

Individuelle praktische Arbeit

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

7. April 2021



Version: 0.2.0

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: 0.2.0
Datum: 7. 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

KURZFASSUNG

Ausgangssituation

Pellentesque cursus luctus mauris. Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam.

Umsetzung

Pellentesque cursus luctus mauris. Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam.

Ergebnis

Pellentesque cursus luctus mauris. Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam.

Inhaltsverzeichnis

I Obligatorische Kapitel	7
1 Obligatorische Dokumentation	8
1.1 Ausgangslage	8
1.2 Detaillierte Aufgabenstellung	9
1.3 Mittel und Methoden	10
1.4 Vorkenntnisse	10
1.5 Vorarbeiten	10
1.6 Neue Lerninhalte	11
1.7 Arbeiten in den letzten sechs Monaten	11
1.8 Projektaufbauorganisation	11
1.9 Durchführungsblock	12
1.10 IPA-Termine	12
1.11 Projektplan	13
1.12 Arbeitsjournale	15
1.12.1 Arbeitsjournal vom 29.03.2021	15
1.12.2 Arbeitsjournal vom 30.03.2021	16
1.12.3 Arbeitsjournal vom 31.03.2021	17
1.12.4 Arbeitsjournal vom 01.04.2021	18
1.12.5 Arbeitsjournal vom 06.04.2021	19
II Dokumentation	20
2 Projektmanagement	21
2.1 Entscheidung	21
2.2 Wasserfallmodell	22
2.2.1 Initialisierung des Dokuments	23
2.2.2 Analyse	23
2.2.3 Entwurf	23
2.2.4 Implementierung	24
2.2.5 Testen	24
2.2.6 Abschluss	24

3 Analyse	25
3.1 Systembeschreibung	25
3.1.1 Systemarchitektur	25
3.1.2 One Story Exhibit	25
3.1.3 Netilion	26
3.1.4 OSE-Dashboard: Backend	27
3.1.5 OSE-Dashboard: Frontend	29
3.2 Ist/Soll-Vergleich	29
3.2.1 Istzustand	30
3.2.2 Sollzustand	30
3.3 Namenskonzept	31
3.3.1 User-Stories	31
3.3.2 Akzeptanzkriterien	31
3.4 Versionskontrolle	32
3.4.1 Git Flow	32
3.4.2 Pipelines zu Vercel und Heroku	33
3.4.3 Dokumentation	33
3.5 Backupkonzept	34
3.5.1 Sicherheit der Daten	34
3.6 Personas	34
3.6.1 Endnutzer	34
3.6.2 MVT	35
3.6.3 OSE Verantwortlicher	35
3.6.4 Entwickler	35
3.7 User Stories	35
3.7.1 US-01	35
3.7.2 US-02	35
3.7.3 US-03	35
3.7.4 US-04	36
3.7.5 US-05	36
3.7.6 US-06	36
3.8 OAuth2 Strategie	36
3.8.1 Was ist OAuth2?	36
3.8.2 Anwendung im OSE-Dashboard	38
4 Entwurf	40
4.1 Systementwurf	40
4.1.1 Generelles Data Fetching	40
4.1.2 Zugriffskontrolle	42
4.2 Mockups	43
4.2.1 Index Page	44

4.2.2 Registration	45
4.2.3 Einstellungsmenü	51
4.3 Testkonzept	54
4.3.1 Fehlerklassen	54
4.3.2 Test Case Schema	55
4.3.3 User-Stories Abdeckung	55
5 Implementierung	56
5.1 OAuth2	56
5.1.1 Frontend	56
5.1.2 Backend	56
5.2 Account löschen	56
5.3 Modelauswahl	56
5.3.1 Auflistung mit Landesflaggen	57
5.3.2 Interaktive Weltkarte	57
III Anhang	59
A Abkürzungsverzeichnis	60
B Glossar	61
C Abbildungsverzeichnis	62
D Quellenverzeichnis	64

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

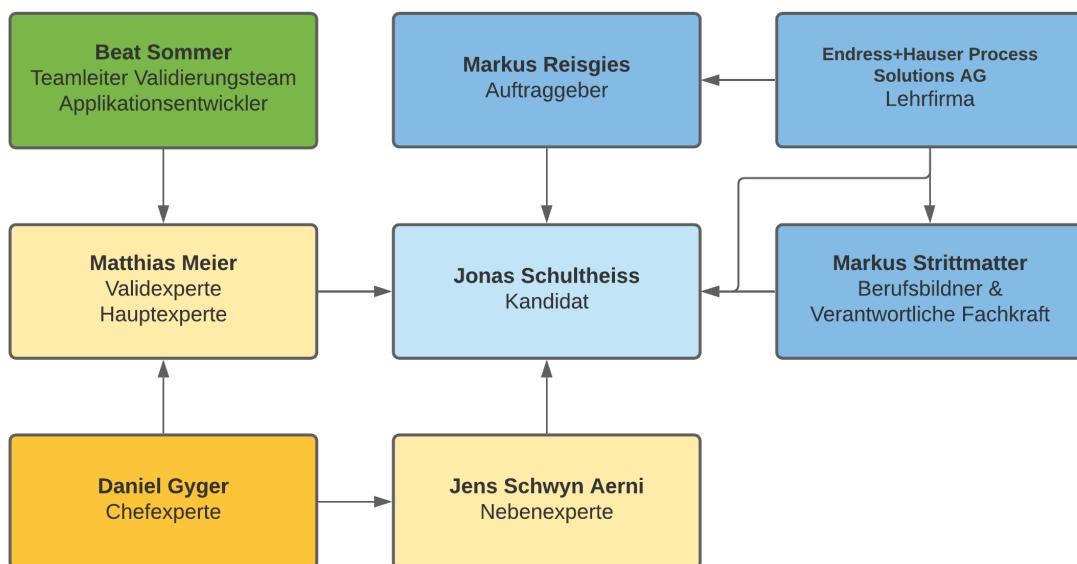


Abbildung 1.1.: Diagramm der Projektaufbauorganisation

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

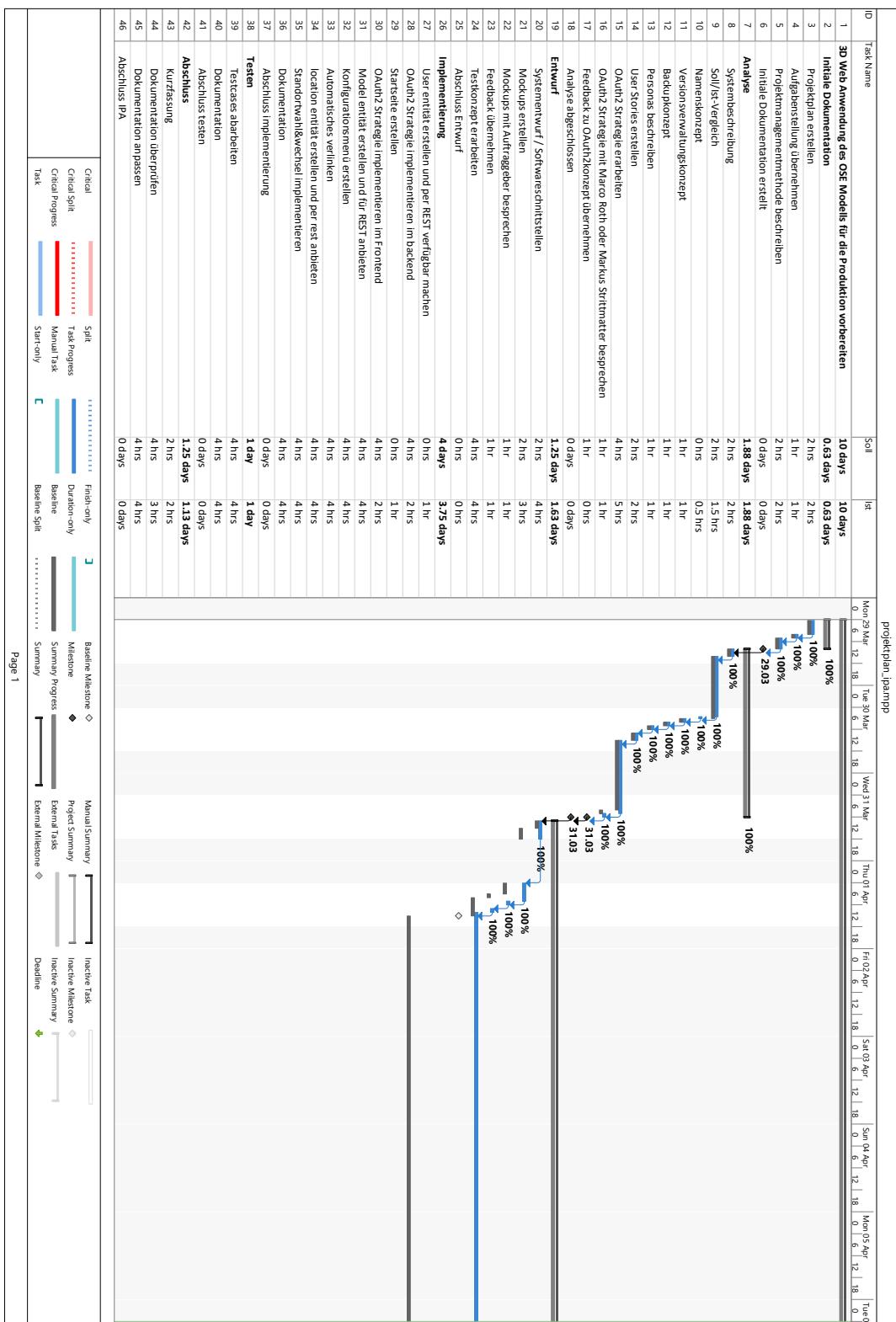
Antrittsgespräch: 23.03.2021 08:30

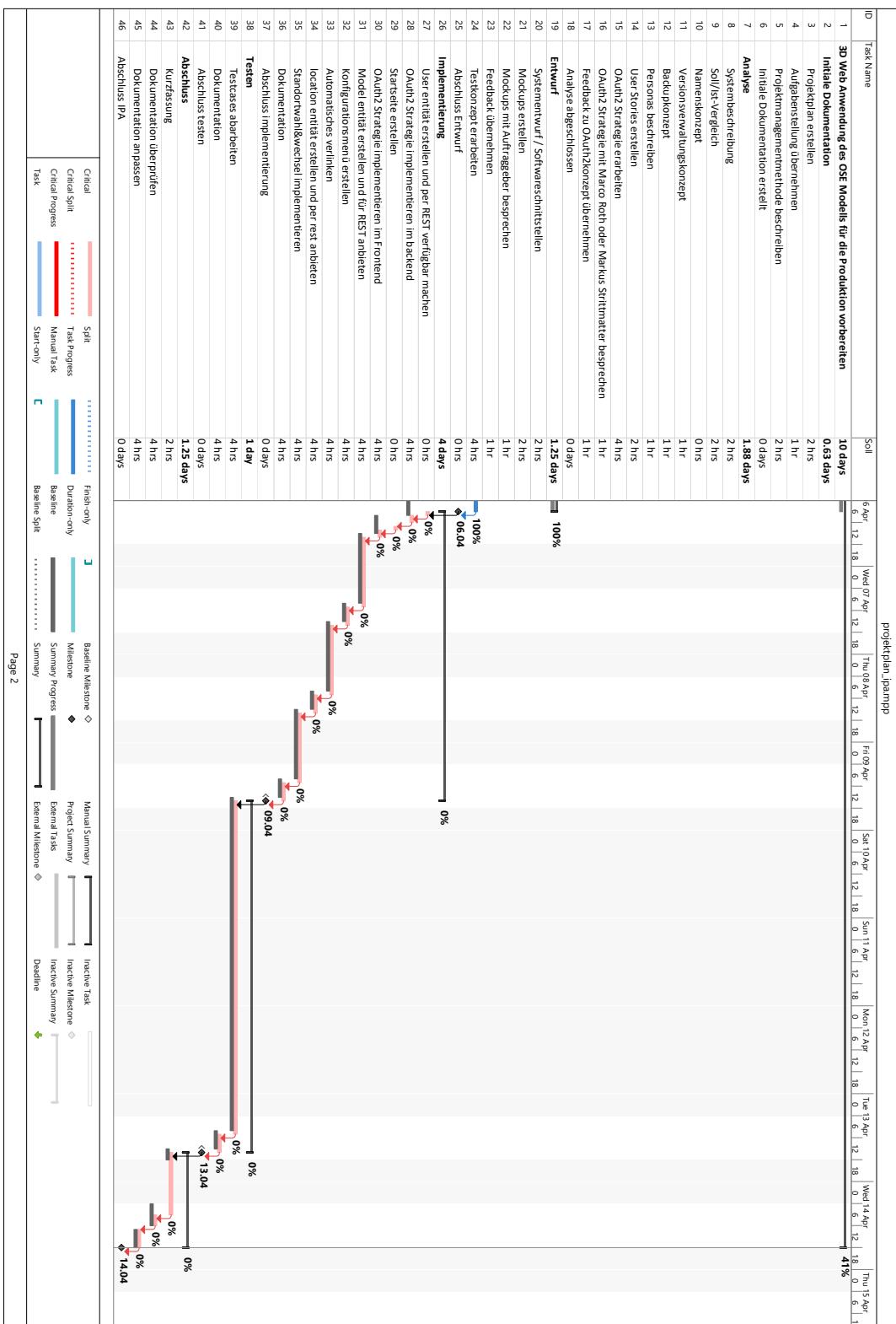
Erster Zwischenbesuch: 01.04.2021 09:00

Zweiter Zwischenbesuch: 01.04.2021 00:00

Endbesuch: 28.04.2021 09:00

1.11. Projektplan





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 gut und schnell 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 entpuppt sich später nicht als 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. Ich bin mir nicht ganz sicher ob ich bei jedem Abschnitt so in die Tiefe musste, wie ich es ging. Da es allerdings zeitlich passt, gebe ich mich zufrieden

Ziel 5: Soll/Ist-Vergleich

Diesen Task gilt es heute anzufangen, sodass es mir morgen sicherlich in einer Stunde fertig reicht. 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 entgegnen 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 nachführen muss.

J. Schultheiss

M. Strittmatter

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

Danach habe ich mich ans Namenskonzept gemacht. 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 kommen wird.

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 UserStories 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, dass ganze 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, nachdem ich gestern das Namenskonzept vergessen hatte im Projektplan einzutragen. Nun hoffe ich einfach, dass das fertigstellen des OAuth2konzeptes morgen nicht länger braucht als Anfangsgedacht.

Reinach, 30.03.2021

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 ein Fehler auf, welchen ich in der Grafik korrigierte. Ich musste jedes Mal die Nummerierung aller nachfolgenden Schritte manuell ändern, was mir Zeit gekostet hat. Ich habe mir allerdings das Thema OAuth2 nochmals nahegebracht, 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 erledigt 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 zwar auf, allerdings ist das nun etwas Pokern. 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 rechte 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 besprochen werden.

Ziel 3

Auch die Erstellung des Testkonzeptes war sehr Zeit konsumierend. Ich habe dabei zuerst definiert, wie der Test benannt und strukturiert sind und habe dann Anhand den User-Stories die Test-Cases entworfen. So ging ich mir sicher, dass alle Anforderungen der Anspruchsgruppen durch Tests gedeckt wurden. Ich werde diesen Task am Dienstag noch kurz fertigstellen, da es mir heute nicht ganz fertig gereicht hat.

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 ausserdem noch 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 Arbeitsjournalen und dem aktuellen Projektplan. Wir haben noch etwas gequatscht und das war es dann auch schon. Ich denke sie haben einen positiven Eindruck vom aktuellen Stand erhalten.

Reflexion

Heute war ein Tag voller Fleissarbeiten. Obwohl ich eine Stunde hinterher bin, bin ich mit meiner Leistung zufrieden du 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.

J. Schultheiss

Reinach, 01.04.2021

Ort, Datum

Kandidat: Jonas Schultheiss

Fachvorgesetzter: Markus Strittmatter

1.12.5. Arbeitsjournal vom 06.04.2021

Ziel	Beschreibung	Geplante Zeit	Benötigte Zeit	Abweichung
1	Testkonzept fertiggestellt	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 EH 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 echt nur dokumentiert habe.

Reinach, 06.04.2021



Ort, Datum

Kandidat: Jonas Schultheiss

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, wie klar der Auftrag ist. Das heißt, ob alle Anforderungen gegeben sind oder noch welche zu einem späteren Zeitpunkt dazu kommen. Die y-Achse beschreibt dagegen, wie deutlich der Weg zu diesen Kriterien ist. Dabei fragt sich zum Beispiel, ob es klar ist, wie man den Auftrag mit dieser Programmiersprache/diesem Framework 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 nachdem, wo man landet, sollte man eine andere Projektmanagementmethode anwenden.

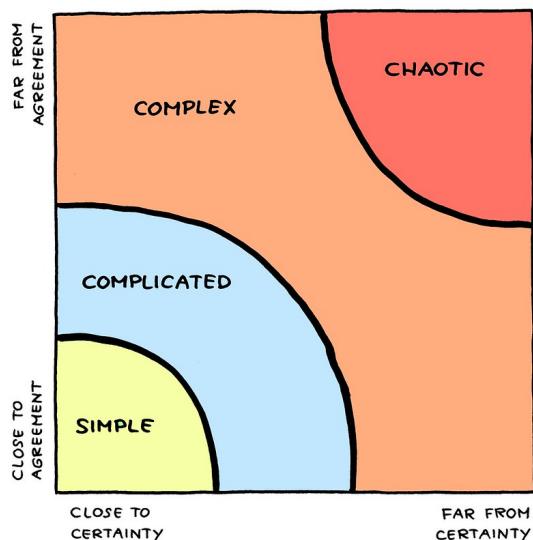


Abbildung 2.1.: Stacey Matrix

- **Simpel** - Das Ziel und der Weg dahin sind klar. In diesem Fall sollte das Wasserfallmodell oder eine ähnliche Methode gewählt werden.
- **Kompliziert** - Sollte entweder das Ziel oder der Lösungsweg etwas unklar 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 ist für wechselnde Anforderungen gemacht.
- **Chaotisch** - Das Ziel und der Weg dahin sind unklar. In diesem Fall sollte das Projekt noch nicht gestartet werden. Stattdessen sollte für mehr Klarheit an beiden Achsen gesorgt werden.

2.2. Wasserfallmodell

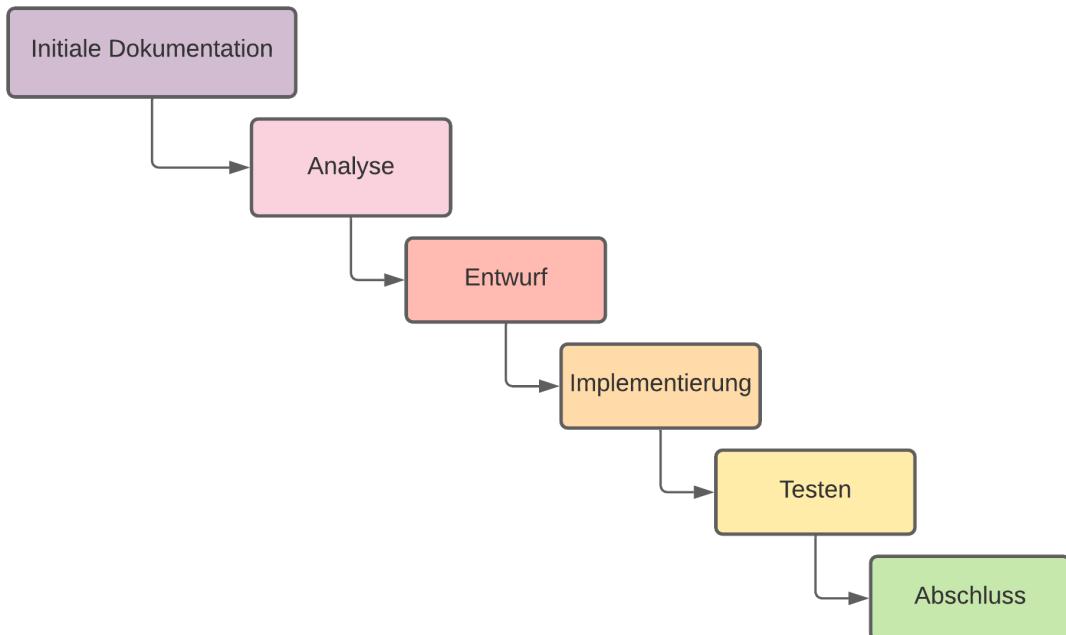


Abbildung 2.2.: Wasserfall Modell

Das Wasserfallmodell arbeitet mit auf sich selber aufbauenden sequenziellen Phasen. Man beginnt erst mit dem nächsten Stadium, sobald die Letzte komplett abgeschlossen ist. Dies erfordert, dass das Ziel und der Weg dorthin klar sein muss. Ein streng geregelter Arbeitsablauf mit Meilensteinen kann dadurch ermöglicht werden, was vor allem für kleine Projekte wie dieses sehr nützlich ist.

2.2.1. Initialisierung des Dokuments

Als Erstes soll das Dokument für die kommenden Arbeitstage so vorbereitet werden, dass mir nachher nichts mehr im Weg steht. Zuerst habe ich die LaTeX-Vorlage nochmals angesehen, 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. Dies habe ich soweit verfasst. Nun fehlt nur noch dieses Kapitel und dieser Arbeitsschritt ist abgeschlossen.

2.2.2. Analyse

Nachdem die Initialisierung komplett abgeschlossen ist, geht es dann an die Analyse. Diese beinhaltet unter anderem eine Systembeschreibung, ein Ist/Soll-Vergleich, Konzepte wie ich die Sicherung der Arbeitsergebnisse sicherstelle und die OAuth2-Strategie. Ersteres beschreibt, wie genau unser Netilion-Ökosystem funktioniert und wie dieses Projekt darin integriert werden soll. Der Ist/Soll-Vergleich zeigt nochmals die zu erarbeitende Lösung auf und wie sie sich mit 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 in kleinere Stücke aufgebrochen werden, damit sich der Entwickler auf einzelne Entwicklungsabschnitte konzentrieren kann und der Fortschritt messbarer wird.

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. Ohne dabei zu sehr in technische Details zu gehen, beschreibt sie, was genau von der Software gefordert ist. Dies ermöglicht eine einfachere Absprache mit den Anspruchsgruppen. Wie dabei diese Anforderungen implementiert werden, spielt keine Rolle. Sehr oft wird diese kurze Beschreibung als Titel für eine Story im Projektmanagementtool genommen.

Beispiel einer User Story:

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

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. Teil davon ist auch, welche Mittel er einsetzt, wieso er diese verwendet und wie er sie anwenden will. Dazu gehören Schnittstellen, Bibliotheken, Datenbank und so weiter. Das Ergebnis dieser Phase ist ein ausgearbeitetes UI/UX 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 von den vorherigen Kapiteln beachtet werden. Nachdem eine Story abgeschlossen ist, wird die Änderung auf einen Branch gepusht und automatisch deployed. So können die Anspruchsgruppen zeitnahe Rückmeldungen geben. Das Produkt der 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, soll die restliche Zeit für die Überarbeitung/Verbesserung der Dokumentation eingesetzt werden. Anschließend soll die IPA am 14.04.2021 abgegeben werden.

KAPITEL 3

ANALYSE

3.1. Systembeschreibung

3.1.1. Systemarchitektur

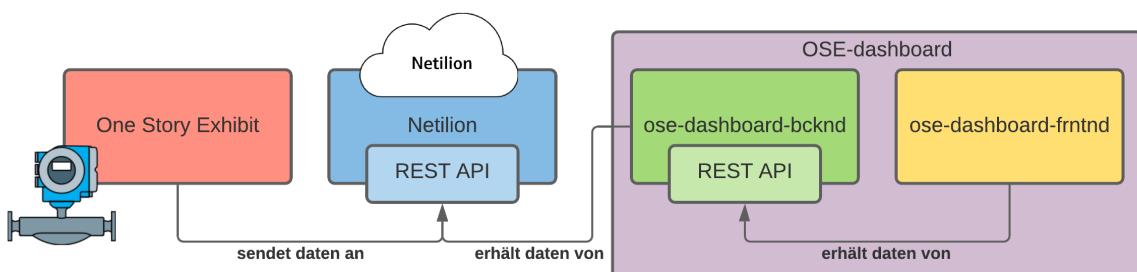


Abbildung 3.1.: Grobe Systemarchitektur

Das System des OSE-Dashboards besteht aktuell aus vier essenziellen Teilen. Die Systemteile werden in den nachfolgenden Kapiteln 3.1.2, 3.1.3, 3.1.4 und 3.1.5 erläutert.

3.1.2. One Story Exhibit

Die Anlage/-n deren Daten von der Applikation ausgewertet werden sollen. In diesem Projekt sind dies Ausstellungsmodelle von Endress+Hauser selbst. Die Gruppe verwendet diese an Messen oder in Verkaufsstellen, um den Kunden unser umfassendes Portfolio an Messgeräten näher zu bringen. Diese Applikation soll diesem Prozess helfen und zeigen, was mit unserem IIoT-Angebot möglich ist.

3.1.3. Netilion

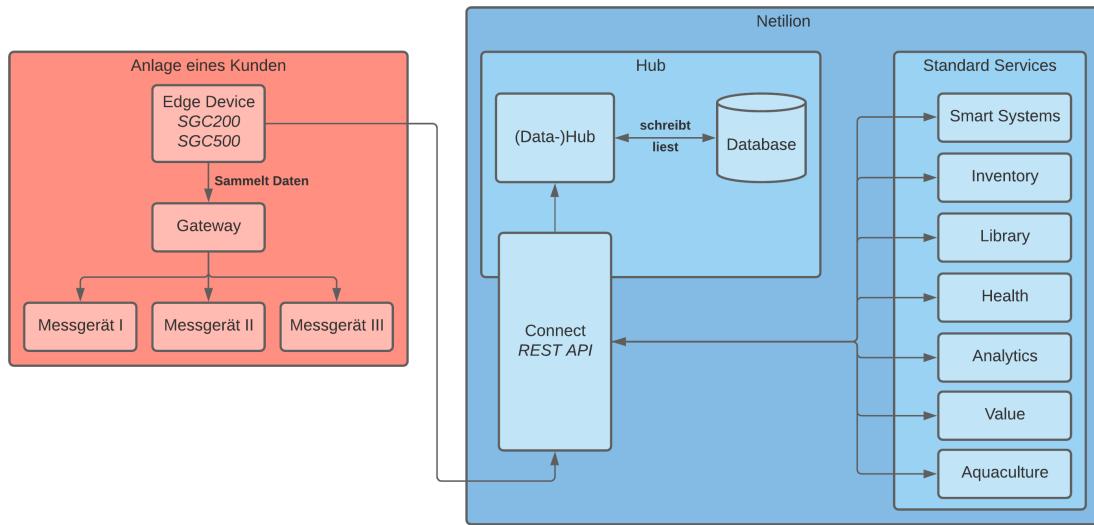


Abbildung 3.2.: Netilion Ökosystem

Netilion ist das IIoT Angebot welches vor einigen Jahren von der Endress+Hauser Digital Solutions ins Leben gerufen wurde. Zuvor hatten Kunden keinen direkten Einblick in die genauen Daten der Messgeräte. Techniker mussten regelmäßig vorbei kommen und die Geräte prüfen, auch wenn es noch optimal lief. Nun ist die ganze Anlage des Kunden ans Internet gebunden, was die Möglichkeiten praktisch grenzenlos macht.

Die Daten der Messgeräte werden mithilfe eines Edge Devices ausgelesen und zuerst lokal gespeichert. In Intervallen sendet es diese dann an die REST-API von Netilion. Da werden sie vom Hub empfangen, validiert und serialisiert. Anschliessend kann der Kunde seine Webapplikationen öffnen 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 das Ganze in einer Datenbank.

Standard Services

Die Standard Services sind Webapplikationen, welche die Daten der Anlage entgegennehmen, passend darstellt und mit anderen Daten integriert. So zeigt zum Beispiel die Applikation *Health* die NE107 Status der einzelnen Geräte an. *Value* hingegen stellt unter anderem die Messwerte der Geräte dar. Diese Services sind das Herz des IIoT-Angebotes. Sie sollen dem Kunden Kosten sparen und Prozesse vereinfachen.

Connect

Netilion Connect ist ein neues Angebot des Ökosystems. Grosse Kunden wollen entweder eigene eingekauft 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. OSE-Dashboard: Backend

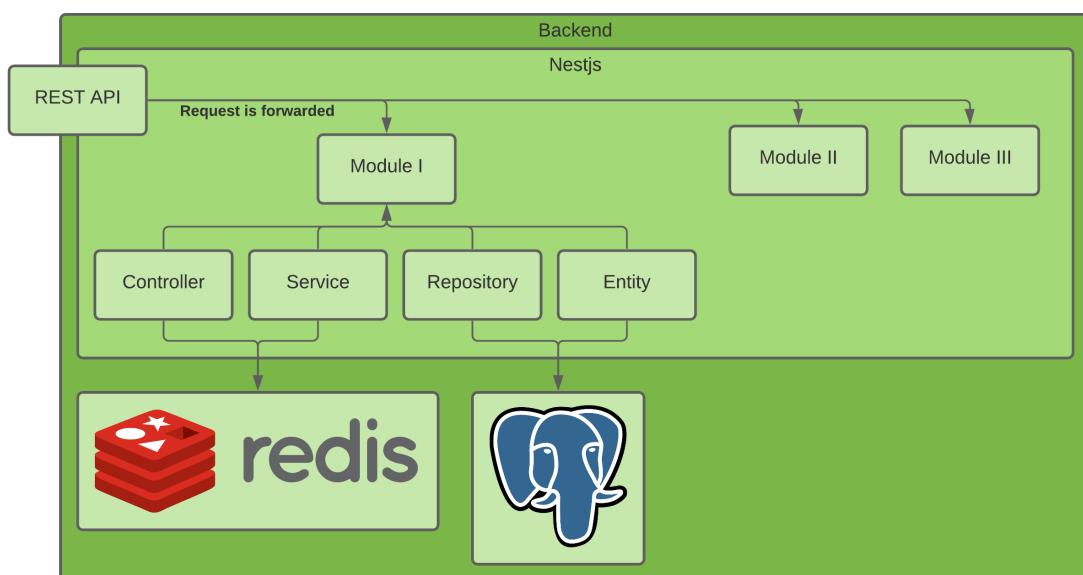


Abbildung 3.3.: OSE-Dashboard Backend

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 "Übersöder Assets". Wie in Angular werden sie zu einem Modul gebündelt.

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.

3.1.5. OSE-Dashboard: Frontend

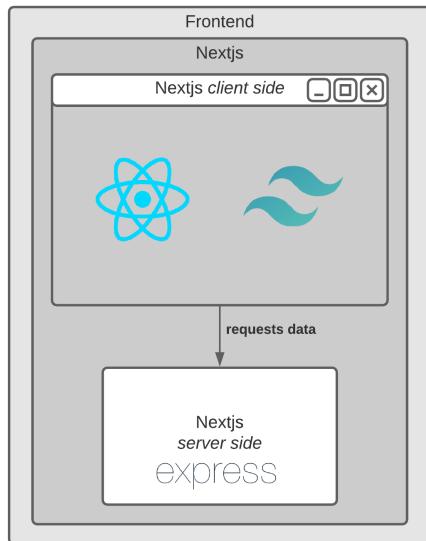


Abbildung 3.4.: OSE-Dashboard Frontend

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. Istzustand

Model

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 Model 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 Model 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. Sollzustand

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 Models 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-Models Einstellungen vornehmen kann, muss ein Konfigurationsmenü her. Ein Teil der IPA ist es, die digitalen Zwillinge automatisch mit den Meshes des 3D-Models 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

3.4.1. Git Flow

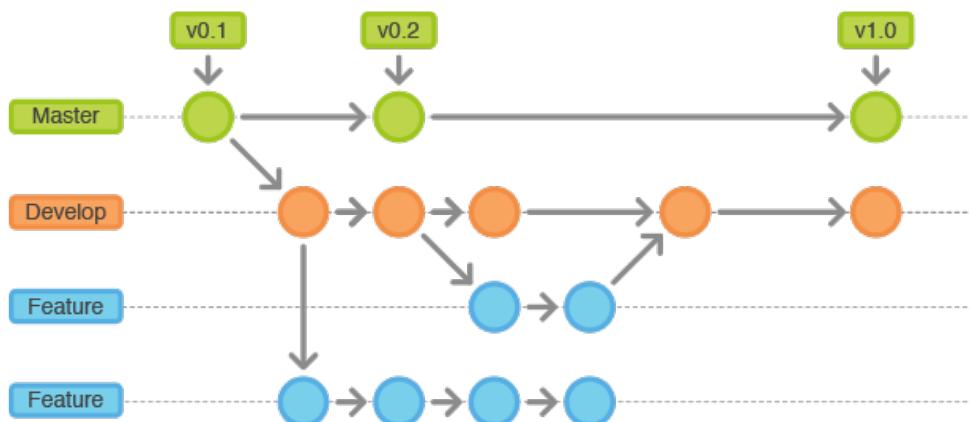


Abbildung 3.5.: Git Flow visualisiert

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 einsetzte**, 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 [4] und die externe SSD [5] sind verschlüsselt.

3.6. Personas

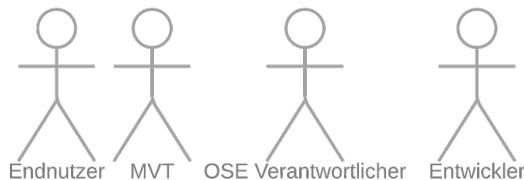


Abbildung 3.6.: Personas

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-Models 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 Model 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 Model 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

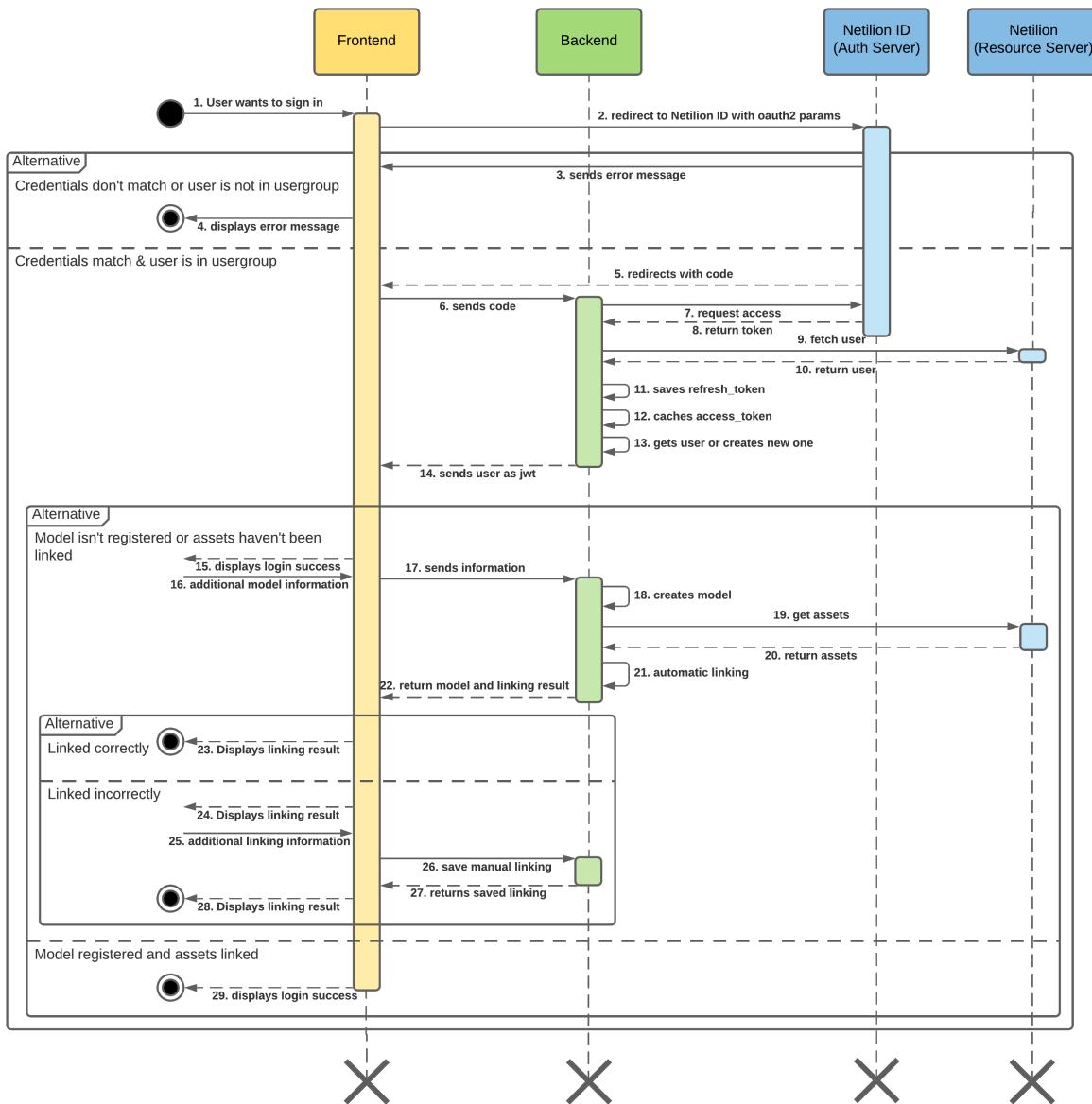


Abbildung 3.7.: OAuth2 visualisiert durch ein Sequenzdiagramm

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 Model 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 Model 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 Model 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 Model bereits registriert und die Messgeräte verlinkt, ist der Prozess beendet.

KAPITEL 4

ENTWURF

4.1. Systementwurf

4.1.1. Generelles Data Fetching

Dieses Abschnitt baut auf das Kapitel 3.8 auf. Nachdem sich ein OSE Verantwortlicher das erste mal 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[2] 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.

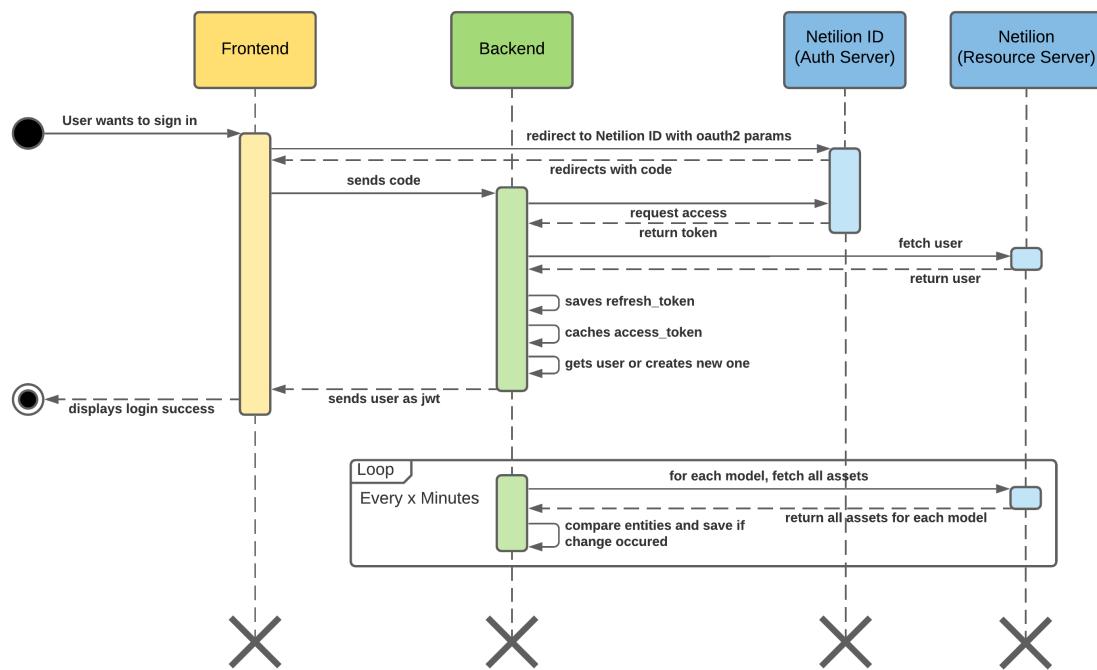


Abbildung 4.1.: Data fetching in Intervallen

4.1.2. Zugriffskontrolle

OSE Verantwortlicher

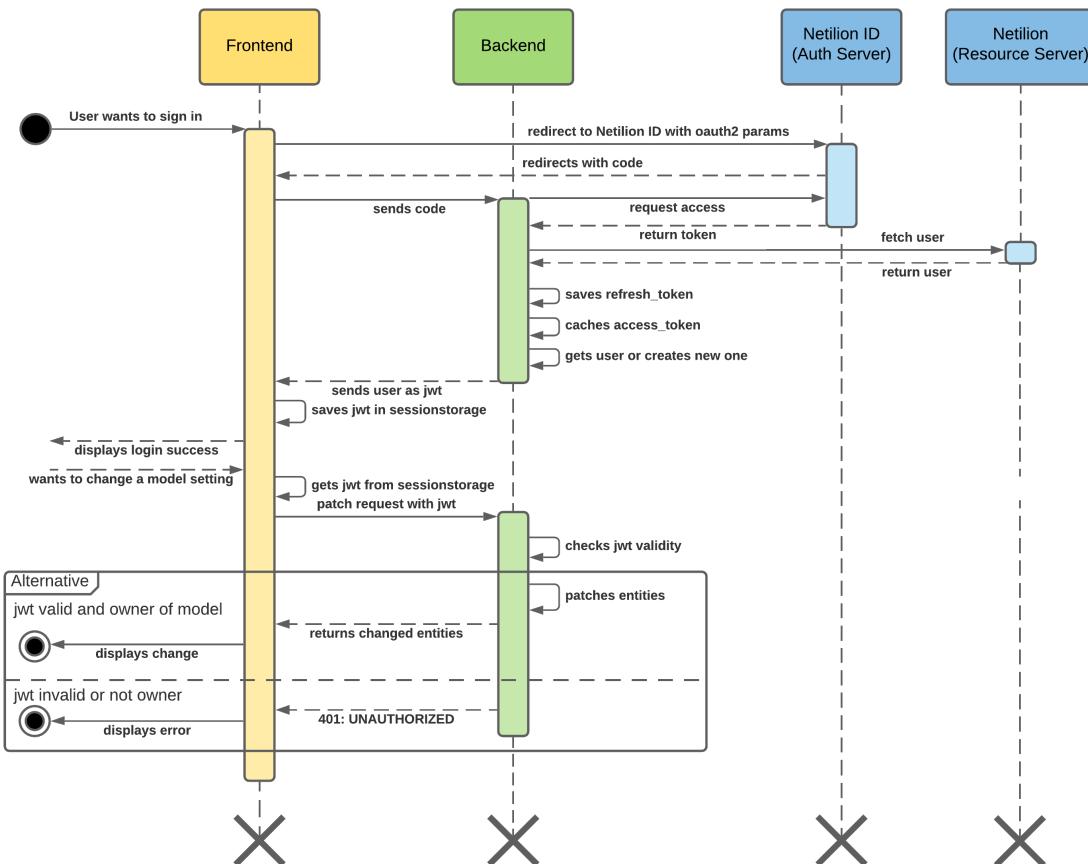


Abbildung 4.2.: Zugriffskontrolle OSE Verantwortlicher

Ein OSE Verantwortlicher soll das Konfigurationsmenü seines Models 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[3] geschützt. Daraufhin wird direkt geprüft werden, ob das Model auch wirklich dem User gehört. Sollte dies nicht der Fall sein, wird die Anfrage als unauthorized zurückgeschickt.

OSE-Dashboard Nutzer

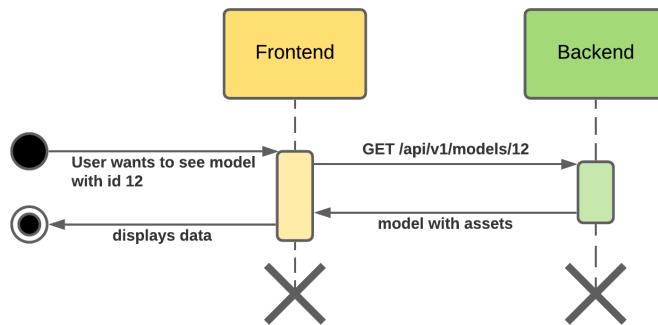


Abbildung 4.3.: Zugriffskontrolle OSE-Dashboard Nutzer

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 4.1.1 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 3.1.4 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.

4.2. 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.

4.2.1. Index Page

Momentan ist die Index Page des OSE-Dashboards das Reinacher Model 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-Model zuständig ist. Sollte dies der Fall sein, könnte er sich einloggen und so die Daten für das Dashboard zur Verfügung stellen.

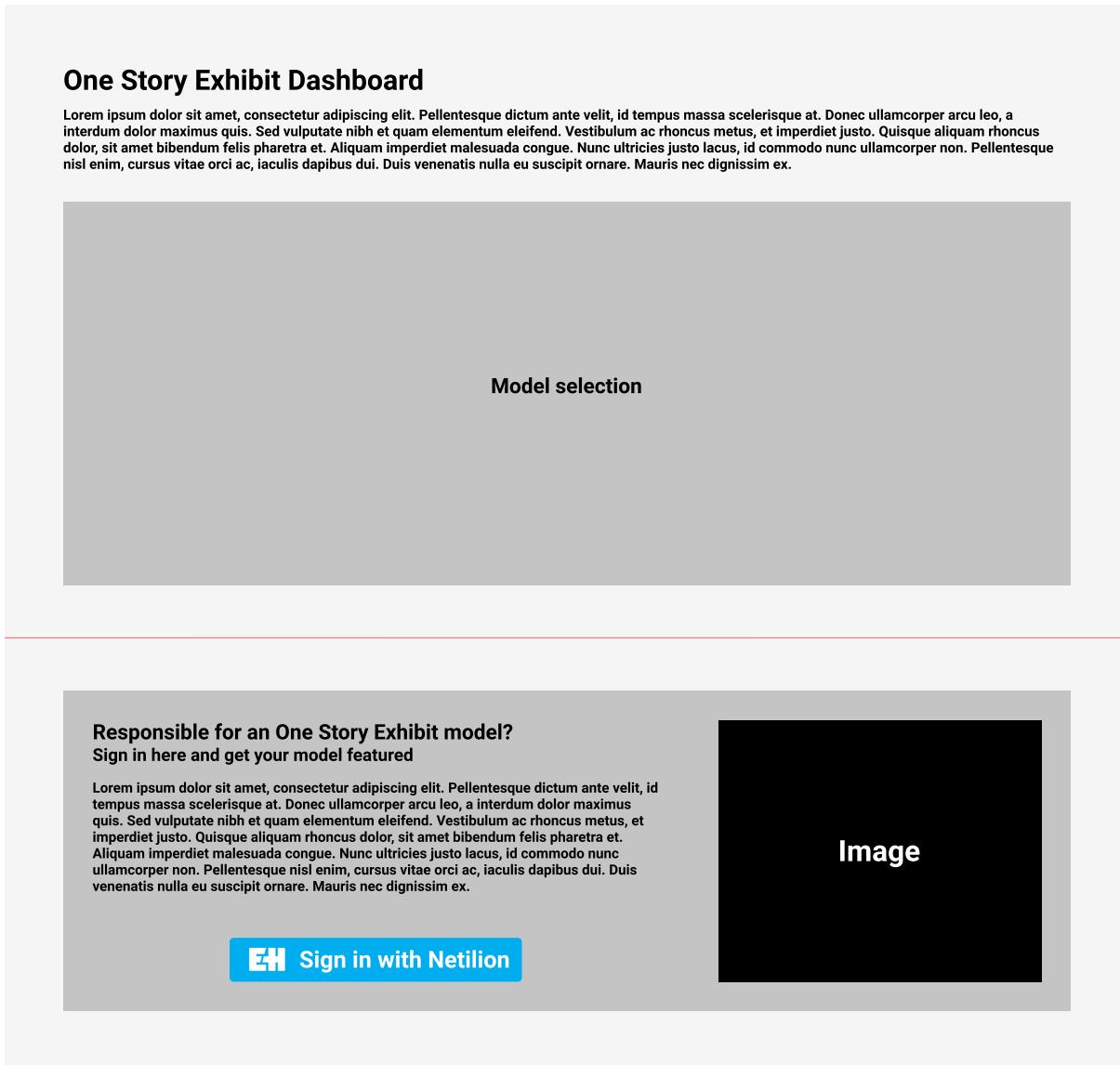


Abbildung 4.4.: Mockup der Index Page

4.2.2. 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.

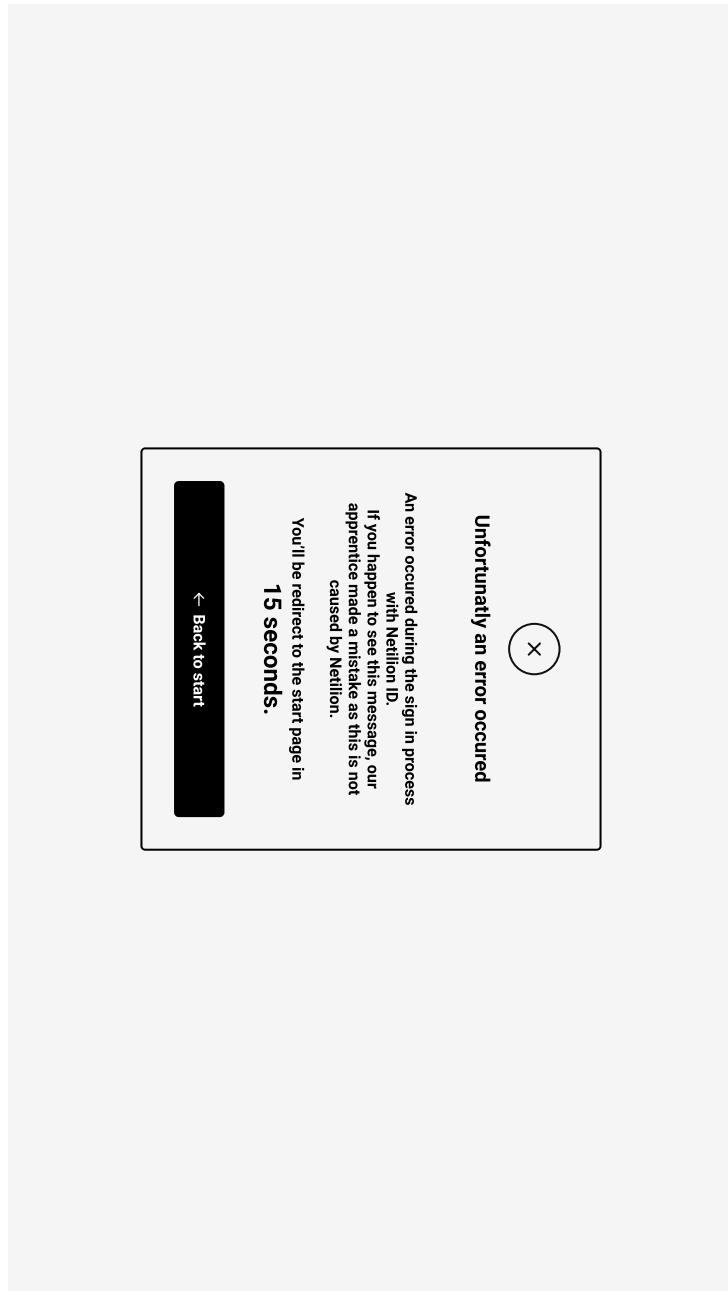


Abbildung 4.5.: Mockup einer Fehlermeldung

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.

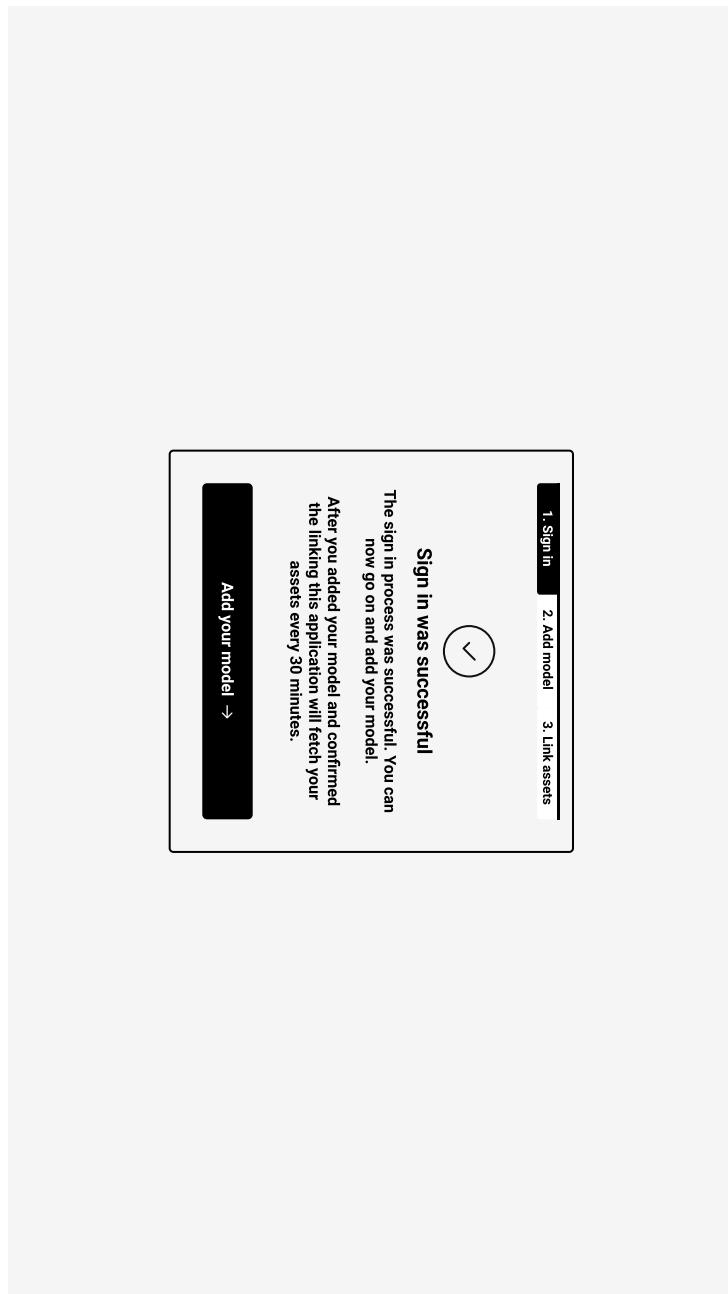


Abbildung 4.6.: Mockup einer erfolgreichen Anmeldung

Model registrieren

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

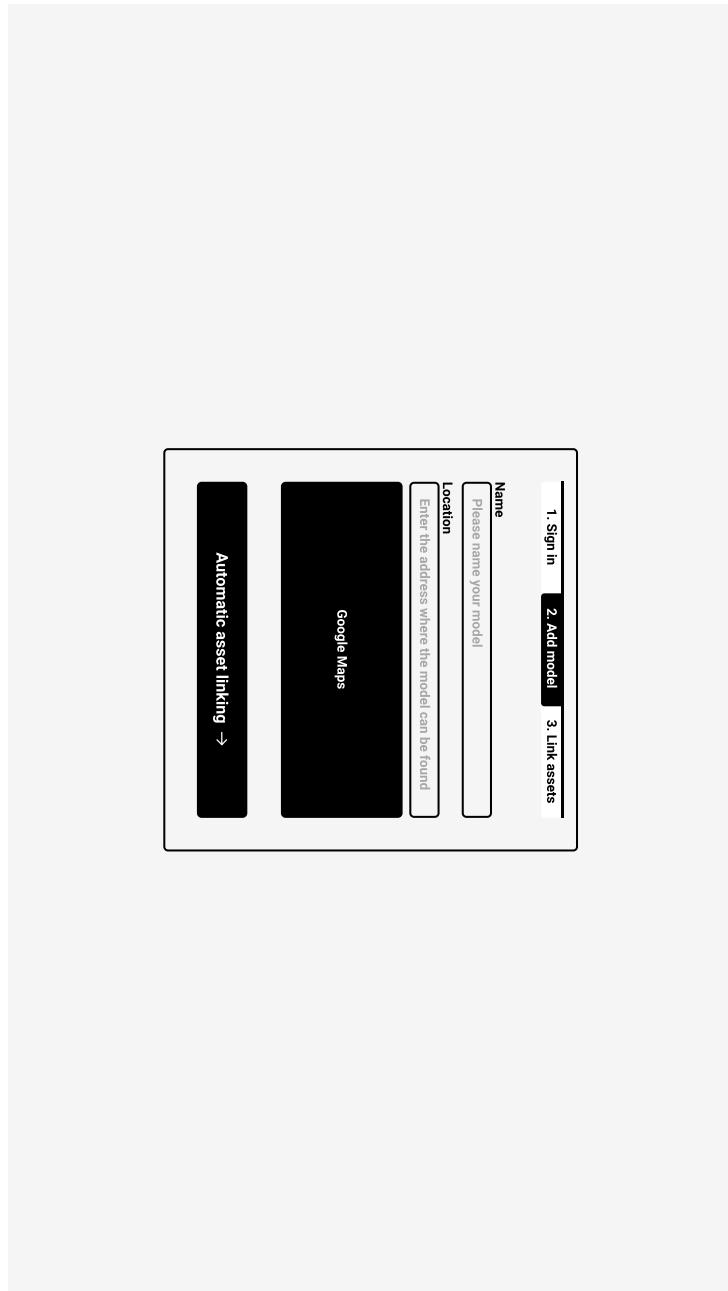


Abbildung 4.7.: Mockup der Registration

Asset verlinkung

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

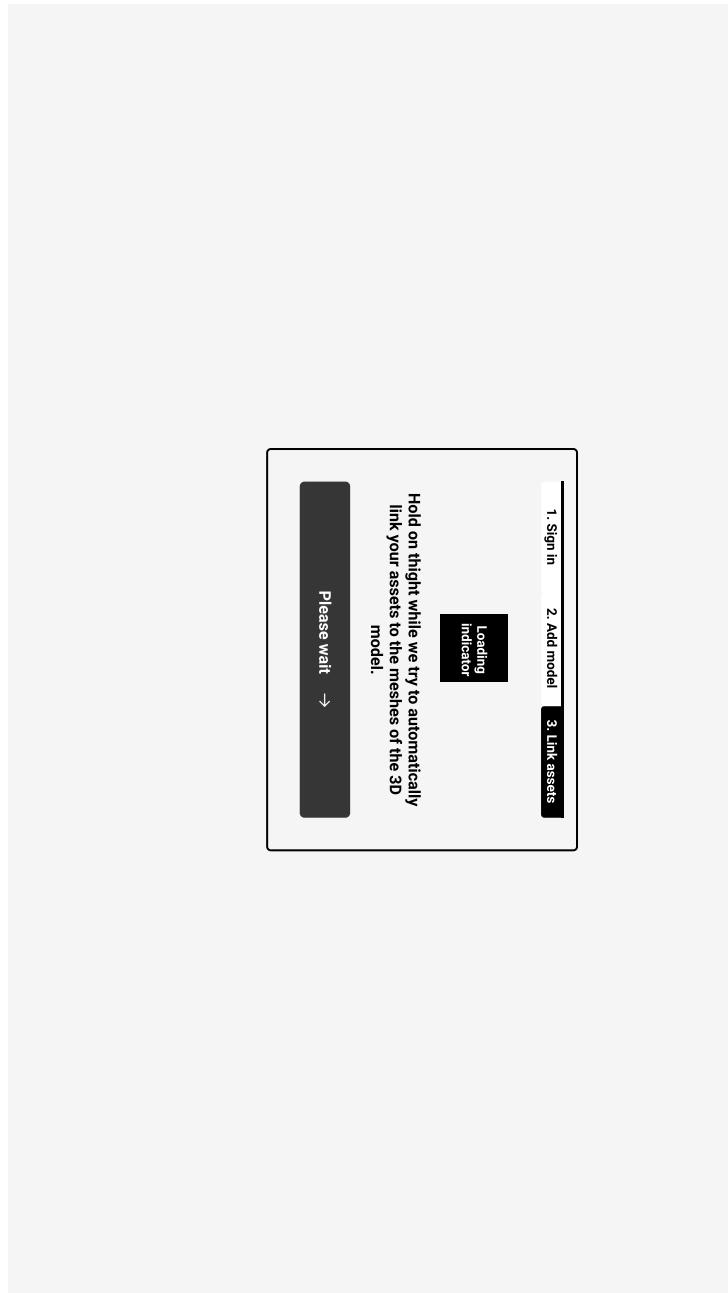


Abbildung 4.8.: Mockup der Asset verlinkung

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.

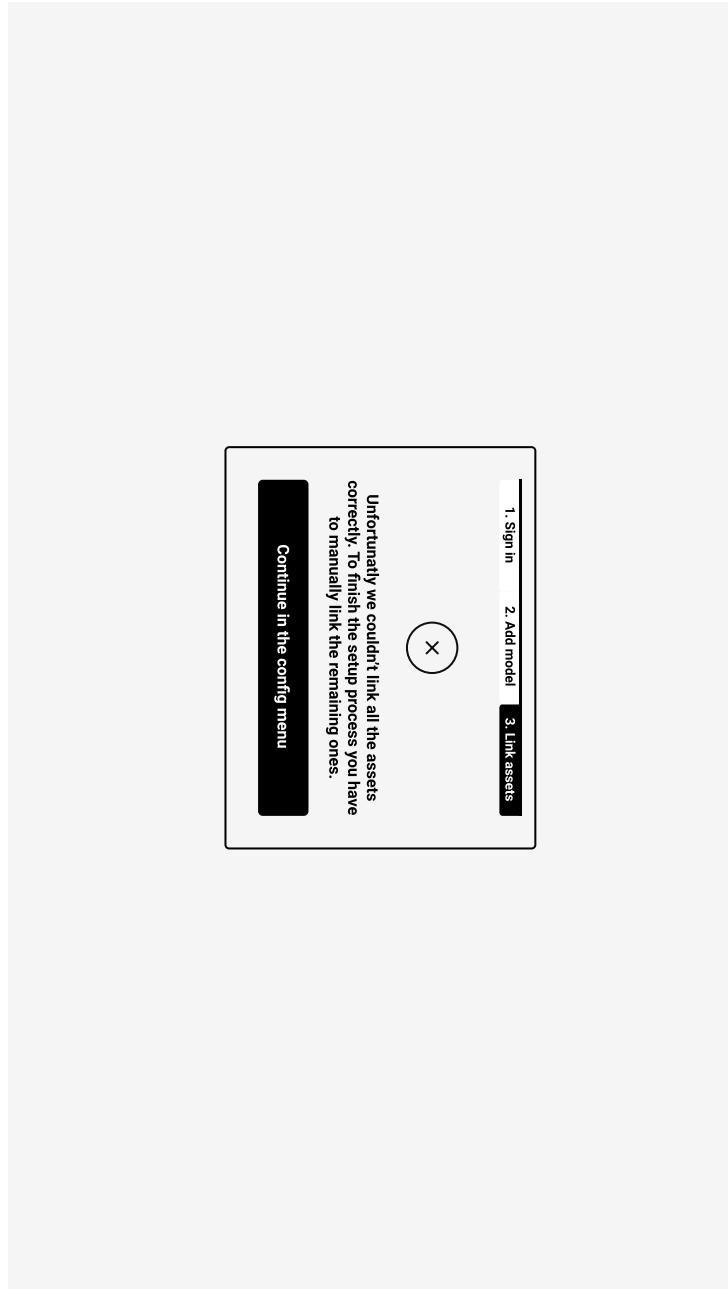


Abbildung 4.9.: Mockup einer fehlgeschlagenen Verlinkung

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.

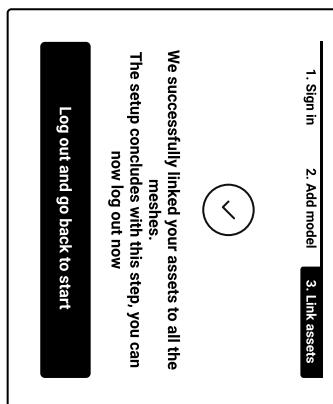


Abbildung 4.10.: Mockup einer erfolgreichen Verlinkung

4.2.3. Einstellungsmenü

Modeleinstellungen

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

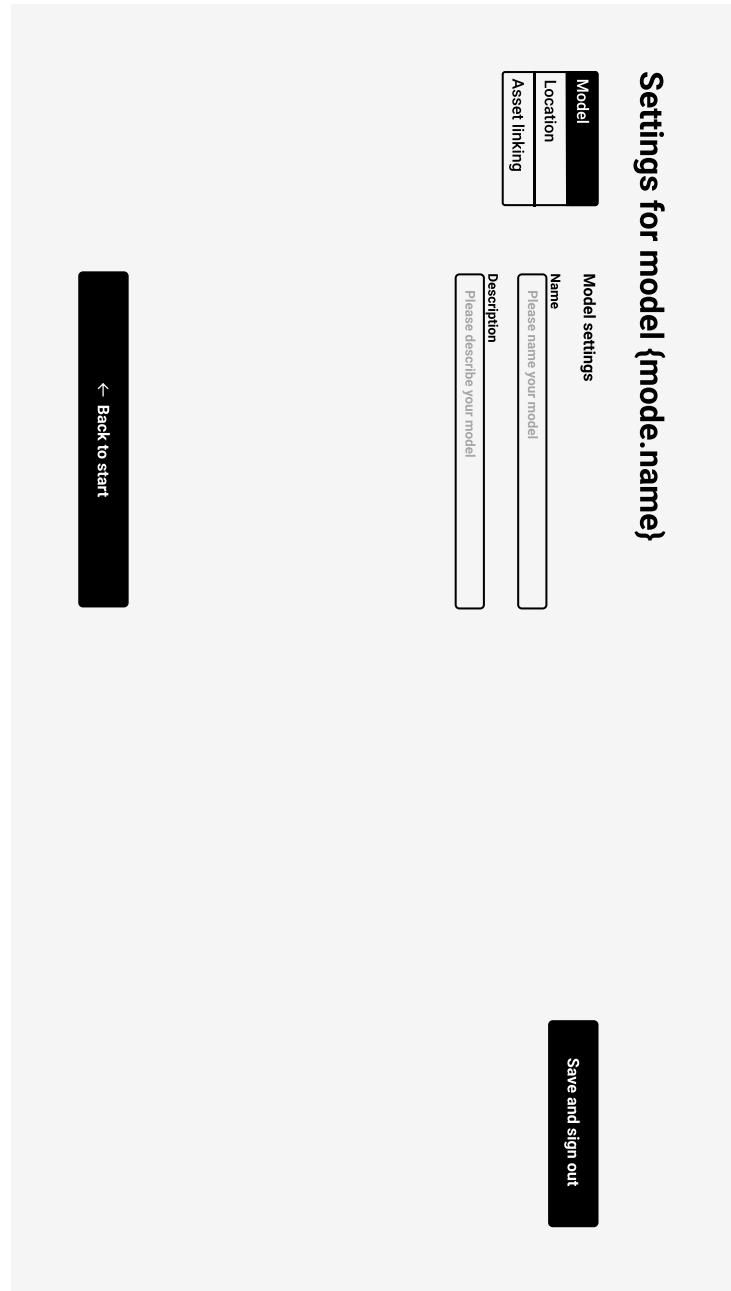


Abbildung 4.11.: Mockup des Modeleinstellungsmenü

Standorteinstellungen

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

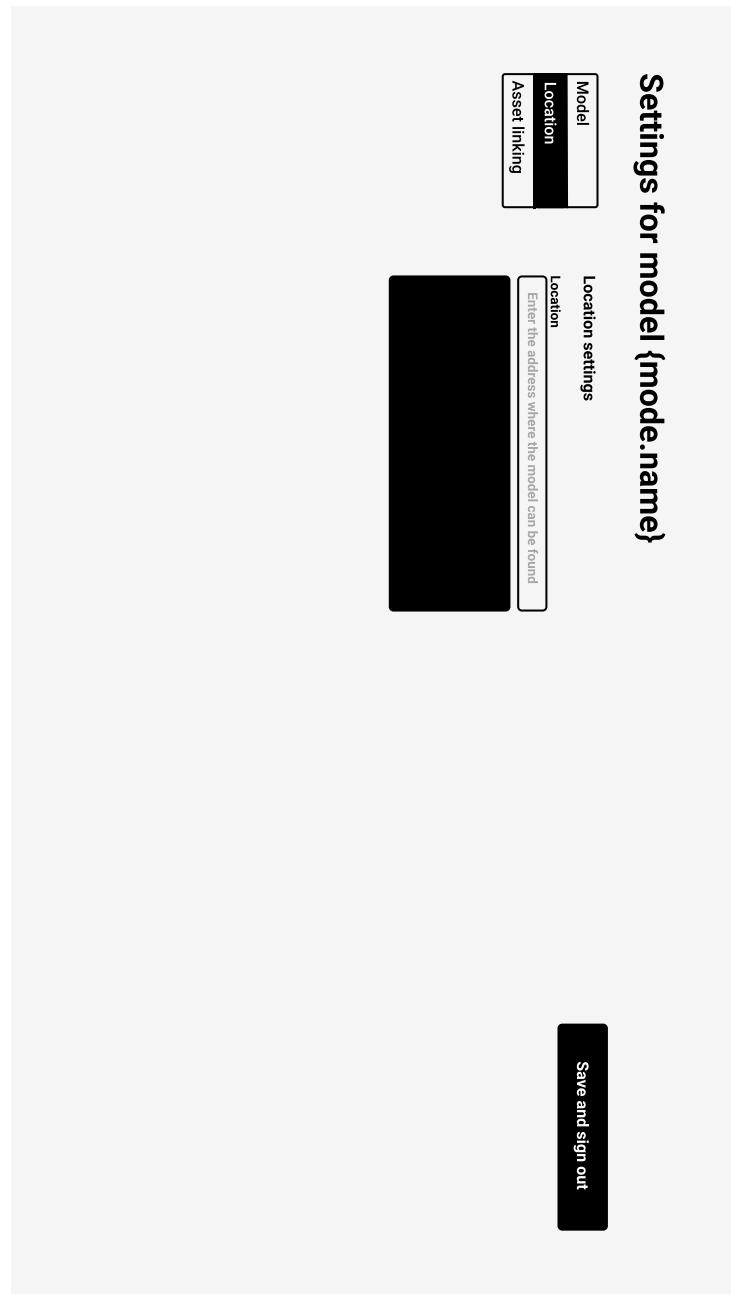


Abbildung 4.12.: Mockup des Standorteinstellungsmenü

Verlinkungseinstellungen

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

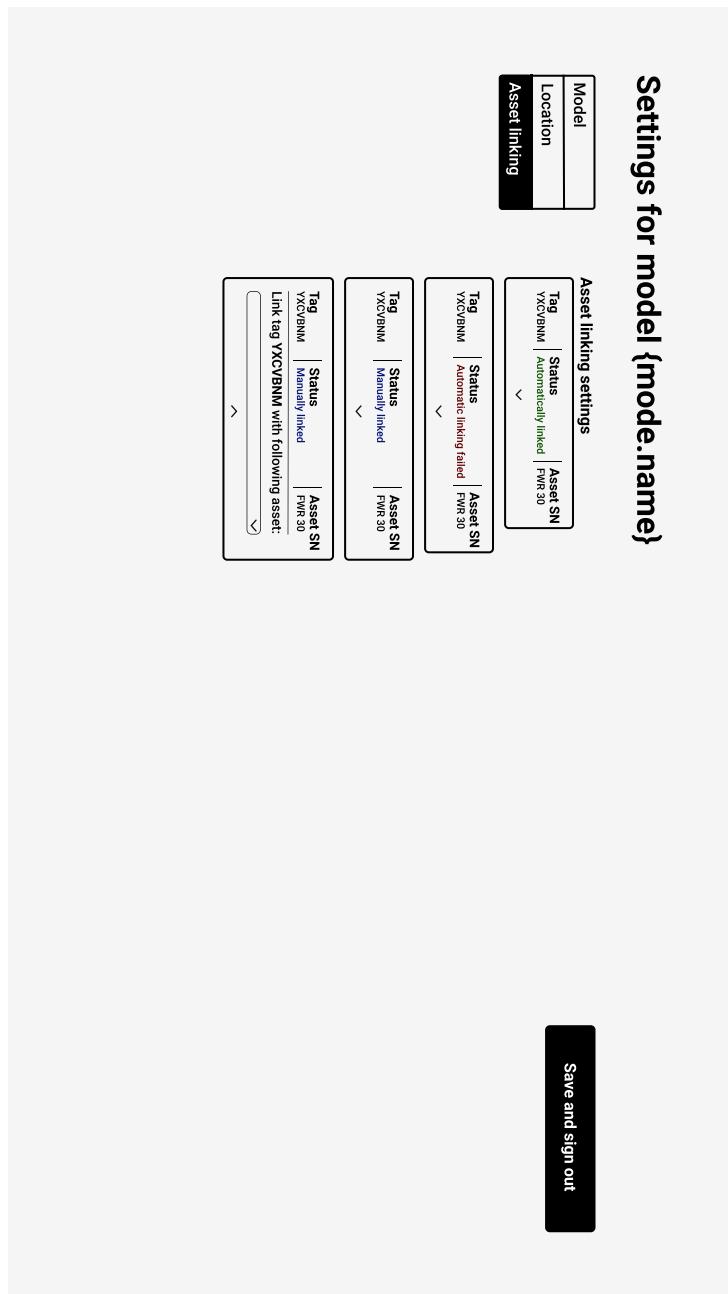


Abbildung 4.13.: Mockup des Verlinkungseinstellungsmenü

4.3. 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

4.3.1. 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

4.3.2. 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

4.3.3. 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

KAPITEL 5

IMPLEMENTIERUNG

5.1. OAuth2

5.1.1. Frontend

5.1.2. Backend

Log-in

Assets

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 dannach 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. Modelauswahl

Es wurde schon bei Kapitel 4.2.1 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 einfach 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 Modelauswahl 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 Models 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

Teil III.

Anhang

ANHANG A

ABKÜRZUNGSVERZEICHNIS

ANHANG B

GLOSSAR

ANHANG C

ABBILDUNGSVERZEICHNIS

1.1	Diagram der Projektaufbauorganisation	11
2.1	Stacey Matrix Grafik von Jurgen Appello.	21
2.2	Ein von mir mit Lucidchart erstelltes Wasserfallmodell	22
3.1	Eine von mir mit Lucidchart erstellte grobe Systemarchitektur	25
3.2	Diagramm Netilion Ökosystem von Jonas Schultheiss	26
3.3	Diagramm OSE-Dashboard Backend von Jonas Schultheiss	27
3.4	Diagramm OSE-Dashboard Frontend von Jonas Schultheiss	29
3.5	Grafik, welche den Git Flow Arbeitsablauf visualisiert	32
3.6	Personas	34
3.7	OAuth2 visualisiert durch ein Sequenzdiagramm	38
4.1	Sequenzdiagram, welches das automatische fetching Erklärt	41
4.2	Sequenzdiagram, welches die Zugriffskontrolle für den OSE Verantwortlichen beschreibt	42
4.3	Sequenzdiagram, welches die Zugriffskontrolle für den OSE-Dashboard Nutzer beschreibt	43
4.4	Mockup der Index Page	44
4.5	Mockup einer Fehlermeldung	45
4.6	Mockup einer erfolgreichen Anmeldung	46

4.7 Mockup der Registration	47
4.8 Mockup der Asset verlinkung	48
4.9 Mockup einer fehlgeschlagenen Verlinkung	49
4.10 Mockup einer erfolgreichen Verlinkung	50
4.11 Mockup des Modeleinstellungsmenü	51
4.12 Mockup des Standorteinstellungsmenü	52
4.13 Mockup des Verlinkungseinstellungsmenü	53

ANHANG D

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] 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).
- [3] 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).
- [4] Hardware security overview. Apple Support. 2021. URL: <https://support.apple.com/en-gb/guide/security/secf020d1074/web> (besucht am 30.03.2021).
- [5] 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).