|  |  |
| --- | --- |
| AVEC_LOGO_R_RGB.pdf | |
| IPA-Bericht 2020 | |
|  | |
| BBB Gebäude Management Pilot Versuch | |
| \\vfbdn571\home70$\REINEGGE\My Documents\AXI NEU\AVECTRIS_Benutzeranleitung_A4\Links\image001.png | |
|  | |
| Nils Egger  Projektleiter | |
| 23.03.2020 |  |
|  | |

Änderungsindex

|  |  |  |  |
| --- | --- | --- | --- |
| Revision | Beschreibung | Erstellt | Datum |
| 0.1 | Erstellung des Dokumentes | Nils Egger | 23.03.2020 |
|  |  |  |  |
|  |  |  |  |

Inhaltsverzeichnis

[1 Aufgabenstellung 5](#_Toc35872889)

[1.1 Titel der Arbeit 5](#_Toc35872890)

[1.2 Ausgangslage 5](#_Toc35872891)

[1.3 Detaillierte Aufgabenstellung 5](#_Toc35872892)

[1.3.1 Beschreibung 5](#_Toc35872893)

[1.3.2 Generelle Eigenschaften 5](#_Toc35872894)

[1.3.3 Anforderungen an die Webapplikation 5](#_Toc35872895)

[1.4 Mittel und Methoden 7](#_Toc35872896)

[1.5 Vorkenntnisse 7](#_Toc35872897)

[1.6 Vorarbeiten 7](#_Toc35872898)

[1.7 Neue Lerninhalte 7](#_Toc35872899)

[1.8 Arbeiten in den letzten 6 Monaten 7](#_Toc35872900)

[1.9 Individuelle Bewertungskriterien 7](#_Toc35872901)

[1.9.1 Responsive Design 7](#_Toc35872902)

[1.9.2 Browserkompatibilität 8](#_Toc35872903)

[1.9.3 Einhaltung Corporate Design 8](#_Toc35872904)

[1.9.4 Codingstyle – Dokumentation 8](#_Toc35872905)

[1.9.5 Validierung Eingaben 9](#_Toc35872906)

[1.9.6 Anzeige der Luftqualitäts-Messwerte 9](#_Toc35872907)

[1.9.7 Grafische Darstellung der Messwerte 10](#_Toc35872908)

[2 Projektorganisation 10](#_Toc35872909)

[2.1 Methode 10](#_Toc35872910)

[2.2 Auftraggeber 10](#_Toc35872911)

[2.3 Beteiligte Personen 10](#_Toc35872912)

[2.4 Durchführungsort 10](#_Toc35872913)

[3 Deklaration der Vorkentnisse 11](#_Toc35872914)

[4 Deklaration der Vorarbeiten 11](#_Toc35872915)

[5 Deklaration der benützten Firmenstandarts 12](#_Toc35872916)

[6 Zeitplan 12](#_Toc35872917)

[6.1 Bemerkungen 12](#_Toc35872918)

[6.2 Legende 12](#_Toc35872919)

[6.3 Meilensteine 12](#_Toc35872920)

[7 Organisation der Arbeitsergebnisse 14](#_Toc35872921)

[8 Informieren 14](#_Toc35872922)

[8.1 Aufgabe 14](#_Toc35872923)

[8.2 Deadline 14](#_Toc35872924)

[8.3 Herausforderungen 14](#_Toc35872925)

[8.3.1 JavaScript Diagramme 14](#_Toc35872926)

[8.3.2 Sensor Wert Beobachter 15](#_Toc35872927)

[8.3.3 Responsive Design 16](#_Toc35872928)

[8.3.4 Sphinx Dokumentation 17](#_Toc35872929)

[8.3.5 Python API Framework 17](#_Toc35872930)

[8.3.6 HTML Präprozessor 17](#_Toc35872931)

[9 Planen 18](#_Toc35872932)

[9.1 System 18](#_Toc35872933)

[9.2 Authentifizierung 19](#_Toc35872934)

[10 Glossar 19](#_Toc35872935)

[11 Literaturverzeichnis 20](#_Toc35872936)

[12 Abbildungsverzeichnis 20](#_Toc35872937)

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?

|  |  |
| --- | --- |
| 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.

|  |  |
| --- | --- |
| 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?

|  |  |
| --- | --- |
| 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?

|  |  |
| --- | --- |
| 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?

|  |  |
| --- | --- |
| 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.

|  |  |
| --- | --- |
| 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?

|  |  |
| --- | --- |
| 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. |

# Projektorganisation

## Methode

Die gesamte IPA wird mit IPERKA (Informieren, Planen, Entscheiden, Realisieren, Kontrollieren und Auswerten) durchgeführt. IPERKA ist einfach anzuwenden und wurde oft während meiner Lehre benutzt, daher ist diese Projektmethode die einzige welche für mich in Frage kommt.

## 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 Vorkentnisse

|  |  |
| --- | --- |
| Kenntnis | Bemerkung |
| Python | Mit Python automatisiere ich meist 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 allgemeine SQL Fähigkeiten aber noch keine vertieften Kenntnisse. Ich arbeite erste 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 Firmenstandarts

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

|  |  |
| --- | --- |
| 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.



# Organisation der Arbeitsergebnisse

Meine gesamte Arbeit ist in einem privaten GIT Repository auf github abgespeichert. Ich werde während der IPA all zwei Stunden meine Arbeit speichern und auf github hochladen.

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

# Informieren

## Aufgabe

Das Ziel ist es für die BBB 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 dann vom Reinigungspersonal abgearbeitet werden. Hierbei wäre damit Fenster öffnen gemeint.

## Deadline

Diese Applikation soll bis und mit dem 08.04.2020 geschrieben sein.

## Herausforderungen

### JavaScript Diagramme

Die Sensordaten müssen auf eine Art Visuell dargestellt werden. Ich stelle mir dies anhand einer JavaScript Bibliothek vor. Ich kenne noch keine und muss mir eine Liste mit Google beschaffen.

Zwei Bibliotheken welche mir sympathisch vorkommen sind folgend aufgelistet. 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 scheinte, die Diagramm Grösse zu verändern. Die Lösung hierzu war es, das Diagramm in einem div zu verpacken und die Grösse des divs 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 1 Chart.js Resultat

Abbildung 2 Chart.js Code

* [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 das wichtigste zu sein.



Abbildung 3 Chartist.js Resultat

Abbildung 4 Chartis.js Code

### 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.

Im Prinzip stelle ich es mir das so vor:

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 5 Sensor Wert Ablauf

### 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 entsprechen. 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. Dies ist für mich auch eher Neuland und werde vermutlich meine Browser Ansicht auf die Grösse eines Tablets ändern damit ich ständig das Resultat sehe, welches schlussendlich auch wirklich benutzt wird. In Chromium basierenden Browsers kann man dies mit der F12 Taste erreichen. 

Abbildung 6 Chromium Entwickler Modus



Abbildung 7 Chromium Responsive Modus

### Sphinx Dokumentation

Python lässt sich anhand von sogenannten Docstrings dokumentieren. Diese kann man dann einfach mit Sphinx zu einer Webseite generieren lassen. Meine Herausforderung hier ist, dass ich diese gründlich nachführe und gleich nach der Erstellung einer Funktion diese auch wirklich schreibe. $

Hierbei gilt, dass meine Docstrings dem Standard von Google folgen, und folgenderweise aussehen sollten.



Abbildung 8 Google Style Docstrings

### Python API Framework

Innerhalb des letzten Jahres hatte ich ein Python Framework geschrieben welches auf Google Firestore Daten abspeichert. Dies wurde mir jedoch zur Hürde und ich habe das komplette Framework SQL basierend umgeschrieben. Ich werde dieses Framework als Gerüst für meine IPA benützen. Eine vollständige Dokumentation zu diesem Framework lässt sich hier finden. Die IPA wird das erste produktive Produkt werden. Es kann sich daher herausstellen, dass wichtige Aspekte eines Backend Frameworks in meinem Fehlen und mir die ganze Arbeit nicht erleichtert, sondern nur erschwert.

Das Framework hilft beim Aufbau eines ASGI Servers welches dazu fähig ist, Daten entgegenzunehmen, validieren und speichern, diese wieder zu verändern und auszulesen. Bei einer API sind Access Tokens eine bevorzugte Wahl für Authentifizierung, dafür bietet das Framework eine komplette Lösung. Zudem ist es möglich Bilder abzuspeichern und wieder darzustellen.

Für mich wird es noch eine kleine Herausforderung mit purem JavaScript auf die API zuzugreifen und dies möglichst so, dass ich nicht Code wiederholend schreiben muss. Zudem muss jeder Aufruf auf die API mit einem Authentifizierungstoken begleitet sein.

### 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 noch [Jekyll](https://jekyllrb.com/) angeschaut. Dieses Tool bietet genau was ich brauche und wird mich beim Aufbau der Seite unterstützen.

### Tür Schaltungssensor Wert

Die Werte des Adeunis RF und der Elsys CO2 Sensoren konnte ich bereits als Vorarbeit dekodieren. Für den Tabs Sensor reichte die Zeit nicht mehr.

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.

|  |  |  |  |
| --- | --- | --- | --- |
| 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 9 Use Case

## System

### Allgemein

Das System ist so aufgebaut, dass ein Benutzer zuerst die statische Webseite von dem IIS Server ladet und danach 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 man muss nicht warten bis zum Beispiel PHP alle Daten in das HTML gerendert hat, sondern man bekommt direkt die Webseite mit einer kleinen Ladeanimation und weiss bereits, dass etwas passiert, danach geht es meistens schnell bis die Daten von der API geladen und dargestellt werden. Zudem muss man nicht jedes Mal eine Seite neu laden um Daten abzuspeichern, denn alle Zugriffe auf die API werden mit Ajax Aufrufe ausgeführt.

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

### Uvicorn Server

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

### Websockets

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

### PostgreSQL Datenbank

PostgreSQL ist ein leistungsstarkes, objektrelationales Open-Source-Datenbanksystem mit mehr als 30 Jahren aktiver Entwicklung. Diese Datenbank ist Zuverlässig und Leistungsstark.



Abbildung 10 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 wurde dies bereits im Vorfeld gemacht.

Damit man weiss welcher Sensor von welchem Skript dekodiert werden soll, muss eine Verlinkung, welche von den Benutzern der Webseite erstellt werden kann, gemacht werden. Jeder Sensor hat eine eindeutige EUI, welche für die Verlinkung perfekt ist.

## Wert Beobachter

Jedem Sensor soll man mehrere Wert Beobachter anhängen können, da ein Sensor auch 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 kann man einen Temperatur Sensor nehmen. Setzte man den Richtwert auf 22 Grad und der Sensor meldet eine Temperatur von 24, so würde eine Meldung ausgehen.

1. Zählerstand Beobachter

Beim Zählerstand soll gemessen werden wie oft etwas gemacht wurde. Dies ist perfekt für die Messung wie oft eine Toilette benutzt wurde. So könnte man 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 11 Authentifizierung Ablauf

## Webseite

### Admin Dashboard

In der Admin Dashboard Ansicht sollen Administratoren sehen, welche Räume welche Meldungen 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 12 Admin Dashboard

### Personal Verwaltung

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



Abbildung 13 Personal Seite

### Räume Verwaltung

In der Räume Verwaltung kann das Gebäude aufgebaut werden. Räume werden in Stockwerken erstellt welche in einem Gebäude erstellt wurden. So gibt es für die Zukunft die Möglichkeit die Applikation auf mehrere Gebäude zu verteilen.



Abbildung 14 Räume Verwaltung

### Raum Ansicht

In der Raum Ansicht sieht man alle Sensoren und Meldungen eines Raums.

Abbildung 15 Raum Ansicht

### Sensoren Verwaltung

In der Sensoren Verwaltung soll die Art der Sensoren eingesehen werden können und die Sensoren selbst.

Für jede Art von Sensor soll definiert werden können, welches und wie viel Material für eine Meldung vorgeschlagen werden soll.

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.

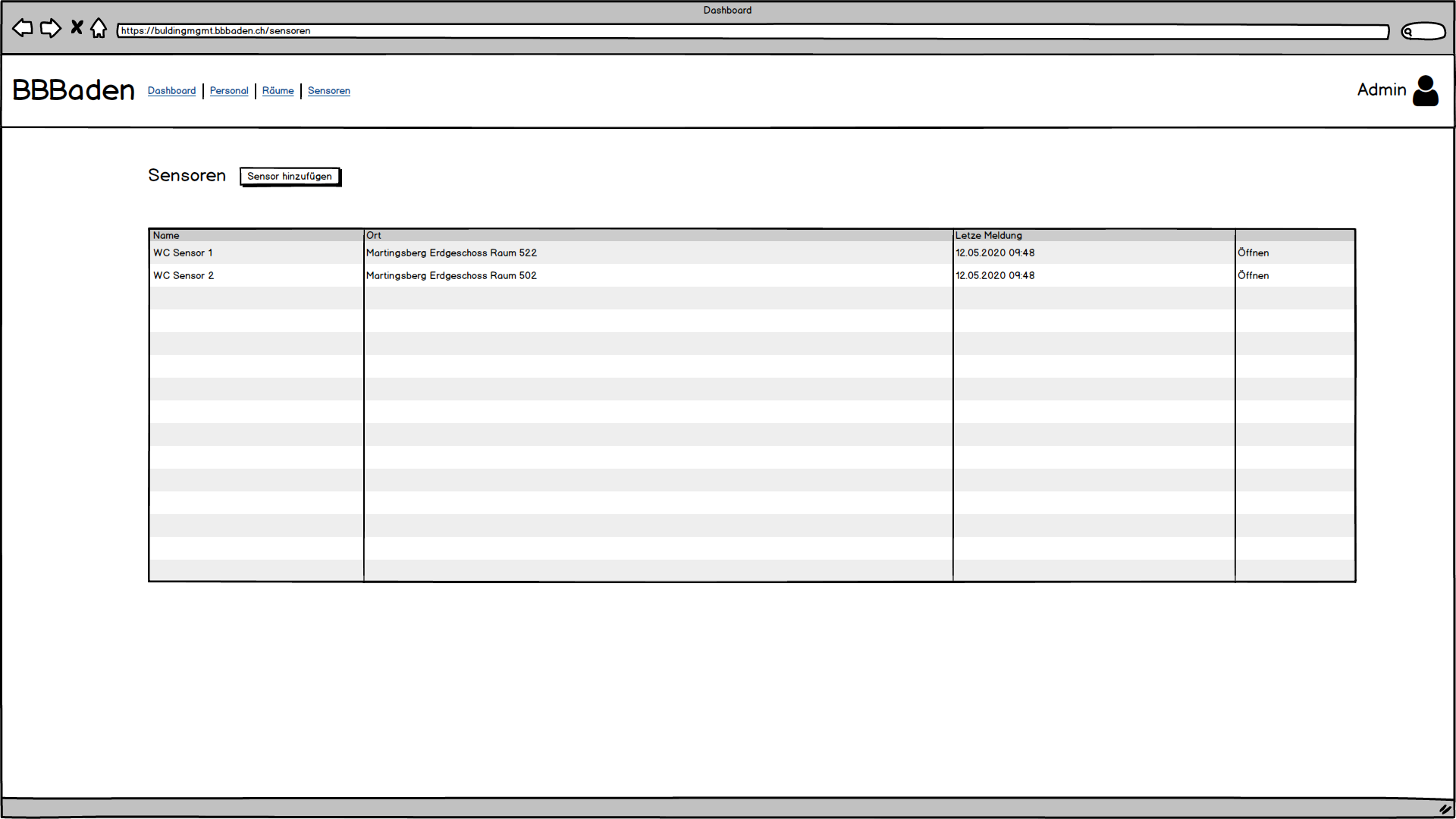


Abbildung 16 Sensoren Verwaltung

## Sensor Ansicht

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



Abbildung 17 Sensor Ansicht

### Reinigungspersonal Dashboard

Der Sinn dieses Dashboards ist, dass das Reinigungspersonal sich eine Route vorstellen kann. Die Räume mit Meldungen sind nach Gebäude und Stockwerk aufgelistet. Nachdem ein Raum geputzt wurde, kann der Raum als gelöst markiert werden. Auch hier verschwinden dann die Meldungen in eine separate Liste. Wenn gelöst geklickt wird, soll auch 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 18 Dashboard Reinigungspersonal

## Konzeptionelles Modell



Abbildung 19 Konzeptionelles Model

## Logisches Modell

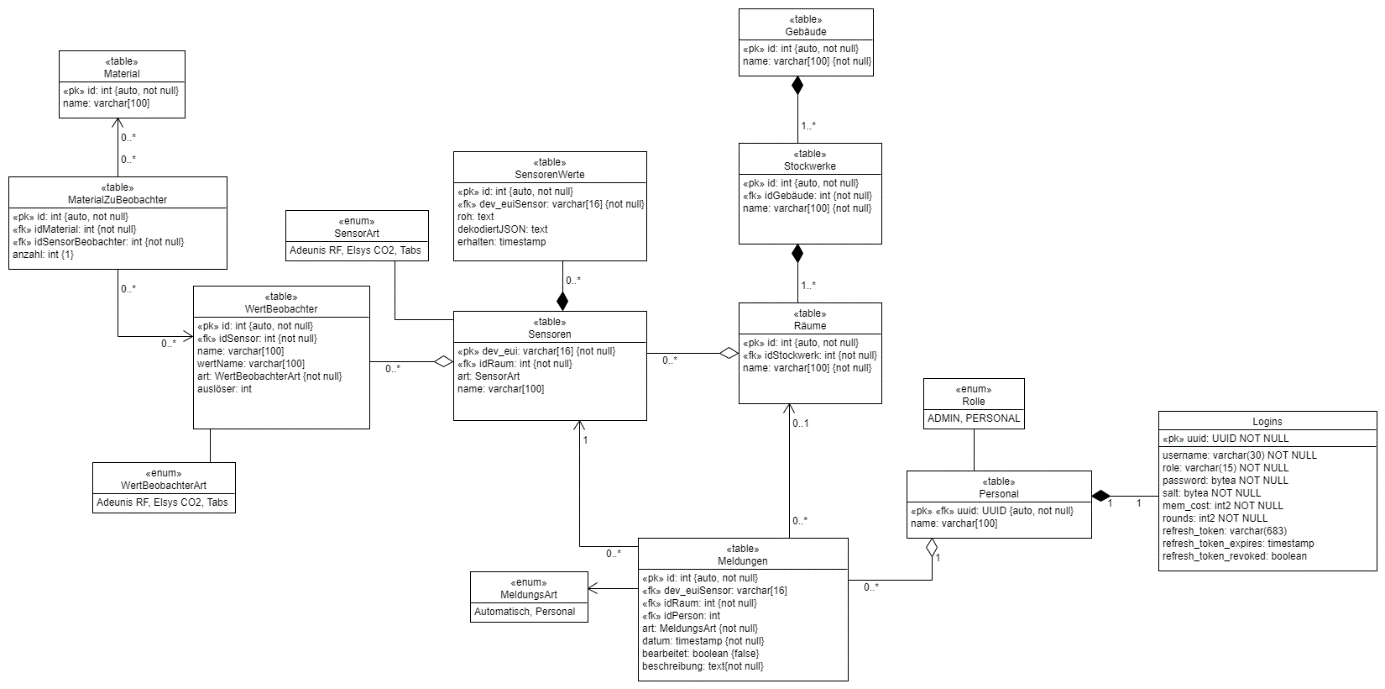


Abbildung 20 Logisches Modell

## 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.

|  |  |  |
| --- | --- | --- |
| 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/sensoren | GET | Auflistung aller Sensoren eines Raumes |
| /raume/<id>/sensoren | POST | Erstellung eines Sensors für den Raum mit ID |
| /raume/<id>/meldungen | GET | Auflistung aller Meldungen eines Raumes |
| /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 |
| /sensoren/<eui>/beobachter | POST | Erstellung eines Beobachter für einen Sensor |
| /sensoren/<eui>/beobachter/<id> | PUT | Aktualisierung eines Beobachters |
| /sensoren/<eui>/beobachter/<id> | DELETE | Löscht einen Beobachter |
| /sensoren/<eui>/meldungen | GET | Auflistung aller Meldungen eines Sensors |
| /sensoren/<eui>/beobachter/<id>/meldungen | GET | Auflistung aller Meldungen eines spezifischen Beobachters |
| /sensoren/<eui>/beobachter/<id>/meldungen | POST | Manuelle Auslösung einer Meldung |
| /views/dashboard | GET | Antwortet mit allen nötigen Daten für das Dashboard je nach Rolle |
| /views/personal | GET | Antwortet mit allen nötigen Daten für die Personal Verwaltung |
| /views/personal/<id> | GET | Antwortet mit allen nötigen Daten für die Person Ansicht |
| /views/raeume | GET | Antwortet mit allen Daten für die Räume Verwaltung |
| /views/raum/<id> | GET | Antwortet mit allen Daten für die Raum Ansicht |
| /views/sensoren | GET | Antwortet mit allen Daten für die Sensoren Verwaltung |
| /views/sensoren/<eui> | GET | Antwortet mit allen Daten für eine Sensor Ansicht. |

## Sicherheit

### Feld Validierung

Allgemein müssen alle Felder auf Inhalt und Länge validiert werden. Alle Eingaben eines Benutzers sollten von Tags und dergleichen 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 so konfiguriert, dass keine inline Scripts und nur JavaScript Files von derselben Domäne erlaubt sind. Der einzige Nachteil hier ist, dass man keinen CDN für JQuery oder dergleichen benutzen kann. So verhindert man aber auch das Schadcode von einem gehackten CDN auf die eigene Seite geladen wird.

## Testen

### Testkonzept

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

### Testfallspezifikationen

#### Authentifizierung

|  |  |  |  |
| --- | --- | --- | --- |
| Testfallnummer | Beschreibung | Voraussetzungen | Eingabe |
| 1.1 | Meldet sich ein Benutzer an, so erhält er eine Access Token und Refresh Token. | Backend und Frontend gestartet. | Benutzername und Passwort. |
| 1.2 | Sendet der Benutzer ein Refresh Token, so erhält er einen frischen Access Token. | In Besitz eines Refresh Token. | Lebensdauer eines Access-Token im Backend auf 1 Minute setzen, einloggen, eine Minute warte und schlussendlich Refresh Token schicken. |
| 1.3 | Bei invaliden Anmelde Daten ergibt sich einen Fehler. | Applikation gestartet. | Falscher Benutzername und Passwort. |
| 1.4 | Bei falschem Refresh Token ergibt sich einen Fehler. | Applikation gestartet. | Falschen Refresh Token an die API schicken. |

#### Sensor Wert Dekodierung

|  |  |  |  |
| --- | --- | --- | --- |
| Testfallnummer | Beschreibung | Voraussetzungen | Eingabe |
| 2.1 | Bei Erhalt eines Sensor Wertes wird dieser durch den richtigen Dekodierer in JSON umgeschrieben. | Loriot Gateway aktiv und verbunden per Websockets. | Daten Senden mit dem Adeunis RF. |
| 2.2 | Ein Wert des Adeunis RF Sensors wird korrekt in folgende Werte umgewandelt.   * Temperatur vorhanden * Übertragung durch Klick ausgelöst * Übertragung durch Bewegung ausgelöst * GPS vorhanden * Uplink Counter * Downlink Counter * Batterie Status * - (RSSI, SNR) | Loriot Gateway aktiv und verbunden per Websockets. | 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 per Websockets. | 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 * Time * Zähler * - (Temp PCB) | Loriot Gateway aktiv und verbunden per Websockets. | Den Status eines Sensors ändern in dem man den Magnet wegnimmt. |

#### Sensor Wert Beobachter

|  |  |  |  |
| --- | --- | --- | --- |
| Testfallnummer | Beschreibung | Voraussetzungen | Eingabe |
| 3.1 | Es können einem Sensor einen Wert Beobachter zugefügt werden. | Applikation gestartet, Sensor erstellt. | 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. | Applikation gestartet. CO2 Sensor hinzugefügt. | 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. | 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. | Elsys CO2 Sensor hinzugefügt und Richtwert Beobachter für Licht und Temperatur erstellt. | Sensor erwärmen und abdunkeln. |

#### Gebäude Verwaltung

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

#### Eingabe Validierung

|  |  |  |  |
| --- | --- | --- | --- |
| Testfallnummer | Beschreibung | Voraussetzungen | Eingabe |
|  |  |  |  |

#### Design und Browserkompatibilität

|  |  |  |  |
| --- | --- | --- | --- |
| Testfallnummer | Beschreibung | Voraussetzungen | Eingabe |
|  |  |  |  |

# Glossar

|  |  |
| --- | --- |
| Begriff | Definition |
| ASGI | Asynchronous Server Gateway Interface, Nachfolger des Web Server Gateway Interface ist ein Web Server welches die Bibliothek asyncio von Python unterstützt. |
| Docstring | Ein Docstring ist ein einzeiliger oder mehrzeiliger Kommentar in Python welcher die Funktion einer Funktion beschreibt. |
| Sphinx | Sphinx ist ein Dokumentation Tool für Python, welches automatisch Docstrings zusammenstellen kann. |
| Mockups | Skizzen des Endresultats |
| Loriot |  |
| Websockets |  |
| Uvicorn |  |
| Nginx |  |
| Apache |  |

# Literaturverzeichnis

|  |  |  |
| --- | --- | --- |
| 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 |

# Abbildungsverzeichnis

[Abbildung 1 Chart.js Resultat 15](#_Toc35934774)

[Abbildung 2 Chart.js Code 15](#_Toc35934775)

[Abbildung 3 Chartist.js Resultat 15](#_Toc35934776)

[Abbildung 4 Chartis.js Code 15](#_Toc35934777)

[Abbildung 5 Sensor Wert Ablauf 16](#_Toc35934778)

[Abbildung 6 Chromium Entwickler Modus 16](#_Toc35934779)

[Abbildung 7 Chromium Responsive Modus 17](#_Toc35934780)

[Abbildung 8 Google Style Docstrings 17](#_Toc35934781)

[Abbildung 9 Use Case 19](#_Toc35934782)

[Abbildung 10 System Aufbau 20](#_Toc35934783)

[Abbildung 11 Authentifizierung Ablauf 22](#_Toc35934784)

[Abbildung 12 Admin Dashboard 22](#_Toc35934785)

[Abbildung 13 Personal Seite 23](#_Toc35934786)

[Abbildung 14 Räume Verwaltung 24](#_Toc35934787)

[Abbildung 15 Raum Ansicht 25](#_Toc35934788)

[Abbildung 16 Sensoren Verwaltung 26](#_Toc35934789)

[Abbildung 17 Sensor Ansicht 27](#_Toc35934790)

[Abbildung 18 Dashboard Reinigungspersonal 28](#_Toc35934791)