.de/produkt/tuer-und-fenstersensor)

Für jeden dieser Sensoren muss je ein Dekodierung Algorithmus geschrieben werden. Für den Adeunis RF und Elsys ERS CO2 Sensor wurde der Algorithmus im Vorfeld geschrieben.

Ein Sensor muss jeweils von einem Skript dekodiert werden. Damit dies geschehen kann ist eine Verlinkung von Sensor zu Skript benötigt. Für die Webseite ist es wichtig, dass Benutzer diese Verlinkungen erstellen können. Für die Umsetzung dieser Verlinkung ist die eindeutige Sensor EUI geeignet.

## Wert Beobachter

Jedem Sensor sollen mehrere Wert Beobachter angehängt werden können. Dies ist nötig, da ein Sensor mehrere Werte messen kann.

Es soll zwei Arten von Wert Beobachtern geben.

1. Richtwert Beobachter

Beim Richtwert wird eine Meldung ausgegeben, wenn der Sensor diesen Richtwert überschreitet. Als Beispiel einen Temperatur Sensor: wird der Richtwert auf 22 Grad gesetzt und der Sensor meldet eine Temperatur von 24, dann würde eine Meldung ausgehen.

1. Zählerstand Beobachter

Beim Zählerstand soll gemessen werden wie oft etwas gemacht wurde. Als Beispiel eignet sich dies für die Messung wie oft eine Toilette benutzt wurde. Als Folge könnte nach 10 Benutzungen eine Meldung ausgeben.

## Authentifizierung

Die Authentifizierung wird per Access und Refresh Tokens funktionieren. Nachdem ein Benutzer mit seinen Login Daten sich einloggte, erhält er je eine der beiden genannten Tokens.

**Access Token**

Dieser Schlüssel hat eine kurze Lebensdauer (15 – 30 min) und beinhaltet Informationen über den Benutzer, diese Informationen können die Rolle und ID sein. Der Token hat eine Signatur angehängt, welche versichert, dass dieser nicht verändert wurde. Bei jedem Aufruf auf die API wird dieser Token mitgeschickt und auf Alter und Signatur geprüft. Der Vorteil ist, dass die API nicht bei jedem Aufruf eine Datenbank Abfrage durchführen muss, sondern direkt die Rolle aus dem Token herauslesen kann.

**Refresh Token**

Der Refresh Token ist ein kryptografisch sicherer zufällig generierter String. Dieser kann dazu verwendet werden, einen neuen Access Token anzufordern. Bei jeder Anfrage für einen neuen Access Token sollte die Aktivität des Benutzers auf verdächtige Aktionen überprüft werden. Ein Refresh Token ist gleich viel Wert wie ein Login und muss daher sicher gespeichert werden.

Die Authentifizierung hat somit folgenden Ablauf. Das Diagramm ist von links nach rechts zu lesen.



Abbildung 12 Ablauf für die Authentifizierung während der Verwendung der Webseite

## Frontend

### Admin Dashboard

In der Admin Dashboard Ansicht sollen Administratoren eine Übersicht sehen, welche Räume welche Meldungen generiert haben. Eine Meldung kann automatisch von einem Wert Beobachter ausgelöst werden oder auch Rückmeldungen des Reinigungspersonals sein. Alle Meldungen eines Raums können als gelöst markieren, was als Effekt hat, dass diese in eine andere Liste verschoben werden und nicht mehr auf dem Dashboard erscheinen, oder gelöscht werden, was dazu führt, dass die Meldung aus der Datenbank gelöscht wird.



Abbildung 13 Admin Dashboard Skizze

### Personal Verwaltung

Auf dieser Seite können Administratoren das Personal verwalten. Benutzer können entweder die Rolle Personal oder Admin haben.



Abbildung 14 Personal Seite Skizze

### Räume Verwaltung

In der Räume Verwaltung kann das Gebäude hinterlegt werden. Räume werden in Stockwerken hinterlegt, welche in einem Gebäude erstellt wurden. Mit dieser Variante, gibt es die Möglichkeit, in der Zukunft, die Applikation auf mehrere Gebäude zu verteilen.



Abbildung 15 Räume Verwaltung Skizze

### Raum Ansicht

In der Raum Ansicht können alle Sensoren und Meldungen eines Raums eingesehen werden.

Abbildung 16 Raum Ansicht Skizze

### Sensoren Verwaltung

Beim Erstellen eines physischen Sensors muss unbedingt eine EUI und Art des Sensors mitgegeben werden. Die Art des Sensors bestimmt welcher Dekodierung Algorithmus verwendet werden soll. Zudem soll dem Sensor mehrere Wert Beobachter angehängt werden können.



Abbildung 17 Sensoren Verwaltung Skizze

## Sensor Ansicht

In der Sensor Ansicht können die Wertbeobachter verändert werden oder die Sensordaten eingesehen werden.



Abbildung 18 Sensor Ansicht Skizze

### Reinigungspersonal Dashboard

Der Sinn und Zweck dieses Dashboards ist, dass das Reinigungspersonal eine Route als Vorschlag erhaltet. Die Räume mit Meldungen sind nach Gebäude und Stockwerk aufgelistet. Nachdem ein Raum geputzt wurde, kann er als „gelöst“ markiert werden. Anschliessend verschwinden die Meldungen in eine separate Liste. Wenn gelöst geklickt wird, soll gleich gefragt werden, ob noch eine Rückmeldung gegeben werden möchte. Eine solches Feedback kann Schadensmeldungen beinhalten oder ein simples ok.

Damit das Reinigungspersonal auch weiss, wie viel Material mitgenommen werden soll, wird zuoberst auf der Seite noch eine vorgeschlagene Materialliste angezeigt. Diese soll Anhand der offenen Meldungen berechnet werden.



Abbildung 19 Dashboard Reinigungspersonal Skizze

## Konzeptionelles Modell



Abbildung 20 Konzeptionelles Modell UML

## Logisches Modell

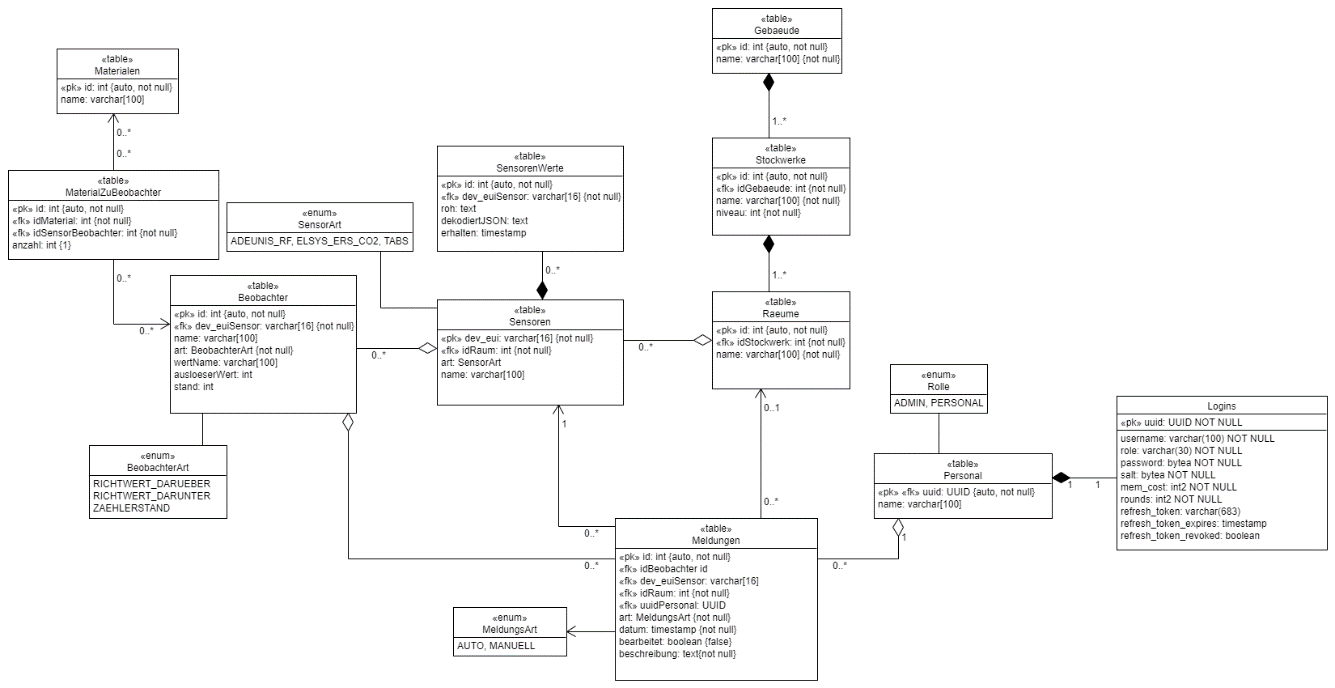


Abbildung 21 Logisches Modell UML

## API Struktur

Die API Struktur definiert auf welchen URLs, mit welchen http Methoden, welche Aktion ausgeführt wird.

Die http Methoden sind GET, POST, PUT und DELETE.

* GET wird für das Abrufen von Daten verwendet
* POST ist für die Erstellung von Daten
* PUT ist für die Aktualisierung von Daten
* DELETE ist für die Endgültige Löschung von Daten

Tabelle 12 API Routen mit beschriebener Aktion welche diese ausführen.

|  |  |  |
| --- | --- | --- |
| Pfad | Methode | Aktion |
| /login | POST | Entnimmt Login Daten aus dem Request Body und antwortet mit dem Access Token und Refresh Token. |
| /login | PUT | Entnimmt Refresh Token aus dem Request Body und antwortet mit aktualisiertem Access Token. |
| /login | DELETE | Macht den Refresh Token des Benutzers ungültig. |
| /personal | GET | Auflistung des Personals. |
| /personal | POST | Erstellt eine neue Person und Login aus den Daten des Request Body. |
| /personal/<id> | PUT | Aktualisiert Person mit den Daten des Request Body. |
| /personal/<id> | DELETE | Löscht Person aus der Datenbank. Alle Meldungen dieser Person werden somit auch gelöscht. |
| /gebaeude | GET | Auflistung der Gebäude. |
| /gebaude | POST | Erstellung eines Gebäude. |
| /gebaude/<id> | PUT | Aktualisierung eines Gebäude. |
| /gebaude/<id> | DELETE | Löschung eines Gebäude. |
| /stockwerke | GET | Auflistung der Stockwerke. |
| /stockwerke | POST | Erstellung eines Stockwerks. |
| /stockwerke /<id> | PUT | Aktualisierung eines Stockwerks. |
| /stockwerke /<id> | DELETE | Löschung eines Stockwerks. |
| /raume | GET | Auflistung der Räume. |
| /raume | POST | Erstellung eines Raums. |
| /raume /<id> | PUT | Aktualisierung eines Raums. |
| /raume /<id> | DELETE | Löschung eines Raums. |
| /raume/<id>/sensoren | GET | Auflistung aller Sensoren eines Raumes. |
| /raume/<id>/meldungen | GET | Auflistung aller Meldungen eines Raumes. |
| /sensoren | POST | Erstellung eines Sensors für den Raum mit ID. |
| /sensoren/<eui> | PUT | Aktualisierung eines Sensors. |
| /sensoren/<eui> | DELETE | Löschung eines Sensors. |
| /sensoren/<eui> | GET | Auflistung aller Werte eines Sensors. |
| /sensoren/<eui>/beobachter | GET | Auflistung aller Beobachter eines Sensors. |
| /beobachter | POST | Erstellung eines Beobachter für einen Sensor. |
| /beobachter/<id> | PUT | Aktualisierung eines Beobachters. |
| /beobachter/<id> | DELETE | Löscht einen Beobachter. |
| /sensoren/<eui>/meldungen | GET | Auflistung aller Meldungen eines Sensors. |
| /beobachter/<id>/meldungen | GET | Auflistung aller Meldungen eines spezifischen Beobachters. |
| /beobachter/<id>/meldungen | POST | Manuelle Auslösung einer Meldung. |
| /materialien | GET | Auflistung aller Materialien. |
| /materialien | POST | Erstellung eines Material. |
| /materialien/<id> | PUT | Aktualisierung eines Material. |
| /materialien/<id> | DELETE | Löschung von Material. |
| /beobachter/<id>/maeterialien | POST | Material zu Beobachter hinzufügen. |
| /beobachter/materialien/<id> | DELETE | Entfernen von Material von Beobachtern. |

## Sicherheit

### Feld Validierung

Allgemein, müssen alle Benutzer Eingabefelder auf Inhalt und Länge validiert werden. Damit es Benutzer nicht möglich ist, Cross Site Skripting zu missbrauchen, muss der Inhalt eines Feldes von HTML Tags bereinigt werden.

### SQL Injection

SQL Abfragen enthalten keine Rohdaten der Benutzer. Die Werte für die Abfrage werden separat der Datenbank geschickt und von PostgreSQL überprüft. Somit ist es nicht möglich, Eingabefelder zu missbrauchen.

### Cross Site Scripting

Damit Cross Site Scripting nicht möglich ist, wird der Meta Tag Content-Security-Policy konfiguriert, dass keine inline Scripts und nur JavaScript Files von derselben Domäne erlaubt sind. Der einzige Nachteil ist, dass keinen CDN für JQuery oder dergleichen benutzt werden darf. Damit wird verhindert das Schadcode von einem gehackten CDN auf die eigene Seite geladen wird.

## Testen

### Testmethoden

Beim Testen wird zwischen Unit Tests und Integration Tests unterschieden. Alle unten aufgeführten Tests sind Integration Tests. Gut zu wissen ist, dass Integration Tests prüfen mehrere verknüpfte Funktionen auf einmal, während Unit Tests nur eine einzige Methode testet. Die Unit Tests sind eher für das Backend und sollen für alle Public Methoden geschrieben werden.

### Testkonzept

Tabelle 13 Testkonzept mit Nummer und Gebiet.

|  |  |
| --- | --- |
| Testnummer | Testgebiet |
| 1 | Authentifizierung |
| 2 | Sensor Wert Dekodierung |
| 3 | Sensor Wert Beobachter |
| 4 | Gebäude Verwaltung |
| 5 | Eingabe Validierung |
| 6 | Design und Browserkompatibilität |

### Testfallspezifikationen

#### Authentifizierung

Tabelle 14 Testfallspezifikationen für die Authentifizierung

|  |  |  |  |
| --- | --- | --- | --- |
| Testfallnummer | Beschreibung | Voraussetzungen | Eingabe |
| 1.1 | Meldet sich ein Benutzer an, so erhält er eine Access Token und Refresh Token. | Das Backend und Frontend muss gestartet sein. Login Seite muss geöffnet sein. | Korrekter Benutzername und Passwort eingeben. |
| 1.2 | Sendet der Benutzer ein Refresh Token, so erhält er einen frischen Access Token. | Das Backend und Frontend muss gestartet sein. Erstes Login muss bereits getätigt sein. Warten auf Ablauf des ersten Access Token. | Neuer Aufruf auf die API. Es wird danach automatisch erkannt, dass der Refresh Token gesendet werden muss. |
| 1.3 | Bei invaliden Anmelde daten ergibt sich einen Fehler. | Das Backend und Frontend muss gestartet sein. Login Seite muss geöffnet sein. | Falscher Benutzername und falsches Passwort eingeben. |
| 1.4 | Bei falschem Refresh Token ergibt sich einen Fehler. | Das Backend und Frontend muss gestartet sein. Benutzer muss eingeloggt sein. Refresh Token muss fälschlicherweise ausgetauscht werden | Aufruf auf die API ausführen. |

#### Sensorwert Dekodierung

Tabelle 15 Testfallspezifikationen für die Sensorwert Dekodierung

|  |  |  |  |
| --- | --- | --- | --- |
| Testfallnummer | Beschreibung | Voraussetzungen | Eingabe |
| 2.1 | Bei Erhalt eines Sensor Wertes wird die richtige Dekodierung ausgewählt und der Wert in JSON umgewandelt. | Loriot Gateway aktiv und verbunden mit Applikation über Websockets. Datenbank geöffnet (pgAdmin). | Druck auf Button des Adeunis RF Sensors. |
| 2.2 | Ein Wert des Adeunis RF Sensors wird korrekt in folgende Werte umgewandelt.   * Temperatur vorhanden * Übertragung durch Klick ausgelöst * Übertragung durch Bewegung ausgelöst * Temperatur * Batterie Status | Loriot Gateway aktiv und verbunden mit Applikation über Websockets. Datenbank geöffnet (pgAdmin). | Klick auf Adeunis RF. |
| 2.3 | Ein Wert eines Elsys CO2 Sensors wird korrekt in folgende Werte umgewandelt.   * Temperatur * Feuchtigkeit * Licht * Bewegung * CO2 * Batterie Spannung | Loriot Gateway aktiv und verbunden mit Applikation über Websockets. Datenbank geöffnet (pgAdmin). | Warten auf neue Daten des Sensors. Elsys schickt alle 5 Minuten. |
| 2.4 | Ein Wert eines Tabs Sensors wird korrekt in folgende Werte umgewandelt.   * Status * Batterie Spannung | Loriot Gateway aktiv und verbunden mit Applikation über Websockets. Datenbank geöffnet (pgAdmin). | Den Status eines Sensors ändern in dem der Magnet entfernt wird. |

#### Sensorwert Beobachter

Tabelle 16 Testfallspezifikationen für die Sensorwert Beobachter

|  |  |  |  |
| --- | --- | --- | --- |
| Testfallnummer | Beschreibung | Voraussetzungen | Eingabe |
| 3.1 | Es können einem Sensor einen Wert Beobachter zugefügt werden. | Web Applikation mit Backend gestartet. Sensor Werte von Loriot abhören. | Wert Beobachter erstellt. Richtwert oder Zählerstand. |
| 3.2 | Ein Richtwert Beobachter erstellt eine Meldung für den Elsys CO2 Temperatur Wert, wenn sich dieser um 1 Grad erwärmt hat. | Web Applikation mit Backend gestartet. Sensor Werte von Loriot abhören. Elsys CO2 Sensor muss hinzugefügt sein. | Beobachter hinzufügen mit einem Richtwert ein Grad wärmer als die zuletzt gemeldete Wärme. Danach Sensor in mit der Hand erwärmen. |
| 3.3 | Ein Zählerstand Beobachter erstellt eine Meldung wenn der Zählerstand eines Adeunis RF zehn erreicht. | Web Applikation mit Backend gestartet. Sensor Werte von Loriot abhören. Adeunis Sensor hinzugefügt mit Zählerstand Beobachter. | Zehn Mal auf den Knopf des Sensors drücken. |
| 3.4 | Es können mehrere Beobachter für den Elsys CO2 erstellt werden. Einen für die Temperatur und einen für das Licht. | Web Applikation mit Backend gestartet. Sensor Werte von Loriot abhören. Elsys CO2 Sensor hinzugefügt und Richtwert Beobachter für Licht und Temperatur erstellt. | Sensor erwärmen und abdunkeln. |

#### Gebäude Verwaltung

Tabelle 17 Testfallspezifikationen für die Gebäude Verwaltung

|  |  |  |  |
| --- | --- | --- | --- |
| Testfallnummer | Beschreibung | Voraussetzungen | Eingabe |
| 4.1 | Es kann ein Gebäude erstellt werden. | Web Applikation mit Backend gestartet. | Gebäude Name eingeben und erstellen. |
| 4.2 | Es können Stockwerke einem Gebäude hinzugefügt werden. | Web Applikation mit Backend gestartet. Mindestens ein Gebäude erstellt. | Gebäude auswählen und Stockwerk Name, Niveau eingeben. |
| 4.3 | Es können Räume einem Stockwerk hinzugefügt werden. | Web Applikation mit Backend gestartet und mindestens ein Stockwerk bereits erstellt. | Raum Name eingeben und Stockwerk auswählen. |
| 4.4 | Räume werden nach Stockwerk Niveau und Gebäude sortiert. | Web Applikation mit Backend gestartet und Raum Verwaltung geöffnet. | Keine. |

#### Eingabe Validierung

Tabelle 18 Testfallspezifikationen für die Eingabe Validierung

|  |  |  |  |
| --- | --- | --- | --- |
| Testfallnummer | Beschreibung | Voraussetzungen | Eingabe |
| 5.1 | Leere Eingaben werden abgefangen. Leer ist auch ein String der Länge 0 und nicht nur None. | Web Applikation mit Backend gestartet. | Zum Beispiel ein neues Gebäude erstellen mit leerem Name. |
| 5.2 | JavaScript Code Eingaben werden nicht ausführbar in der Datenbank abgespeichert. | Web Applikation mit Backend gestartet. | Neues Gebäude erstellen mit dem Namen <script>alert(«Not Safe»);</script>. |
| 5.3 | Die Eingabe eines Gebäude Namens mit einer Länge von mehr als 100 wird nicht akzeptiert. | Web Applikation mit Backend gestartet. | 101 Charakter String eingeben als Gebäudenamen. |

#### Design und Browserkompatibilität

Tabelle 19 Testfallspezifikationen für das Design und die Browserkompatibilität

|  |  |  |  |
| --- | --- | --- | --- |
| Testfallnummer | Beschreibung | Voraussetzungen | Eingabe |
| 6.1 | Das Design passt sich für den Desktop an. | Applikation auf einem Monitor mit der Grösse 1920x1080 geöffnet. | Keine. |
| 6.2 | Das Design passt sich für das Tablet an. | Applikation auf einem Monitor mit der Grösse 1024x768 geöffnet. | Keine. |
| 6.3 | Die Webseite wird im Edge unterstützt. | Applikation in genannten Browser geöffnet. | Durch die Seite navigieren. Auf fehlende Anzeigen achten. |
| 6.4 | Die Webseite wird im Firefox unterstützt. | Applikation in genannten Browser geöffnet. | Durch die Seite navigieren. Auf fehlende Anzeigen achten. |
| 6.5 | Die Webseite wird im Chrome unterstützt. | Applikation in genannten Browser geöffnet. | Durch die Seite navigieren. Auf fehlende Anzeigen achten. |
| 6.6 | Werden die Farben und Schriften gemäss BBB\_CD\_Manual.pdf eingehalten. | Applikation geöffnet. | Keine. |
| 6.7 | Die Luftqualitäts-Messwerte sollen korrekt dargestellt. | Sensor Ansicht eines Elsys CO2 Sensors geöffnet. | Keine. |
| 6.8 | Verfügt die grafische Darstellung der Messwerte über einen Zeitfilter? | Applikation gestartet. Sensor Ansicht geöffnet. | Zeitfilter betätigen. |

## Vorgehen

Nach dem alle Teilelementen des Projektes bekannt sind, kann das Vorgehen beschrieben werden.

Zuerst wird das Backend vollständig realisiert, dies beinhaltet auch Unit Tests. Folgend werden alle API Aufrufe in JavaScript umgesetzt und letztendlich wird das Frontendumgesetzt, welches mit dem zuvor umgesetzten JavaScript Code sich mit dem Backend verbindet.

# Entscheiden

## Allgemein

Die Entscheidung für die JavaScript Diagramm Bibliothek muss als einziges nocht festgelegt werden.

Es steht nur noch die Entscheidung für die JavaScript Diagramm Bibliothek offen. Die Optionen sind Chart.js und Chartist.js. Abgesehen von der Vielfältigkeit haben die zwei Bibliotheken keine grösseren Unterschiede. Als Entscheidungskriterium zwischen Chart.js und Chartist.js wird *daher* wird eine Nutzwertanalyse erstellt.

## Kriterien

Die zu bewertenden Aspekte sind:

* Einfachheit der Anwendung

Je einfacher die Bibliothek anzuwenden ist, desto schneller werde ich vorankommen.

* Unterstützt Darstellung von verschiedenen Werten in einem

Da ein Sensor nicht nur einen Wert misst, wäre es von Vorteil, könnte die Bibliothek mehrere Werte auf einem Diagramm darstellen.

* Vielfältigkeit

Je mehr «Out of the box» funktioniert, desto weniger muss selbst geschrieben werden.

* Grösse der Bibliothek

Je weniger eine Webseite laden muss, desto schneller wird sie. Da sich aber solche Bibliotheken cachen lassen, spielt dies keine grosse Rolle.

* Browserkompatibilität

Chrome, Firefox und Edge müssen unterstützt sein.

* Responsiveness

Ein Muss für Tablets.

## Nutzwertanalyse



Abbildung 22 Nutzwertanalyse für die Entscheidung zwischen Chart.js und Chartist.js

Aus der Nutzwertanalyse ergibt sich das Chart.js der Sieger ist. Mit Chart.js hatte ich am Anfang Probleme mit dem konfigurieren der Grösse eines Diagramms, da diese Bibliothek jedoch einiges mehr bietet als Chartist.js und mir somit Aufwand spart, wird Chart.js die richtige Option sein.

# Realisieren

## Backend

### Backend Aufsetzen

#### Quellcode

Der Quellcode lässt sich über das GIT Repository installieren. Der Quellcode befindet sich unter realisieren/backend.

#### Datenbank

Falls PostgreSQL nicht installiert ist, sollte dies zusätzlich noch heruntergeladen werden. Damit die Datenbank angeschaut werden kann, sollte PgAdmin zusätzlich installiert werden.

Danach habe ich das «db\_setup.py» Skript im Repository unter realisieren/backend/ erstellt, mit welchem die Datenbank gemäss dem logischen Modell aufgebaut wird. Bevor dieses Skript jedoch ausgeführt wird, muss die Datenbank „bbb“ über PgAdmin erstellt werden.

#### Python Packages

Für das ausführen des Backend müssen noch Python Bibliotheken installiert werden. Als erstes muss das von mir entwickelte Framework installiert werden. Mit Git lässt sich das Repository von <https://github.com/nilsegger/tedious.git> clonen. Danach muss im neu erstellten tedious Ordner folgende Befehle ausgeführt werden.

* Python setup.py sdist && pip install dist/tedious-1.0.tar.gz

Danach müssen nur noch alle zusätzlich benützten Packages installiert werden. Dafür müssen die Befehle im Projekt Ordner unter /realisieren/backend/ ausgeführt werden.

* python setup.py sdist
* pip install dist/bbbapi-1.0.tar.gz

#### Server starten

Letztendlich wird der Server mit ‘uvicorn main:app’ im /realisieren/backend Ordner gestartet.

Damit die Authentifizierung reibungslos funktioniert, müssen die Public und Private Keys im RSA Format erstellt werden. Diese werden gebraucht, um die Signatur eines Access Tokens zu erstellen und validieren. Zusätzlich müssen die Dateinamen des Public und Private Keys identisch sein und eine Länge welche durch sechzehn teilbar ist haben. Dies ist, weil im Header eines Access Tokens die Public Key Kennung vorhanden sein muss, damit das Backend weiss mit welchem Schlüssel die Signatur validiert werden kann. Im config.ini File sind die Schlüssel unter assets/keys/public und assets/keys/private verlinkt. Mehr über die Konfiguration folgt gleich im Kapitel Konfiguration.

### Konfiguration

Das Backend lässt sich minimal konfigurieren, die Werte und ihre Auswirkung sind in folgender Tabelle beschrieben.

Das Format von config.ini sieht folgenerweise aus:

[TITEL]  
name=wert   
[KEYS]   
private-keys=assets/keys/private

#### Keys

Tabelle 20 Keys Konfiguration

|  |  |
| --- | --- |
| Name | Beschreibung |
| private-keys | Pfad zu den privaten RSA Schlüsseln. |
| public-keys | Pfad zu den passenden RSA öffentlichen Schlüsseln. |
| Identifier-secret | 32 Charakter langer String welcher den Dateinamen von privaten Schlüsseln verschlüsselt damit er ohne Erkennungsproblem in den Header eines Access Token geschrieben werden kann. |

#### Token

Tabelle 21 Token Konfiguration

|  |  |
| --- | --- |
| Name | Beschreibung |
| expire | Zeit in Sekunden bis ein Access Token nach Erstellung abläuft. |
| issuer | Domain des Servers welcher den Access Token erstellt hat. |
| audience | Domain des Servers für welchen einen Access Token anwendbar ist. |
| refresh-token-bytes | Grösse des zu erstellenden Refresh Token in bytes. |

#### Scrypt

Scrypt ist der Algorithmus mit welchem die Passwörter gehashed werden.

Tabelle 22 Scrypt Hash Algorithmus Konfiguration

|  |  |
| --- | --- |
| Name | Beschreibung |
| mem\_cost | Wie viel Memory die Generierung des Hashes benutzen sollte. Hierbei sollte ein Wert von 16 genügen. |
| rounds | Wie oft gehashed werden soll. Hierbei sollte ein Wert von 8 genügen. |

#### DB\_CREDENTIALS

Tabelle 23 Datenbank Verbindung Konfiguration

|  |  |
| --- | --- |
| Name | Beschreibung |
| database | Name der Datenbank. |
| user | Benutzername des Datenbank Benutzers. |
| password | Passwort des Datenbank Benutzers. |

#### DB

Tabelle 24 Datenbank Verbindungsabbau Konfiguration

|  |  |
| --- | --- |
| Name | Beschreibung |
| close-timeout | Zeit in Sekunden wie lange gewartet werden soll bis die Verbindung zur Datenbank terminiert werden soll nach Schliessung des Servers. |

#### ASGI

Tabelle 25 ASGI Server Konfiguration

|  |  |
| --- | --- |
| Name | Beschreibung |
| max-body-size | Grösse in Bytes welche ein Aufruf auf die Datenbank nicht überschreiten darf. |

#### Beobachter

Tabelle 26 Beobachter Konfiguration

|  |  |
| --- | --- |
| Name | Beschreibung |
| timeout | Wert in Sekunden, wie lange Beobachter warten müssen, bis sie eine zweite Meldung ausschreiben dürfen. |

### Tabs Sensor Dekodierung

#### Status

Für den Tabs Sensor war ausschlieslich wichtig zu wissen, ob die Türe offen oder geschlossen ist.

Ein Sensor Wert kann folgenderweise Aussehen ‘00fc35fa25060000’. Dieser steht in Hex. Wenn wir dem Datenblatt folgen, wissen wir, dass das erste Byte den Status wiederspiegelt. Ein Byte hat acht Bits. Ein Charakter in Hex ist ein Nibble und ein Nibble ist 4 Bits. Somit entsprechen die ersten zwei 0 unserem Status.

Ausgeschrieben in Bits entspricht dies 0000 0000, unserem Byte. Wäre die letzte null eine 1, so wäre die Türe offen. Dies sähe so aus 0000 0001 und in einem Sensor Wert Beispiel ‘01fc35fa25060000’.

### Loriot Listener

Der Loriot Listener ist das Skript, welches die Daten der Sensoren von Loriot holt, dekodiert und die angehängten Beobachter des Sensors informiert. Das Skript ist so geschrieben, dass es sich bei Errors nach 5 Sekunden automatisch neu startet. Dies ist eine einfache Vorbeugung, falls die Verbindung zu Loriot unterbrochen wird. Das Skript besteht aus 3 Hauptfunktionen. Main, Listen und Handle.

* **Main** verbindet sich mit der Datenbank und dient als Wrapper für Listen. Passiert in Listen ein Fehler, so wird Listen nach 5 Sekunden wieder erneut aufgerufen.
* **Listen** verbindet sich per Websocket mit Loriot und empfängt die rohen Sensor Daten und gibt diese den passenden Dekodierer. Damit das Skript weiss, welcher Dekodierer verwendet werden soll, muss der Sensor in der Sensoren Tabelle erscheinen. In dieser Tabelle gibt es die Spalte Art, welche aussagt, was für ein Sensor es ist. Der Sensor wird durch die Device EUI identifiziert, diese ist bei den Rohdaten enthalten. Nach der Dekodierung wird der Sensor und Wert der Handle Methode übergeben.
* **Handle** lädt zuerst die Daten in die Sensor Wert Tabelle und alarmiert danach die Beobachter, welche in der Beobachter Tabelle zum Sensor mit passender Device EUI referenziert sind.

Vereinfacht sieht dies folgenderweise aus:



Abbildung 23 Loriot Listener Ablauf

### Modelle

Ein Modell besteht aus Feldern und repräsentiert eine Reihe aus einer Datenbank Tabelle.

Als Beispiel ein Screenshot des Gebäude Modell.

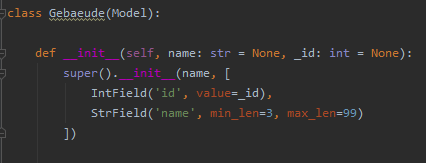


Abbildung 24 Gebäude Modell Code Beispiel

Es besteht aus einer ID und einem Namen, der Name wird beim einlesen von Benutzerdaten auf die Mindestlänge von 3 und Maximum Länge von 99 geprüft. Modelle können auch verschachtelt werden, so wird ein Foreign Key zu einem Modell.

So wurde beim Stockwerk, anstatt von einem IntField für den Fremdschlüssel für idGebaeude ein Gebäude Modell eingefügt.

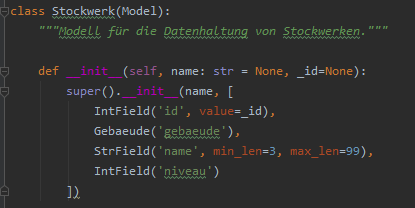


Abbildung 25 Stockwerk Modell Code Beispiel

Daten werden von der Datenbank in der Form eines Dictionary zurückgegeben. Ein Dictionary ist ein Key zu Wert Objekt. Als Keys für diese Dictionary können daher Key Pfade gesetzt werden. Solche sehen wie folgt aus für ein Stockwerk: ‘id’, ‘name’, ‘niveau’, ‘gebaeude.id’ und ‘gebaeude.name’. Einen solchen Pfad kann abgelaufen werden und somit den Wert eines Feldes setzen.

Bei einer Stockwerk Modell Instanz, würde ‚gebaeude.id‘ dem ID Feld des Gebaeude Feldes entsprechen.

Werden die Schlüssel des Antwort Dictionary der Datenbank mit Key Pfaden gesetzt, so werden die Werte automatisch in das Modell eingelesen.

Beispiels Abfrage mit Gelb angestrichenen Werte als Key Pfad.

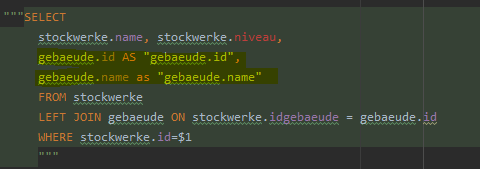


Abbildung 26 Code Beispiel Abfrage mit Key Pfaden

Das Resultat dieser Abfrage würde folgenderweise aussehen und automatisch anhand von einer Methode, welche die Key Pfade abläuft, eingelesen werden

{

‘name’: ‘Stockwerk Name’,

‘niveau’: 0,

‘gebaeude.id’: 0,

‘gebauede.name’: ‘Gebäude Name’

}

**Implementierte Modelle**

* Beobachter
* Gebäude
* Material
* Meldung
* Personal
* Raum
* Sensor
* Stockwerk

### Modell Controller

Ein Model Controller ist zuständig für die Erstellung, Aktualisierung und Löschung von Modellen. Vor jeder Aktion werden die Modells auf Inhalt überprüft.

Die Modell Controller verfügen über die einfachen Methoden Create, Update und Delete.

Als Beispiel die Mehode Create des Personal Controller.

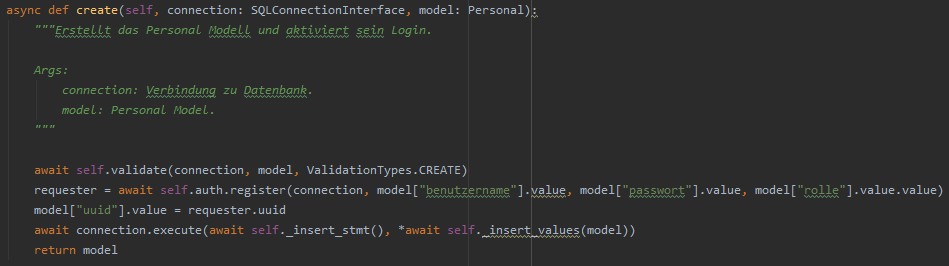


Abbildung 27 Code Beispiel für die Erstellung eines Personal per Controller

**Implementierte Modell Controller**

* Beobachter
* Gebäude
* Material
* Meldung
* Personal
* Raum
* Sensor
* Stockwerk

### Auflistungen

Für die Auflistung von Modellen wird die ListController Klasse benutzt. Diese teilt die Daten in der Tabelle auf, dass nicht alles auf einmal geholt werden muss.

So sieht die Klasse für die Gebäude Auflistung aus.

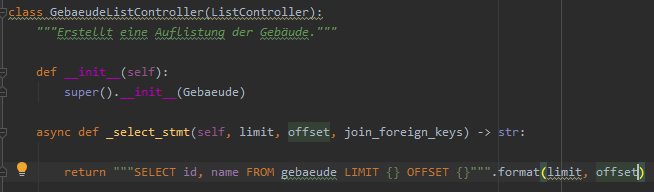


Abbildung 28 Gebäude List Controller Code Beispiel

Die List Controller Klasse wurde mit Hinsicht darauf erstellt, dass nur noch die Methode \_select\_stmt überschrieben werden muss.

**Implementierte Auflistungen**

* Gebäude
* Personal
* Sensoren
* Sensorwerte
* Beobachter eines Sensors
* Stockwerk
* Räume
* Material
* Material für die Meldung eines Beobachters

### Beobachter

Die Beobachter sind dafür zuständig, die Sensor Werte auf gewisse Bedingungen zu prüfen. Damit die Beobachter nicht ständig eine Meldung ausgeben, haben sie ein Timeout von einer Stunde (konfigurierbar), bis sie eine neue Meldung ausgeben können.

#### Richtwert Darüber

Ein Richtwert darüber misst, ob ein gewisser Wert einen definierten Wert überschritten hat. Beispiel:

Der Beobachter misst den Wert Temperatur eines Adeunis Sensors und gibt eine Meldung, wenn dieser den Wert 30 überschritten hat. Wenn der Sensor einen Wert von 31 meldet, wird eine Meldung ausgegeben.

#### Richtwert Darunter

Dieser Beobachter ist einfach das Gegenteil des letzten Beobachters, anstatt dass der Wert darüber liegen muss, muss dieser darunter liegen. Beispiel:

Der Beobachter misst den Wert Temperatur eines Adeunis Sensors und gibt eine Meldung, wenn dieser den Wert von 18 unterschritten hat. Wenn der Sensor einen Wert von 17 meldet, wird eine Meldung ausgegeben.

#### Zählerstand

Der Zählerstand misst wie oft der Sensor sich meldet und inkrementiert seinen Stand, Überschreitet nun der Sensor einen definierten Wert mit seinem Stand, wird eine Meldung ausgegeben und der Stand wird auf null zurückgesetzt. Beispiel:

Der Beobachter zählt den Stand des Adeunis Sensors und soll nach der zehnten Nachricht eine Meldung ausgeben. Hat sich der Sensor zehn Mal gemeldet, und egal welche Werte er meldete, wird eine Meldung ausgegeben.

#### Beobachter mit Sensor verknüpfen

Zuerst muss ein Beobachter Modell erstellt werden. Dabei wird die Sensor Dev EUI auf die des gewünschten Sensors gesetzt und anschliessend wird das Modell mit dem Beobachter Controller in der Datenbank erstellt. Dies könnte folgenderweise aussehen.

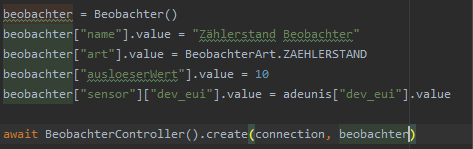


Abbildung 29 Beobachter erstellen Code Beispiel

Mit diesem Zählerstand Beobachter wird nach zehn Nachrichten eines des Adeunis Sensors eine Meldung ausgeben.

#### Beobachter hinzufügen

Möchte ein neuer Beobachter hinzugefügt werden, ist es wichtig, dass zuerst einmal die Enumeration BeobachterArt in common\_types.py erweitert wird. Danach muss die Klasse BeobachterInterface erweitert werden und davon die Methode watch implementiert werden. In dieser Methode werden die Werte des Sensors überprüft und wenn die erwünschten Bedingungen nicht erfüllt werden, wird eine Meldung ausgeschrieben. Schlussendlich muss nur noch im loriot\_listener.py Skript folgender Variable mit einer Instanz des Beobachters erweitern.

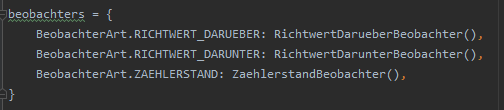


Abbildung 30 Loriot Listener Beobachter Variable

Jedes Mal, wenn ein Sensor einen Wert schickt, werden die Beobachter aus der Beobachter Tabelle mit der richtigen Sensor Dev EUI ausgelesen und anhand von der oben gezeigten Variable und der Spalte ‘art’ in der Tabelle zu den korrekten Instanzen geleitet.

### Resources

Eine Resource nimmt jeweils die Anfrage eines Benutzers entgegen, prüft den Benutzer auf Rolle und bearbeitet die gewünschten Aktionen anhand von Controllern.

Für Modell Controllern und List Controllern gibt es bereits vom benutzen Framework her Resources welche die Benützung dieser vereinfacht.

Als Beispiel wird hier die Resource dargestellt, welche dafür zuständig ist, ein Material einem Beobachter zu zuweisen. Dieses Material wird dann jedes Mal vorgeschlagen, wenn dieser Beobachter eine Meldung ausgibt.

Die Resource hat eine on\_post und on\_delete Methoden, hier wird die on\_post gezeigt.

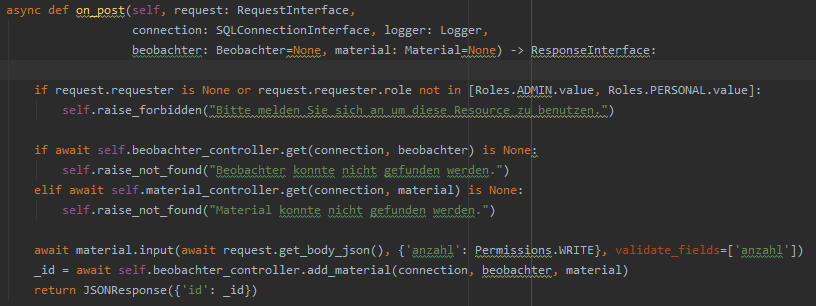


Abbildung 31 on\_post Material zu Beobachter Resource Code Beispiel

Die Resource prüft zuerst, ob der Benutzer, welcher die Anfrage geschickt hat über die notwendigen Rechte besitzt, diese entsprechen hier den Rollen ADMIN und PERSONAL. Danach wird geschaut, dass der Beobachter und das Material existieren und folgend wird noch der JSON im Body der Anfrage in das Material Modell eingelesen. Dabei wird geprüft, dass das Feld Anzahl nicht leer ist. Zu aller Letzt wird der Eintrag in die MaterialZuBeobachter Tabelle anhand des Beobachter Controllers geschrieben. Als Antwort schickt die Resource die ID des neuen Eintrags, damit diese falls gewünscht wieder gelöscht werden kann.

Anhand von solchen Resourcen können alle Anfragen auf das Backend eines Benutzers abgearbeitet werden.

### Benutzer Eingaben Validierung

#### Escaping von HTML

Für die Bereinigung von HTML in Strings wird eine Bibliothek benutzt, nämlich [Bleach](https://github.com/mozilla/bleach) von Mozilla. Mit einer einfachen Methode wandelt Sie mögliche böse Tags in sichere HTML Encodings um.

Beispiel: Ein ‘<’ wird zu ‘&lt;’. Dieses Zeichen wird dann von Browsern nicht als ein Tag angeschaut.

Als Beispiel den Unit Test welche die Bereinigungsfunktion testet.



Abbildung 32 HTML Bereinigungsbeispiel

#### Länge, Not Null Validierungen

Bei einem String muss natürlich geprüft werden, dass dieser nicht null ist. Darum wird vor dem Erstellen eines Modells die Felder noch auf ‘not None’ und mit ‘len(string)’ auf Länge geprüft.

## Frontend

### Auth

Für den Unautorisierten Zugriff auf das Backend wird die «$.ajax» Funktion von JQuery benutzt. Bei einem Status von 200 war das Login erfolgreich, bei 404 existiert die Benutzername- und Passwortkombination nicht und eine Fehlermeldung wird angezeigt. Ansonsten wird von dem Backend erhaltenen Accesstoken die Rolle des Benutzers herausgelesen. Der Accesstoken besteht aus Header, Payload und Signature, für diesen Zweck ist lediglich der Payload wichtig. Dieser ist in Base64 enkodiert und lässt sich mit der von JavaScript eingebauten Funktion «atob» zu einem String umwandeln. Schlussendlich ist der Payload noch JSON und kann anhand von «JSON.parse» in ein Dictionary umgeformt werden. In diesem Dict befindet sich der Key «role», welcher als Wert die Rolle des Benutzers enthält.

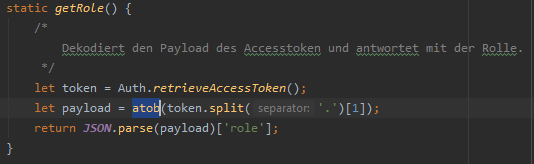


Abbildung 33 Code Beispiel für das Herauslesen der Rolle aus dem Accesstoken.

Anhand der Rolle wird der Benutzer schlussendlich auf die Rolle spezifische Daschboard Ansicht weitergeleitet. Für ADMIN wäre dies dashboard.html und für PERSONAL dashboard\_personal.html.

### Verbindung zum Backend

Bei der Verbindung zum Backend darf der Access Token nicht fehlen. Nach dem Login wurde dieser in der Session Storage abgespeichert. Als Randnotiz ist wichtig zu wissen, dass der Session Storage durch Cross Site Scripting ausgelesen werden kann. Darum ist es sehr elementar, dass sichergestellt ist, dass dies nicht möglich ist.

Das Backend erwartet den Accesstoken im Authorization Header als Bearer Token. Der Code für eine Anfrage an die API sieht folgend aus.



Abbildung 34 API Anfragen Code Beispiel

Die URL ist die natürlich den Link zur Resource welche angefragt werden möchte. Die Methode kann GET, POST, PUT oder DELETE sein. Nachdem die Methode eine Antwort bekommt wird der Callback mit der Response als Parameter aufgerufen. Für POST und PUT kann der Body der Anfrage mit dem data Parameter gesetzt werden. Falls die Anfrage in einem 401 Status (Unauthorized) resultiert, wird versucht, einen frischen Accesstoken anzufragen. Mit dem neuen Autorisierungsschlüssel wird die gleiche Anfrage nochmals versucht, dabei wird für die Methode der Parameter isNested auf Wahr gesetzt. Funktioniert es wieder nicht, wird angenommen, dass die Autorisierungsdaten nicht stimmen und der Benutzer wird ausgeloggt.

Die \_request Methode ist in der Klasse Client, welche noch die statischen Methoden get, post, put und delete anbietet.

### Listen

Eine Auflistung aus dem Backend enthaltet die Werte «list», «is\_end» und «offset». Die letzten zwei sind wichtig, wenn nicht alle Daten auf einmal geladen werden dürfen. Bei dieser Applikation ist das der Fall bei den Meldungen. Wenn das Frontend auf einmal 1000 Meldungen laden müsste, würde vielleicht der Browser abstürzen. Darum werden die Meldungen in Gruppen von 25 aufgeteilt, möchte der Benutzer mehr sehen, muss dieser einen Knopf drücken und die nächsten 25 werden geladen.

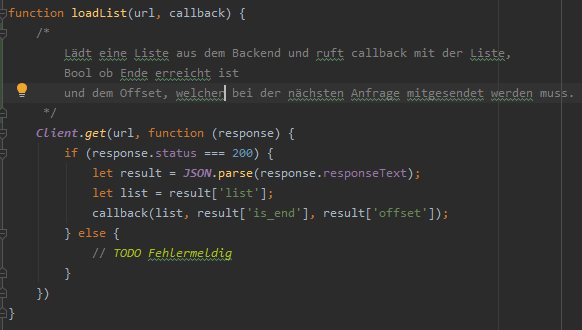


Abbildung 35 Code für das Laden einer Auflistung

Nachdem der Callback mit der Liste als Parameter aufgerufen wirdkann mithilfe einer For Schleife diese durchgelaufen und damit kann zum Beispiel die Reihen einer Tabelle gesetzt werden.

### Formen

Bevor ein neues Modell an das Backend gesendet werden kann, soll der Inhalt auf die Länge überprüft werden. Die Daten dürfen erst abgesendet werden, wenn alle nötigen Felder ausgefüllt sind, ansonsten soll eine Meldung angezeigt werden.

Für die Überprüfung habe ich eine Funktion geschrieben, bei welcher als Parameter das JQuery Objekt folgend mit erwarteter Mindest- und Maximum Länge übergeben werden kann. Ist das Input Feld leer oder kann die Länge Bedingungen nicht einhalten, wird die Bootstrap CSS Klasse «is-invalid» dem Input Feld hinzugefügt. Dies führt dazu, dass das HTML Div mit der CSS Klasse «invalid-feedback» unter dem Feld rot angezeigt wird. Dieses beinhaltet eine Meldung ähnlich wie «Der Name darf nicht leer sein.».

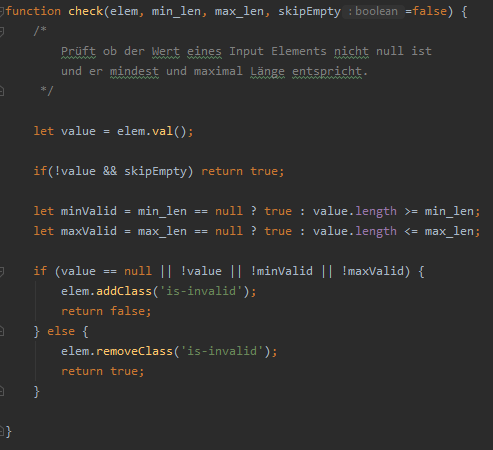


Abbildung 36 Inhaltsüberprüfung eines Inputs Feldes Code Beispiel

Diese Überprüfung findet für alle Felder einer Form statt. Nachdem der Benutzer valide Werte eingegeben hat, werden alle Werte in ein Dict verpackt und mit JSON enkodiert. Soll das Objekt erstellt werden, muss die Anfrage mit der Methode POST stattfinden, ansonsten für die Aktualisierungen PUT.

Als Beispiel die Raum Form:

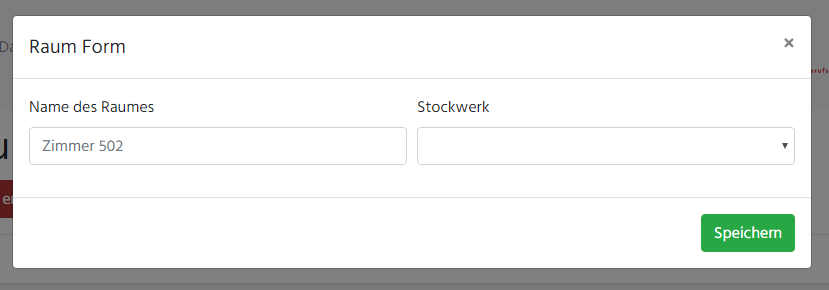


Abbildung 37 Raum Form Ansicht

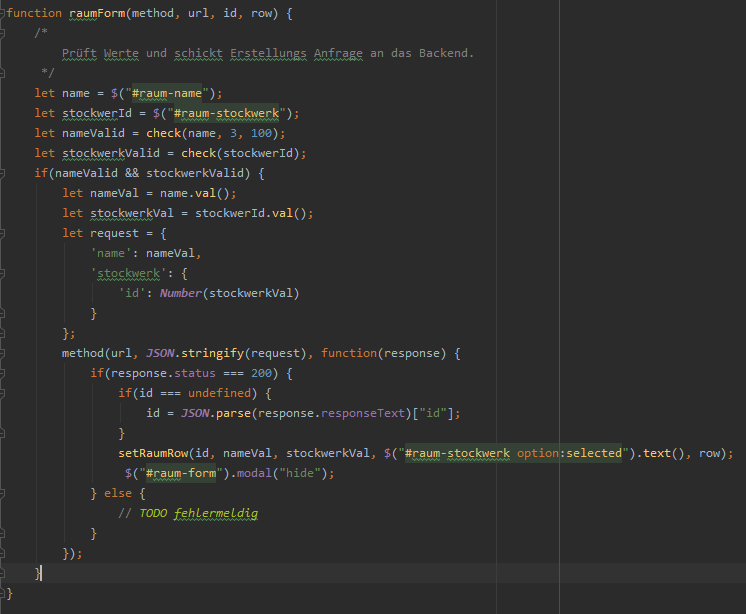


Abbildung 38 Raum Form Code Beispiel

Als erstes werden die Input Felder definiert und validiert. Der Name wird auf die Mindestlänge von drei und Maximum Länge von hundert geprüft. Die StockwerkID wird lediglich darauf getestet, dass sie nicht leer ist. Danach wird der Body der Abfrage mit dem JSON Objekt gesetzt und schlussendlich mit der gewünschten Methode ausgeführt. Die Methoden können Client.post oder Client.put sein. War die Erstellung oder Aktualisierung erfolgreich, wird eine Reihe in der Tabelle hinzugefügt, oder die bereits existierende mit den neuen Werten ergänzt.

### Reinigungspersonal Ansicht

Bei der Reinigungspersonal Ansicht war es wichtig, dass die Meldungen nach Zimmer, Stockwerk und Gebäude sortiert werden. Zusätzlich sollte das Personal eine Liste von vorgeschlagenem Material sehen.

Für die Sortierung ist es von Vorteil zu wissen, wie eine solche Meldung überhaupt aussieht.

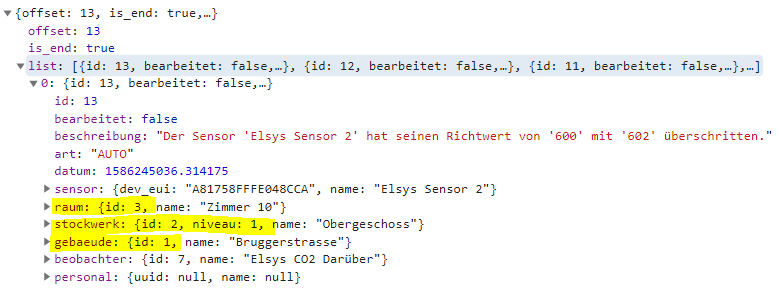


Abbildung 39 Meldung in JSON

Wir entnehmen bei jeder Meldung die Raum ID, Stockwerk Niveau und Gebäude ID. Die Sortierung funktioniert für jede Ebene gleich. Zuerst wird ein Dictionary erstellt, bei welchem die Meldungen per Gebäude ID eingefüllt werden. Damit anschliessend per Stockwerk Niveau sortiert werden kann, wird ein Dictionary im Dictionary erstellt. Für die erste Meldung, welche nur nach Gebäude ID sortiert ist, kämen wir zu folgendem Stand. Die eins entpricht der Gebäude ID.

{

1: {}

}

Danach wird das gleiche Prinzip für die Sortierung nach Stockwerk und Raum angewendet. Am Schluss wird beim Raum kein Dictionary erstellt, sondern ein Array mit allen Meldungen des Raumes ergänzt. Nach dem Sortierungsdurchlauf des oben erwähnten Beispiels entsteht ein Objekt welches wie folgt aussieht:

{

1: {

1: {

3: [ (Meldung Objekt 1), (Meldung Objekt N)]

}

}

}

Die erste 1 entspricht wieder der Gebäude ID, die zweite 1 dem Stockwerk Niveau und die 3 der Raum ID. Alle Meldungen, welche im gleichen Raum sind, werden dem gleichen Array hinzugefügt. Beim Darstellen muss darauf geachtet werden, dass man die Stockwerk Niveaus von klein nach gross auflistet. Dies kann mit «Array().sort()» bewerkstelligt werden.

Nachdem alle Meldungen geladen wurden, müssen die Materialien für jeden Beobachter geholt werden. Dafür wird die Anzahl von jedem Beobachter gezählt und im Backend abgefragt, ob für die Reinigung einer Meldung dieses Beobachters Material benutzt wird. Die Materialien werden gruppiert, gezählt und dargestellt in der Form einer Checkliste. Diese Checkliste bietet keine Funktion, sondern soll nur beim Bereitstellen der Tour zu einem Überblick verhelfen.

### Darstellung der Sensor Daten

Bei den Sensor Daten war die Herausforderung, diese nach Benutzer definierten Zeitabstände zu gruppieren. Würde jedes einzelne Packet eines Sensors dargestellt werden, dann wären die Diagramme überflüssig, da nichts erkennbar ist. Ein Sensor kann pro Tag über 150 Pakete schicken, würde einen Zeitabschnitt von sieben Tagen betrachtet werden, wären dies 1050 Punkte auf einem Diagramm mit einer Breite von 800 Pixeln auf einem Tablet. Dies hat kein Sinn und daher müssen die Pakete gruppiert werden.

Jedes Paket ist mit einem Timestamp (Sekunden seit Epoch) markiert. Wird dieser in Millisekunden konvertiert, kann damit eine JavaScript Date Objekt erstellt werden. Dieses hat die Funktion «date.toLocaleString('de-CH')» mit welchem das Datum in ausgeschriebenem Text (07.04.2020 12:33) konvertiert werden kann. Damit die Pakete gruppiert werden können, muss bekannt sein, in welchen Zeitabständen. Danach wird mit einer For Schleife bei jedem Paket geprüft, ob dieser den Zeitabstand zur letzten Gruppierung übertrifft oder einhaltet. Wenn der Zeitabstand eingehalten wird, wird das Paket in den Array der letzten Gruppe hinzugefügt. Ansonsten wird eine neue Gruppe erstellt, welche als Label das ausgeschriebene Datum des Paketes trägt. Dieses Label wird schlussendlich im Diagramm auf der X-Achse dargestellt.

In Code sieht dies folgenderweise aus.

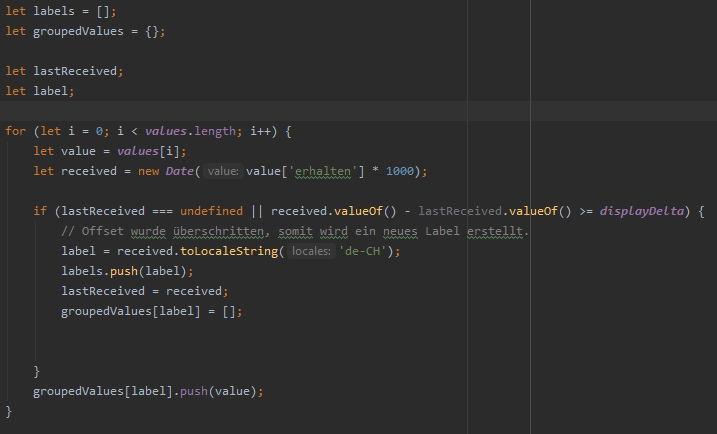


Abbildung 40 Gruppierungsalgorithmus für Sensor Pakete Code Beispiel

Nach dem Durchlauf dieser Schleife, sehen die Daten bei einem Zeitabschnitt einer halben folgendermassen aus.

{

‘07.04.2020 12:33’: [ (Paket 1), (Paket N) ],

‘07.04.2020 13:03’: [ (Paket 1), (Paket N) ]

}

Anschliessend, nach dem die Pakete in definierte Zeitabstände gruppiert wurden, können die Sensordaten, wie beispielsweise Temperatur oder CO2 ausgelesen werden. Für das Verständnis ist gut zu wissen, dass die Pakete erst gruppiert wurden und es immer noch gleich viele hat wie am Anfang. Das Ziel ist es jetzt, pro Sensor Wert (Temperatur, CO2, …) pro Label ( ‘07.04.2020 12:33’, 07.04.2020 13:03, …) immer nur drei Werte zu haben. Den kleinsten, grössten und den Durchschnitts Wert pro Zeitabschnitt. Dafür werden pro Zeitabschnittsgruppe alle Werte durch iteriert und die verschiedenen Sensorwerte mit JSON in ein Dictionary konvertiert. Diese werden nach Sensorwert Name in einem Dictionary gruppiert und schlussendlich nochmals durchiteriert damit der Min, Max und Durchschnitts Wert gefunden werden kann. Zu allerletzt werden für alle Vorkommenden Sensorwertenamen ein eigenes Diagramm erstellt, bei welchem die Labels mit den drei Werten dargestellt werden.

Der unten aufgelistete Code ist die Fortsetzung des oben gezeigten Codes.

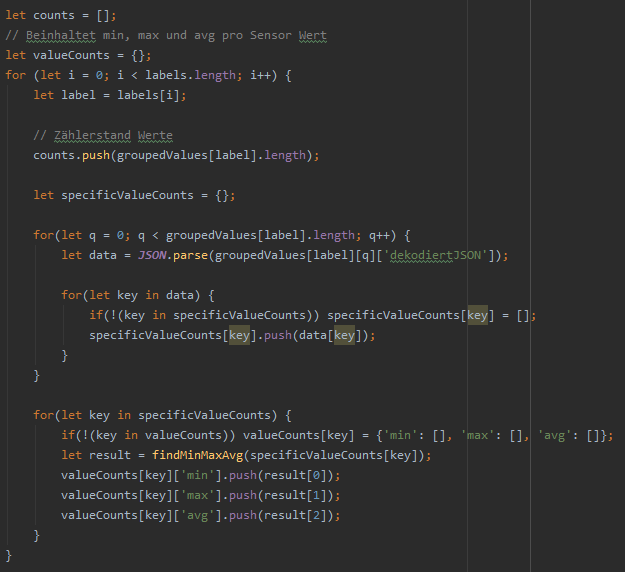


Abbildung 41 Gruppierung Code für die Sensor Werte

### Corporate Design

Das Corporate Design besteht aus drei Dingen. Die Farben sollen schwarz oder Rot (#c84646) sein, die Schriften «Hind Madurai Semi Bold» und «ROBOTO Slab Bold» dürfen benutzt werden und das Logo muss rechts den Abstand der Breite eines «B» und oben und unten die Höhe eines «B» zum Rand haben. Die Einhaltung der Farben und Schriften ist mit CSS sehr einfach und kann in den Bootstrap Dateien verändert werden.

Bei der Logo Grösse und dem Abstand sind jedoch ein paar Rechnungen nötig. Aus dem BBB\_CD\_Manual.pdf ist bekannt, dass die Grösse des Logos bei 100% 48x30 mm beträgt. Ein «B» entspricht 11x15 mm. Mit einem Dreisatz kann mit diesen Angaben das Logo Responsiv dargestellt werden. Der SCSS Code dafür folgt hier.

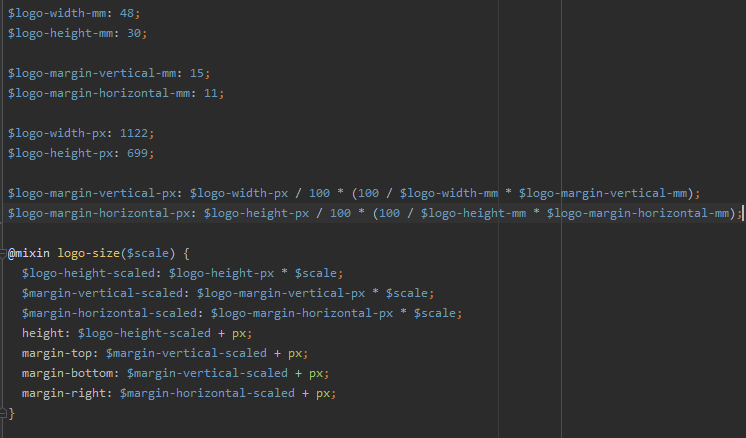


Abbildung 42 SCSS Code für Responsiveness des Logo

# Kontrollieren

## Testanlage

Die Tests, welche im Frontend passieren, werden jeweils auf drei verschiedenen Browsern durchgeführt. Chrome (Version 80.0.3987.163), Edge (Version 44.17763.831.0) und Firefox (Version 72.0.1). Die Webseite wird von einem Microsoft Windows Server 2019 Standard Version 10.0.17763 Build 17763 gehostet und ist über das Intranet aufrufbar.

## Testmethoden

Die Frontend Tests werden durch Aufrufen der zu prüfenden Seiten getätigt und durch vergleich mit dem erwarteten Resultat validiert. Die Tests werden jeweils, falls diese geklappt haben, mit einem «Ok» pro Browser markiert, ansonsten «Fehlerhaft».

Die Tests, welche nicht auf etwas visuellem basieren, werden durch ausführen der Funktionen oder Sensor Mitteilungen getestet. Diese werden nur mit einem «Ok» oder «Fehlerhaft» bewertet.

## Testprotokoll

Tabelle 26 Ausführung des Testprotokolls

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Testfallnummer | Erwartetes Ergebnis | Tatsächliches Ergebnis | Status | Datum | Tester |
| 1.1 | Der Benutzer erhält nach Login einen Refresh und Access Token und diese sind im Session Storage aufzufinden. | Die Tokens werden nach erfolgreichem Login in den Session Storage geschrieben. | Chrome: Ok  Edge: Ok  Firefox: Ok | 07.04.2020 | Nils Egger |
| 1.2 | Der Browser erhält einen neuen Access Token von der API. | Alle Browser erhielten einen neuen Access Token. | Chrome: Ok  Edge: Ok  Firefox: Ok | 07.04.2020 | Nils Egger |
| 1.3 | Der Browser zeigt eine Meldung bei der Eingabe von invaliden Login Daten. | Alle Browser zeigten die Meldung nach Eingabe von falschen Login Daten. | Chrome: Ok  Edge: Ok  Firefox: Ok | 07.04.2020 | Nils Egger |
| 1.4 | Der Browser loggt den Benutzer nach Anfrage zu einem neuen Access Token mit falschem Refresh Token aus. | Alle Browser leerten den Storage Token und leiteten den Benutzer zur Login Seite. | Chrome: Ok  Edge: Ok  Firefox: Ok | 07.04.2020 | Nils Egger |
| 2.1 | Der Loriot Listener kann Sensoren anhand der Art zu den richtigen Dekodierer zuweisen. | Alle Sensoren können dekodiert werden. Der Loriot Listener gibt die Sensor Daten den richtigen Dekodierer weiter. | Ok | 07.04.2020 | Nils Egger |
| 2.2 | Ein Paket eines Adeunis RF Sensors wird in die Werte Temperatur vorhanden, Übertragung durch Klick ausgelöst, Übertragung durch Bewegung ausgelöst, Temperatur und VDD umgewandelt. | Das Paket wird in die erwartete Werte umgewandelt. | Ok | 07.04.2020 | Nils Egger |
| 2.3 | Ein Paket eines Elsys ERS CO2 Sensor wird in die Werte Temperatur, Feuchtigkeit, Licht, Bewegungen, CO2 und VDD umgewandelt. | Das Paket wird in die erwartete Werte umgewandelt. | Ok | 07.04.2020 | Nils Egger |
| 2.4 | Ein Paket eines Tabs Sensors wird in die Werte Status und VDD umgewandelt. | Das Paket wird in die erwartete Werte umgewandelt. | Ok | 07.04.2020 | Nils Egger |
| 3.1 | Auf der Seite <http://si1010023/beobachter.html> kann ein Beobachter einem Sensor hinzugefügt werden. | In jedem Browser kann ein Beobachter, oder mehrere, einem Sensor hinzugefügt werden. | Chrome: Ok  Edge: Ok  Firefox: Ok | 07.04.2020 | Nils Egger |
| 3.2 | Der Richtwert Beobachter meldet sich nach der Erwärmung des Sensors. | Eine Meldung des Beobachters wird in allen Browsern im Dashboard angezeigt. | Chrome: Ok  Edge: Ok  Firefox: Ok | 07.04.2020 | Nils Egger |
| 3.3 | Der Zählerstand schreibt eine Meldung ins Dashboard nach 10 Paketen (Knopf drücke auf Adeunis Sensor). | Die Meldung wird in allen drei Browsern im Dashboard angezeigt. | Chrome: Ok  Edge: Ok  Firefox: Ok | 07.04.2020 | Nils Egger |
| 3.4 | Es ist möglich für den Elsys ERS CO2 Sensor mehrere Beobachter zu erstellen, welche alle auf ihre Werte reagieren. | In den drei Browsern lassen sich die Beobachter erstellen und die Meldungen dieser werden auch dargestellt. | Chrome: Ok  Edge: Ok  Firefox: Ok | 07.04.2020 | Nils Egger |
| 4.1 | Gebäude lassen sich auf der Seite <http://si1010023/gebaeude.html> erstellen. | In den drei Browser lassen sich die Gebäude erstellen. | Chrome: Ok  Edge: Ok  Firefox: Ok | 07.04.2020 | Nils Egger |
| 4.2 | Auf der Seite <http://si1010023/stockwerke.html> lassen sich Stockwerke einem Gebäude hinzufügen | Stockwerke können in den drei Browsers erstellt und einem Gebäude hinzugefügt werden. | Chrome: Ok  Edge: Ok  Firefox: Ok | 07.04.2020 | Nils Egger |
| 4.3 | Auf der Seite <http://si1010023/raeume.html> lassen sich Räume einem Stockwerk hinzufügen. | Räume können in den drei Browsers erstellt und einem Stockwerk hinzugefügt werden. | Chrome: Ok  Edge: Ok  Firefox: Ok | 07.04.2020 | Nils Egger |
| 4.4 | In der Raum Verwaltung werden die Räume nach Stockwerk Niveau und Gebäude sortiert. | In den drei Browser werden die Räume korrekt sortiert. | Chrome: Ok  Edge: Ok  Firefox: Ok | 07.04.2020 | Nils Egger |
| 5.1 | Bei den Formen werden leere Eingaben abgefangen und eine Fehlermeldung wird dargestellt. | In den drei Browser werden leere Eingaben abgefangen und es wird immer eine Fehlermeldung dargestellt. | Chrome: Ok  Edge: Ok  Firefox: Ok | 07.04.2020 | Nils Egger |
| 5.2 | Eine Code Eingabe wie «<script>alert(«hello»);</script>» wird nicht ausgeführt. | Der JavaScript Code wird in keinem Browser ausgeführt. Jedoch wird er erst nach einem Reload dargestellt. | Chrome: Ok  Edge: Ok  Firefox: Ok | 07.04.2020 | Nils Egger |
| 5.3 | Es wird bei einer Fehleingabe bei einer Form eine Fehlermeldung dargestellt, in der zusätzlich die Max Anzahl an Charakter dargestellt wird. | In allen Browser wird eine korrekte Meldung dargestellt. | Chrome: Ok  Edge: Ok  Firefox: Ok | 07.04.2020 | Nils Egger |
| 6.1 | Das Design vergrössert sich, damit es auf dem grossen Bildschirm korrekt dargestellt wird. | In allen Browsern wird die Webseite korrekt dargestellt. | Chrome: Ok  Edge: Ok  Firefox: Ok | 07.04.2020 | Nils Egger |
| 6.2 | Das Design verkleinert sich für die Tablet Ansicht. | In allen Browsern wird die Webseite korrekt dargestellt. | Chrome: Ok  Edge: Ok  Firefox: Ok | 07.04.2020 | Nils Egger |
| 6.3 | Die Webseite wird von Edge ohne Probleme unterstützt. | Der Browser unterstützt die Webseite. | Ok | 07.04.2020 | Nils Egger |
| 6.4 | Die Webseite wird von Firefox ohne Probleme unterstützt. | Der Browser unterstützt die Webseite. | Ok | 07.04.2020 | Nils Egger |
| 6.5 | Die Webseite wird von Chrome ohne Probleme unterstützt. | Der Browser unterstützt die Webseite. | Ok | 07.04.2020 | Nils Egger |
| 6.6 | Die Schriften werden korrekt von Google Fonts geladen und mit richtiger Farbe dargestellt. | Es werden die korrekte Fonts und Farben dargestellt. | Chrome: Ok  Edge: Ok  Firefox: Ok | 07.04.2020 | Nils Egger |
| 6.7 | Die Luftqualität wird unter Elsys Sensors angezeigt. | In allen Browsern wird die Luftqualität unter Elsys Sensoren dargestellt. | Chrome: Ok  Edge: Ok  Firefox: Ok | 07.04.2020 | Nils Egger |
| 6.8 | Die Sensordaten Ansicht kann durch einen Zeitfilter begrenzt werden. | In allen Browsern wird die Filterfunktion angezeigt und kann verwendet werden. | Chrome: Ok  Edge: Ok  Firefox: Ok | 07.04.2020 | Nils Egger |

## Testbericht

Alle Tests funktionierten beim ersten Versuch. Dies liegt daran, dass das Testkonzept bei der Realisierung stets im Vordergrund stand und somit während der Entwicklung darauf geschaut wurde.

# Auswerten

## Reflexion

Die IPA war bis auf die HTML Formen und den Tabs Sensor gut vorbereitet. Wären die Formen auch vorbereitet gewesen, hätte ich sehr viel Zeit gespart und mein Stress schon fast eliminiert. Für das nächste Projekt wird dies sicherlich besser im Voraus abgeklärt, vielleicht hätte sogar ein Framework für das Frontend vorbereitet werden sollen.

Aus Zeitlichen Gründen konnten nicht alle Seiten aus den Mockups umgesetzt werden, diese hätten von Beginn an auf das minimale reduziert werden müssen.

Der Sinn der Backend und Frontend Aufteilung ist eigentlich, dass all Logik im Backend geschieht und im Frontend lediglich die Daten dargestellt werden. Aus Zeitlichen Gründen wurde der Code für das Sortieren von Sensor Daten im Frontend umgesetzt. Allgemein könnte im Backend sogenannte Views erstellt werden, welche alle Daten für eine Seite vorbereiten, anstatt das alle Daten separat geladen werden.

Das Backend und Frontend laden beide sehr schnell, bei diesem Aspekt hat sich die Aufteilung sehr gelohnt.

## Fazit

Die IPA war sehr intensiv und ich hatte stets das Gefühl, dass die Zeit an mir vorbeifliege. Hätte ich nicht durchgehend konzentriert gearbeitet, wäre ich definitiv nicht fertig geworden. Das Endprodukt entspricht dem Ziel, könnte jedoch vom Frontend Code her noch verbessert werden.

Das Backend aufzubauen war nicht sehr schwierig und konnte auch gut nach Zeitplan passieren. Die Challenge kam erst beim Erstellen des Frontend. Die Hürden lösten sich jedoch schnell und es waren nur noch die HTML Formen welche mir die Zeit stahlen. Ich bin der Meinung, dass das Endprodukt ein Erfolg ist. Ich bin vor allem glücklich mit dem Resultat der Diagramme und wie die Zeitabstände zwischen den Punkten definiert werden können.

Damit die Applikation noch besser wird, sollte all Logik aus dem Frontend entfernt und Views erstellt werden. Falls die Applikation vergrössert werden sollte, ist es am gescheitesten, wenn ein Framework wie Angular oder Vue.js im Frontend benutzt wird. Das Backend ist im aktuellen Zustand sehr gut erweiterbar und es sind keine Änderungen nötig.

Bei diesem Projekt habe ich gelernt, wie Sensordaten sortiert werden können und mit Chart.js visuell schön dargestellt werden.

Ich bin zuversichtlich, dass diese Applikation intensiv von der BBB eingesetzt werden kann.

## Vergleich mit dem Zeitplan

Der Zeitplan konnte eingehalten werden, dort wo ich dem Zeitplan hinterher fiel, konnte ich an einem anderen Ort diesen wieder aufholen. Bei den HTML Formen wurde am meisten Zeit verloren, während beim Testen und API Schnittstellen programmieren am meisten gespart werden konnte.

# Glossar

Tabelle 27 Glossar mit Definition

|  |  |
| --- | --- |
| Begriff | Definition |
| Apache | Webserver |
| ASGI | Asynchronous Server Gateway Interface, Nachfolger des Web Server Gateway Interface ist ein Web Server welches die Bibliothek asyncio von Python unterstützt. |
| Dictionary | Ein Dictionary enthält mehrere Werte, welche alle durch einen Key abgefragt werden können. Vergleichbar mit einem Array, aber anstatt von einem Index, hat jeder Wert seinen Key (String oder Nummer). |
| Docstring | Ein Docstring ist ein einzeiliger oder mehrzeiliger Kommentar in Python welcher die Funktion einer Funktion beschreibt. |
| Loriot | Loriot ist Webapplikation von welcher die Sensor Daten geholt werden können. |
| Mockups | Skizzen des Endresultats |
| Nginx | Statischer Webserver |
| Sphinx | Sphinx ist ein Dokumentation Tool für Python, welches automatisch Docstrings zusammenstellen kann. |
| Uvicorn | ASGI Server |
| Websockets | http Protokoll |

# Literaturverzeichnis

Tabelle 28 Literaturverzeichnis mit Datum und Begründung

|  |  |  |
| --- | --- | --- |
| Abgerufener Link | Wann | Begründung |
| <https://www.chartjs.org/docs/latest/charts/line.html> | 23.03.2020 | Hilfestellung für Chart.js Beispiel. |
| <https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/MIME_types/Common_types> | 23.03.2020 | Mime Types wurden für verschiedene Diagramme verwendet. |
| <https://www.slant.co/options/10578/alternatives/~chart-js-alternatives> | 23.03.2020 | Hilfe Auswahl von JavaScript Diagramm Bibliotheken. |
| <https://gionkunz.github.io/chartist-js/getting-started.html> | 23.03.2020 | Hilfestellung für Chartist.js |
| <https://channels.readthedocs.io/en/latest/asgi.html> | 23.03.2020 | Hilfestellung für eigene Erklärung |
| <https://de.wikipedia.org/wiki/Web_Server_Gateway_Interface> | 23.03.2020 | Hilfestellung für eigene Erklärung |
| <https://iot-shop.de/wp-content/uploads/2020/03/RM_Door-_-Window-Sensor_20200205_v2.pdf> | 24.03.2020 | Hilfestellung für Dekodierung des Sensor Wertes |
| <https://www.hacksplaining.com/> | 24.03.2020 | Hilfestellung für wichtigste Sicherheitsaspekte der Webapplikation |
| <https://www.postgresql.org/docs/9.2/plpython-data.html> | 24.03.2020 | Datentypen von PostgreSQL |
| <https://stackoverflow.com/questions/9002901/how-to-import-module-to-sphinx> | 25.03.2020 | Hilfestellung für Problem mit Sphinx |
| <https://stackoverflow.com/questions/5464627/how-to-have-same-text-in-two-links-with-restructured-text> | 25.03.2020 | Problembehebung |
| <https://github.com/mozilla/bleach> | 27.03.2020 | Lösung für die Bereinigung von HTML in Strings. |
| <https://stackoverflow.com/questions/1957273/how-do-i-generate-a-random-string-of-length-x-a-z-only-in-python> | 30.03.2020 | Hilfestellung für die Erstellung eines Random Strings einer gewissen Länge. |
| <https://stackoverflow.com/questions/10654992/how-to-get-collection-of-keys-in-javascript-dictionary> | 03.04.2020 | Antwort für die Frage, wie alle Keys eines Dictionaries aufgelistet werden können in JavaScript. |
| <https://stackoverflow.com/questions/14182079/delete-rows-with-foreign-key-in-postgresql> | 03.04.2020 | Hilfestellung für die Löschung aller Reihen, welche von einem Foreign Key abhängig sein. |
| <https://stackoverflow.com/questions/28729634/set-values-in-input-type-date-and-time-in-javascript> | 06.04.2020 | Hilfestellung für das Setzen des Values eines Date Inputfeldes. |
| <https://stackoverflow.com/questions/33659508/javascript-get-a-date-from-an-html-input> | 06.04.2020 | Hilfestellung für das Herauslesen eines Datums von einem Inputfeld. |
| <https://stackoverflow.com/questions/9575790/how-to-get-time-in-milliseconds-since-the-unix-epoch-in-javascript> | 06.04.2020 | JavaScript Date Objekt in Sekunden seit Epoch umwandeln. |
| <chrome://history/?q=https%3A//stackoverflow.com/questions/25981703/pip-install-fails-with-connection-error-ssl-certificate-verify-failed-certi> | 07.04.2020 | Problemlösung für das Umgehen von SSL bei der Installation von Python Packages. |

# Tabellenverzeichnis

[Tabelle 1 Änderungsindex 2](#_Toc37235412)

[Tabelle 2 Responsive Design Bewertungskriterium 8](#_Toc37235413)

[Tabelle 3 Browserkompatibilität Bewertungskriterium 9](#_Toc37235414)

[Tabelle 4 Corporate Design Bewertungskriterium 9](#_Toc37235415)

[Tabelle 5 Codingstyle Bewertungskriterium 10](#_Toc37235416)

[Tabelle 6 Validierung Bewertungskriterium 10](#_Toc37235417)

[Tabelle 7 Anzeige Luftqualität Bewertungskriterium 10](#_Toc37235418)

[Tabelle 8 Grafische Darstellung der Messwerte Bewertungskriterium 11](#_Toc37235419)

[Tabelle 9 Deklaration der Vorkenntnisse mit Bemerkung 12](#_Toc37235420)

[Tabelle 10 Legende der gefärbten Felder im Zeitplan 13](#_Toc37235421)

[Tabelle 11 Bedeutung eines Wertes eines Tabs Sensors 20](#_Toc37235422)

[Tabelle 12 API Routen mit beschriebener Aktion welche diese ausführen. 33](#_Toc37235423)

[Tabelle 13 Testkonzept mit Nummer und Gebiet. 36](#_Toc37235424)

[Tabelle 14 Testfallspezifikationen für die Authentifizierung 36](#_Toc37235425)

[Tabelle 15 Testfallspezifikationen für die Sensorwert Dekodierung 37](#_Toc37235426)

[Tabelle 16 Testfallspezifikationen für die Sensorwert Beobachter 38](#_Toc37235427)

[Tabelle 17 Testfallspezifikationen für die Gebäude Verwaltung 39](#_Toc37235428)

[Tabelle 18 Testfallspezifikationen für die Eingabe Validierung 40](#_Toc37235429)

[Tabelle 19 Testfallspezifikationen für das Design und die Browserkompatibilität 40](#_Toc37235430)

[Tabelle 20 Keys Konfiguration 43](#_Toc37235431)

[Tabelle 21 Token Konfiguration 43](#_Toc37235432)

[Tabelle 22 Scrypt Hash Algorithmus Konfiguration 44](#_Toc37235433)

[Tabelle 23 Datenbank Verbindung Konfiguration 44](#_Toc37235434)

[Tabelle 24 Datenbank Verbindungsabbau Konfiguration 44](#_Toc37235435)

[Tabelle 25 ASGI Server Konfiguration 45](#_Toc37235436)

[Tabelle 26 Ausführung des Testprotokolls 61](#_Toc37235437)

[Tabelle 27 Glossar mit Definition 67](#_Toc37235438)

[Tabelle 28 Literaturverzeichnis mit Datum und Begründung 68](#_Toc37235439)

# Abbildungsverzeichnis

[Abbildung 1 Chart.js Resultat des Beispielcodes 17](#_Toc37235440)

[Abbildung 2 Chart.js Code für die Darstellung eines Liniendiagramms 17](#_Toc37235441)

[Abbildung 3 Chartist.js Resultat des Beispielcodse 17](#_Toc37235442)

[Abbildung 4 Chartis.js Code für die Darstellung eines Liniendiagramms 17](#_Toc37235443)

[Abbildung 5 Prototyp Algorithmus für die Dekodierung und Meldungserstellung eines Sensorwertes. 18](#_Toc37235444)

[Abbildung 6 Chromium Entwickler Modus Tablet Ansicht aktivieren 19](#_Toc37235445)

[Abbildung 7 Chromium Responsive Modus Menu 19](#_Toc37235446)

[Abbildung 8 Beispiel eines Google Style Docstrings 19](#_Toc37235447)

[Abbildung 9 Use Case für die Plattform 21](#_Toc37235448)

[Abbildung 10 System Aufbau 23](#_Toc37235449)

[Abbildung 11 Ablauf für die Authentifizierung während der Verwendung der Webseite 25](#_Toc37235450)

[Abbildung 12 Admin Dashboard Skizze 25](#_Toc37235451)

[Abbildung 13 Personal Seite Skizze 26](#_Toc37235452)

[Abbildung 14 Räume Verwaltung Skizze 27](#_Toc37235453)

[Abbildung 15 Raum Ansicht Skizze 28](#_Toc37235454)

[Abbildung 16 Sensoren Verwaltung Skizze 29](#_Toc37235455)

[Abbildung 17 Sensor Ansicht Skizze 30](#_Toc37235456)

[Abbildung 18 Dashboard Reinigungspersonal Skizze 31](#_Toc37235457)

[Abbildung 19 Konzeptionelles Modell UML 32](#_Toc37235458)

[Abbildung 20 Logisches Modell UML 33](#_Toc37235459)

[Abbildung 21 Nutzwertanalyse für die Entscheidung zwischen Chart.js und Chartist.js 42](#_Toc37235460)

[Abbildung 22 Loriot Listener Ablauf 46](#_Toc37235461)

[Abbildung 23 Gebäude Modell Code Beispiel 47](#_Toc37235462)

[Abbildung 24 Stockwerk Modell Code Beispiel 47](#_Toc37235463)

[Abbildung 25 Code Beispiel Abfrage mit Key Pfaden 48](#_Toc37235464)

[Abbildung 26 Gebäude List Controller Code Beispiel 49](#_Toc37235465)

[Abbildung 27 Beobachter erstellen Code Beispiel 50](#_Toc37235466)

[Abbildung 28 Loriot Listener Beobachter Variable 50](#_Toc37235467)

[Abbildung 29 on\_post Material zu Beobachter Resource Code Beispiel 51](#_Toc37235468)

[Abbildung 30 HTML Bereinigungsbeispiel 52](#_Toc37235469)

[Abbildung 31 Code Beispiel für das Herauslesen der Rolle aus dem Accesstoken. 52](#_Toc37235470)

[Abbildung 32 API Anfragen Code Beispiel 53](#_Toc37235471)

[Abbildung 33 Code für das Laden einer Auflistung 54](#_Toc37235472)

[Abbildung 34 Inhaltsüberprüfung eines Inputs Feldes Code Beispiel 55](#_Toc37235473)

[Abbildung 35 Raum Form Ansicht 55](#_Toc37235474)

[Abbildung 36 Raum Form Code Beispiel 56](#_Toc37235475)

[Abbildung 37 Meldung in JSON 57](#_Toc37235476)

[Abbildung 38 Gruppierungsalgorithmus für Sensor Pakete Code Beispiel 58](#_Toc37235477)

[Abbildung 39 Gruppierung Code für die Sensor Werte 60](#_Toc37235478)

[Abbildung 40 SCSS Code für Responsiveness des Logo 61](#_Toc37235479)