

Individuelle praktische Arbeit

**3D Web Anwendung des OSE Modells für die Produktion
vorbereiten**

14. April 2021



Version: 0.5.2

Durchführung: 29.03.2021 - 14.04.2021

Verantwortliche Fachkraft
Markus Strittmatter
markus.strittmatter@endress.com

Hauptexperte
Matthias Meier

Kandidat
Jonas Schultheiss
jonas.schultheiss@endress.com

Nebenexperte
Jens Schwyn Aerni

DOKUMENTMANAGEMENT

Autor: Jonas Schultheiss
Version: 0.5.2
Datum: 14. April 2021
Status: In Progress
Dateiname: ipa_jsc_ose_2021.pdf

Version	Datum	Änderung
----------------	--------------	-----------------

0.0.1	29.03.2021	Initialisierung des Dokuments
0.0.2	29.03.2021	Projektmanagementmethode beschrieben
0.0.3	29.03.2021	Systembeschreibung abgeschlossen
0.0.4	30.03.2021	Ist/Soll-Vergleich
0.0.5	30.03.2021	Namenskonzept
0.0.6	30.03.2021	Personas, Versionsverwaltungs- und Backupkonzept
0.0.7	31.03.2021	OAuth2 Strategie erstellt
0.1.0	31.03.2021	Analyse abgeschlossen
0.1.1	31.03.2021	Systementwurf begonnen
0.1.2	01.04.2021	Diagramme und Arbeitsjournale
0.1.3	01.04.2021	Systementwurf abgeschlossen
0.1.4	01.04.2021	Mockups erstellt und dokumentiert
0.1.5	06.04.2021	Testkonzept erarbeitet
0.2.0	07.04.2021	Entwurf finalisiert und Analyse korrigiert
0.2.1	09.04.2021	Implementation begonnen

Version	Datum	Änderung
----------------	--------------	-----------------

0.3.0	13.04.2021	Implementation fertiggestellt
0.3.1	13.04.2021	Testing begonnen
0.4.0	13.04.2021	Testing abgeschlossen
0.5.0	13.04.2021	Abschluss erstellt
0.5.1	14.04.2021	Glossar erstellt
0.5.2	14.04.2021	Korrekturen einarbeitun begonnen

DANKSAGUNG

Zunächst möchte ich Markus Strittmatter und Endress+Hauser für diese Ausbildung danken. Nur durch Sie konnte ich in den letzten vier Jahren meine Leidenschaft und mein Können für das Programmieren weiterentwickeln. Dafür bin ich froh, da ich nun weiss, dass Informatik das Richtige für mich ist. Es ist auch nicht selbstverständlich, einen Lehrling die Fachrichtung wechseln zu lassen. Dies erlaubte mir mich auch in der Schule voll und ganz auf das Programmieren zu fokussieren.

Gerne würde ich mich auch bei Valentino Rusconi und Marco Roth bedanken. Konnte ich etwas nicht alleine bewältigen, nahmen Sie sich die Zeit, um mir komplexe Konzepte zu erklären oder bei schwierigen Fehlern zu helfen.

Abschliessend möchte ich mich auch recht herzlich bei Lisa Marie Hüglin, Robert Kölblin und Simon Jäggi für das Korrekturlesen bedanken.

KURZFASSUNG

Kurze Ausgangssituation

Die Endress+Hauser Gruppe verwendet Austellungsmodelle, um verschiedene Messgeräte aus dem eigenen Produktpotfolio an Messen oder anderen offiziellen Anlässen vorzustellen. Die Messdaten der an den Modellen gezeigten Geräte können von den Kunden mittels dem IIoT Angebot «Netilion» angezeigt und ausgewertet werden. Ergänzend zu den durch Endress+Hauser angebotenen Standart Services, können Kunden zusätzlich eine Netilion Connect Subscription abschliessen. Eine solche Subscription erlaubt es dem Kunden auf die REST API der Netilion Cloud zuzugreifen, wodurch eigene Applikationen mit den Daten der Messgeräte erstellt werden können. Um den Kunden die Möglichkeiten der Netilion Connect Subscription zu zeigen, wurde das «OSE-Dashboard» erstellt. Dieses zeigt eine 3D Ansicht eines Austellungsmodells, das mittels Kameraführung betrachtet werden kann. Um zusätzliche Informationen über das dargestellte Modell zu zeigen, kann man mit den einzelnen Messgeräten interagieren, um beispielsweise deren aktuellen Status anzuzeigen. Dieses Dashboard wurde statisch für ein bestimmtes Austellungsmodell in Reinach aufgebaut und ist nicht für andere Austellungsmodelle benutzbar. Damit entsteht die Problematik, dass bei Messen mit anderen Austellungsmodellen das Dashboard nicht gezeigt werden kann.

Umsetzung

Um das Dashboard für alle weltweit verteilten Ausstellungsmodelle verwenden zu können, wurde mittels NextJS im Frontend und NestJS im Backend die Webapplikation erweitert. Zuerst wurden neue Entitäten in NestJS erstellt und bereits vorhandene angepasst. Anschliessend wurden die passenden Controller und Services erstellt, um dem Frontend eine Schnittstelle anbieten zu können. Dieses fragt Daten davon ab und visualisiert sie. Dadurch wurde eine Weltkarte, die Registrierung, das Konfigurationsmenü und die dynamische Darstellung des 3D Modells möglich.

Ergebnis

Das Ergebnis ist ein erweitertes OSE Dashboard. Modellbetreiber auf der ganzen Welt können sich mittels OAuth2 anmelden und die eigenen Austellungsmodelle und deren Messgeräte erfassen. Somit kann das Dashboard dynamisch für alle Messen der Endress+Hauser Gruppe verwendet werden.

Inhaltsverzeichnis

I Obligatorische Kapitel	1
1 Obligatorische Dokumentation	2
1.1 Ausgangslage	2
1.2 Detaillierte Aufgabenstellung	3
1.3 Mittel und Methoden	4
1.4 Vorkenntnisse	4
1.5 Vorarbeiten	4
1.6 Neue Lerninhalte	5
1.7 Arbeiten in den letzten sechs Monaten	5
1.8 Projektaufbauorganisation	5
1.9 Durchführungsblock	6
1.10 IPA-Termine	6
1.11 Projektplan	7
1.12 Arbeitsjournale	8
1.12.1 Arbeitsjournal vom 29.03.2021	8
1.12.2 Arbeitsjournal vom 30.03.2021	9
1.12.3 Arbeitsjournal vom 31.03.2021	10
1.12.4 Arbeitsjournal vom 01.04.2021	11
1.12.5 Arbeitsjournal vom 06.04.2021	12
1.12.6 Arbeitsjournal vom 07.04.2021	13
1.12.7 Arbeitsjournal vom 08.04.2021	14
1.12.8 Arbeitsjournal vom 09.04.2021	15
1.12.9 Arbeitsjournal vom 13.04.2021	16
II Dokumentation	17
2 Projektmanagement	18
2.1 Entscheidung	18
2.2 Wasserfallmodell	19
2.2.1 Initialisierung des Dokuments	20
2.2.2 Analyse	20
2.2.3 Entwurf	20

2.2.4	Implementierung	21
2.2.5	Testen	21
2.2.6	Abschluss	21
3	Analyse	22
3.1	Systembeschreibung	22
3.1.1	Systemarchitektur	22
3.1.2	One Story Exhibit	23
3.1.3	Netilion	23
3.1.4	here developer API	24
3.1.5	OSE-Dashboard: Backend	25
3.1.6	OSE-Dashboard: Frontend	27
3.2	Ist/Soll-Vergleich	28
3.2.1	Ist Zustand	28
3.2.2	Soll Zustand	29
3.3	Namenskonzept	30
3.3.1	User-Stories	30
3.3.2	Akzeptanzkriterien	30
3.4	Versionskontrolle	30
3.4.1	Git Flow	31
3.4.2	Pipelines zu Vercel und Heroku	32
3.4.3	Dokumentation	32
3.5	Backupkonzept	32
3.5.1	Sicherheit der Daten	33
3.6	Personas	33
3.6.1	Endnutzer	33
3.6.2	MVT	33
3.6.3	OSE Verantwortlicher	34
3.6.4	Entwickler	34
3.7	User Stories	34
3.7.1	US-01	34
3.7.2	US-02	34
3.7.3	US-03	34
3.7.4	US-04	35
3.7.5	US-05	35
3.7.6	US-06	35
3.8	OAuth2 Strategie	35
3.8.1	Was ist OAuth2?	35
3.8.2	Anwendung im OSE-Dashboard	36
4	Entwurf	38

4.1	Generelles Abfragen von Daten	38
4.2	Zugriffskontrolle	40
4.2.1	OSE Verantwortlicher	40
4.2.2	OSE-Dashboard Nutzer	41
4.3	Softwareschnittstellen	41
4.3.1	Schnittstellen im OSE	42
4.3.2	Netilion	42
4.3.3	Backend	46
4.3.4	here developer API	47
4.3.5	Frontend	48
4.4	Mockups	48
4.4.1	Index Page	49
4.4.2	Registration	50
4.4.3	Einstellungsmenü	56
4.5	Testkonzept	58
4.5.1	Fehlerklassen	59
4.5.2	Test Case Schema	59
4.5.3	User-Stories Abdeckung	60
4.5.4	Test-Cases	61
4.6	Datenbankerweiterung	69
4.6.1	User	69
4.6.2	Model	69
4.6.3	Mesh	70
4.7	Authentifizierung mit JWT	70
4.7.1	Was ist ein JWT	70
4.7.2	Anwendung	71
5	Implementierung	72
5.1	OAuth2	73
5.1.1	Frontend	73
5.1.2	Backend	74
5.2	Account löschen	76
5.3	Modellauswahl	77
5.3.1	Auflistung mit Landesflaggen	77
5.3.2	Interaktive Weltkarte	77
5.3.3	Tatsächliche Auswahl	78
5.4	Automatische Verlinkung von Assets und Meshes	78
5.5	Einhaltung der Coding Guidelines	79
5.6	Umgebungsvariablen	80
5.6.1	Umgebungsvariablen - Frontend	80
5.6.2	Umgebungsvariablen - Backend	81

5.7 3D Modell auswechseln	81
6 Testen	84
6.1 Testergebnisse	84
7 Abschluss	85
7.1 Bekannte Fehler	85
7.1.1 Konfigurationsmenü	85
7.1.2 Verlinkung von Assets zu Meshes	85
7.2 Verbesserungsmöglichkeiten	86
7.2.1 Layout verschiebungen	86
7.2.2 Refresh Token verschlüsseln	86
7.2.3 Unit test/-s	86
7.3 Verstöße gegen die Coding Guidelines	87
7.3.1 Frontend	87
7.3.2 Backend	88
7.4 Schlussbetrachtung	88
7.4.1 Reflexion	88
7.4.2 Schlusswort	89
7.4.3 Persönliche Bilanz	89
III Anhang	90
A Glossar	91
B Abbildungsverzeichnis	95
C Quellenverzeichnis	97
D Coding Guidelines	98

Teil I.

Obligatorische Kapitel

KAPITEL 1

OBLIGATORISCHE DOKUMENTATION

1.1. Ausgangslage

In der Endress+Hauser Gruppe gibt es ein Messemodell mit dem Namen One Story Exhibit (OSE Modell). Dieses OSE Modell gibt es mehrfach in der gleichen Ausführung. Diese Modelle sind weltweit in den Endress+Hauser Sales Center verteilt. Dieses Modell bietet einen interaktiven Einblick in das Produkt Portfolio von Endress+Hauser Digital Solutions. Der Kunde kann unter anderem unser IIOT Angebot Netilion interaktiv erleben. Aktuell werden dafür unsere Netilion Standard Services wie z.B. Analytics, Health und Value verwendet. Dabei werden die Daten der Messgeräte via Edge Device in unserer IIOT Cloud gespeichert. Via Netilion Standard Web Applikationen kann man z.B. den aktuellen Health Status der Messgeräte sowie den aktuellen Messwert sehen.

Die Web Applikation „OSE Dashboard“ soll dem Kunden ein Beispiel für die Verwendung der Netilion Connect Subscription zeigen. Mit einer Netilion Connect Subscription erhält der Kunde Zugriff auf die REST API unserer Netilion Cloud und kann somit eigene IIOT Anwendungen entwickeln oder die Daten aus Netilion in seine eigene Cloud importieren. Die Applikation „OSE Dashboard“ zeigt das OSE Modell in einer 3D Ansicht. Man kann über eine Kameraführung an die Messgeräte heran zoomen und den Health Status des Messgerätes anzeigen lassen. Aktuell ist die Applikation nur mit dem OSE Modell in Reinach nutzbar, weil die Applikation nur mit den Daten der Messgeräte dieses Modells in Netilion verlinkt ist.

Mit dieser IPA soll die Web Applikation „OSE Dashboard“ so erweitert werden, dass sie für andere OSE Modelle mit gleichem Aufbau verwendet werden kann.

1.2. Detaillierte Aufgabenstellung

Die bestehende Web Anwendung „OSE Dashboard“ zeigt das One Story Exhibit Model (OSE Modell), welches in Reinach steht, in einer 3D Ansicht an. Darin werden die Daten der Messgeräte aus der Netilion Cloud gelesen und ebenfalls dargestellt.

Das Ziel dieses Projekts ist, dass die Web Anwendung auch für andere OSE Modelle, welche dem gleichen Aufbau haben, verwendet werden kann, ohne dass zukünftig eine Änderung an der Applikation notwendig ist.

Die bestehende Anwendung muss dafür so erweitert werden, dass Verlinkung des 3D Modells mit den Daten der Messgeräte nicht mehr im Source Code hinterlegt ist. Es muss ein Weg gefunden werden, diese Verlinkung für alle bestehenden und zukünftigen OSE Modelle zu speichern.

Laut Autraggeber haben alle OSE Modelle den gleichen Aufbau mit Messgeräten vom gleichen Typ. Auch die Bezeichnung der Messgeräte sollte bei allen Modellen gleich sein. Wird die Anwendung das erste mal für ein OSE Modell verwendet, muss für alle Geräte aus diesem OSE Modell die Verlinkung mit dem 3D Modell der Anwendung automatisch erfolgen und gespeichert werden. Sollte es aber Messgeräte geben, die nicht vom gleichen Typ sind, weil sie z.B. durch eine neuere Version ersetzt wurden, dann soll der Anwender die Möglichkeit haben über ein Konfigurationsmenü die Verlinkung vorzunehmen.

Änderungen an der Konfiguration dürfen nicht von jedem User vorgenommen werden. Das Konfigurationsmenü darf nur von Usern geöffnet werden, welche die entsprechende Berechtigung haben. Dafür soll in Netilion eine User Gruppe erstellt werden. Alle User dieser Gruppe dürfen dann die Konfiguration ändern.

Die Anwendung selber soll aber weiterhin ohne Login aufrufbar sein. Der Anwender soll als Datenquelle für das 3D Modell zwischen den verschiedenen integrierten Standorten wählen können. Da jedes OSE Modell einen eigenen User hat, müssen die User Credentials der einzelnen OSE Modelle ebenfalls in der Applikation gespeichert werden. Dabei muss darauf geachtet werden, dass diese Daten sicher gespeichert werden und der Anwender keinen Zugriff darauf hat.

Die Methoden mit der Logik für die automatische Verlinkung der Messgeräte mit dem 3D Modell soll automatisiert getestet werden.

Für den Test der kompletten Applikation sind manuelle Tests ausreichend. Dabei sind folgende Testfälle zu beachten:

- Konfigurationsmenu nur mit entsprechender Berechtigung aufrufbar
- Alle Messgeräte werden automatisch verlinkt.
- Messgeräte können nicht automatisch verlinkt werden .
- User kann ohne Login zwischen integrierten OSE Modellen wechseln.
- Credentials der OSE Modelle sind sicher gespeichert

Die Tests sollen zuerst an dem OSE Modell in Reinach durchgeführt werden. Bei diesem Modell können auch für Tests die Daten in Netilion mal abgeändert werden. Zum Abschluss sollen 2 weitere OSE Modelle integriert werden um die Funktionalität der Anwendung mit mehreren OSE Modellen zu testen.

1.3. Mittel und Methoden

Anbei werden Mittel und Methoden aufgelistet, welche von dieser IPA gefordert werden:

- HTML & CSS
- JavaScript
- Node.js JavaScript Runtime Environment für Desktop und Server
- React Eine von FaceBook entwickelte JavaScript Bibliothek, welche die Strukturierung von Komponenten basierten Benutzeroberflächen erleichtert.
- JSX Ein Syntax welcher das übliche JavaScript erweitert. Es ermöglicht das Vermischen von HTML mit JavaScript und wird von React verwendet
- Three JavaScript Bibliothek, welche es ermöglicht 3D Modelle in einem HTML Canvas darzustellen
- react-three-fiber JavaScript Bibliothek, welche auf Three aufbaut und das ganze in React verfügbar macht
- GLTF Dateiformat für 3D Modelle, welches sich am besten für das Web eignet

Hosting der Web Applikation

- Heroku
- Vercel

Entwicklungsumgebung

- Macbook Pro 2018 mit Dockingstation und 2 externen Monitoren. (Ersatzweise steht ein Windows Laptop zur Verfügung)
- MacOS Big Sur
- Visual Studio Code
- GitHub

1.4. Vorkenntnisse

Alle Tools und Techniken sind dem Lernenden schon bekannt. Er hat alle Tools, Programmiersprachen und Techniken schon in mehreren Projekten während der Ausbildung angewendet.

1.5. Vorarbeiten

Die Version 1 des OSE Dashboards wurde als Vorarbeit zu dieser IPA vom Lernenden entwickelt.

1.6. Neue Lerninhalte

In dieser IPA gibt es keine neuen Lerninhalte.

1.7. Arbeiten in den letzten sechs Monaten

Entwickeln einer Webanwendung, welche Daten aus der Endress+Hauser IIOT Plattform Netilion darstellt. Diese Anwendung zeigt den Wasserverbrauch der Kaffeemaschinen in der Pausenzone an und berechnet die Anzahl der konsumierten Tassen. Diese Anwendung dient dazu, unseren Kunden das Angebot Netilion Connect zu erklären.

Entwickeln einer weiteren Webanwendung, welche in Verbindung mit Netilion Connect Daten aus der Endress+Hauser IIOT Plattform darstellt. Dies ist die erste Version des OSE Dashboards, welches ein Messemodell in 3D darstellt und den „Gesundheitszustand“ der Messgeräte anzeigt. Diese Webanwendung wird im Rahmen dieser IPA erweitert.

1.8. Projektaufbauorganisation

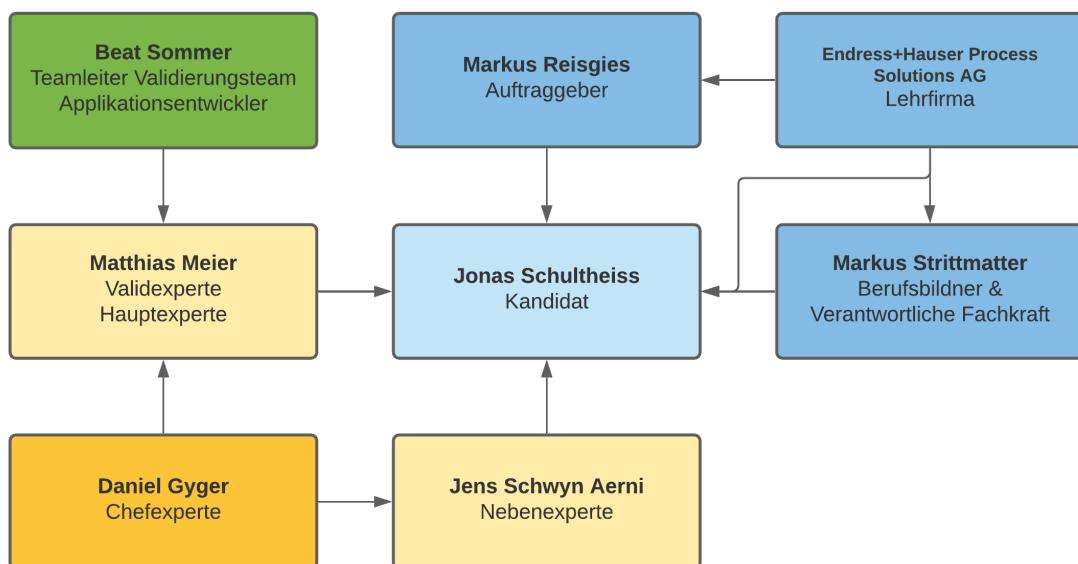


Abbildung 1.1.: Diagramm der Projektaufbauorganisation

1.9. Durchführungsblock

Startblock 2: 29.03.2021 – 16.04.2021

PA-Durchführung: 29.03.2021 – 14.05.2021

Einreichung bis: 31.01.2021

1.10. IPA-Termine

Antrittsgespräch: 23.03.2021 08:30

Erster Zwischenbesuch: 01.04.2021 09:00

Zweiter Zwischenbesuch: 08.04.2021 09:45

Endbesuch: 28.04.2021 09:00

1.11. Projektplan

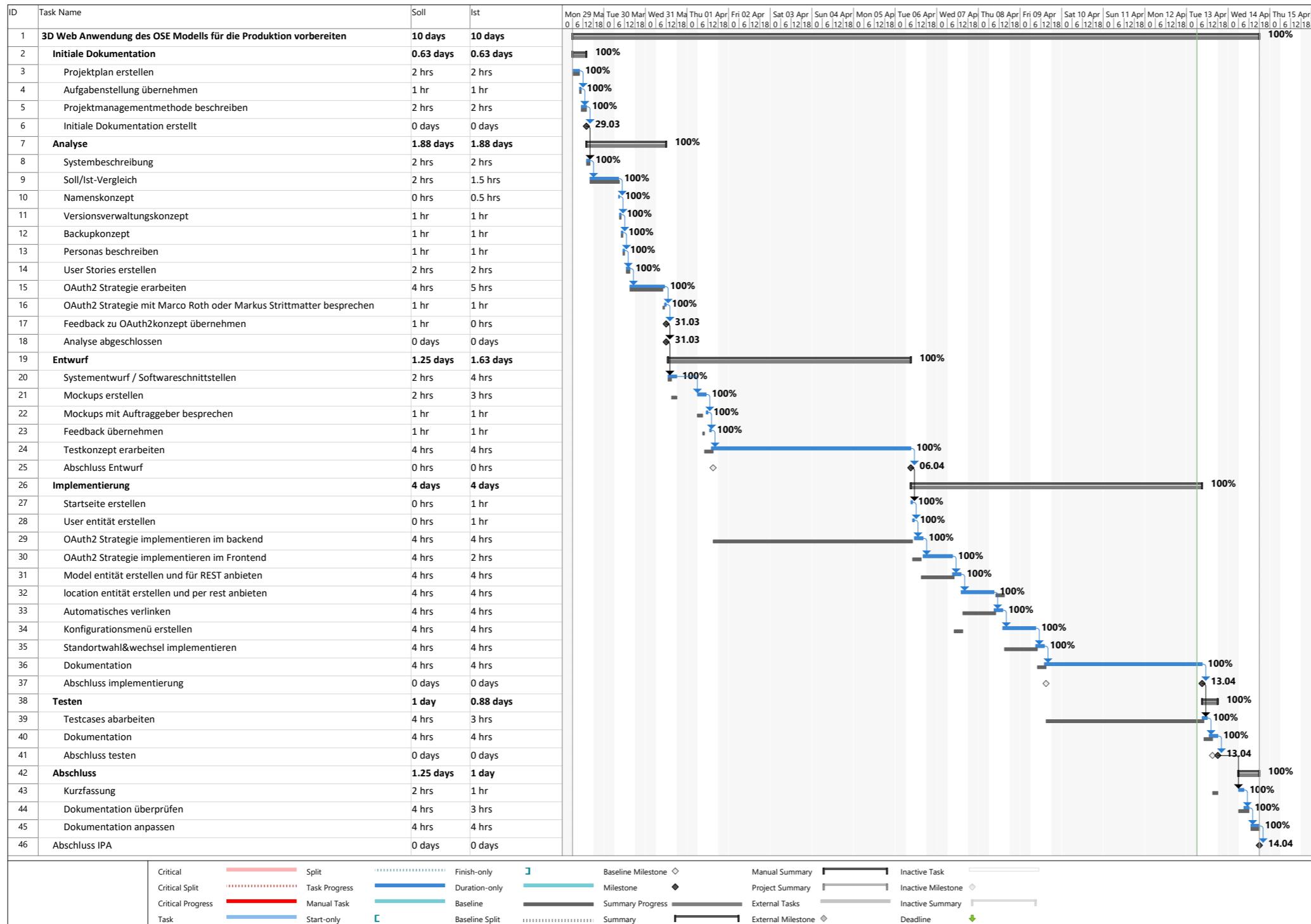


Abbildung 1.2.: Projektplan, erstellt mit Microsoft Projekts

1.12. Arbeitsjournale

1.12.1. Arbeitsjournal vom 29.03.2021

Ziel	Beschreibung	Geplante Zeit	Benötigte Zeit	Abweichung
1	Projektplan erstellen	2 hrs	2 hrs	0 hrs
2	Aufgabenstellung übernehmen	1 hrs	1 hrs	0 hrs
3	Projektmanagementmethode beschreiben	2 hrs	2 hrs	0 hrs
4	Systembeschreibung	2 hrs	2 hrs	0 hrs
5	Soll/Ist-Vergleich	1 hrs	1 hrs	0 hrs
Total		8 hrs	8 hrs	0 hrs

Ziel 1: Projektplan erstellen

Die IPA begann mit der Erstellung des detaillierten Projektplans. Dies konnte ich zügig während zwei Stunden machen, da ich vor der IPA schon Erfahrungen mit Microsoft Projekt sammeln konnte.

Ziel 2: Aufgabenstellung übernehmen

Dieser Task war sehr leicht, da man einfach nur Texte kopieren und einfügen musste. Ich habe die Texte eins zu eins gleich gelassen, ausser, dass ich aus den Auflistungen normale LaTeX Auflistungen gemacht habe. Ich hoffe dies wird später nicht zu einem Problem.

Ziel 3: Projektmanagementmethode beschreiben

Als nächstes sollte ich die Projektmanagementmethode beschreiben. Ich habe mich im Vorfeld bewusst auf die Wasserfallmethode entschieden, da das Ziel und die Aufgaben klar sind in diesem Auftrag. Trotzdem habe ich im ersten Teil nochmals beschrieben, wieso ich zu dieser Entscheidung kam, mithilfe der Stacey Matrix. Darauf folgte dann mein Wasserfall mit passenden Erklärungen, was zu welchem Abschnitt dazugehört.

Ziel 4: Systembeschreibung

In diesem Abschnitt habe ich sehr ausführlich beschrieben wie das System aussieht und aus welchen Teilsystemen es besteht. Da ich es zeitlich ermöglichen konnte, habe ich die einzelnen Abschnitte vertieft bearbeitet.

Ziel 5: Soll/Ist-Vergleich

Ich konnte mich bei einigen Punkten auf die ausführliche Erklärung des vorherigen Kapitels beziehen, was das Ganze etwas einfacher machte.

Reflexion

Ich stand gestern Abend und heute Morgen sehr unter Stress, da ich nicht genau wusste, was mich erwartet oder ob ich unerwarteten Problemen entgegen werde. Nun, am Ende des ersten Tages bin ich beruhigt und zufrieden mit meinem Fortschritt. Mir ist allerdings aufgefallen, dass ich das Namenskonzept und die Erstellung von User Stories im Projektplan vergessen habe und morgen noch nachholen muss.

Reinach, 29.03.2021

Ort, Datum

Kandidat: Jonas Schultheiss

Fachvorgesetzter: Markus Strittmatter

1.12.2. Arbeitsjournal vom 30.03.2021

Ziel	Beschreibung	Geplante Zeit	Benötigte Zeit	Abweichung
1	Ist/Soll-Vergleich	1 hrs	1.5 hrs	- 0.5 hrs
2	Namenskonzept	0 hrs	0.5 hrs	+0.5 hrs
3	Versionsverwaltungskonzept	1 hrs	1 hrs	0 hrs
4	Backupkonzept	1 hrs	1 hrs	0 hrs
5	Personas beschreiben	1 hrs	1 hrs	0 hrs
6	User Stories erstellen	2 hrs	2 hrs	0 hrs
7	OAuth2 Strategie erarbeiten	2 h	2 hrs	0 hrs
Total		8 hrs	8 hrs	0 hrs

Ziel 1

Der heutige Tag began, wie gestern der Letzte endete: nämlich mit dem Ist/Soll-Vergleich. Ich konnte diesen sogar in einer kürzeren Zeit fertigstellen als gedacht.

Ziel 2

Im Anschluss habe ich mit dem Namenskonzept angefangen. Dies war nicht im originalen Projektplan aufgelistet. Da ich im vorherigen Task eine halbe Stunde gespart habe, habe ich mich sehr angestrengt, um diesen Task in der restlichen halben Stunde zu erledigen, wodurch es nicht zu weiteren Verschiebungen gekommen ist.

Ziel 3 und 4

Beide Ziele sind mir gut gelungen. Mir gefällt sehr, dass falls etwas schieflaufen würde, ich immer ein Stündliches Backup zur Verfügung habe.

Ziel 5

Darauffolgend habe ich die Personas beschrieben. Dies sind die Anspruchsgruppen des Projektes.

Ziel 6

Bei den User-Stories hatte ich etwas Probleme, die ich mir selbst machte. Ich konnte die Stories nie komplett aus den Augen der User erstellen und musste so mehrmals, alles nochmals durchlesen.

Ziel 7

Beendet habe ich den Tag mit dem OAuth2konzept. Bisher habe ich mein theoretisches Wissen nochmals aufbereitet und an einem Sequenzdiagramm gearbeitet, welches jedoch noch nicht fertiggestellt ist. Ich denke ich werde morgen auf die Deadline des Tasks fertig. Jedoch kann es sein, dass ich morgen einen Fehler im Diagramm finde. Das Diagramm ist schwer zu editieren, was je nach dem andere Tasks nach hinten verschieben kann.

Reflexion

Ich war heute zum Glück nicht mehr so gestresst wie gestern. So konnte ich den Morgen direkt gut nutzen, um vorwärts zu kommen. Ich bin auch froh, dass mein Zeitplan wieder im grünen ist. Gestern ist mir aufgefallen, dass ich das Namenskonzept vergessen hatte im Projektplan einzutragen. Nun hoffe ich einfach, dass das Fertigstellen des OAuth2konzeptes morgen nicht länger braucht als anfangs gedacht.

J. Schultheiss

Reinach, 30.03.2021

M. Strittmatter

Ort, Datum

Kandidat: Jonas Schultheiss

Fachvorgesetzter: Markus Strittmatter

1.12.3. Arbeitsjournal vom 31.03.2021

Ziel	Beschreibung	Geplante Zeit	Benötigte Zeit	Abweichung
1	OAuth2 Strategie erarbeiten	2 hrs	3 hrs	+1 hrs
2	OAuth2 besprechen mit Markus	1 hrs	1 hrs	0 hrs
3	Mockups erstellen	0 hrs	1 hrs	+ 1 hrs
4	Feedback übernehmen	1 hrs	0 hrs	-1 hrs
5	Systementwurf erstellen	2 hrs	3 hrs	+1 hrs
Total		6 hrs	8 hrs	+3 hrs

Ziel 1

Heute Morgen habe ich als erstes die OAuth2-Strategie fertiggestellt. Dafür habe ich ein Sequenzdiagramm erstellt. Die einzelnen Schritte habe ich dabei nummeriert und wollte diese im Anschluss in der Dokumentation beschreiben. Leider kann die von mir verwendete Software (lucidchart.com) diese Schritte nicht selbst nummerieren. Während dem Beschreiben der Schritte fiel mir mehrmals Fehler auf, welche ich in der Grafik korrigierte. Ich musste jedes Mal die Nummerierung aller nachfolgenden Schritte manuell ändern, was mich Zeit gekostet hat. Ich habe mir allerdings das Thema OAuth2 nochmals angeeignet, weswegen ich die Implementierungszeit bei beiden Tasks um eine Stunde gekürzt habe.

Ziel 2

Anschliessend habe ich das Ganze mit Markus Strittmatter besprochen, welcher mit dem Resultat einverstanden war.

Ziel 3

Ich musste zuerst warten, bis Markus mit einem Meeting fertig war. Die Zeit habe ich genutzt, um schon einmal mit den Mockups zu beginnen. Diese Stunde kann ich morgen an den Mockups sparen.

Ziel 4

Nichtig, da keine Änderungen gewünscht waren.

Ziel 5

Dieser Task hatte original den Namen «Softwareschnittstellen». Jedoch habe ich bei der originalen Erstellung des Zeitplans vergessen, dass ich laut Leitfrage eins, einen ausführlichen Systementwurf brauche. Ich habe nun den Task verändert, dass er «Systementwurf» heisst, aber die Softwareschnittstellen und mehr enthält. Ich habe ihn dabei auf vier Stunde eingeschätzt, wovon ich drei gleich heute genutzt habe.

Reflexion

Heute war wieder ein stressiger Tag. Ich habe nun zwei Tasks, welche länger brauchen/brauchten als ich original geplant habe. Bei der Erstellung des Projektplans habe ich keine Buffers, ausser vielleicht den letzten Tag, eingebaut, was mich jetzt etwas einengt (und stresst). Solange ich morgen die Mockups in einer Stunde fertig bekomme geht es auf. Ich werde mich morgens nochmals anstrengen, damit ich im Zeitplan bleibe.




Reinach, 31.03.2021

Ort, Datum

Kandidat: Jonas Schultheiss

Fachvorgesetzter: Markus Strittmatter

1.12.4. Arbeitsjournal vom 01.04.2021

Ziel	Beschreibung	Geplante Zeit	Benötigte Zeit	Abweichung
1	Systementwurf	1 hrs	1 hrs	0 hrs
2	Mockups erstellen	2 hrs	3 hrs	+1 hrs
3	Testkonzept	4 hrs	4 hrs	0 hrs
Total		7 hrs	8 hrs	+1 hrs

Ziel 1

Als erstes habe ich den Systementwurf fertiggestellt. Dafür habe ich das ganze System nochmals im Detail mit Texten und Diagrammen erklärt.

Ziel 2

Anschliessend habe ich die Mockups erstellt. Hiermit hatte ich keine Probleme. Es war allerdings eine Fleissarbeit und hat etwas mehr Zeit benötigt, als ich original eingeplant hatte. Nun sind sie meiner Meinung nach fertig und müssen nur noch mit Markus Strittmatter besprochen werden.

Ziel 3

Auch die Erstellung des Testkonzeptes hat viel Zeit in Anspruch genommen. Ich habe dabei zuerst definiert, wie der Test benannt und strukturiert ist und habe dann Anhand den User-Stories die Test-Cases entworfen. So bin ich mir sicher, dass alle Anforderungen der Anspruchsgruppen durch Tests gedeckt sind. Ich werde diesen Task am Dienstag noch kurz fertigstellen, da ich heute nicht vollständig fertig wurde.

Leitfrage 5

Im Kapitel «Systementwurf» habe ich die Leitfrage 5 beantwortet und dies später mit Markus Strittmatter besprochen. Er meinte, dass meine Texte mit Diagrammen verständlicher gemacht werden können. Im Anschluss habe ich ihm nochmals den Fortschritt gezeigt, woraufhin er zufrieden war.

Besuch der beiden Experten

Heute bekam ich Besuch von meinen Experten. Ich habe sie am Empfang abgeholt und zu meinem Arbeitsplatz geführt. Als wir dort ankamen fragten sie mich über den aktuellen Stand, den Arbeitsjournals und dem aktuellen Projektplan. Wir unterhielten uns noch ein wenig. Ich denke sie haben einen positiven Eindruck vom aktuellen Stand erhalten.

Reflexion

Heute war ein Tag voller Fleissarbeiten. Obwohl ich eine Stunde hintendran bin, bin ich mit meiner Leistung zufrieden - ich freue mich darauf, am kommenden Dienstag endlich mit dem Implementieren beginnen zu können. (Ausserdem freue ich mich auf das Wochenende). Da Markus heute etwas früher gehen musste, werde ich die Mockups mit ihm am Montag besprechen. Ausserdem sollte ich am Dienstagmorgen nochmals das Dokument durchgehen, damit ich es den beiden Gegenlesern zustellen kann. Zudem muss ich die Programmietasks noch etwas aufbrechen.

Reinach, 01.04.2021

Ort, Datum

J. Schultheiss

Kandidat: Jonas Schultheiss

M. Strittmatter

Fachvorgesetzter: Markus Strittmatter

1.12.5. Arbeitsjournal vom 06.04.2021

Ziel	Beschreibung	Geplante Zeit	Benötigte Zeit	Abweichung
1	Testkonzept fertigstellen	1 hrs	1 hrs	0 hrs
2	Startseite erstellen	1 hrs	1 hrs	0 hrs
3	User Entität erstellen	1 hrs	1 hrs	0 hrs
4	OAuth2 Backend	3 hrs	4 hrs	+1 hrs
5	OAuth2 Frontend	2 hrs	1 hrs	0 hrs
Total		8 hrs	8 hrs	+1 hrs

Ziel 1

Als erstes habe ich das Testkonzept fertiggestellt, der Aufwand betrug dabei nur noch eine Stunde.

Ziel 2

Danach habe ich die Startseite im Frontend ersetzt. Vorher war das 3D-Model direkt zu sehen. Dies habe ich nun geändert und durch die im Mockup beschriebene Startseite ersetzt. Ich hatte etwas Probleme mit der Image Componente von Nextjs, weswegen der «Sign in with Netilion» Button nun kein E+H Logo besitzt.

Ziel 3

Anschliessend habe ich die User Entität und der dazugehörende Service im Backend erstellt. Dies stellte kein Problem dar. Diese Entität wird im nächsten Schritt verwendet.

Ziel 4

Nachdem ich den User erstellt hatte, habe ich mich an OAuth2 gemacht. Begonnen habe ich dabei im Backend. Ich hatte dabei einmal ein Verständnisproblem mit der Netilion API und zwei Mal Probleme mit den Modulen. Dies hat mich insgesamt eine Stunde lang aufgehalten, weswegen ich für den ganzen Task etwas länger brauchte.

Ziel 5

Diesen Task konnte ich heute leider nicht fertig machen, da ich im Backend eine Stunde länger brauchte als gedacht. Die momentan bestehenden Views/Komponenten sehen schön und minimalistisch aus und funktionieren. Ich denke nicht, dass ich morgen länger als eine Stunde aufwand habe, um diesen Task fertigzustellen.

Reflexion

Heute war der erste Tag der IPA, an dem ich programmieren konnte. Dies hat mir sehr gefallen, da sich die letzten Tage nur dokumentiert habe.

Reinach, 06.04.2021

Ort, Datum

J. Schultheiss

Kandidat: Jonas Schultheiss

M. Strittmatter

Fachvorgesetzter: Markus Strittmatter

1.12.6. Arbeitsjournal vom 07.04.2021

Ziel	Beschreibung	Geplante Zeit	Benötigte Zeit	Abweichung
1	OAuth2 Frontend	1 hrs	1 hrs	0 hrs
2	Model entität	4 hrs	4 hrs	0 hrs
3	Location entität	3 hrs	3 hrs	0 hrs
Total		8 hrs	8 hrs	+0 hrs

Ziel 1

Heute habe ich als erstes den OAuth2 teil im Frontend abgeschlossen. Ich hatte Probleme, welche nun etwas schwierig zu beschreiben sind. Lief etwas schief, habe ich auf /register/error weitergeleitet. Unter anderem habe ich die useEffect Hook verwendet, um gewisse Prozesse mit dem Lifecycle der Komponente zu steuern. Einige dieser Prozesse enthielten Weiterleitungen. Long Story short: Weiterleitungen haben die «router» Instanz verändert, wodurch redender getriggert wurden, welche Weiterleitungen verursachten usw. Ich habe so mehrere undendliche Schleifen verursacht. Diese sind schwer zu debuggen, da der Browser nach kurzer Zeit mein ganzer Prozessor besetzt.

Ziel 2

Nachdem ich dieses Problem gelöst habe, habe ich mich an die Modelkomponente gemacht. Die Schwierigkeit hierbei war, dass die Modelle in einem Intervall mit unterschiedlichen access_tokens aktualisiert werden sollten. Ich behielt jedoch die Übersicht und konnte dies im geplanten Zeitrahmen erledigen.

Ziel 3

Der letzte Task für heute war die Erstellung der Location Entität, welche vom Model benötigt wird. Dank meiner Nestjs Erfahrung war dies eher eine Fleissarbeit. Auch die Einbindung der Adressen API verlief ohne grosse Probleme und war eher eine Fleissarbeit.

Reflexion

Ich bin heute wieder erfolgreich in den «Developer-Mindset» gekommen. Ich konnte heute dadurch viel Erreichen, ohne Minus zu machen. Morgen kommen nochmals die Experten vorbei. Ich werde mich morgen nochmals auf ihren Besuch vorbereiten und eventuell die Dokumentation als Ganzes nochmals ausdrucken.

Ausserdem habe ich einige Tasks im Projektplan verschoben, damit es vom Implementieren her mehr Sinn macht.

Reinach, 07.04.2021

Ort, Datum

J. Schultheiss

Kandidat: Jonas Schultheiss

M. Strittmatter

Fachvorgesetzter: Markus Strittmatter

1.12.7. Arbeitsjournal vom 08.04.2021

Ziel	Beschreibung	Geplante Zeit	Benötigte Zeit	Abweichung
1	Location entität	1 hrs	1 hrs	0 hrs
2	Automatisches verlinken	4 hrs	4 hrs	0 hrs
3	Konfigurationsmenü	3 hrs	3 hrs	0 hrs
Total		8 hrs	8 hrs	+0 hrs

Ziel 1

Ich habe heute die Location Resource fertiggestellt. Dabei hatte ich keine Probleme oder Schwierigkeiten.

Ziel 2

Danach habe ich die automatische Verlinkung implementiert. Schwierigkeiten hatte ich nicht wirklich. Ein Problem welches nun allerdings besteht ist folgendes: Ich muss die Namen der Messstellen in meiner Datenbank haben, damit ich sie mit den Messgeräten verlinken kann. Das verwendete ORM bietet kein Seeding an, weswegen ich manuell eine Migration erstellt habe, welche die Daten einfügt. **Dies sollte nicht so gelöst werden und muss nach der IPA geändert werden.** Stattdessen soll richtiges Seeding verwendet werden oder die ganze Resource vom Frontend steuerbar sein.

Ziel 3

Der letzte Task für heute war ein Frontendtask. Dies bietet eine Abwechslung zu den letzten beiden Tagen. Probleme oder Schwierigkeiten hatte ich keine.

Expertenbesuch

Die beiden Experten besuchten mich heute das zweite Mal. Dabei haben sie sich das Arbeitsjournal und den Projektplan angesehen. Zudem wollten sie wissen, ob ich etwas zeigen kann was ich implementiert habe. Ich wollte ihnen den Anmeldeprozess zeigen. Beim ersten Versuch wurde ein Fehler angezeigt. Verwundert habe ich zuerst meine Experten angesehen und dann die Logs des Backends. Es stellte sich heraus, dass Redis nicht gestartet war. Dies hielt das Backend vom Starten ab, woraufhin das Frontend beim Login einen 404 Fehler erhielt. Daraufhin habe ich Redis gestartet. Daraufhin gab ich dem Anmeldeprozess noch einen Versuch und wieder funktionierte dieser nicht. Ich hatte eine Ahnung an was dies liegen könnte, jedoch wollte ich nicht auf die Zeit der Experten debuggen. Ich habe ihnen kurzerhand eine Bildschirmaufnahme gezeigt, wo es noch funktionierte.

Das Problem war folgendes. ich befand mich auf einem Feature Branch als die Experten vorbeikamen. Ich wechselte für die Demo auf den Develop Branch und «stash»-te meine anderen Änderungen. Aus irgendeinem Grund fehlte der Controller, welcher für den Anmeldeprozess verantwortlich war, weswegen das Frontend immer einen 404 erhielt, wenn es eine POST-Request an /auth/login sendete.

Korrektur von Lisa Marie Hüglin

Das Kapitel «Analyse» wurde von Lisa durchgelesen und korrigiert. Die Korrektur habe ich in meine Dokumentation übernommen.

Reflexion

Heute war ein spannender Tag. Ich konnte interessante Dinge implementieren und wieder etwas mehr Gestalten im Frontend. Ich bin zuversichtlich, dass die restlichen Tage stressig werden, allerdings weiß ich, das es gut kommen wird.

Reinach, 08.04.2021

Ort, Datum

J. Schultheiss

Kandidat: Jonas Schultheiss

M. Strittmatter

Fachvorgesetzter: Markus Strittmatter

1.12.8. Arbeitsjournal vom 09.04.2021

Ziel	Beschreibung	Geplante Zeit	Benötigte Zeit	Abweichung
1	Konfigurationsmenü fertigstellen	1 hrs	2 hrs	0 hrs
2	Standortwahl & Wechsel	4 hrs	4 hrs	0 hrs
3	Dokumentation begonnen	3 hrs	2 hrs	0 hrs
Total		8 hrs	8 hrs	+0 hrs
<u>Ziel 1</u>				

Heute habe ich als erstes das Konfigurationsmenü fertiggestellt. Die Menüs Model und Location waren gestern leicht umzusetzen. Einzig der Assets Tab war etwas nervig zum Implementieren. Es war nicht schwer umzusetzen dank meiner Erfahrung mit React. Jedoch war eine grosse Fleissarbeit. Schlussendlich habe ich es im geplanten Zeitrahmen geschafft.

Ziel 2

Als nächstes habe ich weiter gemacht mit der Standortwahl und dem Wechsel. Dabei konnte ich auf bestehende Ressourcen der vorherigen Schritte zurückgreifen und etwas Zeit sparen. Diese gesparte Zeit kam mir später zu Nutze, als ich etwas Probleme mit dem Abfragen der Preview des Standortes hatte. Die API von «HERE Developer» nimmt Koordinaten entgegen und sendet als Antwort ein jpeg zurück. Ich route/proxy die Request des Frontends über mein eigenes Frontend, damit der APIKey nicht leaked wird. Das Problem war folgendes. Ich musste zuerst das Bild als stream entgegennehmen, als Buffer speichern und als Observable Stream zurück ans Frontend schicken. Dies war mir Anfangs nicht klar. Mir war bewusst, was Write-/ReadStreams und Buffers sind. Allerdings hatte ich nur begrenzte Erfahrungen damit in Node.js. Dazu kommt das NestJS nicht die normalen Node.js Streams verwendet, sondern eine eigene Abstraktion verwendet. Nach gründlichem Recherchieren konnte ich diesen Fehler lösen und es im geplanten Zeitrahmen abschliessen.

Ziel 3

Nun geht es darum den erreichten Fortschritt zu dokumentieren. Ich bin gut vorangekommen und denke das es mir morgen in einer Stunde fertig reichen wird.

Reflexion

Heute konnte ich den Implementierungsteil abschliessen. Ich bin zufrieden mit dem Ergebnis, auch wenn es noch einige Ecken und Kanten hat. Schade finde ich es jetzt jedoch, dass die kommenden Tage wieder Dokumentation lastig sind. Jedoch muss ich jetzt noch die letzten beide Tage durchstehen und dann bin ich fertig.

Reinach, 09.04.2021

Ort, Datum

J. Schultheiss

Kandidat: Jonas Schultheiss

M. Strittmatter

Fachvorgesetzter: Markus Strittmatter

1.12.9. Arbeitsjournal vom 13.04.2021

Ziel	Beschreibung	Geplante Zeit	Benötigte Zeit	Abweichung
1	Dokumentation fertigstellen	1 hrs	1 hrs	0 hrs
2	Testpläne abarbeiten	3 hrs	1 hrs	-2 hrs
3	Dokumentieren	4 hrs	6 hrs	+2 hrs
Total		8 hrs	8 hrs	+0 hrs
Ziel 1				

Als erstes habe ich den Implementationsteil in der Dokumentation vollendet. Dabei habe ich detailliert beschrieben, wie das zuvor erstellte OAuth2konzept umgesetzt wurde. Leider hatte ich keine Zeit, um den Refresh Token zu verschlüsseln. Ich werde im Kapitel «Abschluss» weiter darauf eingehen und beschreiben, wie ich damit fortfahren werde. Als weiteres habe ich beschrieben, dass es in der momentanen Umsetzung nicht möglich ist, seinen Benutzer zu löschen. Ich konnte die gewünschten Features umsetzen und denke es reicht, wenn ich dies noch nach der IPA umsetze.

Ausserdem enthalten ist die Modellauswahl, beziehungsweise mein Vorgehen, wie ich mich zwischen zwei Darstellungsvarianten entschieden habe.

Ziel 2

Danach habe ich die Test-Cases abgearbeitet. Dies stellte sich als nicht sehr aufwendig heraus, wodurch ich diese Zeit als Buffer nutzen konnte, um meine Dokumentation im Allgemeinen zu verbessern.

Ziel 3

Wie gerade angesprochen, habe ich als nächstes an der Dokumentation weitergearbeitet. Zuerst habe ich den «Testing» Abschnitt erstellt. Da ich bereits alles zum Testing in der Entwurfsphase definiert habe, besteht dieses Kapitel nun aus einer Tabelle der Resultate der abgehandelten Test-Cases.

Die restliche Zeit wurde verwendet, um das Dokument allgemein zu verbessern und kleinere Kapitel noch hinzuzufügen. So habe ich noch bei der Implementierung die «Umgebungsvariablen» hinzugefügt. Dies zu dokumentieren ist meiner Meinung nach wichtig, damit andere Entwickler keine Probleme haben meine Software bei ihnen laufen zu lassen.

Reflexion

Heute war ein echt trockener Tag voll mit dokumentieren. Mir fällt es mittlerweile schwieriger mich eine längere Zeit zu konzentrieren. Vor allem nach dem Mittag fällt mir dies auf, wenn ich beginne das Mittagsessen zu verdauen.

Ich biss allerdings durch und konnte einen grossen Fortschritt erreichen. Dies fiel mir leichter mit dem Gedanken, das es ja nur noch zwei Tage sind, bis ich diese Arbeit abgeschlossen habe.

Reinach, 13.04.2021

Ort, Datum

J. Schultheiss

Kandidat: Jonas Schultheiss

M. Strittmatter

Fachvorgesetzter: Markus Strittmatter

Teil II.

Dokumentation

KAPITEL 2

PROJEKTMANAGEMENT

2.1. Entscheidung

Die Entscheidung der Projektmanagementmethode ist sehr wichtig und sollte sorgfältig gemacht werden. Sollte eine Methode falsch angewandt werden, kann sie das Team verlangsamen und bei der Arbeit hindern. Bei dieser Entscheidung hilft die Stacey Matrix.

Die x-Achse beschreibt die Verständlichkeit des Auftrags. Das heisst, ob alle Anforderungen gegeben sind oder noch welche zu einem späteren Zeitpunkt dazu kommen können. Die y-Achse beschreibt dagegen, wie deutlich der Weg zu diesen Anforderungen ist. Dabei fragt sich zum Beispiel, wie man den Auftrag mit dieser Programmiersprache lösen kann.

Die ganze Auswertung beruht auf Schätzungen. Diese subjektiven Beobachtungen werden allerdings präziser, je mehr Berufserfahrung man sammelt.

In der Abbildung 2.1 gibt es vier Abschnitte. Je Kategorie sollte man eine andere Projektmanagementmethode anwenden.

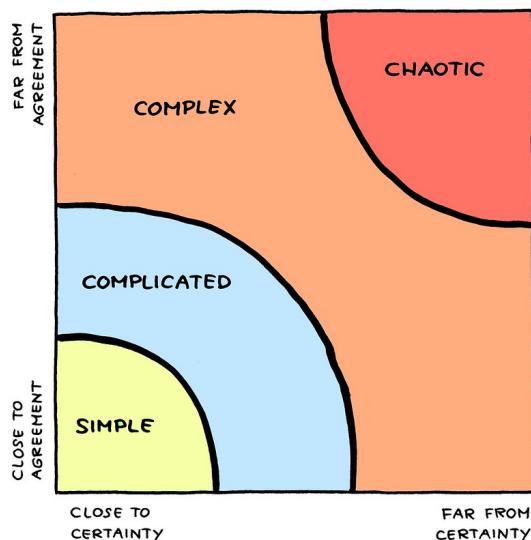


Abbildung 2.1.: Stacey Matrix

- **Simpel** - Das Ziel und der Weg sind klar. In diesem Fall sollte das Wasserfallmodell oder eine ähnliche Methode gewählt werden.
- **Kompliziert** - Sollte entweder das Ziel oder noch nicht ganz klar sein, sollte man Kanban anwenden.
- **Komplex** - Sind beide Achsen unklar oder ändern sich Kriterien, sollte etwas Agiles wie Scrum gewählt werden. Eine Methode wie diese eignet sich für wechselnde Anforderungen.
- **Chaotisch** - Das Ziel und der Weg sind unklar. In diesem Fall sollte das Projekt noch nicht gestartet werden. Stattdessen sollte für mehr Klarheit bezüglich Weg und Ziel geschaffen werden.

2.2. Wasserfallmodell

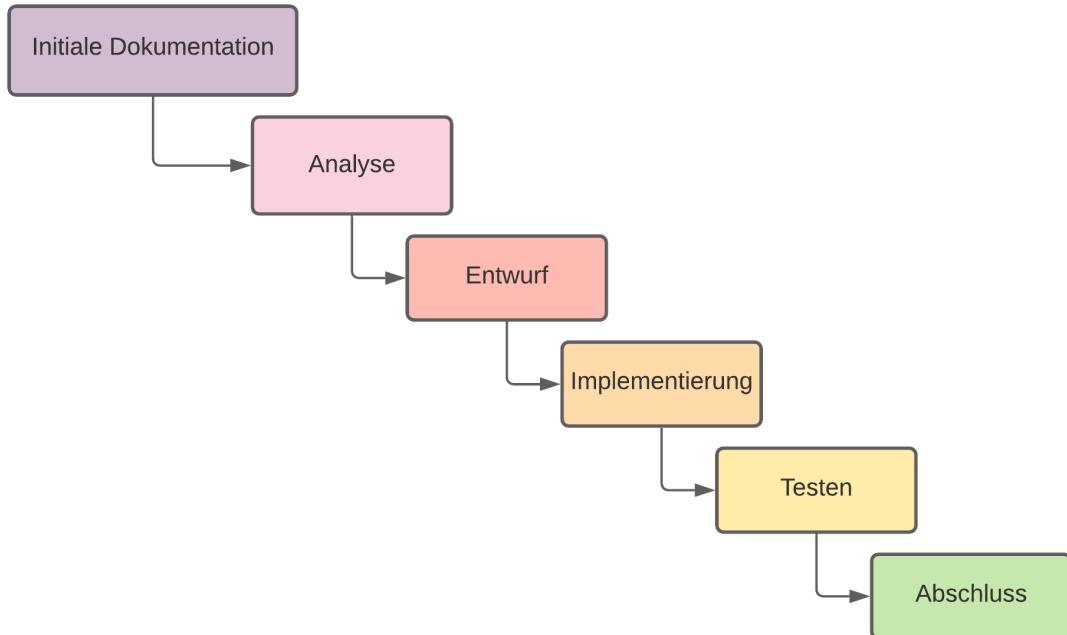


Abbildung 2.2.: Wasserfall Modell

Das Wasserfallmodell ist eine Abfolge sequenzieller Phase, wobei man erst mit der nächsten Stufe beginnt, wenn die davorliegende abgeschlossen ist. Dies erfordert, dass das Ziel und der Weg klar sein muss. Ein streng geregelter Arbeitsablauf mit Meilensteinen kann dadurch ermöglicht werden, was vor allem für kleine Projekte sehr nützlich ist. Dies ist auch der Grund, weshalb man sich bei dieser Arbeit für das Wasserfallmodell entschieden hat.

2.2.1. Initialisierung des Dokuments

Als Erstes soll das Dokument für die kommenden Arbeitstage so vorbereitet werden, damit zukünftige zeitaufwendige Anpassungen vermeiden werden können. Zuerst soll die LaTeX-Vorlage nochmals angesehen werden, um sicherzugehen, dass alles stimmt und richtig konfiguriert ist. Danach sollten essenzielle Bestandteile wie der detaillierte Zeitplan, das Arbeitsjournal und der obligatorische Teil erstellt werden.

2.2.2. Analyse

Nachdem die Initialisierung komplett abgeschlossen ist, geht es an die Analyse. Diese beinhaltet unter Anderem eine Systembeschreibung, ein Ist/Soll-Vergleich, Konzepte wie die Arbeitsergebnisse gesichert werden und die OAuth2-Strategie. Ersteres beschreibt, wie genau unser Netilion-Ökosystem funktioniert und wie dieses Projekt integriert werden soll. Der Ist/Soll-Vergleich zeigt nochmals die zu erarbeitende Lösung auf und wie sie sich von dem bereits existierenden Produkt differenziert. Die OAuth2-Strategie soll genau beschreiben, wie der Anmeldeprozess abläuft und was ihn sicher macht.

Ausserdem enthalten sind Personas und detaillierte User-Stories. Diese werden gebildet, indem die Anforderung des Auftraggebers unterteilt werden, damit sich der Entwickler auf einzelne Entwicklungsabschnitte konzentrieren kann und der Fortschritt messbarer ist.

User-Stories

Die User Story beschreibt in kurzer Form eine Anforderung einer Anspruchsgruppe an die Software. Dabei soll geklärt werden, **wer** welche **Funktionalität** aus welchem **Grund** implementiert haben möchte[2]. Ohne dabei zu sehr in technische Details zu gehen, beschreibt sie, was genau von der Software gefordert ist. Dadurch wird die Absprache mit den Anspruchsgruppen erleichter. Wie die Anforderungen implementiert werden, liegt im Ermessen des Entwicklers. Sehr oft wird diese kurze Beschreibung als Titel für eine Story im Projektmanagementtool angegeben.

Beispiel einer User Story:

Aufbau: Als <Anspruchsgruppe> möchte ich <Feature>, damit <Anwendungsfall> erreicht wird.

Beispiel: Als Nutzer möchte ich, dass die *ne107 Werte grafisch dargestellt werden*, damit *ich die Status der Messgeräte direkt sehen kann*.

2.2.3. Entwurf

In der Entwurfsphase wird der genaue Lösungsweg ermittelt. Der Entwickler beschreibt, wie er mit welchen technischen Mitteln die zuvor definierten Ziele erreichen kann. Dazu gehören Schnittstellen, Bibliotheken, Datenbank und Weiteres. Das Ergebnis dieser Phase ist ein ausgearbeitetes User Interface/User Experience Design, genaue Pläne, wie die Software aufgebaut werden soll und das Testkonzept.

2.2.4. Implementierung

In diesem Abschnitt gilt es, die dokumentierten Anforderungen mit den gewählten Technologien umzusetzen. Dabei sollen die User-Stories und Testfälle der vorherigen Kapiteln berücksichtigt werden. Nachdem eine Story abgeschlossen ist, wird die Änderung auf einen Branch gepusht und automatisch deployed. So können die Anspruchsgruppen dem Entwickler zeitnahe Rückmeldungen geben. Das Produkt der späteren Implementierungsphase ist die fertige Software.

2.2.5. Testen

In dieser Phase wird nochmals sorgfältig durchgegangen, ob alle Kriterien erfüllt wurden und ob die Tests wie gewünscht ablaufen. Stimmt das Ergebnis mit den gesetzten Erwartungen, gilt die Software als fertiggestellt.

2.2.6. Abschluss

Nachdem alle Ziele erreicht und mit dem Testprotokoll verifiziert wurden, wird die verbleibende Zeit für die Überarbeitung der Dokumentation eingesetzt werden. Anschließend soll die IPA am 14.04.2021 abgegeben werden.

KAPITEL 3

ANALYSE

3.1. Systembeschreibung

3.1.1. Systemarchitektur

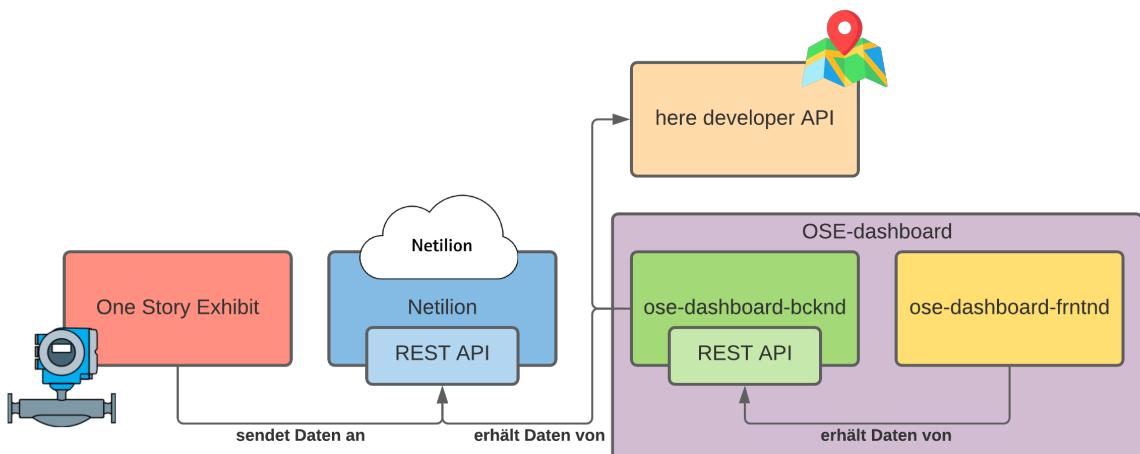


Abbildung 3.1.: Grobe Systemarchitektur

[6] Das System des OSE-Dashboards besteht aktuell aus fünf essenziellen Teilen. Die Systemteile werden in den nachfolgenden Kapiteln 3.1.2 bis 3.1.6 erläutert.

3.1.2. One Story Exhibit

In diesem Projekt sollen die Anlagedaten der Endress+Hauser Ausstellungsmodelle von der Applikation ausgewertet werden. Endress+Hauser verwendet diese Modelle an Messen oder lokalen Vertriebsorganisationen, um den Kunden das umfassende Portfolio an Messgeräten näher zu bringen. Diese Applikation soll unterstützend demonstrieren, was mit dem IIoT Service «Netilion» für Kunden möglich ist.

3.1.3. Netilion

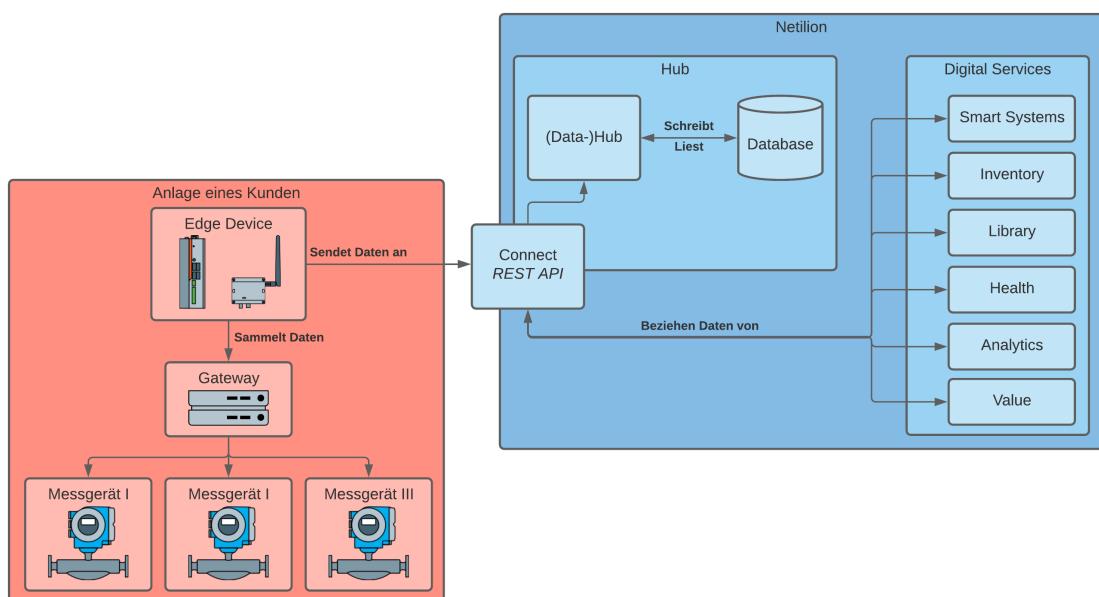


Abbildung 3.2.: Netilion Ökosystem

Netilion ist das IIoT Ökosystem welches vor einigen Jahren von der Endress+Hauser Digital Solutions ins Leben gerufen wurde. Zuvor hatten Kunden keinen direkten Einblick in die Daten der Messgeräte. Techniker mussten regelmässig vorbei kommen und die Geräte prüfen, auch wenn diese funktionierten. Wenn die Anlage des Kunden an das Internet angebunden ist, erschlossen sich neue Informationsmöglichkeiten über die eingebauten Messgeräte. Genau das ist der Punkt, an dem Netilion mit seinen digitalen Services für den Kunden Mehrwerte liefert.

Die Daten der Messgeräte werden mithilfe eines Edge Devices ausgelesen und zuerst lokal gespeichert, bevor sie dann in regelmässigen Intervallen an die REST-API von Netilion gesendet werden. Dort werden sie vom Hub empfangen, validiert und serialisiert. Anschliessend kann der Kunde eine der digitalen Services nutzen und die verarbeiteten Daten grafisch aufbereitet ansehen.

Hub

Der Hub ist das zentrale Lager aller Daten, welche im Ökosystem verwendet werden. Er bietet die REST-API an, validiert Daten, regelt das Benutzer- und Zugriffssystem und speichert alles in einer Datenbank.

Digitalen Services

Die Digitalen Services sind Webapplikationen, welche die Daten der Anlage entgegennehmen, aufbereiten und mit anderen Daten integrieren. So zeigt zum Beispiel die Applikation «Health» die NE107 Status der einzelnen Geräte an. «Value» hingegen stellt unter anderem die Messwerte der Geräte dar. Daneben gibt es noch weitere Services wie in Abbildung 3.2 genannt.

Ziel ist es mit den digitalen Services den Kunden zu helfen Ihre Anlagenverfügbarkeit zu erhöhen, Prozesse zu vereinfachen und damit Kosten zu sparen.

Connect

Netilion Connect ist ein neues Angebot des Ökosystems. Grosse Kunden wollen entweder eigene eingekaufte oder von Ihren Entwickler hergestellte Systeme verwenden. Deshalb sollten sie auch ausserhalb unserer Applikation an ihre Daten kommen können. So ergibt sich zum Beispiel die Möglichkeit, ein Dashboard mit den Firmen Guidelines zu erstellen.

3.1.4. here developer API

Als Vorarbeit dieser Arbeit wurde recherchiert, wie die Ortung der Modelle implementiert werden kann. Die erste option war Google Maps. Dies stellte sich allerdings als zu aufwändig heraus. Daher wurde nach alternativen gesucht. Mit der «here developer API» wurde diese schlussendlich gefunden. Die Ortungsdaten sollen im Backend mit den Daten der Messgeräte von Netilion integriert.

3.1.5. OSE-Dashboard: Backend

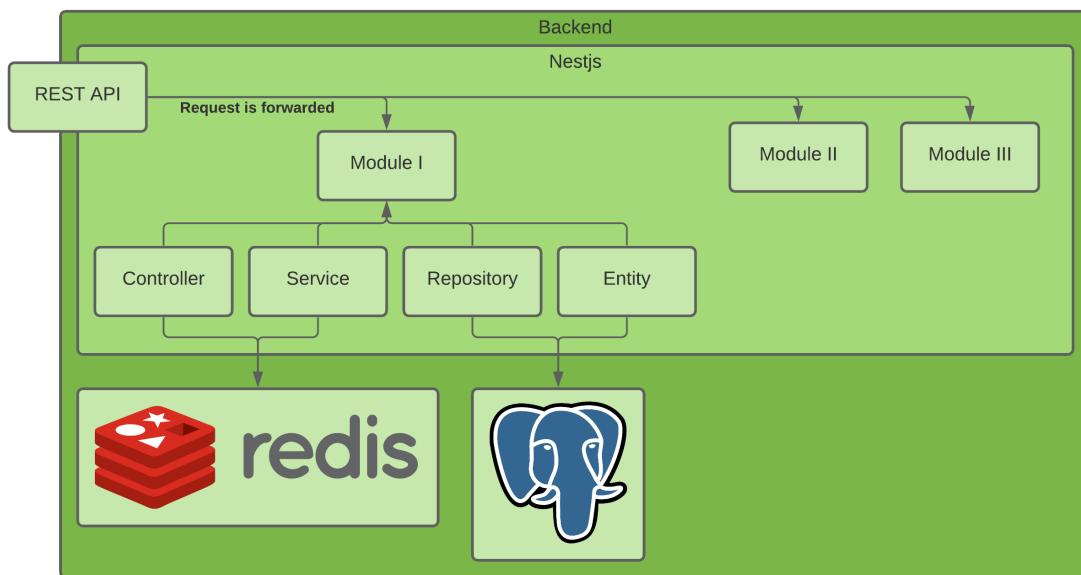


Abbildung 3.3.: OSE-Dashboard Backend

Die Aufgabe des Backends war primär das Cachen der Daten. Ohne Caching würde jede Instanz des Frontends direkt Anfragen an Netilion gesendet. Auch wenn ein Client nur alle 30 Minuten die Daten aktualisiert, wir wissen trotzdem nicht, auf wie vielen Clients unser Frontend gerade aktiv ist. Das Abonnement der REST-API besitzt eine maximale Zahl an Anfragen, welche gemacht werden können, bevor auf die nächste teurere Stufe aufgerüstet werden muss. Diese Nummer nicht zu überschreiten war ein Kriterium des Product-Owners. Da dies schwierig abzuschätzen oder auszurechnen ist mit einer unbekannten Zahl, wurde das Backend erstellt.

Ein eigenes Backend zu erstellen hat den Vorteil, dass wir zusätzliche Daten anfordern können und diese direkt mit den Netilion-Daten verlinken/integrieren können.

Kurz vor der IPA hat das Team, welches hinter der Entwicklung von Netilion steht, angekündigt, dass Webhooks nun in der Produktion verfügbar sind. Mithilfe von Webhooks kann ein Client einzelne Events abonnieren und erhält direkt Bescheid, wenn sich ein Wert ändert. Mit diesen Webhooks müsste ich keine Intervalle/Cron-Jobs verwenden müssen, wodurch die Anzahl der Anfragen deutlich vermindert werden würde.

Auch wenn ein Lösungsansatz mit Webhooks optimaler wäre, habe ich mich dagegen entschieden, es an dieser IPA so zu lösen. Das Feature wurde kurz vor der IPA angekündigt und ich habe bisher keine theoretische oder praktische Erfahrung mit Webhooks. Gerne werde ich es nach dem Abschluss der IPA angehen.

Nestjs

Seit dem zweiten Lehrjahr arbeite ich mit Javascript. Die Sprache hat mich von Anfang an angesprochen. Der Hauptgrund war, dass man mit Javascript nicht nur ein Frontend erstellen kann, sondern mittlerweile dank Node.js auch Backends/Servers und sogar Apps fürs Smartphone. Wenn ein Produkt im ganzen Stack Javascript verwendet, müssen Entwickler nicht mehrere Sprachen lernen, sondern können sich ganz auf eine Fokussieren, Stücke vom Quellcode können ausgetauscht werden und so weiter.

Ich habe Erfahrungen mit praktisch allen populären Node.js-Backendframeworks, konnte mich allerdings nie ganz mit einem anfreunden. Die meisten Frameworks sind unopinionated. Dies gefällt mir im Backend nicht, da nach einer kurzen Zeit ein Durcheinander entsteht, welches man dann selber aufräumen kann. Sobald man einen passenden Platz für alles gefunden hat, kann man das gleiche nochmals beim nächsten Projekt wiederholen.

NestJS geht komplett in eine andere Richtung. Es ist sehr inspiriert von Java Spring und Angular, ist extrem opinionated und das meiste kommt out of the box, ohne das man selber viele Konfigurationen machen muss.

Inspiration

Wie in Java Spring Boot gibt es Controller, Services, Repositories und Entitäten. Diese werden jeweils nur für eine Ressource erstellt. Mögliche Ressourcen sind zum Beispiel «Users» oder «Assets». Wie in Angular werden sie zu einem Modul gebündelt.

Komponenten einer Nestjs Applikation

Nestjs ist highly opinionated. Dabei ist vordefiniert, wie der Code strukturiert ist. Entwickler sollten folgendem Schema folgen:

Komponente Beschreibung

Entity	Enthält die Entität, welche vom ORM verwendet wird und als Basisklasse dient
Repository	Erledigt alle direkten Interaktionen mit den Entitäten
Service	Handhabt die Geschäftslogik
Controller	Dient dazu, Anfragen entgegenzunehmen, validieren und den verantwortlichen Service aufzurufen
Module	Bündelt die Komponenten einer Funktion oder Resource, reguliert Imports von anderen Services und Exports von eigenen Services

Redis & Postgresql

Ich habe mich bei temporärem Caching für Redis entschieden, da ich keine Alternative dazu kenne und da es auch sonst intern verwendet wird. PostgreSQL habe ich genommen, da Netilion und das dahinterstehende Team dies nutzt

Hosting

Das Backend wird auf Heroku gehostet. Gründe dafür sind folgende:

- Praktische Erfahrung seit dem zweiten Lehrjahr.
- Wird intern und bei Netilion verwendet.
- Bietet gratis Redis & PostgreSQL Instanz an, welche für diesen Use-Case komplett ausreichen.

3.1.6. OSE-Dashboard: Frontend

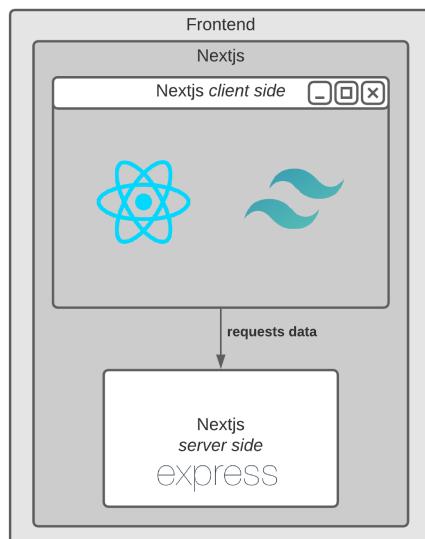


Abbildung 3.4.: OSE-Dashboard Frontend

Hosting

Das Hosting wird mit Vercel gemacht. Vercel ist eine amerikanische Firma, die sich auf den JAM-Stack fokussiert. Seit 2016 bietet sie eine sehr vereinfachte Möglichkeit, moderne Javascript Frontends zu bilden und deployen. Zeitgleich arbeiten sie auch an Next.js, welches kurze Zeit später veröffentlicht wurde. Next.js ist ein Frontend Framework, welches Routing, Server-side-rendering und viele weitere Features mit React verbindet. Next.js ist für das Hosting auf Vercel optimiert. So ist das Deployment noch einfacher als bei vergleichbaren Frameworks, und es können tiefere Analysewerte gemessen werden.

Server-side-rendering

Eines der Features, welches Next.js ausmacht, ist das Server-side-rendering, welches es out of the box anbietet. Dadurch kann man beim Entwickeln angeben, welche Seiten statisch und welche dynamisch sein sollen. So wird nicht nur die Erfahrung des Users besser, man steigert auch das SEO Ranking. Das Framework baut auf express auf. Dies ermöglicht es einem, ein minimales backend in das Frontend einzubauen. Die Vorteile davon sind, dass sensible Daten so nicht beim Benutzer landen.

3.2. Ist/Soll-Vergleich

Ein meiner Meinung nach wichtiger Teil der IPA ist der Vergleich zwischen dem momentan vorhandenen Stand und wie das fertige Produkt aussehen sollte. In den folgenden Kapiteln 3.2.1 und 3.2.2 gehe ich auf diese beiden Themen ein.

3.2.1. Ist Zustand

Modell

Bei der Erstellung der Vorarbeit war die Verwendung von mehreren Modellen nie ein Thema. Es sollte extra für Reinach angefertigt werden. Zu diesem Zeitpunkt wusste ich nicht einmal, dass die Endress+Hauser Gruppe noch mehr von ihnen besass, geschweige den, dass sie intern standardisiert wurde. Das heisst, dass immer die gleichen Messgeräte auf den Modellen sind.

Da es sich nur um ein einziges Modell handelte, habe ich die IDs der digitalen Zwillinge der Messgeräte per Hand herausgesucht und dann statisch in den Quellcode geschrieben. Dies war der einfachste Weg, um das Projekt wie gewünscht umzusetzen. Es gab mir zu diesem Zeitpunkt auch keinen Sinn, das Ganze dynamisch zu gestalten.

Credentials

Damit ich die Daten der Messgeräte erhalten konnte, musste ich geheime Zugangsdaten verwenden. Dies wurde von Anfang mit Environmentvariablen gelöst, da ich leider in der Vergangenheit schon einmal erfahren musste, was es heisst, die geheimen Zugangsdaten auf GitHub zu pushen. Anfangs wurde dies mit BasicAuth im Server-Side Teil von Nextjs erledigt. Später habe ich es dann allerdings ins Backend verschoben. Momentan verwendet das Projekt noch diesen statischen Lösungsweg mit BasicAuth.

Konfiguration

Da es sich nur um ein Modell handelte, habe ich gewünschte Konfigurationen entweder in Environmentvariablen gespeichert oder direkt in den Sourcecode geschrieben. Da es nun mehrere Modelle sind, muss ein Konfigurationsmenü her.

3.2.2. Soll Zustand

Modelle

Es sollen nun mehrere Modelle eingebunden werden können und ein User soll zwischen den Standorten wechseln können. Dafür müssen neue Entitäten im Backend und neue Tabellen in der Datenbank angelegt werden. Einerseits soll eine «models»Tabelle her, welche die «assets», also Messgeräte, bündelt. Andererseits muss auch der Standort des Modells abrufbar sein können, weswegen auch eine «locations»Tabelle erstellt werden sollte.

OAuth2

In dieser Erweiterung ist es notwendig, dass sich mehrere Netilion-Accounts sicher anmelden können. Ich habe mich in der Vergangenheit sehr intensiv mit OAuth2 beschäftigt und finde, dass ich es optimal für diesen Use-Case umsetzen kann. Sprechen wir allerdings über die Entscheidung, wieso OAuth2 verwendet werden sollte und nicht einfach das bestehende BasicAuth Konstrukt erweitert werden sollte.

- BasicAuth wird in der nahen Zukunft nicht mehr unterstützt
- OAuth2 verbessert die User Experience, da ein Nutzer nur noch seinen Log-in braucht
- OAuth2 vermindert den administrativen Aufwand
- Die Lösung ist optimaler und schöner

Konfigurationsmenü

Damit der Verantwortliche eines OSE-Modells Einstellungen vornehmen kann, muss ein Konfigurationsmenü her. Ein Teil der IPA ist es, die digitalen Zwillinge automatisch mit den Meshes des 3D-Modells zu verlinken. Sollte dies aus irgendeinem Grund nicht möglich sein, soll der User dies selbst manuell verlinken können. Damit diese wichtige Einstellung aber nicht von jedem User vorgenommen werden kann, darf nur dies nur ein eingeloggter User, welcher sich in einer Usergroup befindet, vornehmen.

3.3. Namenskonzept

3.3.1. User-Stories

Segment	Abkürzung	Beschreibung
----------------	------------------	---------------------

1	US	Abkürzung für User-Story
2	-	Trennzeichen
3	01	Fortlaufende Kennzahl

3.3.2. Akzeptanzkriterien

Segment	Abkürzung	Beschreibung
----------------	------------------	---------------------

1	AK	Abkürzung für Akzeptanzkriterium
2	-	Trennzeichen
3	01	Fortlaufende Kennzahl

3.4. Versionskontrolle

Ich werde Git mit GitHub verwenden, da ich nun drei Jahre damit gearbeitet habe und mein Lehrbetrieb die gleiche Strategie verfolgt. Dabei werde ich immer nach Abschluss eines Abschnittes einen Commit erstellen. Wichtig ist, dass der Master Branch immer problemlos läuft und deployed werden kann.

Diese IPA ist in drei Teile aufgebrochen: das Frontend, das Backend und die Dokumentation. Für alle drei Teile verwende ich Git & GitHub. Hier ist eine Auflistung aller Repositories:

Frontend: EndressHauser-ProcessSolutions/ose-dashboard-frntnd
 Backend: EndressHauser-ProcessSolutions/ose-dashboard-bcknd
 Dokumentation: EndressHauser-ProcessSolutions/ose-dashboard-docs

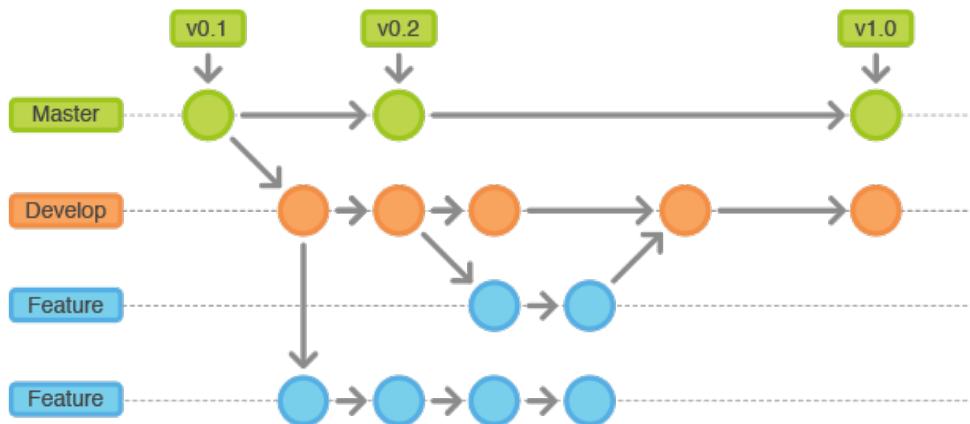


Abbildung 3.5.: Git Flow visualisiert

3.4.1. Git Flow

Mit dem Git Flow definiert ein Team oder ein Entwickler, wie sie/er die Versionskontrolle in Git handhabt. Es gibt drei Arten von Branches:

- **Master** - Enthält immer die lauffähige und stabile Version, welche gerade deployed ist.
- **Develop** - Dies ist der Branch, auf dem entwickelt wird. Er soll immer lauffähig sein, damit neue Features getestet werden können, bevor er mit dem Master zusammengeführt wird.
- **Feature** - Die Einzelnen Stories und Tasks werden in einem Feature Branch entwickelt. Sobald das Feature fertig ist, kann es in den Develop Branch gemerged werden.

Diese Strategie bietet einige Vorteile:

- Der Master Branch kann mithilfe von CI/CD automatisch deployed werden. So erreichen neue Features schneller den Kunden
- Der Develop Branch kann auch mit CI/CD automatisch deployed werden, damit Features intern einfacher und schneller besprochen werden können.
- Durch die klare Abtrennung dieser Branches erhöht sich die Stabilität der Software, was die Kundenzufriedenheit erhöht.
- Features können abgekapselt entwickelt werden.
 - Develop Branch bleibt dadurch immer lauffähig
 - Mehrere Features können gleichzeitig entwickelt werden, ohne sich in die Quere zu kommen
 - Bietet bessere Übersicht auf GitHub

3.4.2. Pipelines zu Vercel und Heroku

Vercel und Heroku bieten Build-Pipelines an. Mit diesen kann ein Branch mit Vercel oder Heroku verbunden werden. Wird ein neuer Commit gepusht, triggert dies den Buildprozess von Vercel oder Heroku, woraufhin man kurze Zeit später eine live Version hat.

Ich werde jeweils zwei Pipelines einrichten. Einmal die Produktionsumgebung mit dem Master Branch und einmal die Testumgebung mit dem Develop Branch.

3.4.3. Dokumentation

Die Dokumentation ist in LaTeX geschrieben und wird regelmässig auf GitHub in ein separates Repository gepusht. Im GitHub habe ich ausserdem ein GitHub Action eingerichtet, welche nach jedem push getriggert wird, das Dokument als PDF rendert und zum Download zur Verfügung stellt.

Bei der Versionierung dieses Dokumentes orientiere ich mich an meiner Einschätzung verbunden mit folgendem System:

Versionsartefakt	Beispiel	Beschreibung
1	0	Major Grosse Abschlüsse des Dokumentes
2	.	Trennzeichen
3	1	Minor Eine Änderung wie z.B. ein neues Kapitel
4	.	Trennzeichen
5	3	Patch Kleine Änderung / Neue Texte in einem Kapitel

Wichtig zu beachten ist, dass ich bei der Dokumentation **nicht GitFlow einsetze**, da es keinen Vorteil für den Aufwand bietet.

3.5. Backupkonzept

Die ganze Dokumentation und jeglicher Code wird mindestens einmal täglich auf GitHub gepusht. Optimal ist, wenn der Code im Falle vom Front- und Backend nach jedem Feature gepusht wird und bei der Dokumentation nach der fertigstellung eines Unterkapitels. Sollte was verloren gehen kann man so also immer mindestens auf den letzten Stand des Vorabends zurückzuspringen.

Das MacBook wird mithilfe vom vorinstallierten Tool «Time Machine»ständig inkrementell auf einer externen SSD gesichert. Time Machine speichert dabei Folgendes [1]:

- Lokale Schnappschüsse, solange Speicherplatz vorhanden ist
- Stündliche Backups der letzten 24 Stunden
- Tägliche Backups des letzten Monats
- Wöchentliche Backups aller vorherigen Monate

Somit ist eine schnelle Weiterarbeit trotz Hardwareproblemen möglich.

3.5.1. Sicherheit der Daten

Die Repositories befinden sich auf dem GitHub Team der Firma. Dieses Team ist so sicher eingerichtet, wie es GitHub erlaubt. Meine Accounts verwenden beide sichere Passwörter und Two-Way-Authenticator.

Die interne SSD des MacBooks [7] und die externe SSD [8] sind verschlüsselt.

3.6. Personas

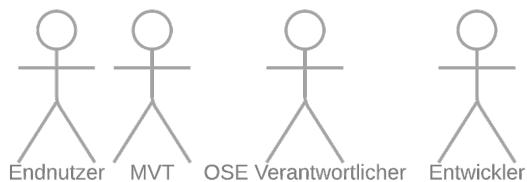


Abbildung 3.6.: Personas

3.6.1. Endnutzer

Der Endnutzer ist schlussendlich der Besucher, welcher am Fernseher mit dem Dashboard interagiert. Er möchte möglich einfach die Daten des OSE-Modells ansehen können, ohne sich dabei einloggen zu müssen.

3.6.2. MVT

Das MVT Team ist für interne und externe Trainings unseres Portfolios zuständig. Markus Reisgis von MVT war der originale Auftraggeber. Für sie ist wichtig, dass alle gewünschten Daten korrekt angezeigt werden und das der Endnutzer eine gute Erfahrung macht.

3.6.3. OSE Verantwortlicher

Der OSE Verantwortliche ist die Person, welche für ein Modell verantwortlich ist. Dies ist zum Beispiel MVT in Reinach. Es gibt einige Weitere in Deutschland, Frankreich und so weiter. Für sie ist wichtig, dass sie wenig Mehraufwand haben. Im optimalen Fall sollte Log-in und die eventuelle Konfiguration möglichst einfach und fehlerfrei ablaufen.

3.6.4. Entwickler

Der Entwickler bin ich. Mir ist es wichtig, alle gewünschten Features kompetent und wie gewünscht umzusetzen, und dabei den Fortschritt korrekt zu dokumentieren.

3.7. User Stories

3.7.1. US-01

Als OSE Verantwortlicher möchte ich mich einloggen können, damit ich mein Modell registrieren kann.

- AK-01 Log-in möglich
- AK-02 Log-in Daten sind geschützt und nicht von einem Benutzer auslesbar
- AK-03 Bei einem Fehlversuch wird eine passende Fehlermeldung angezeigt

3.7.2. US-02

Als OSE Verantwortlicher möchte ich, dass die digitalen Zwillinge der Messgeräte automatisch verlinkt werden, ohne das ich etwas machen muss, solange sie standardisiert sind.

- AK-01 Automatische Verlinkung bei Registrierung funktioniert, solange die Assets nach dem Standard benannt sind
- AK-02 Methode, welche die Verlinkung vornimmt, ist mit automatischen Tests abgedeckt

3.7.3. US-03

Als OSE Verantwortlicher möchte ich ein Konfigurationsmenü, mit welchem ich die Verlinkung manuell ändern kann, sollte ich ein Gerät austauschen.

- AK-01 Konfigurationsmenü ist aufrufbar
- AK-02 Verlinkungen sind manuell veränderbar

3.7.4. US-04

Als OSE Verantwortlicher möchte ich sicher sein, dass kein Endnutzer die Verlinkung ändern kann.

- AK-01 Konfigurationsmenü ist nur aufrufbar, solange der OSE Verantwortliche eingeloggt ist
- AK-02 Konfigurationsmenü ist nur aufrufbar, solange sich der Account in einer spezifischen User Gruppe befindet

3.7.5. US-05

Als Endnutzer möchte ich die Applikation nutzen können, ohne mich einloggen zu müssen.

- AK-01 Applikation ist weiterhin ohne Log-in nutzbar

3.7.6. US-06

Als Endnutzer möchte ich die Datenquelle des OSE-Dashboards aus den verfügbaren Standorten auswählen können, damit ich auch Daten anderer Modelle sehen kann.

- AK-01 Standort kann geändert werden
- AK-02 Übersicht über alle verfügbaren Standorte ist implementiert

3.8. OAuth2 Strategie

3.8.1. Was ist OAuth2?

OAuth2 wurde ins Leben gerufen, um einen spezifischen Use-Case zu decken. Eine Applikation eines Drittanbieters möchte zum Beispiel Daten des Users von Spotify abrufen. Vor OAuth2 musste der User seine Log-in-Daten an den Drittanbieter senden, welcher diese dann in Klartextformat abspeichern musste, damit er später selber die Daten abrufen kann.

Der User hatte so keine Kontrolle über die Aufbewahrung der Log-in-Daten und wusste schon gar nicht was der Drittanbieter alles mit den Log-in-Daten anstellt. Zusammengefasst war der ganze Prozess sehr unsicher und der User übergab immer alle Rechte des Accounts.

Mit OAuth2 konnte dieser Prozess nun sicher gestaltet werden. Der User wird auf die Anmeldeseite des OAuth2-Providers weitergeleitet und meldet sich dort an. So kommt die Applikation des Drittanbieters zu keinem Zeitpunkt an die Log-in-Daten. Als nächstes wird dem User gezeigt, auf was für Daten der Drittanbieter zugriff haben möchte. Akzeptiert der User, wird er wieder zurück an den Drittanbieter weitergeleitet, welcher mit diesem Schritt einen Zugriffstoken erhält. Dieser ist nur von dieser Applikation und auf diesen User nutzbar, um die Daten, die der User freigegeben hat, anzufragen.

3.8.2. Anwendung im OSE-Dashboard

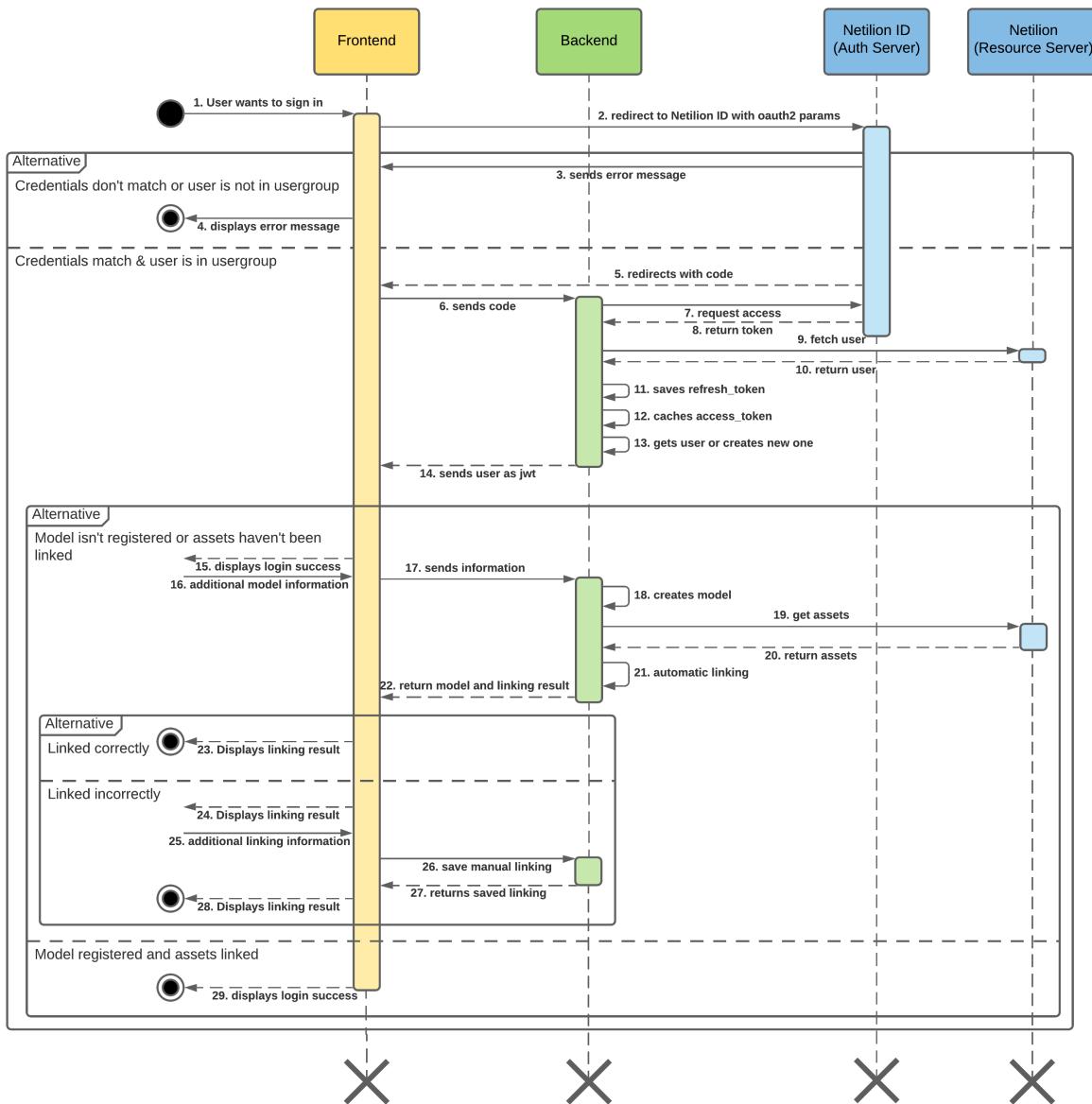


Abbildung 3.7.: OAuth2 visualisiert durch ein Sequenzdiagramm

Folgend ist eine Auflistung der einzelnen Schritte, welche in der Abbildung 3.7 dargestellt sind.

1. Der User will sich einloggen und klickt auf ein `<a>` Tag.
2. Daraufhin wird er zu Netilion ID weitergeleitet. Die URL enthält dabei die `client_id` und `redirect_uri`. Dadurch weiss Netilion, welche Applikation die Anfrage macht und wohin der User nach der Anmeldung weitergeleitet werden möchte.
3. Gibt es ein Problem mit dem Log-in, erhält die `redirect uri`, also das Frontend, eine Fehlermeldung.

4. Das Frontend nimmt die Fehlermeldung entgegen und stellt sie für den User dar.
5. Wenn alles korrekt abläuft, wird der Client mit einem code wieder an unser Frontend weitergeleitet.
6. Dieser Code wird direkt an das Backend gesendet.
7. Mit diesem Code kann das Backend dann einen Zugriffstoken anfordern.
8. Dieser wird von Netilion ID an unser Backend geschickt.
9. Mit diesem Zugriffstoken ist es dann möglich, Daten von Netilion abzufragen, in diesem Fall der User.
10. Die Daten des Users werden anschliessend zurückgeschickt.
11. Der `refresh_token` wird verschlüsselt und in der Datenbank gespeichert.
12. Der `access_token` wird in redis gecached.
13. Existiert der User, wird dieser aus der Datenbank gelesen. Existiert er nicht, wird ein neuer erstellt.
14. Der User wird als JWT zurück ans Frontend geschickt.
15. Das Frontend zeigt an, dass die Anmeldung erfolgreich verlief.
16. Wenn das Modell noch nicht registriert wurde oder die Messgeräte noch nicht verlinkt wurden, wird der User nach zusätzlichen Daten wie dem Standort gefragt.
17. Das Frontend schickt diese Daten mit dem JWT an das Backend.
18. Das Backend erstellt ein neues Modell mit den Daten des Users.
19. Anschliessend fragt es bei Netilion nach allen Messgeräten nach.
20. Netilion antwortet mit allen Messgeräten des Users.
21. Daraufhin erfolgt die automatische Verlinkung.
22. Das Modell mitsamt verlinkten Messgeräten wird ans Frontend weitergeleitet.
23. Stimmt die Verlinkung ist der Nutzer zufrieden und der Prozess beendet.
24. Stimmt die Verlinkung nicht, da der Nutzer nicht die standardisierten Geräte oder benennungen verwendet, wird ihm das Resultat angezeigt.
25. Daraufhin verlinkt der User selbst die Geräte.
26. Dies wird ans Backend gesendet, damit die Änderungen gespeichert werden können.
27. Die gespeicherten Änderungen werden an das Frontend geschickt.
28. Der User wird benachrichtigt, dass seine Änderungen übernommen wurden. Der User ist zufrieden und der Prozess beendet.
29. Ist das Modell bereits registriert und die Messgeräte verlinkt, ist der Prozess beendet.

KAPITEL 4

ENTWURF

4.1. Generelles Abfragen von Daten

Dieses Abschnitt baut auf das Kapitel 3.8 auf. Nachdem sich ein OSE Verantwortlicher das erste mal

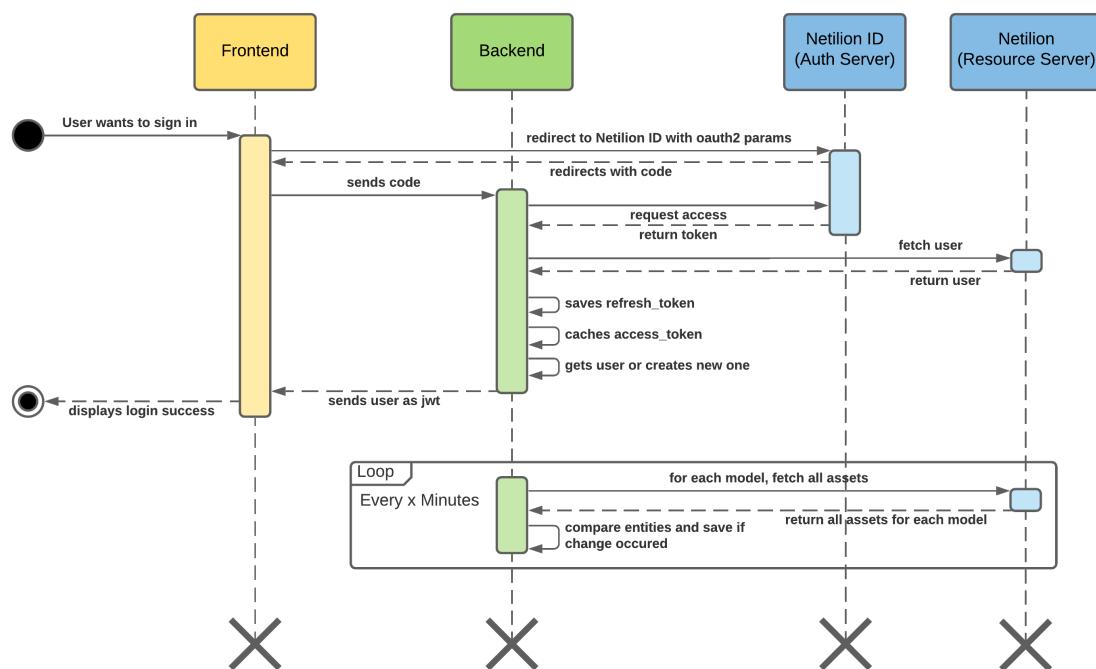


Abbildung 4.1.: Data fetching in Intervallen

eingeloggt hat, wird der `access_token` in Redis gecached und der `refresh_token` in der Datenbank gespeichert. Die Funktion soll entweder den gecachten Token nehmen und direkt zurückgeben oder mit dem `refresh_token` einen neuen Anfordern, welcher wiederum gecached und zurückgegeben werden soll. Damit ist es dann möglich, dass das Backend selbst die Daten auffrischen kann. Dafür werde ich das Task-Scheduling[4] feature von Nestjs verwenden. Damit ist es möglich eine Funktion wiederholt laufen zu lassen.

Bevor ich allerdings das ganze automatisieren kann, brauche ich eine Helperfunktion. Denn jede Anfrage an Netilion braucht einen validen `access_token` und er muss auch dem Netilion Benutzer gehören, unter welchem die Messgeräte registriert sind. Mit dieser Function wird es möglich sein das ganze zu vollautomatisieren. Wenn nun ein Benutzer des OSE-Dashboards sich die Daten nun ansehen möchte, kann er dies machen, ohne den Login des jeweiligen OSE Verantwortlichen wissen zu müssen.

4.2. Zugriffskontrolle

4.2.1. OSE Verantwortlicher

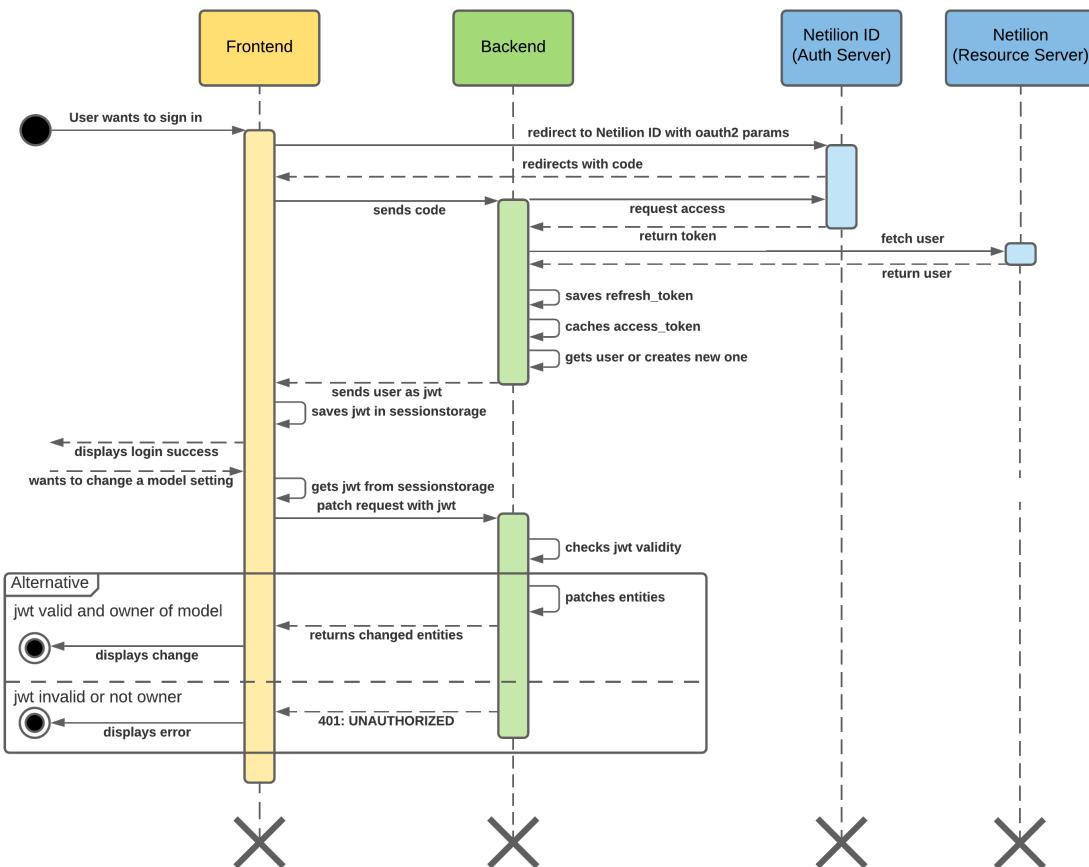


Abbildung 4.2.: Zugriffskontrolle OSE Verantwortlicher

Ein OSE Verantwortlicher soll das Konfigurationsmenü seines Modells öffnen können. Verantwortlicher anderer Modelle oder normale User sollen dies nicht können. Das ganze sollte folgenderweise gelöst werden:

Das Frontend erhält nach dem erfolgreichen Login einen JWT. Dieser wird zuerst benötigt, um den Link zum Konfigurationsmenü überhaupt erst im Frontend anzuzeigen. Außerdem wird er als Authentifizierungsmethode zum Backend verwendet. Der REST Endpoint vom Backend wird durch eine Nestjs Guard[5] geschützt. Daraufhin wird direkt geprüft werden, ob das Modell auch wirklich dem User gehört. Sollte dies nicht der Fall sein, wird die Anfrage als unauthorized zurückgeschickt.

4.2.2. OSE-Dashboard Nutzer

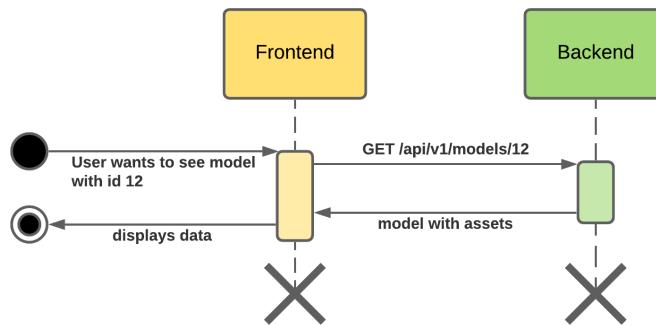


Abbildung 4.3.: Zugriffskontrolle OSE-Dashboard Nutzer

Der OSE-Dashboard möchte sich alle OSE-Modelle ansehen können, ohne sich dabei einloggen zu müssen. Das ganze sollte folgenderweise gelöst werden:

Wie in Kapitel 4.1 angesprochen, sind die aktuellen Daten im Backend verfügbar. So ist es also gar nicht nötig Netilion direkt anzufragen. Das Backend soll stattdessen die Daten mit einem public Endpoint verfügbar machen. Auf diesem Weg umgehen wir auch die maximale Anfragen an Netilion, welche in Kapitel 3.1.5 beschrieben wurde.

So kann das Frontend beliebig oft die Daten abfragen, ohne dass sich ein User einloggen muss oder das wir die Grenze der Connect Subscription übertreten.

4.3. Softwareschnittstellen

Die zu erweiterbare Webapplikation ist in mehrere logische Schichten unterteilt. Jede Schicht ist für einen konkreten Teil der kompletten Lösung verantwortlich. Eine vereinfachte Darstellung der Lösung ist bereits in der Abbildung 3.1 dargestellt. Folgend sind alle Schichten und deren Zuständigkeit zusammengefasst:

Schicht	Zuständigkeit
OSE	Daten der Messgeräte in der Anlage werden von einem Edge Device gesammelt und an Netilion gesendet
Netilion	IIoT-Service von Endress+Hauser
Backend	Backend des OSE Dashboards
here developer API	API eines Drittanbieters. Wird verwendet für das Abfragen von Koordinaten oder Adressen und für die Darstellung der Karte.
Frontend	Frontend des OSE Dashboards

4.3.1. Schnittstellen im OSE

Wie und welche Daten das OSE Modell an Netilion schickt ist für diese Arbeit nicht relevant und wird daher nicht dokumentiert. Allerdings ist es wichtig zu wissen, dass die Daten gesammelt und in Intervallen an Netilion gesendet werden.

4.3.2. Netilion

Folgend sind die verwendeten Schnittstellen von Netilion aufgelistet. Von den angeforderten Entitäten werden nur die verwendeten Attribute vermerkt und beschrieben. Um zusätzliche Informationen über die Entitäten zu erhalten, sollten die offizielle Netilion API Dokumentation aufgesucht werden.

API Routes

Anfrage	Pfad	Antwort
HTTP:GET	/users/current	NetilionUser
HTTP:GET	/assets	Array of Asset

NetilionUser

Attribut	Datentyp	Beschreibung
----------	----------	--------------

id	number	Primary Key
email	string	Email mit der sich der Nutzer auf Netilion registriert hat

Asset

Attribut	Datentyp	Beschreibung
----------	----------	--------------

id	number	Primary Key
serial_number	string	Seriennummer des Messgerätes
description	string	Beschreibung des Messgerätes
production_date	string	Produktionsdatum des Messgerätes
last_seen_at	string	Wann sich das Messgerät zuletzt bei Netilion gemeldet hat
product	number	Foreign Key
status	number	Foreign Key
instrumentations	number	Foreign Key

Status

Der Health-Status des Messgerätes. Zeigt zum Beispiel an, ob das Gerät gewartet werden muss.

Attribut	Datentyp	Beschreibung
----------	----------	--------------

id	number	Primary Key
code	string	Standardisierter Status Code
description	string	Beschreibung des Status
name	string	Name des standardisierten Status Codes

Instrumentations

Instrumentations ist der interne Begriff für die Messstelle, an der sich das Messgerät befindet. Wird auch Tag genannt.

Attribut	Datentyp	Beschreibung
id	number	Primary Key
tag	string	Bezeichnung der Messstelle
description	string	Beschreibung der Messstelle
criticality	string	Kritikalität der Messstelle
accessibility	string	Erreichbarkeit der Messstelle

Product

Ein Product ist ein Typ von Messgerät. Product bildet dabei das allgemeine Produkt ab, während dem Asset ein Zwilling eines hergestellten Messgerätes vom Typ Product ist.

Attribut	Datentyp	Beschreibung
id	number	Primary Key
product_code	string	Code des Produktes
name	string	Name des Produktes
description	string	Beschreibung des Produktes
manufacturer	number	Foreign Key

Manufacturer

Der Manufacturer ist der Hersteller eines Messgerätes.

Attribut Datentyp Beschreibung

id	number	Primary Key
name	string	Name des Herstellers

4.3.3. Backend

Folgend sind die verwendeten Schnittstellen von `ose-dashboard-bcknd` aufgelistet.

API Routes

Anfrage	Pfad	Antwort
Models		
HTTP:POST	/models/:modelId/assets/:assetId	Void
HTTP:POST	/models/:id/autolink	Void
HTTP:POST	/register	Model
HTTP:GET	/models	Array of Models
HTTP:GET	/models/:id	Model
HTTP:GET	/models/:id/location	Location
HTTP:GET	/models/:id/assets	Array of Asset
HTTP:PUT	/models/:id/location	Location
HTTP:PATCH	/models/:id	Model
Assets		
HTTP:GET	/assets	Array of Assets
HTTP:GET	/assets/:id	Asset
Users		
HTTP:GET	/users/current	Array of Assets
HTTP:GET	/users/:id/model	Model
HTTP:PATCH	/users/:id/finished-setup	Void
Location		
HTTP:GET	/mapview	image/jpeg; charset=UTF-8
HTTP:GET	/geolocation	Array of Locations
Authentication		
HTTP:POST	/auth/login	Void

Model

Digitale Abbildung eines OSE Modelles

Attribut	Datentyp	Beschreibung
id	number	Primary Key
name	string	Name des Modelles
description	string	Beschreibung des Modelles
location	string	Id der Adresse des Modelles
owner	number	Foreign Key
assets	number	Foreign Key

4.3.4. here developer API

Diese API ermöglicht die Ortung der Modelle in diesem Projekt. Sie ermöglicht das Abfragen von Adressen und das rendern von Satellitenkarten. Um diese Daten abzufragen, wird allerdings ein Api-Key benötigt. Diese API soll nicht im Frontend abgefragt werden, da ansonsten dieser Key abgegriffen werden kann. Die beiden folgend beschriebenen API-Routen sollen daher über das Backend proxied werden, wo der APIKey im nachhinein der Anfrage beigefügt wird.

API Routes

Anfrage	Pfad	Antwort
HTTP:GET	/v1/geocode	Array of Locations
HTTP:GET	/mia/1.6/mapview	image/jpeg; charset=UTF-8

Location

Attribut	Datentyp	Beschreibung
id	string	Eindeutige Identifikationsnummer
title	string	Ausgeschriebene Adresse
position	object	Objekt, welches den Längengrad und Breitengrad enthält

4.3.5. Frontend

Das Frontend ist mit Nextjs gemacht und baut auf React auf. Es konsumiert Daten des Backends und stellt diese passend dar. Vorhandene Schnittstellen dienen nicht für andere Software, sondern für die Interaktion mit Kunden.

Geschützt?	Pfad	Funktionalität
Nein	/	Zeigt die Startseite an
Nein	/models/:id	Zeigt selektierte 3D Model an
Nein	/register	Zeigt selektierte 3D Model an
Nein	/register/error	Zeigt Fehlermeldung an und leitet Benutzer nach 15 Sekunden weiter an die Startseite
Nein	/register/unauthorized	Zeigt Fehlermeldung an und leitet Benutzer nach 15 Sekunden weiter an die Startseite
Ja	/settings	Leitet weiter auf /settings/model
Ja	/settings/model	Zeigt Einstellungsmenü des Modelles an
Ja	/settings/location	Zeigt Einstellungsmenü des Standortes an
Ja	/settings/assets	Zeigt Einstellungsmenü der Verlinkungen an

4.4. Mockups

Mockups dienen zur Planung der Darstellung des Frontends. Es ist viel sinnvoller das Layout und den Inhalt der Website im Vorfeld zu skizzieren und damit herumzuspielen, als während dem Implementieren. Weiss man während der Implementierungsphase noch nicht wie das Produkt grob aussehen sollte verschwendet man meist wertvolle Zeit.

Oft wird vor der Erstellung der Website ein UX-Design erstellt. Dieses gibt grob das Layout und den Inhalt. Ein UI-Design wird angewandt wenn das detaillierte Design der Website wichtig ist. Dabei werden die ganzen Komponenten der Website bis ins Detail im Vorfeld designed, sodass dies den Anspruchsgruppen und den Programmierern klar ist.

Bei dieser Arbeit ist nur die Erarbeitung eines UX-Designs wichtig. Dies wurde soweit erstellt und ist nachfolgend in den Kapiteln dokumentiert.

4.4.1. Index Page

Momentan ist die Index Page des OSE-Dashboards das Reinacher Modell selbst. Dies soll sich nun ändern. Die rote Linie markiert die initiale Sicht des User. Diese enthält den Titel und eine Beschreibung der Webapplikationen, gefolgt von der Auswahlmöglichkeit der ganzen Modelle. Unter der initialen Sicht ist ein Abschnitt, welcher den User fragt, ob er für ein OSE-Modell zuständig ist. Sollte dies der Fall sein, könnte er sich einloggen und so die Daten für das Dashboard zur Verfügung stellen.

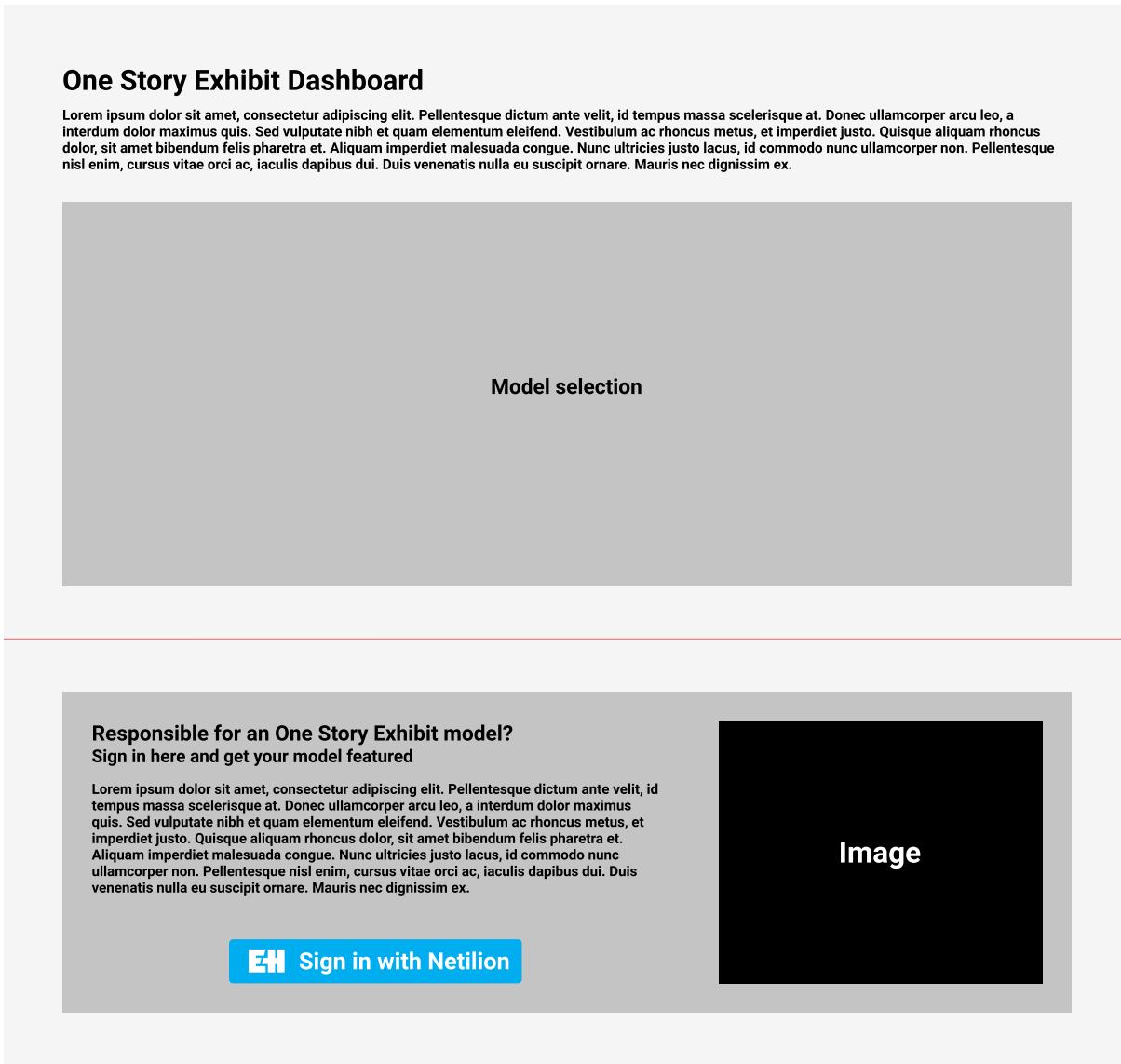


Abbildung 4.4.: Mockup der Index Page

4.4.2. Registration

Fehlermeldung

Sollte beim Loginprozess etwas schiefgehen, wird die unten dargestellte Fehlermeldung angezeigt. Dabei wird erwähnt, dass dieser Fehler sehr wahrscheinlich durch den Kandidaten geschah und nicht durch Netilion. Darunter befindet sich die Meldung, dass der User in x Sekunden wieder auf die Startseite weitergeleitet wird.

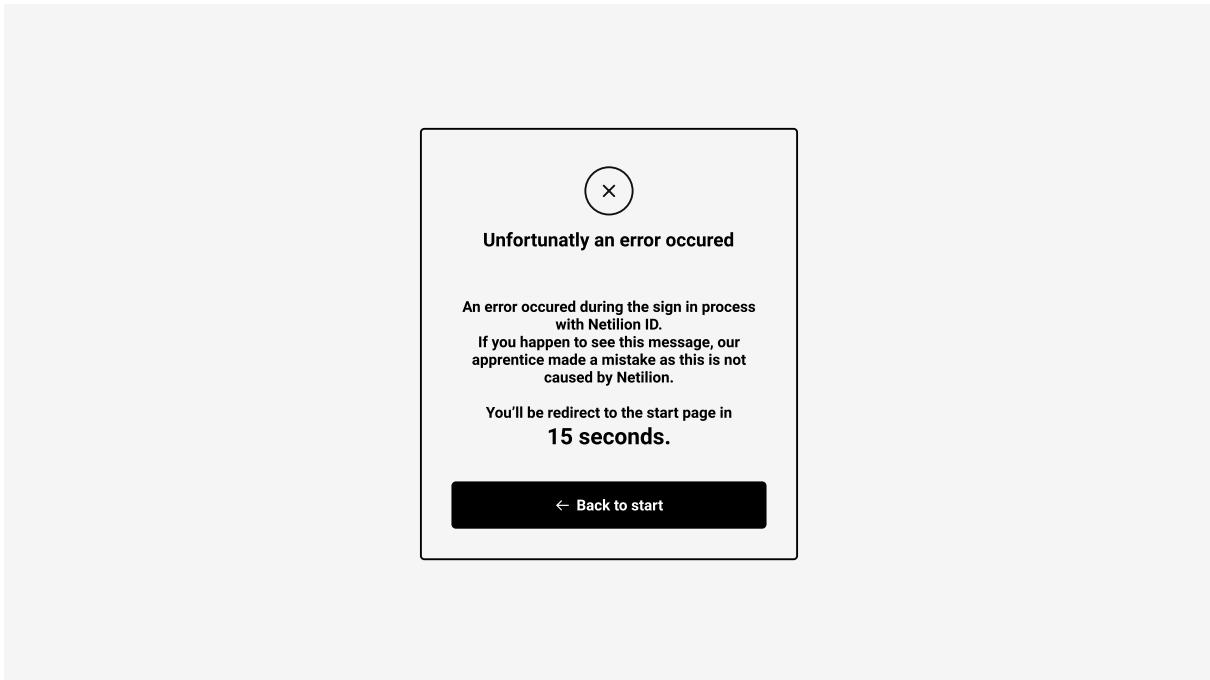


Abbildung 4.5.: Mockup einer Fehlermeldung

Anmeldung erfolgreich

Diese Ansicht wird angezeigt, wenn der Loginprozess erfolgreich abläuft. Dabei wird im oberen Teil angezeigt, dass dies der Erste der insgesamt drei Schritte ist, welche für die erfolgreiche Registration benötigt werden.

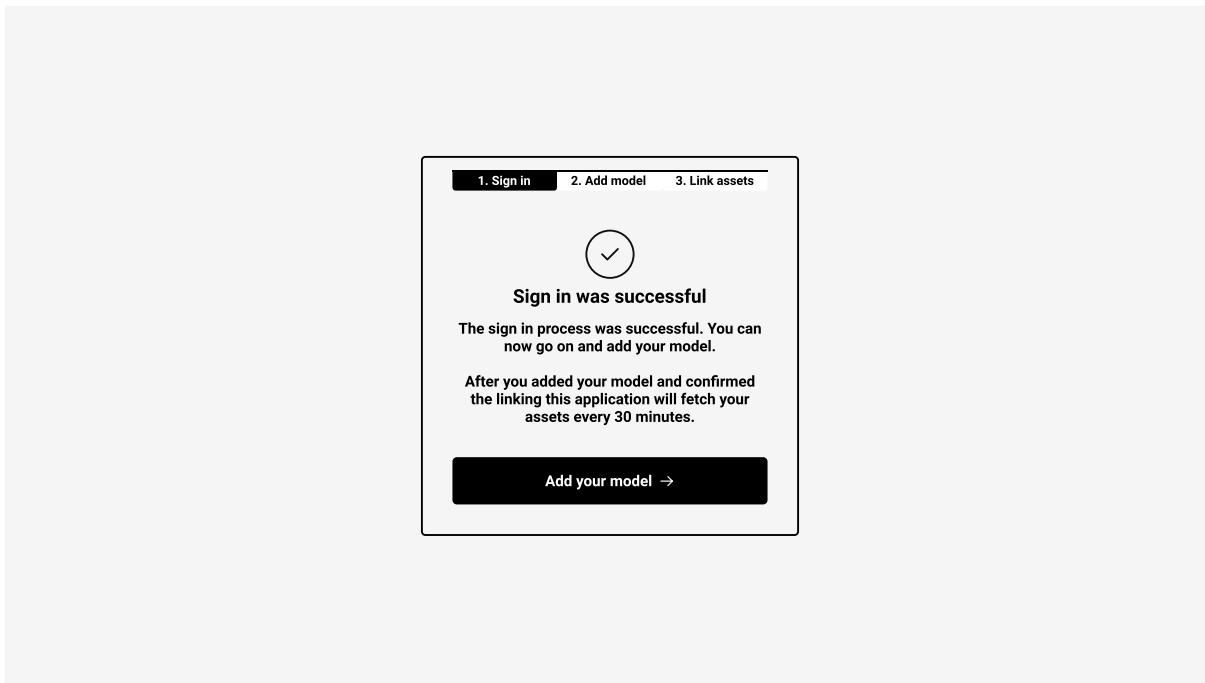


Abbildung 4.6.: Mockup einer erfolgreichen Anmeldung

Modell registrieren

Nun wird der OSE-Verantwortlicher darum gebeten, seinem Modell einen Namen zu geben und den Standort anzugeben, damit die User sein Modell dann über die Index Page finden können.

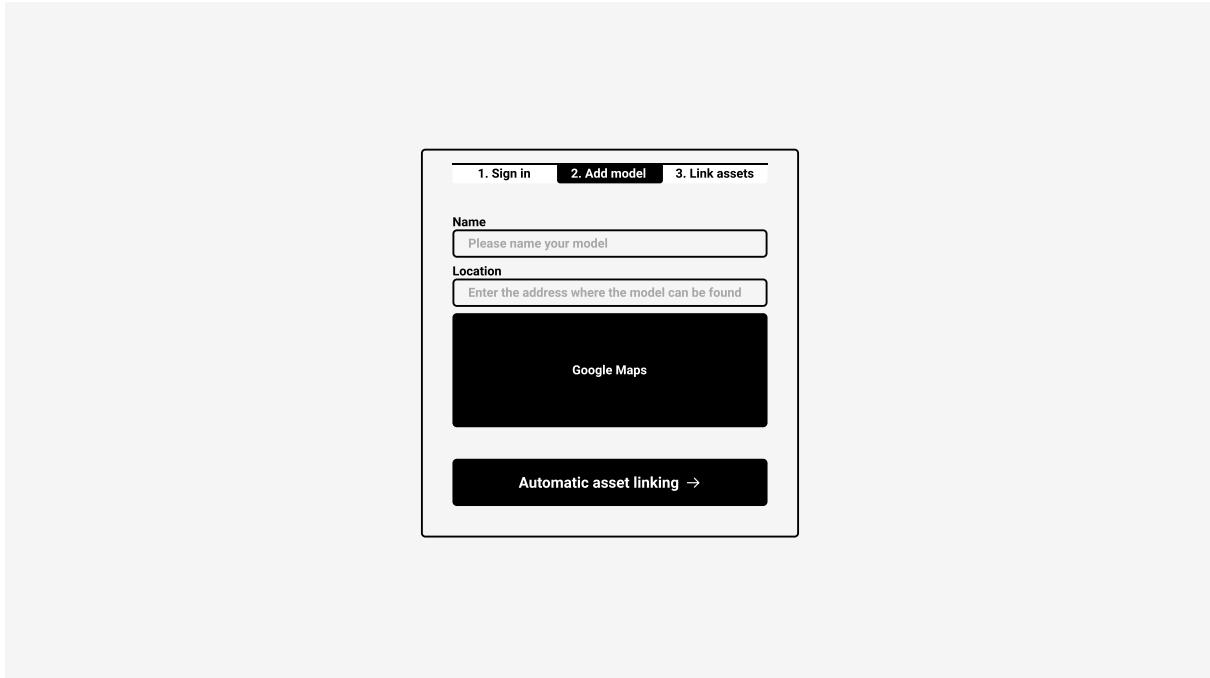


Abbildung 4.7.: Mockup der Registration

Asset verlinkung

Der letzte Schritt der Registrierung ist das Verlinken der Assets mit den Meshes des 3D Modells. Da dies automatisch geschieht, wird der User an dieser stelle gebeten zu warten, bis der Prozess abgeschlossen ist.

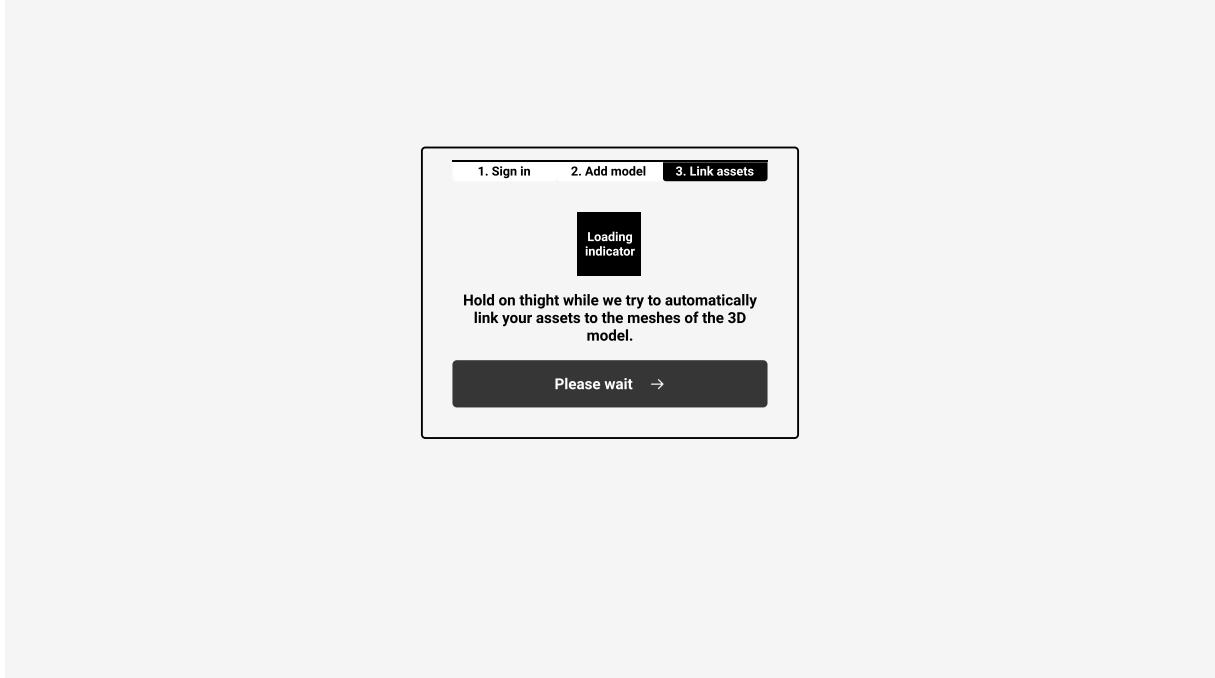


Abbildung 4.8.: Mockup der Asset verlinkung

Asset verlinkung fehlgeschlagen

Konnten nicht alle Assets verlinkt werden, wird dieses Fenster angezeigt. Es erklärt dem User was geschah und das er die Verlinkung im nächsten Schritt manuell anpassen kann.

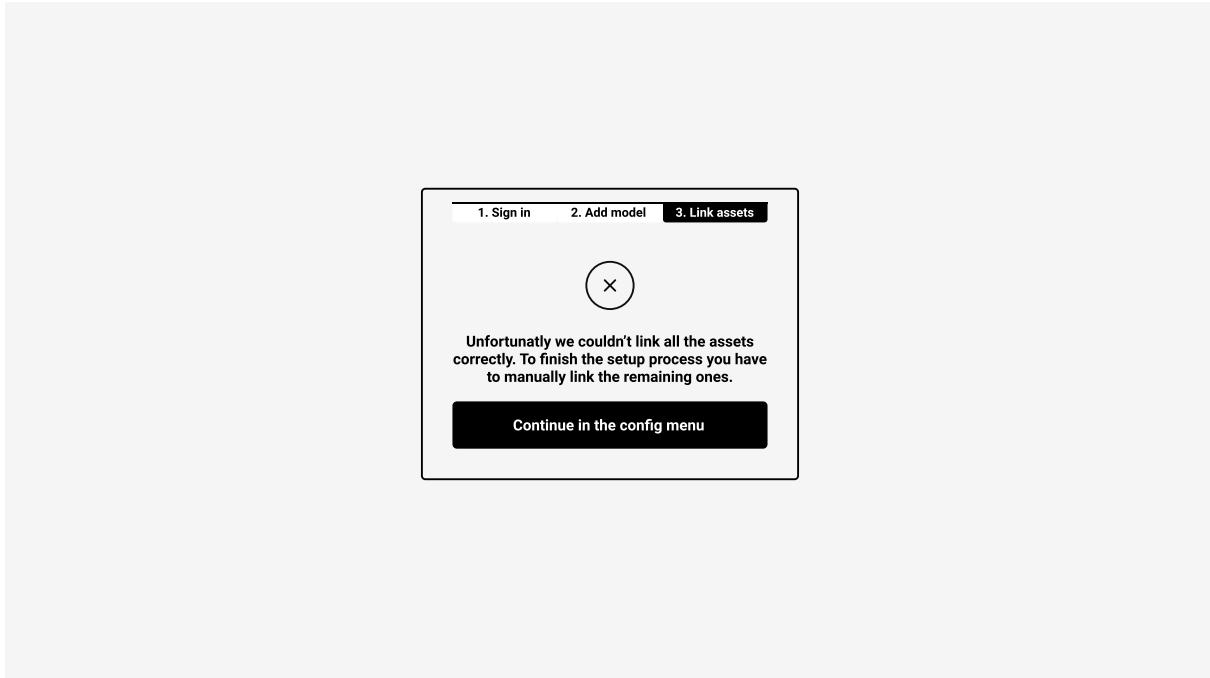


Abbildung 4.9.: Mockup einer fehlgeschlagenen Verlinkung

Asset verlinkung erfolgreich

Konnten alle Assets verlinkt werden, wird dieses Fenster angezeigt. Es informiert den User, dass der Registrationsprozess hiermit abgeschlossen ist. Mit dem Knopf am unteren Rand wird er ausgeloggt und kehrt zur Startseite zurück.

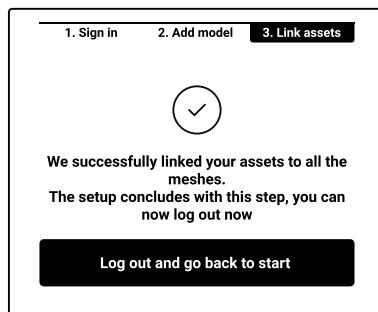


Abbildung 4.10.: Mockup einer erfolgreichen Verlinkung

4.4.3. Einstellungsmenü

Modelleinstellungen

Nachfolgend ist ein Design für die Modelleinstellungsseite. Hier kann der OSE Verantwortlicher den Namen und die Beschreibung des Modells ändern.

The mockup shows a user interface for 'Settings for model {mode.name}'. On the left is a sidebar with three items: 'Model' (selected), 'Location', and 'Asset linking'. The main area contains 'Model settings' with fields for 'Name' (placeholder: 'Please name your model') and 'Description' (placeholder: 'Please describe your model'). A 'Save and sign out' button is at the top right. At the bottom is a 'Back to start' button. The background is light gray.

Abbildung 4.11.: Mockup des Modelleinstellungsmenü

Standorteinstellungen

Dies ist das Design für die Standortseinstellungsseite. Hier kann der Standort im nachhinein verändert werden.

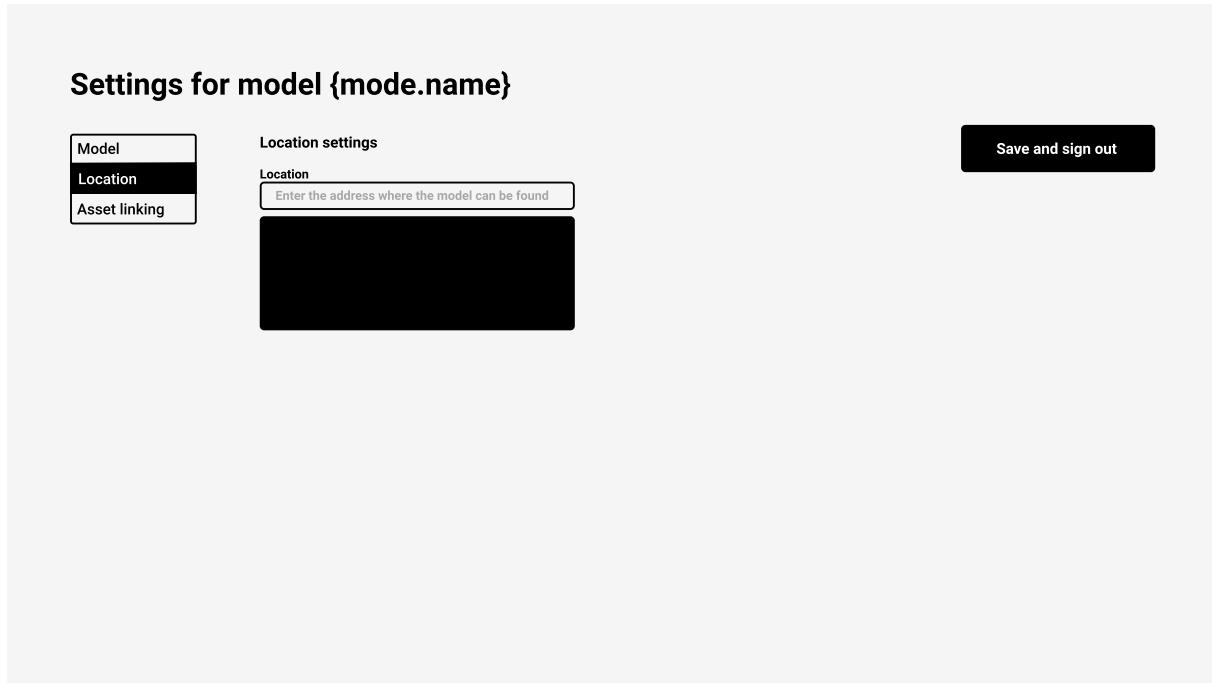


Abbildung 4.12.: Mockup des Standorteinstellungsmenü

Verlinkungseinstellungen

Nachfolgend ist ein Design für die Verlinkungseinstellungsseite. Hier kann der OSE Verantwortlicher Verlinkung des Modells ändern.

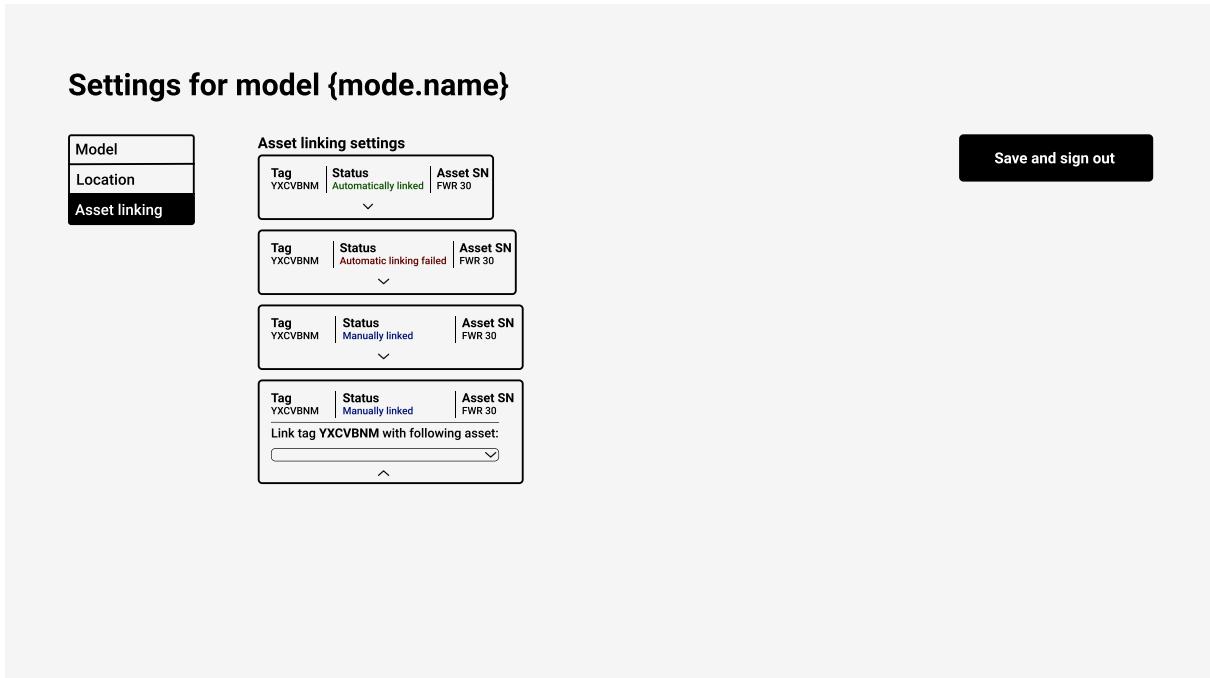


Abbildung 4.13.: Mockup des Verlinkungseinstellungsmenü

4.5. Testkonzept

Damit sich bei der Abnahme des Projektes sicher gegangen werden kann, dass alles wie gewünscht funktioniert, wird ein Testkonzept erarbeitet. Dieses soll sich auf die bereits erstellten User-Stories stützen und so alle Anforderungen der Anspruchsgruppen abdecken. Nach der Implementierungsphase wird dieses Konzept befolgt und differenzen dokumentiert.

Einzelne Test-Cases werden wie folgt benannt:

Segment	Abkürzung	Beschreibung
1	TC	Abkürzung für Test Case
2	-	Trennzeichen
3	01	Fortlaufende Kennzahl

4.5.1. Fehlerklassen

Damit Testergebnisse nach der Prüfung priorisiert werden können, werden ihnen Fehlerklassen zugewiesen. Fehlerklassen sind folgenderweise definiert:

Fehlerklasse	Beschreibung
---------------------	---------------------

- | | |
|---|--|
| 0 | Definiertes Resultat stimmt mit dem getesteten Resultat überein |
| 1 | Definiertes Resultat weicht von dem getestet Resultat ab, beeinträchtigt die Funktionalität allerdings nicht |
| 2 | Definiertes Resultat weicht von dem getestet Resultat ab und beeinträchtigt die Funktionalität |
| 3 | Definiertes Resultat und getestetes Resultat stimmen nicht überein |
-

4.5.2. Test Case Schema

Segment	Beschreibung
----------------	---------------------

- | | |
|---------------------|---|
| Name | Name des Testes, gemäss des Namenskonzeptes |
| Tester | Name der Person, die den Test durchgeführt hat |
| Erwartetes Resultat | Vom Nutzer gewünschtes Resultat |
| Effektives Resultat | Eingetroffenes Resultat |
| Szenario | In Schritten definiertes Testszenario |
| Fehlerklasse | Zugewiesene Fehlerklasse nach der Überprüfung |
| Weiteres Vorgehen | Vorgehensbeschreibung, sollte die Fehlerklasse nicht 0 sein |
| Status | Zeigt, ob der Test Case geprüft wurde |
-

4.5.3. User-Stories Abdeckung

Test-Case	Beschreibung	Betroffene User-Story
TC-01	Login möglich und sicher -> OAuth2konzept befolgt	US-01
TC-02	Fehlermeldung Login	US-01
TC-03	Automatische Verlinkung Assets	US-02
TC-04	Konfigurationsmenü aufrufbar	US-03
TC-05	Verlinkungen manuell änderbar	US-03
TC-06	Konfigurationsmenü nur durch eingeloggten User, welcher sich in der Usergruppe befindet, aufrufbar	US-04
TC-07	Übersicht aller Standorte	US-06
TC-08	Standort änderbar	US-06

4.5.4. Test-Cases

TC-01: Login

Name	TC-01
Tester	Jonas Schultheiss
Erwartetes Resultat	OSE Verantwortlicher kann sich anmelden
Effektives Resultat	Das effektive Resultat stimmt mit dem erwarteten überein
Fehlerklasse	0
Weiteres Vorgehen	Nicht nötig
Status	Durchgeführt am 13.04.2021

Schritt	Beschreibung	Erwartetes Resultat
0	Benutzer befindet sich auf der Startseite	/
1	Benutzer klickt den Knopf «Sign in with Netilion» an	Client des Benutzers wird an Netilion ID weitergeleitet
2	Benutzer gibt seine Daten an und loggt sich ein	Client des Benutzers wird zurück an das OSE-Dashboard geleitet
3	Benutzer nicht registriert	Benutzer wird zur Registration weitergeleitet
Alternative	Benutzer registriert	Benutzer wird zum Konfigurationsmenü weitergeleitet

TC-02: Login falsche Logindaten

Name	TC-02
Tester	Jonas Schultheiss
Erwartetes Resultat	Anmeldeprozess läuft schief, aber wird bei Netilion ID direkt behandelt
Effektives Resultat	Das effektive Resultat stimmt mit dem erwarteten überein
Fehlerklasse	0
Weiteres Vorgehen	Nicht nötig
Status	Durchgeführt am 13.04.2021

Schritt	Beschreibung	Erwartetes Resultat
0	Benutzer befindet sich auf der Startseite	/
1	Benutzer klickt den Knopf «Sign in with Netilion» an	Client des Benutzers wird an Netilion ID weitergeleitet
2	Benutzer gibt seine Daten an und probiert sich anzumelden	Netilion ID zeigt Fehlermeldung an

TC-03: Login mit unberechtigtem Account

Name	TC-03
Tester	Jonas Schultheiss
Erwartetes Resultat	Anmeldeprozess funktioniert und Bentzer erhält passende Fehlermeldung
Effektives Resultat	Das effektive Resultat stimmt mit dem erwarteten überein
Fehlerklasse	0
Weiteres Vorgehen	Nicht nötig
Status	Durchgeführt am 13.04.2021

Schritt	Beschreibung	Erwartetes Resultat
0	Benutzer befindet sich auf der Startseite	/
1	Benutzer klickt den Knopf «Sign in with Netilion» an	Client des Benutzers wird an Netilion ID weitergeleitet
2	Benutzer gibt seine Daten an und meldet sich an	Client des Benutzers wird zurück an das OSE-Dashboard geleitet, wo eine passende Fehlermeldung angezeigt wird

TC-04: Automatische Verlinkung der Assets

Name	TC-04
Tester	Jonas Schultheiss
Erwartetes Resultat	Assets und Meshes werden automatisch verlinkt
Effektives Resultat	Kann ein Asset nicht mit einem Mesh verlinkt werden, bekommt dies der Nutzer nicht mit
Fehlerklasse	1
Weiteres Vorgehen	Wird in Kapitel 7 beschrieben
Status	Durchgeführt am 13.04.2021

Schritt	Beschreibung	Erwartetes Resultat
0	Noch nicht registrierter Benutzer hat sich / bereits angemeldet und befindet sich auf dem zweiten Tab der Registration	
1	Benutzer geht zum nächsten Tab	Client zeigt an, dass die Verlinkung im Gange ist und Backend verlinkt die Entitäten
2A	Alle Assets verlinkt	Benutzer wird benachrichtigt und kann sich ausloggen
Alternative	Nicht alle Assets verlinkt	Benutzer wird benachrichtigt und kann zum Konfigurationsmenü fortfahren

TC-05: Konfigurationsmenü aufrufbar

Name	TC-05
Tester	Jonas Schultheiss
Erwartetes Resultat	Konfigurationsmenü ist aufrufbar
Effektives Resultat	Das effektive Resultat stimmt mit dem erwarteten überein
Fehlerklasse	0
Weiteres Vorgehen	Nicht nötig
Status	Durchgeführt am 13.04.2021

Schritt	Beschreibung	Erwartetes Resultat
0	Bereits registrierter Benutzer befindet sich / auf der Startseite	
1	Benutzer meldet sich an	Client wird zum Konfigurationsmenü weitergeleitet

Schritt	Beschreibung	Erwartetes Resultat
0	Noch nicht registrierter Benutzer befindet sich auf der Startseite	
1	Benutzer meldet sich an	Client wird zur Registration weitergeleitet
2	Benutzer registriert sich	Client wird zum Konfigurationsmenü weitergeleitet

TC-06: Verlinkung manuell änderbar

Name	TC-06
Tester	Jonas Schultheiss
Erwartetes Resultat	Assets und Meshes werden manuell verlinkt
Effektives Resultat	Das effektive Resultat stimmt mit dem erwarteten überein
Fehlerklasse	0
Weiteres Vorgehen	Nicht nötig
Status	Durchgeführt am 13.04.2021

Schritt	Beschreibung	Erwartetes Resultat
0	Bereits registrierter Benutzer befindet sich / im Konfigurationsmenü im Assets Tab	/
1	Benutzer klickt auf eine Verlinkung	Verlinkungskomponente öffnet sich
2	Benutzer klickt auf das Dropdown	Dropdown öffnet und stellt alle verfügbaren Assets dar
3	Benutzer selektiert anderen Mesh	Knopf «Save changes» wird freigeschaltet
4	Benutzer klickt Knopf	Änderungen werden gespeichert

TC-07: Übersicht aller Standorte

Name	TC-07
Tester	Jonas Schultheiss
Erwartetes Resultat	Benutzer sieht übersicht aller Standorte
Effektives Resultat	Das effektive Resultat stimmt mit dem erwarteten überein
Fehlerklasse	0
Weiteres Vorgehen	Nicht nötig
Status	Durchgeführt am 13.04.2021

Schritt	Beschreibung	Erwartetes Resultat
0	Benutzer befindet sich auf der Startseite	/
1	Benutzer öffnet die Augen	Benutzer sieht auf der initialen Ansicht der Startseite die Standortauswahl

TC-08: Standort änderbar

Name	TC-08
Tester	Jonas Schultheiss
Erwartetes Resultat	Standort ist änderbar
Effektives Resultat	Das effektive Resultat stimmt mit dem erwarteten überein
Fehlerklasse	0
Weiteres Vorgehen	Nicht nötig
Status	Durchgeführt am 13.04.2021

Schritt	Beschreibung	Erwartetes Resultat
0	Bereits registrierter Benutzer befindet sich auf der Startseite	/
1	Benutzer meldet sich an	Benutzer wird angemeldet und zum Konfigurationsmenü weitergeleitet
2	Benutzer klickt auf den Tab «Location»	Benutzer wird zu <code>/settings/location</code> weitergeleitet
3	Benutzer gibt im Eingabefeld eine neue Adresse ein	Passende Adressen werden vorgeschlagen
4	Benutzer klickt auf einen Adressenvorschlag	Adresse wird im Eingabefeld vervollständigt
5	Benutzer klickt auf den Knopf «Save changes»	Änderungen werden gespeichert

4.6. Datenbankerweiterung

Damit die gewünschten Änderungen implementiert werden können, muss die Datenbank erweitert werden. Die Änderungen sind in der Abbildung 4.14 grün dargestellt. Grau markierte Tabellen sind bereits bestehende Elemente, welche in der Vorarbeit implementiert wurden.

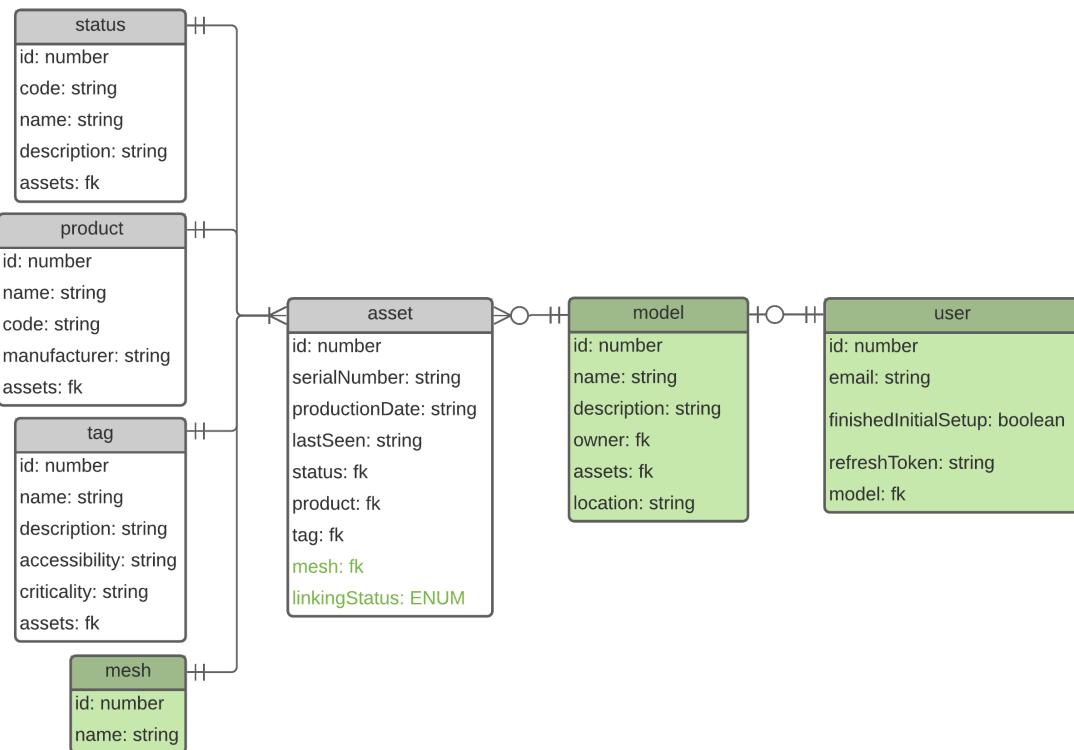


Abbildung 4.14.: ERM welches die Datenbankerweiterung visualisiert

4.6.1. User

Die Userentität hat drei Verwendungen:

- Soll zu einem JWT geformt werden, welcher zur Authentifizierung verwendet werden soll
- Soll den `refresh_token` speichern
- Soll speichern, ob der User die Registration abgeschlossen hat

4.6.2. Model

Das Modell soll einen Namen und eine Beschreibung enthalten. Diese Angaben können später verwendet werden, wenn das jeweilige Modell angezeigt wird. Das Attribut «location» enthält eine Id, mit

welcher die genaue Adresse und Koordinaten beim Backend angefragt werden können. Außerdem dient das Modell als Bindeglied zwischen dem «User» und den «Assets».

4.6.3. Mesh

Das Mesh wird verwendet um die Verlinkung zwischen dem 3D-Modell und dem einzelnen Asset zu ermöglichen. Das Mesh selbst enthält dabei nur einen Namen. Bei der Planung dieser IPA wurde nicht beachtet, dass die Namen der Meshes hinzugefügt werden müssen. Da es keinen Administrator in der Applikation gibt und Nestjs, beziehungsweise das darunterliegende ORM TypeOrm, kein Seeding unterstützt, müssen diese Daten manuell der Datenbank hinzugefügt werden. Im Repository des Backends soll ein SQL und ein CSV hinterlegt werden, womit die Daten dann mit Postico, einem PostgreSQL Client, hinzugefügt werden können.

Damit das Frontend auch anzeigen kann, ob ein Asset nun automatisch oder manuell verlinkt wurde, soll der Assetentität ein enum mit dem Namen «linkingStatus» hinzugefügt werden. Dieses enum soll folgende Auswahl zur Verfügung stellen:

- Nicht verlinkt
- Automatisch verlinkt
- Automatische Verlinkung schlug fehl
- Manuell verlinkt

4.7. Authentifizierung mit JWT

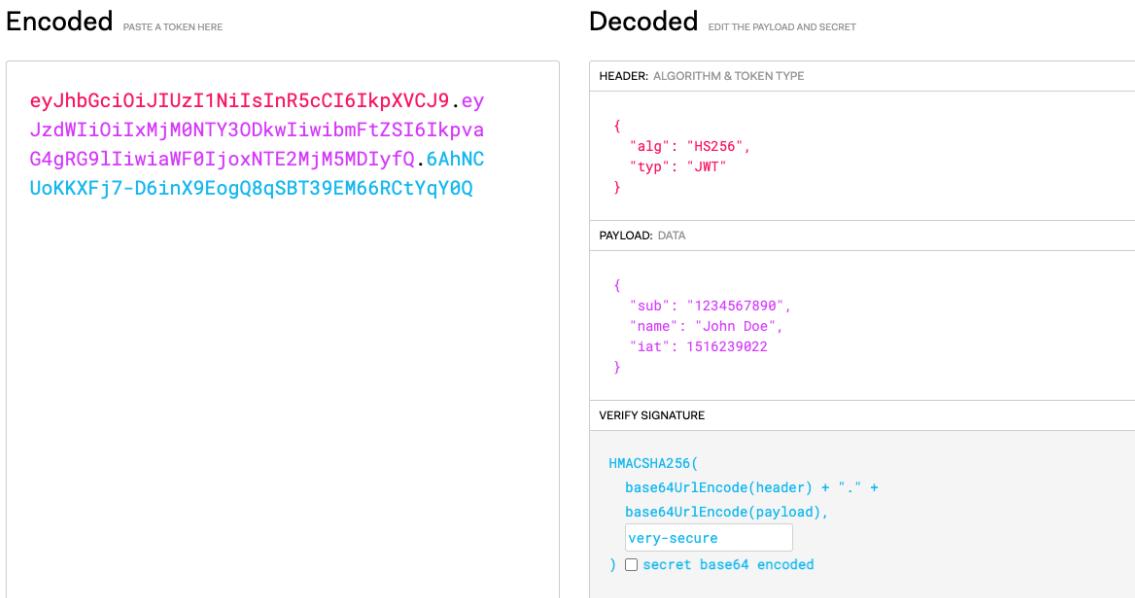
Die Authentifizierung zwischen dem Front- und Backend, wird mit «JSON Web Tokens» gelöst. Dies wurde vom Kandidaten in einigen vorherigen Arbeiten so angewendet und ist für ihn der Standard, wenn die Schichten getrennt werden.

4.7.1. Was ist ein JWT

«JSON Web Token (JWT) ist ein offener Standard (RFC 7519), der eine kompakte und in sich geschlossene Methode zur sicheren Übertragung von Informationen zwischen Parteien als JSON-Objekt definiert. Diese Informationen können verifiziert und vertrauenswürdig sein, da sie digital signiert sind. JWTs können mit einem Geheimnis (mit dem HMAC-Algorithmus) oder einem öffentlichen/privaten Schlüsselpaar mit RSA oder ECDSA signiert werden.

Obwohl JWTs auch verschlüsselt werden können, um die Geheimhaltung zwischen den Parteien zu gewährleisten, werden wir uns auf signierte Token konzentrieren. Signierte Token können die Integrität der darin enthaltenen Ansprüche verifizieren, während verschlüsselte Token diese Ansprüche vor anderen Parteien verborgen. Wenn Token mit öffentlichen/privaten Schlüsselpaaren signiert werden, bescheinigt die Signatur auch, dass nur die Partei, die den privaten Schlüssel besitzt, diejenige ist, die sie signiert hat.»[3].

Ein JWT besteht aus drei Bestandteilen: dem Header, der Payload und der Signatur. Der Header enthält den Algorithmus mit dem der JWT encoded wird und den Typ des Tokens. Der Payload kann von der Applikation selbst bestimmt werden, sollte allerdings klein gehalten werden, da der JWT bei jeder Anfrage im Header als «Authorization» mitgesendet wird. Laut dem Standard müssen die Attribute «sub» und «iat» enthalten sein. Sie sagen nämlich aus, wer das Subjekt ist und wann der JWT ausgestellt wurde. Der letzte Teil des Tokens ist die Signatur, womit der Aussteller des Tokens die Integrität überprüfen kann.



The screenshot shows the jwt.io interface. On the left, under 'Encoded' (PASTE A TOKEN HERE), there is a long string of encoded data in blue. On the right, under 'Decoded' (EDIT THE PAYLOAD AND SECRET), the token is split into three sections:

- HEADER: ALGORITHM & TOKEN TYPE**: Contains the JSON:

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```
- PAYOUT: DATA**: Contains the JSON:

```
{
  "sub": "1234567890",
  "name": "John Doe",
  "iat": 1516239022
}
```
- VERIFY SIGNATURE**: Contains the verification code:

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  very-secure
) □ secret base64 encoded
```

Abbildung 4.15.: Screenshot von JWT.io

4.7.2. Anwendung

Bei einer erfolgreichen Anmeldung erstellt das Backend einen JWT. Dessen Inhalt ist die Id des Nutzers, die Email, ein Boolean womit das Frontend weiss, ob der User bereits ein Modell registriert hat, das Erstellungsdatum und das Verfallsdatum. Dieser wird an das Frontend zurückgegeben. Das Frontend speichert diesen im SessionStorage und sendet ihn bei jeder weiteren Anfrage mit. Somit weiss das Backend immer, wer die Anfrage erstellt hat.

Sollte zum Beispiel die Id des Nutzers im JWT verändert werden, stimmt die Signatur nicht mehr und das Backend lehnt die Anfrage mit dem HTTP Status Code 401 ab.

KAPITEL 5

IMPLEMENTIERUNG

5.1. OAuth2

5.1.1. Frontend

```
1 /*[...]*/
2
3 const AuthContext = createContext();
4
5 /*[...]*/
6
7 export const AuthProvider = ({ children }) => {
8   /*[...]*/
9
10  useEffect(() => {
11    const login = async () => {/*[...]*/};
12
13    login();
14  }, /*[...]*/);
15
16  const loadUserFromSessionStorage = useCallback(async () => {/*[...]*/}
17  }, []);
18
19  useEffect(() => {
20    loadUserFromSessionStorage();
21  }, [loadUserFromSessionStorage]);
22
23  const logout = () => {/*[...]*/};
24
25  return (
26    <AuthContext.Provider value={{ isAuthenticated: !!user, user, isLoading, requestError, logout, JWT }}>
27      {children}
28    </AuthContext.Provider>
29  );
30 };
31
32 export const useAuth = () => useContext(AuthContext);
33
```

Abbildung 5.1.: authContext.js von ose-dashboard-frntnd

Dinge wie der JWT, Nutzerinformationen oder Funktionen wie «login» und «logout» müssen global in der Nextjs Applikation zur verfügung stehen. Vor ein paar Jahren hätte man dies noch mit Redux lösen müssen. Mittlerweile gibt es allerdings in React ein Konzept mit dem Namen «Context». Context erlaubt es einem, einen Wrapper mit State und Funktionen zu schreiben und diese den Subkomponenten zur verfügung zustellen. Werden Nutzerinformationen oder dazugehörige Funktionen in einer Subkomponente verwendet, können diese mit der `useAuth` Hook konsumiert werden. Dies sieht wie folgt aus:

```

1 import Button from '@components/button';
2 import SettingsNav from '@components/settingsNav';
3 import { useAuth } from 'contexts/authContext';
4
5 export default function SettingsLayout({ children, saveDisabled, clicked }) {
6   const { user, logout } = useAuth();
7
8   return (
9     <div className="p-12">
10      <h1 className="text-gray-900 text-2xl pb-2 font-bold md:text-3xl">Settings for model {user?.model?.name}</h1>
11      <div className="flex flex-row justify-between mt-8">
12        <SettingsNav />
13        <div className="flex flex-col w-1/3">{children}</div>
14        <div className="flex flex-col">
15          <Button clicked={logout}>Sign out</Button>
16          <Button clicked={clicked} disabled={saveDisabled}>
17            Save changes
18          </Button>
19        </div>
20      </div>
21    </div>
22  );
23}
24

```

Abbildung 5.2.: settingsLayout.js von `ose-dashboard-frntnd`

In der Zeile 6 wird der «user» und die «logout» Funktion vom Context konsumiert. Dies ermöglicht das anzeigen des Namens des Modelles auf Zeile 10 und das abmelden auf Zeile 15.

5.1.2. Backend

Der schwierige Teil der Implementierung von OAuth2 befindet sich allerdings im Backend. Eine Bedingung ist, dass sich der Nutzer anmelden kann. Die Zweite ist, dass alle Modelle in einem Intervall die Daten ihrer Assets aktualisieren. Diese Funktionalitäten sind in den beiden nachfolgenden Kapiteln beschrieben.

Log-in

```

1 @Injectable()
2 export class AuthService {
3   constructor(
4     private oauth2Service: OAuthService,
5     private netilionRequestService: NetilionRequestService,
6     private usersService: UsersService,
7     private jwtService: JwtService
8   ) {}
9
10  async login(loginDto: LoginDto): Promise<IJWT> {
11    const { code } = loginDto;
12    const token = await this.oauth2Service.getInitialAccessToken(code);
13    const netilionUser = await this.netilionRequestService.getCurrentUser(token);
14    const { id, email, finishedInitialSetup } = await this.usersService.getOrCreate(netilionUser,
15      token.refreshToken);
16    const rawJWT: IJWTBody = { sub: id, email, finishedInitialSetup };
17    return {
18      accessToken: await this.jwtService.signAsync(rawJWT)
19    };
20  }

```

Abbildung 5.3.: auth.service.ts von `ose-dashboard-bcknd`

Das Frontend erhält bei einer erfolgreichen Anmeldung einen Code. Dieser Code sendet es weiter an das Backend. Als erstes wird der Code in einen Token umgewandelt. Mit diesem Token ist es nun möglich die Nutzerinformationen von Netilion abzufragen. Anschliessend wird die Function `getOrCreate` vom `usersService` mit den Nutzerinformationen und dem Token aufgerufen. Diese Funktion fragt zuerst bei der Datenbank nach, ob der Nutzer bereits existiert. Existiert er, wird er direkt zurückgegeben. Existiert er nicht, wird ein neuer erstellt. Danach werden die für den JWT-Payload nötigen Daten in einem Objekt gebündelt und vom `jwtService` in einen signed JWT formattiert.

Der JWT wird als Antwort auf die Anfrage zurück ans Frontend gesendet, wo er fortlaufend, wie in Kapitel 4.7.2 beschrieben, als Authentifizierungsmethode zwischen Front- und Backend verwendet wird.

Assets

```

1 @Injectable()
2 export class ModelsService {
3   constructor(
4     // [...]
5   ) {}
6
7   @Cron('*/30 * * * *')
8   async handleCron() {
9     const models = await this.findAll();
10    if (models) {
11      for await (const model of models) {
12        const user = await this.userService.findOneWithRefreshToken(model.owner.id);
13        await this.createOrUpdateAssets(user, model);
14      }
15    }
16  }
17
18  // [...]
19
20  private async createOrUpdateAssets(user: User, model: Model): Promise<void> {
21    const netilionResponses: NetilionResponseDto[] = await this.fetchAllAssets(user);
22
23    for await (const asset of netilionResponses) {
24      await this.assetsService.createOrUpdateAsset(asset, model);
25    }
26  }
27}
28

```

Abbildung 5.4.: models.service.ts von ose-dashboard-bcknd

Im `ModelsService` ist von Zeile sieben bis 16 ein Cron-Job deklariert. Dieser ist so konfiguriert, dass die `handleCron` Funktion auf Zeile acht alle 30 Minuten aufgerufen wird. Diese Funktion fragt zuerst beim Repository alle Modelle nach. Anschliessend wird in einer Schleife von jedem Modell der dazugehörige Nutzer abgefragt, womit danach die Funktion `createOrUpdateAssets` aufgerufen wird. Diese Funktion fragt zuerst bei Netilion nach allen Assets des Users nach. Nachdem die Antwort erhalten wurde, schleift sie über alle Assets und ob diese erstellt werden müssen oder einfach aktualisiert werden können.

5.2. Account löschen

Normalerweise kann man sich bei einem Dienst wieder abmelden und dank GDDPR auch alle seine Daten löschen lassen. In unserem Fall sollte der `refresh_token` des OSE-Verantwortlichen revoked werden, woraufhin die Entität gelöscht wird. Im optimalen Falle sollte das ORM so konfiguriert sein, dass danach alle Daten kaskadierend gelöscht werden. Das bedeutet, dass alle Einträge in der Datenbank, welche mit dem User in Verbindung gebracht werden können, nacheinander gelöscht werden.

Da dies ein bedeutender Auwand ist, welcher nicht von dieser IPA verlangt wird, werde ich dies nach der Abschlusspräsentation implementieren.

5.3. Modellauswahl

Es wurde schon bei Kapitel 4.4.1 auf die Abbildung 4.4 eingegangen, dabei wurde jedoch ein Punkt bewusst weggelassen. Die Fläche, welche mit «Model selection» markiert ist, soll nicht einfach leer sein, sondern dem Nutzer das Navigieren der Modelle einfacher zu machen.

Bei den Mockups wurde allerdings das ganze noch nicht verfeinert, da mögliche Lösungen zuerst evaluiert werden sollen. Momentan stehen zwei Möglichkeiten zur Auswahl bereit. Diese werden in den Kapiteln 5.3.1 und 5.3.2 beschrieben.

5.3.1. Auflistung mit Landesflaggen

Die initiale Idee der Modellauswahl war eine Auflistung der Modelle nach den Ländern. Damit ich dies im Frontend machen kann, muss ich zuerst im Controller der Modelle eine neue Route implementieren, welche sie nach den Ländern geordnet zurücksendet.

Meiner Meinung nach ist diese Möglichkeit sehr simpel, was nicht negativ gemeint ist. Sollte ich merken, dass die andere Möglichkeit Probleme macht oder machen könnte, werde ich diese implementieren.

Pro

- Einfach zu implementieren
- Erfüllt den Zweck
- Lauft etwas schief, kann ich mir selbst weiterhelfen

Kontra

- Eine dynamische Darstellung von Flaggen, in der Form von Bildern oder Emojis, könnte sich als schwieriger als Gedacht herausstellen
- Schwierig schön darzustellen
- Mehr Fleissarbeit um dies zu implementieren

5.3.2. Interaktive Weltkarte

Eine Idee die von mir kommt, ist die Darstellung mit einer Weltkarte. Ich hatte diese Idee bevor die IPA begann und habe mich in der Freizeit weiter informiert. Zuerst dachte ich, ich könnte eine Karte im SVG-Format nehmen und selbst eine solche React-Komponente erstellen. Schnell wurde klar, dass sich dies als sehr grosser Aufwand herausstellen würde. Dadurch begann ich mit der recherche nach Libraries von Dritten, welche sich diese mühe bereits gemacht haben. Somit stieß ich auf [react-simple-maps](#). Mit dieser Komponente ist es mir möglich, eine Weltkarte mit Markern darzustellen. Die Marker werden dabei mit Koordinaten gesetzt. Da ich bei der Registration die Koordinaten des Modells erhalte, sollte dies kein Problem sein.

Pro

- Ohne Konfigurationen sieht diese Komponente schön aus
- Weniger Fleissarbeit um dies zu implementieren

Kontra

- Ich muss mich etwas in die Dokumentation einlesen
- Lauft etwas schief, kann mir nur Google oder die Dokumentation weiterhelfen

5.3.3. Tatsächliche Auswahl

Bei der Implementierung wird nun zweitere Möglichkeit, welche in Kapitel 5.3.2 beschrieben wurde, implementiert. Grund dafür ist, dass genügen Zeit zur Verfügung ist und nicht eine overengineerte Lösung gebastelt werden muss, welche kurz nach der IPA ersetzt wird. Begründet wird dieser Entscheid damit, dass die Dokumentation im Aspekt der Verwendung in diesem Projekt gründlich durchgelesen wurde.

5.4. Automatische Verlinkung von Assets und Meshes

Eine Anforderung dieser IPA ist, dass die Verlinkun von Assets und Meshes automatisch gemacht werden kann. Dies ist möglich, da sich bereits alle benötigten Informationen in der Datenbank des Backends befinden. Dies soll so die Erfahrung des Benutzer fördern und den Registrationsprozess vereinfachen.

```

1 @Injectable()
2 export class ModelsService {
3   constructor(
4     // [...]
5   ) {}
6
7   // [...]
8
9   async autoLinkAssets(id: number) {
10    const assets = await this.assetsService.getAssetsOfModel(id);
11    for await (const asset of assets) {
12      const { id } = asset;
13      await this.autoLinkAsset(id);
14    }
15  }
16
17  private async autoLinkAsset(id: number) {
18    const asset = await this.assetsService.findOne(id);
19    if (asset && asset.tag) {
20      const mesh = await this.meshesService.findOne(asset.tag.name);
21      if (mesh) {
22        try {
23          await this.assetsService.link(asset.id, mesh, LinkingStatus.AUTOMATICALLY_LINKED);
24        } catch (error) {
25          await this.assetsService.changeLinkingStatus(asset.id, LinkingStatus.AUTOMATIC_LINKING_FAILED);
26        }
27      } else {
28        await this.assetsService.changeLinkingStatus(asset.id, LinkingStatus.NOT_LINKED);
29      }
30    }
31  }
32
33  // [...]
34 }
35

```

Abbildung 5.5.: models.service.ts von ose-dashboard-bcknd

Die Funktion `autoLinkAssets` schleift über alle Assets eines Modells und ruft dabei für jedes einzelne `autoLinkAsset` auf. Diese erstellt die wirkliche Verlinkung. Dafür fragt es das Asset zuerst einzeln beim `assetsService` nach, da auf diesem Weg die Relation zu `Tag` left joined und selektiert wird. Wurde ein Asset mit einem Tag gefunden, wird beim `meshesService` nachgefragt, ob ein Mesh mit dem Namen des Tags existiert. Ist dies der Fall, werden die beiden miteinander verlinkt.

5.5. Einhaltung der Coding Guidelines

Die Coding Guidelines wurden eingehalten. Der Linter zeigt auf dem Front- und Backend weder Errors noch Warnings an. Es ist wichtig zu erwähnen, dass die Regeln nicht angepasst wurden und es keine Instanz gibt, an der eine Regel ignoriert wurde.

Dies kann bei beiden Repositories mit dem Kommand `yarn lint` überprüft werden.

```
1 yarn run v1.22.5
2 warning ../../package.json: No license field
3 $ eslint '**/*.{js,jsx}' --quiet
4 ✨  Done in 4.82s.
```

Abbildung 5.6.: Output von `yarn lint` im Frontend

```
1 yarn run v1.22.5
2 warning ../../package.json: No license field
3 $ eslint "{src,apps,libs,test}/**/*.{ts,tsx}" --quiet
4 ✨  Done in 7.52s.
```

Abbildung 5.7.: Output von `yarn lint` im Backend

Eine Anmerkung muss hierbei gemacht werden. Die «Warning» die in den Abbildungen 5.6 und 5.7 bezieht sich nicht auf das Linting. Es handelt sich dabei um eine Warnung des Packetmanagers «NPM». Dieser wirft die Warnung, da ihm die Lizenz in der Datei `package.json` unbekannt ist. Da es sich bei dieser Arbeit um kein öffentliches Projekt handelt, habe ich «license»: «UNLICENSED» angegeben.

5.6. Umgebungsvariablen

Umgebungsvariablen sind für fast alle Webanwendung nötig. Sie machen es möglich geheime oder heikle Daten der Anwendung erst zur Laufzeit zur Verfügung zu stellen. Auf diese Weise müssen die Daten nicht im Code fest angegeben werden. So kann ein Versionskontrollsystem eingesetzt werden, ohne das Dritte an heikle Daten kommen. Folgend sind die verwendeten Umgebungsvariablen beschrieben.

5.6.1. Umgebungsvariablen - Frontend

Variable	Beschreibung
NEXT_PUBLIC_AUTH_URL	Die URL, auf die der Benutzer während dem Anmeldeprozess weitergeleitet wird
NEXT_PUBLIC_API_BASE_URL	URL, welche auf eine Instanz des Backends zeigt.

5.6.2. Umgebungsvariablen - Backend

Variable	Beschreibung
PORT	Der Port unter dem das Backend erreichbar ist
DATABASE_URL	Datenbankverbindungsadresse
NETILION_AUTHURL	Basis URL für Anfragen an den OAuth2 Endpoint
NETILION_AUTHURL	Basis URL für Anfragen an den OAuth2 Endpoint
NETILION_URL	Basis URL für Anfragen an die Resourcen Endpoints
NETILION_REDIRECT_URI	URI, auf die der User nach dem Anmeldeprozess weitergeleitet wird
NETILION_CLIENT_ID	Id der erstellten OAuth2 Applikation in Netilion
NETILION_CLIENT_SECRET	Secret der erstellten OAuth2 Applikation in Netilion
REDIS_URL	Redis Verbindungsadresse
REDIS_TTL	Gibt an, wie lange Einträge in Redis gespeichert bleiben
SECURITY_JWT_SECRET	String, mit welchem die JWTs signiert werden
SECURITY_JWT_EXPIRES_IN	Gibt an, wie lange JWTs gültig sind
SECURITY_JWT_EXPIRES_IN	String, mit welchem die <code>refresh_token</code> verschlüsselt werden
GEOLOCATION_API_KEY	Zugriffsschlüssel, um auf die «here developer API» zugreifen zu können
PERMITTED_USERGROUP_ID	Id der Netilion UserGroup. Nur die darin enthaltenen Nutzer können sich anmelden
PERMITTED_USERGROUP_NAME	Name der Netilion UserGroup. Nur die darin enthaltenen Nutzer können sich anmelden

5.7. 3D Modell auswechseln

In der ersten Version des OSE-Dashboards stellte sich die korrekte Einbindung des Modells als schwierig heraus. Grund dafür ist, dass die Datei mehrmals mit speziellen Konfigurationen umgewandelt werden muss. Diese Schritte sind in diesem Kapitel beschrieben, damit nachfolgende Entwickler/-innen sich die Mühe und Zweit ersparen können.

1. Das Modell wird von einem Konstrukteur von Endress+Hauser erstellt. Da Konstrukteure CAD-Modelle modellieren, wird die Datei mit hoher Wahrscheinlichkeit vom Typ .stp sein.
2. Wird ein Auftrag beim Konstrukteur erstellt, muss dieser darauf hingewiesen werden, dass die

einzelnen Bestandteile des Modells für uns korrekt benannt werden müssen. Im Schritt 10 ist es sehr wichtig, dass die Benennung keine Umlaute (öäü) und Leerzeichen enthält.

3. Für die erste Umwandlung wird das Program «FreeCAD» verwendet. Dieses muss auf einem Windows Computer installiert werden.
4. Die .stp Datei importieren und als .dae exportieren
5. Für die nächste Umwandlung brauchen wir eine 3D Modellierungssoftware. Bisher wurde Blender verwendet, da dies keine Kosten verursacht.
6. In Blender kann das Modell nun nach belieben angepasst werden. Zum Beispiel können Farben verändert werden.
7. Anschliessend soll das Modell als .gltf exportiert werden. In diesem Schritt kann der Detailgrad bestimmt werden. Je höher der Detailgrad desto grösser ist die Datei. Eine grosse Datei verlangsamt die Ladezeit erheblich und erfordert moderne Hardware um flüssig angezeigt werden zu können.
8. Nun kann die .gltf Datei in den «public» Ordern von React/Nextjs gelegt werden. Dies ist notwendig, da der Client des Benutzers diese Datei herunterladen können muss.
9. Für den nächsten Schritt brauchen wir ein Script. Das Repository dafür ist `pmndrs/gltfjsx`. Dieses muss nicht installiert werden, sondern kann als Remote Script mit NPM ausgeführt werden. In der Dokumentation des Scriptes ist die korrekte Anwendung beschrieben.
10. Nun sollte mit gltfjsx aus der .gltf Datei eine React Komponente generiert werden. Diese sollte in den «components» Ordner verschoben werden.
11. Um das Modell darzustellen, muss das Modell in der Scene Komponente eingebunden werden.
12. Damit das Modell interaktiv wird, müssen noch einige kleine Konfigurationen gemacht werden. Die Szene enthält eine Variable «assetSelected». Diese muss der Modell Komponente als property weitergegeben werden.
13. Anschliessend müssen die Meshes im Modell, welche ein Asset darstellen, durch die Komponente «Asset» ersetzt werden und mit zusätzlichen Informationen ausgestattet werden. Dies wird in der Abbildung 5.8 dargestellt.

```

1 // [...]
2 <mesh
3   material={materials.Material}
4   geometry={nodes.Plexideckel_400mm.geometry}
5   position={[ -0.5, 12.399999618530273, -7.415002346038818]}
6   rotation={[Math.PI / 2, 2.084434279e-7, -1.5594242608490143]}
7   scale={[ [0.0099999997764826, 0.0099999997764826, 0.0099999997764826] ]}
8 >
9 <Asset
10  material={cadMat07}
11  geometry={nodes.TrustSens__1_TM371.geometry}
12  position={[ 2.15198016166687, 17.569725036621094, -14.739999771118164]}
13  rotation={[ 0, 0, -0.0169431638304912 ] }
14  scale={[ [0.0099999997764826, 0.0099999997764826, 0.0099999997764826] ]}
15  meshName={'TT1002'}
16  assetSelected={properties.assetSelected}
17 >
18 // [...]

```

Abbildung 5.8.: Ausschnitt aus der momentanen «Model.js» Komponente

Die «Asset» Komponente besteht aus einem Mesh. Daher werden auch alle Properties des Meshes verlangt. Zusätzlich muss der Mesh mit einem Tag benannt werden. Damit die hierarchisch höher liegende Komponente erkennen kann, wann welches Asset angeklickt wurde, muss noch die Funktion «assetSelected» mitgegeben werden.

Mit diesem Schritt ist die einbundung eines neuen Modelles abgeschlossen.

KAPITEL 6

TESTEN

Das Testen der fertigen Applikation wurde am 13.04.2021, wie im Kapitel 4.5 geplant, durchgeführt. Die Ergebnisse sind im folgendem Kapitel 6.1 dokumentiert.

6.1. Testergebnisse

Kapitel	Test-Case	Fehlerklasse
4.5.4	TC-01: Login	0
4.5.4	TC-02: Login falsche Logindaten	0
4.5.4	TC-03: Login mit unberechtigtem Account	0
4.5.4	TC-04: Automatische Verlinkung der Assets	1
4.5.4	TC-05: Konfigurationsmenü aufrufbar	0
4.5.4	TC-06: Verlinkung manuell änderbar	0
4.5.4	TC-07: Übersicht aller Standorte	0
4.5.4	TC-08: Standort änderbar	0

KAPITEL 7

ABSCHLUSS

7.1. Bekannte Fehler

7.1.1. Konfigurationsmenü

Durch das Konfigurationsmenü kann die Applikation momentan vom einem Benutzer zum abstürzen gebracht werden. Grund dafür ist das komplexere State Management, welches noch nicht 100% durchgedacht wurde. Mit bestimmten Interaktionen kann ein Fehler auftreten, welcher im momentanen Stand nicht behandelt wird. Dies sorgt so für einen Absturz. Allerdings wird die Funktionalität der Applikation dadurch nicht beeinträchtigt, da durch das «normale» interagieren der Fehler nicht auftritt.

7.1.2. Verlinkung von Assets zu Meshes

Die automatische und manuelle Verlinkung funktioniert und die Applikation ist nutzbar. Jedoch gibt es einige Edge Cases durch welche auffällt, dass die Implementation noch nicht ganz perfekt ist. Grund hierfür ist die darunterliegende Datenstruktur.

Die Abbildung 4.14 stellt dar, dass ich in der Vorarbeit die Tabellen «asset», «product», «tag» und «status» bereits erstellt habe. Die Priorität war damals, dass alle Daten in die Datenbank übernommen werden und dadurch so wenig Anfragen wie möglich an Netilion gesendet werden. Dazu kommt noch, dass die Schnittstelle für die Assets gleich noch die Informationen der restlichen drei Tabellen mitsenden kann, ohne das man mehrere Requests machen muss. Umgekehrt, also vom Tag zum Asset geht dies nicht.

Durch diese Voreinstellungen/Erfahrungen spielte die Asset Tabelle eine zentrale Rolle im Datenbankschema. Dies stellte sich später allerdings als negativ heraus. Die vorgeschlagene Lösung enthielt eine

Hilfstabelle namens «mesh», womit ein Asset verlinkt werden sollte. Durch diese Implementation ist es nun möglich, dass mehrere Assets des gleichen Modelles mit demselben Mesh verlinkt sind.

Eine bessere Lösung wäre es, wenn der Tag die zentrale Rolle übernehmen würde. Dadurch würde die Hilfstabelle «mesh» nicht mehr verwendet werden müssen. Außerdem könnte so ein Asset nur mit einem Mesh verlinkt werden, was die Fehleranfälligkeit vermindert.

Dies zu Ändern hätte allerdings den Rahmen einer IPA deutlich gesprengt. Gerne werde ich nach der IPA das ganze genauer analysieren und implementieren.

7.2. Verbesserungsmöglichkeiten

7.2.1. Layout verschiebungen

Es kommt im Frontend momentan sehr oft zu Layout verschiebungen. Damit gemeint ist, dass Elemente, mit denen der User interagieren möchte, durch State Changes die Position verändern. Aus User Experience Sicht ist dies schlecht gelöst. Beispiel hierfür ist das Vorschlagen von Adressen. Sobald der User einen Teil einer Adresse eingibt, erscheinen drei bis fünf Vorschläge. Diese Vorschläge schieben andere Elemente wie Eingabefelder weiter nach unten.

Dieses Problem kann in React sehr schön und einfach mit dem Drittanbieterpacket `framer-motion` gelöst werden. Ich habe bereits Erfahrungen damit gesammelt und würde gerne nach der IPA dies implementieren.

7.2.2. Refresh Token verschlüsseln

Im Dokument ist erwähnt, dass der `refresh_token` des Benutzers verschlüsselt in der Datenbank gesichert wird. Dies hatte während der Implementierungsphase eine niedrige Priorität, da es nicht direkt zur Funktionalität beitrug. Nun bleibt keine Zeit mehr für die Implementation. Ich werde dies baldmöglichst nach der IPA implementieren, da mir die Sicherheit der Applikation sehr wichtig ist.

7.2.3. Unit test-s

In der Aufgabenstellung stand, dass die Methode, welche die Assets mit Meshes verlinkt, mit automatischen Tests abgedeckt werden soll. Mit Tests habe ich in Nestjs sehr wenig Erfahrung, wodurch mich eine Implementation des Testes einiges an Zeit gekostet hätte. Ich habe dies nun bewusst weggelassen, damit ich mich auf andere Aufgaben fokussieren kann.

7.3. Verstösse gegen die Coding Guidelines

7.3.1. Frontend

/pages/settings/location.js:19

```

1  useEffect(() => {
2    const fetchModel = async () => {
3      setIsLoading(true);
4      try {
5        const { data: fetchedModel } = await baseAPI.get(`/users/${user.id}/model`);
6
7        setModel(fetchedModel);
8        const { data: requestedLocation } = await baseAPI.get(`/models/${fetchedModel.id}/location`);
9        setSelectedLocation(requestedLocation);
10       setAddress(requestedLocation.title);
11       setIsLoading(false);
12     } catch (error) {
13       setError(error.response.data.message);
14       setIsLoading(false);
15     }
16   };
17
18   if (user) {
19     fetchModel();
20   }
21 }, [user]);

```

Abbildung 7.1.: Abbildung der useEffect Hook in location.js

Dieser Verstoss ist zu Begründen auf die vielen State updates die innerhalb der useEffect Hook geschehen. Dies ist nicht wirklich sinnvoll auslagbar.

/pages/settings/model.js:16

```

1  useEffect(() => {
2    const fetchModel = async () => {
3      setIsLoading(true);
4      try {
5        const { data: fetchedModel } = await baseAPI.get(`/users/${user?.id}/model`);
6        setModel(fetchedModel);
7        setName(fetchedModel.name);
8        setDescription(fetchedModel.description);
9        setIsLoading(false);
10     } catch (error) {
11       setError(error.response.data.message);
12       setIsLoading(false);
13     }
14   };
15
16   if (user) {
17     fetchModel();
18   }
19 }, [user]);

```

Abbildung 7.2.: Abbildung der useEffect Hook in model.js

Dieser Verstoss folgt der gleich gleichen Begründung wie der erste.

7.3.2. Backend

/src/netilion-request/netilion-request.service.ts:43

```

1  async getAssets(user: User) {
2    const baseUrl = this.configService.get('netilion.url');
3    const token = await this.oauthService.getAccessToken(user);
4    const query =
5      `per_page=100&include=status%20%20pictures%20%20product%2C%20instrumentations%20%20product.manufacturer`;
6    const res = await axios.get(`${baseUrl}/assets?${query}`, {
7      headers: {
8        'Content-Type': 'application/json',
9        Authorization: `Bearer ${token.accessToken}`
10     }
11   });
12   if (!res.data || !res.data.assets) {
13     throw new InternalServerErrorException('Could not fetch assets');
14   }
15 }
16
17 return res.data.assets;
18 }
```

Abbildung 7.3.: Abbildung der getAssets Funktion

Bei dieser Funktion handelt es sich bereits um eine Auslagerung. Jede Zeile muss in dieser Funktion sein und ist sinnlos auszulagern.

/src/netilion-request/oauth.service.ts:18&43

Auch bei diesen Funktionen handelt es sich um ausgelagerte Rest Abfragen. Jegliche Teile auszulagern würde nur den Code schwerer zum Verstehen machen.

7.4. Schlussbetrachtung

7.4.1. Reflexion

Wie man vielleicht den Arbeitsjournalen ablesen konnte, war ich in den vergangen zehn Tagen durchgehend in Stresssituationen. Rückblickend gesehen, hätten wir den Auftrag etwas kürzen können. Ich hatte auch am Anfang der Implementierungsphase extremen Respekt davor, dass ich den Anmeldeprozess mit OAuth2 nicht schaffen werde. Beziehungsweise das es mir viel zu viel Zeit wegnehmen könnte, sodass ich folglich zu wenig Zeit für andere Aufgaben hätte. Dies ist zum Glück nicht aufgetreten und es verlief alles recht positiv.

7.4.2. Schlusswort

Ich konnte mit dieser Arbeit eine grosse Erweiterung des OSE-Dashboards erstellen. Diese kann nun von verschiedenen Endress+Hauser Standorten eingesetzt werden, um den Kunden die Möglichkeiten von Netilion Connect näher zu bringen.

7.4.3. Persönliche Bilanz

Ich bin mir sicher, dass ich einige Erfahrungen dieser Arbeit in mein weiteres Berufsleben mitnehmen kann. Ich konnte unter Druck arbeiten, wobei meine Leistung in den ganzen zehn Tagen nicht abgenommen hat. Ich hatte viele Momente in diesen Tagen, an denen ich am liebsten einfach nichts gemacht hätte für ein bis zwei Stunden, da ich so unter Druck stand. Jedoch habe ich es durchgezogen und merkte dadurch, dass dies nur eine kleine gedankliche Barrier war, die ich überwältigen konnte. Abgesehen davon investierte ich sehr viel Zeit in diese Technische Dokumentation. Ich denke auch hier, dass mir gewisse Erfahrungen und Eindrücke bleiben werden. Ausserdem habe ich mit dieser Arbeit eine nahezu perfekte Implementation von OAuth2 im Bezug auf den Anmeldeprozess erstellt.

Ich bin mehr als nur zufrieden mit meinen erreichten Ergebnissen.

Teil III.

Anhang

ANHANG A

GLOSSAR

Ausdruck	Erklärung
Angular	TypeScript Frontend Framework
API	Applikation Programming Interface - Eine Softwareschnittstelle zwischen getrennten Schichten
Asset	Endress+Hauser interne Bezeichnung für ein Messgerät
Backend	Softwareschicht, welche sich um die Business Logik und serialisierung der Daten kümmert
Branches	Entwicklungslien
Build-Pipelines	Coninuous Delivery Anwendung
CI/CD	Continous Integration/Coninous Delivery
Edge Device	Gerät, welches Daten von den Messgeräten abliest und in Netilion schreibt
Express.js	JavaScript Backend Bibliothek
Frontend	Softwareschicht, welche sich um die Darstellung von Daten und Prozessen kümmert
GDPR	Datenschutzgesetz der EU
Git	Versionskontrollsoftware
HERE Developer API	Drittanbieter Schnittstelle
Heroku	Hostinganbieter des Backends
IIoT	Industrial Internet of Things
IIoT-Ökosystem	Ganzes IIoT angebot von Netilion
IPA	Individuelle Praktische Arbeit
JSX	React Syntax - Vermischung von HTML und JavaScript wird ermöglicht

Ausdruck	Erklärung
Kanban	Projektmanagementmethode
Lucidchart.com	Website mit der Diagramme erstellt werden können
Mesh	Ein Teil eines 3D Modells
Microsoft Projekt	Tool, welches für die Erstellung des Projektplanes eingesetzt wurde
Mockups	Designentwurf einer Website
MVT	Auftraggeber dieses Projektes
NE107 Status	Standartisierter Status, welcher den Zustand eines Messgerätes beschreibt
Netilion	IIoT PaaS Angebot von Endress+Hauser
Netilion-Ökosystem	Ganzes IIoT angebot von Netilion
Node.js	JavaScript Runtime Environment
OAuth2	Standard, für Anmeldung und abschacheln von Nutzerdaten in Drittanbieter Applikationen
opinionated	Sehr viel ist vom Arbeitsablauf ist vorgegeben
ORM	Object-relational mapping
OSE Modell	One Story Exhibit Modell - Austellungsmodell von Endress+Hauser
Personas	Anspruchsgruppen an das Projekt
PostgreSQL	SQL Datenbank
Product Owner	Für ein Projekt zuständige Person
React	Eine von FaceBook entwickelte JavaScript Bibliothek, welche die Strukturierung von Komponenten basierten Benutzeroberflächen erleichtert.

Ausdruck

Erklärung

Redis	Key/Value Pair caching Server
REST API	Webbasierte Programschnittstelle
Scrum	Projektmanagementmethode
SEO-Ranking	Score den eine Website von einer Suchmaschine erhält
StaceyMatrix	Matrix, welche das Wählen einer Projektmanagementmethode vereinfachen soll
UI/UX	User Interface/User Experience
unopinionated	Sehr wenig ist vom Arbeitsablauf vorgegeben
Use-Case	Anwendungsfall
Vercel	Hostinganbieter des Frontends
Wasserfallmethode	Eine Projektmanagementmethode
Webhooks	Simples pub/sub System für Websites

ANHANG B

ABBILDUNGSVERZEICHNIS

1.1	Diagram der Projektaufbauorganisation	5
1.2	Projektplan, erstellt mit Microsoft Projekts	7
2.1	Stacey Matrix Grafik von Jurgen Appello.	18
2.2	Ein von mir mit Lucidchart erstelltes Wasserfallmodell	19
3.1	Eine von mir mit Lucidchart erstellte grobe Systemarchitektur	22
3.2	Diagramm Netilion Ökosystem von Jonas Schultheiss	23
3.3	Diagramm OSE-Dashboard Backend von Jonas Schultheiss	25
3.4	Diagramm OSE-Dashboard Frontend von Jonas Schultheiss	27
3.5	Grafik, welche den Git Flow Arbeitsablauf visualisiert	31
3.6	Personas	33
3.7	OAuth2 visualisiert durch ein Sequenzdiagramm	36
4.1	Sequenzdiagram, welches das automatische fetching Erklärt	38
4.2	Sequenzdiagram, welches die Zugriffskontrolle für den OSE Verantwortlichen beschreibt	40
4.3	Sequenzdiagram, welches die Zugriffskontrolle für den OSE-Dashboard Nutzer beschreibt	41
4.4	Mockup der Index Page	49
4.5	Mockup einer Fehlermeldung	50

4.6 Mockup einer erfolgreichen Anmeldung	51
4.7 Mockup der Registration	52
4.8 Mockup der Asset verlinkung	53
4.9 Mockup einer fehlgeschlagenen Verlinkung	54
4.10 Mockup einer erfolgreichen Verlinkung	55
4.11 Mockup des Modelleinstellungsmenü	56
4.12 Mockup des Standorteinstellungsmenü	57
4.13 Mockup des Verlinkungseinstellungsmenü	58
4.14 ERM welches die Datenbankerweiterung visualisiert	69
4.15 Screenshot von JWT.io	71
5.1 authContext.js von ose-dashboard-frntnd	73
5.2 settingsLayout.js von ose-dashboard-frntnd	74
5.3 auth.service.ts von ose-dashboard-bcknd	75
5.4 models.service.ts von ose-dashboard-bcknd	76
5.5 models.service.ts von ose-dashboard-bcknd	79
5.6 Output von yarn lint im Frontend	80
5.7 Output von yarn lint im Backend	80
5.8 Ausschnitt aus der momentanen «Model.js»Komponente	83
7.1 Abbildung der useEffect Hook in location.js	87
7.2 Abbildung der useEffect Hook in model.js	87
7.3 Abbildung der getAssets Funktion	88

ANHANG C

QUELLENVERZEICHNIS

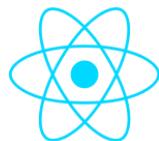
- [1] Apple. *Mit Time Machine ein Backup eines Mac erstellen*. Apple Support. Jan. 2021. URL: <https://support.apple.com/de-ch/HT201250> (besucht am 30.03.2021).
- [2] Atlassian. *User Storys / Beispiele und Vorlage / Atlassian*. Atlassian. 2021. URL: <https://www.atlassian.com/de/agile/project-management/user-stories> (besucht am 08.04.2021).
- [3] auth0.com. *JSON Web Tokens - Jwt.io*. URL: <https://jwt.io/>.
- [4] *Documentation | NestJS - A progressive Node.js framework*. Documentation | NestJS - A progressive Node.js framework. 2021. URL: <https://docs.nestjs.com/techniques/task-scheduling> (besucht am 31.03.2021).
- [5] *Documentation | NestJS - A progressive Node.js framework*. Documentation | NestJS - A progressive Node.js framework. 2021. URL: <https://docs.nestjs.com/guards> (besucht am 31.03.2021).
- [6] flaticon. *User Storys / Beispiele und Vorlage / Atlassian*. flaticon. 2021. URL: https://www.flaticon.com/free-icon/map_854878 (besucht am 08.04.2021).
- [7] *Hardware security overview*. Apple Support. 2021. URL: <https://support.apple.com/en-gb/guide/security/secf020d1074/web> (besucht am 30.03.2021).
- [8] *Keep your Time Machine backup disk for Mac secure*. Apple Support. 2021. URL: <https://support.apple.com/en-gb/guide/mac-help/mh21241/mac> (besucht am 30.03.2021).

ANHANG D

CODING GUIDELINES

Endress+Hauser Digital Solutions Coding Guidelines

Coding Conventions for JavaScript and TypeScript in the context of React, Next and Nest.



1 Preamble

1.1 Exceptions from the E+H Coding Conventions

It is permitted to add project specific rules to the E+H Coding Conventions as long as compliance with the mandatory definitions listed in this document is ensured.

1.2 Eslint

Eslint is our linter of choice because it is practically the standard in the world of JavaScript. It can also be used as a formatter, but we'll primarily use it as a linter.

1.3 Prettier

Prettier is the formatter that should be used. It has also become a standard in the JavaScript world.

1.4 Husky

Husky allows commands to be executed automatically with hooks. We use Husky to make sure that the code is lined and formatted before committing with the pre-commit hook.

1.5 Editor settings

```
1 {
2   "useTabs": false,
3   "tabWidth": 2,
4   "printWidth": 100,
5   "endOfLine": "auto",
6   "arrowParens": "avoid",
7   "semi": true,
8   "singleQuote": true,
9   "bracketSpacing": true,
10  "trailingComma": "none"
11 }
```

2 Generic Coding Conventions

2.1 Blank line after statement block

A statement such as if-cases, switch cases, loops or anything else that defines a new scope within should end with a blank line to aid readability.

```
1 foo.map(bar => {
2   // does something
3 });
4
5 console.log(foo);
6 if (foo) {
7   // does something
8 }
9
10 console.log(foo);
```

2.2 Naming Booleans

A Booleans name should always ask something to which you can get a yes or no answer.

```
1 const isTokenStillValid = token.validUntil > new Date.now();
2
3 // easily readable
4 if (isTokenStillValid) {
5   // easy to understand that in this case the token is still valid
6 } else {
7   // and isn't in this case
8 }
```

2.3 Function length

The length of a function should not exceed 15 lines. Exceptions are functional React components and functions in repositories of ORMs, as in these cases it makes no sense to outsource the logic. If a case arises in which it also makes no sense, this must be documented.

2.4 React components

Class-based components are no longer part of the standard since React 16.8. Since then, functional components have been used. However, these differ from the normal functions in two aspects. The name should be capitalized, and a regular function should be used instead of the ES6 arrow functions.

```
1 // do
2 export default function Header(properties) {
3   return <p>This is a header!</p>;
4 }
5
6 // don't
7 export default header = (properties) => {
8   return <p>This is a header!</p>;
9 }
```

2.5 Avoid single letter names. Be descriptive with your naming

```
1 // bad
2 function q() {
3   // ...
4 }
5
6 // good
7 function query() {
8   // ...
9 }
```

2.6 Use camelCase when naming objects, functions, and instances

```
1 // bad
2 const OBJEcttsssss = {};
3 const this_is_my_object = {};
4 function c() {}
5
6 // good
7 const thisIsMyObject = {};
8 function thisIsMyFunction() {}
```

2.7 Do not use trailing or leading underscores

Why? JavaScript does not have the concept of privacy in terms of properties or methods. Although a leading underscore is a common convention to mean “private”, in fact, these properties are fully public, and as such, are part of your public API contract. This convention might lead developers to wrongly think that a change won’t count as breaking, or that tests aren’t needed. tl;dr: if you want something to be “private”, it must not be observably present.

```
1 // bad
2 this.__firstName__ = 'Panda';
3 this.firstName_ = 'Panda';
4 this._firstName = 'Panda';
5
6 // good
7 this.firstName = 'Panda';
8
9 // good, in environments where WeakMaps are available
10 // see https://kangax.github.io/compat-table/es6/#test-WeakMap
11 const firstNames = new WeakMap();
12 firstNames.set(this, 'Panda');
```

2.8 Don’t save references to this. Use arrow functions or binds.

```
1 // bad
2 function foo() {
3   const self = this;
4   return function () {
5     console.log(self);
6   };
7 }
8
9 // bad
10 function foo() {
11   const that = this;
12   return function () {
13     console.log(that);
14   };
15 }
16
17 // good
18 function foo() {
19   return () => {
20     console.log(this);
21   };
22 }
```

Or https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Function/bind

2.9 Nestjs specific file naming

```
1 // words get split by dashes (-)
2 // generally what-entity.type.ts
3
4 // module
5 // resource(plural).module.ts
6 assets.module.ts
7
8 // controller
9 // resource(plural).controller.ts
10 assets.controller.ts
11
12 // service
13 // resource(plural).service.ts
14 assets.service.ts
15
16 // repository
17 // resource(plural).repository.ts
18 assets.repository.ts
19
20 // entity
21 // resource(singular).entity.ts
22 asset.entity.ts
23
24 // dto / data transfer object
25 // name-of-dto.dto.ts
26 netilion-response.dto.ts
27
28 // interface
29 // name.interface.ts
30 tag.interface.ts
```

2.10 Nextjs specific file naming

```
1 // generally
2 modal.js
3 modalTitle.js
4
5 // some pages (naming required by nextjs)
6 _app.js
7 _document.js
8 /api/assets/[id].js
```