

Individuelle praktische Arbeit

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

30. März 2021



Version: 0.0.6

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: 0.0.6
Datum: 30. März 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

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	6
1 Obligatorische Dokumentation	7
1.1 Ausgangslage	7
1.2 Detaillierte Aufgabenstellung	8
1.3 Mittel und Methoden	9
1.4 Vorkenntnisse	9
1.5 Vorarbeiten	9
1.6 Neue Lerninhalte	10
1.7 Arbeiten in den letzten sechs Monaten	10
1.8 Projektaufbauorganisation	10
1.9 Durchführungsblock	11
1.10 IPA-Termine	11
1.11 Projektplan	11
II Dokumentation	14
2 Projektmanagement	15
2.1 Entscheidung	15
2.2 Wasserfallmodell	16
2.2.1 Initialisierung des Dokuments	17
2.2.2 Analyse	17
2.2.3 Entwurf	17
2.2.4 Implementierung	18
2.2.5 Testen	18
2.2.6 Abschluss	18
3 Analyse	19
3.1 Systembeschreibung	19
3.1.1 Systemarchitektur	19
3.1.2 One Story Exhibit	19
3.1.3 Netilion	20
3.1.4 OSE-Dashboard: Backend	21

3.1.5 OSE-Dashboard: Frontend	23
3.2 Ist/Soll-Vergleich	23
3.2.1 Ist-Zustand	24
3.2.2 Soll-Zustand	24
3.3 Namenskonzept	25
3.3.1 User Stories	25
3.3.2 Akzeptanzkriterien	25
3.4 Versionskontrolle	26
3.4.1 Git Flow	26
3.4.2 Pipelines zu Vercel und Heroku	27
3.4.3 Dokumentation	27
3.5 Backupkonzept	28
3.5.1 Sicherheit der Daten	28
3.6 Personas	28
3.6.1 Endnutzer	28
3.6.2 MVT	29
3.6.3 OSE Verantwortlicher	29
3.6.4 Entwickler	29
3.7 User Stories	29
III Anhang	30
A Abkürzungsverzeichnis	31
B Glossar	32
C Abbildungsverzeichnis	33

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

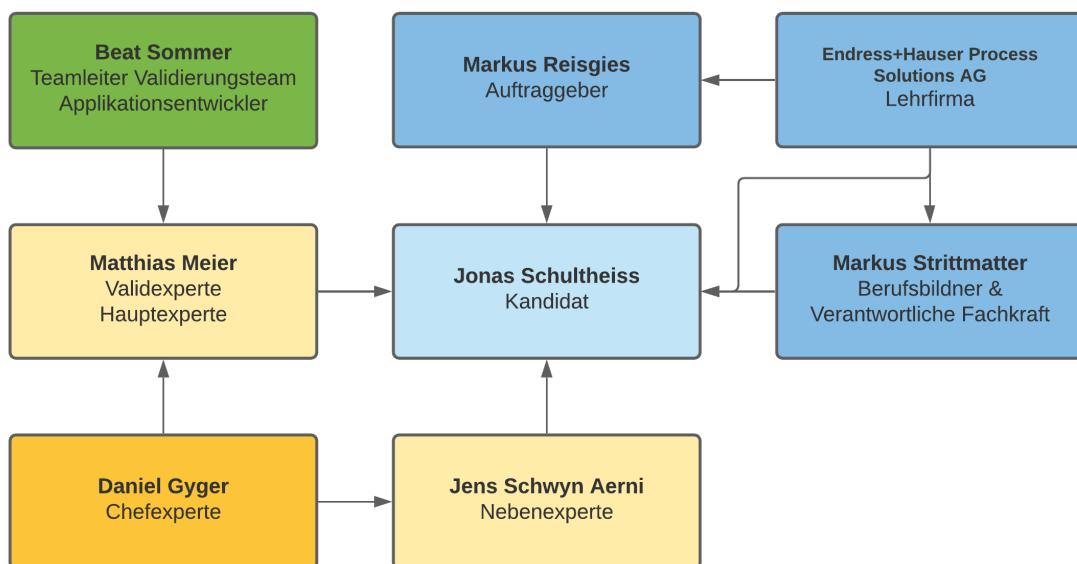


Abbildung 1.1.: Diagramm der Projektaufbauorganisation

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

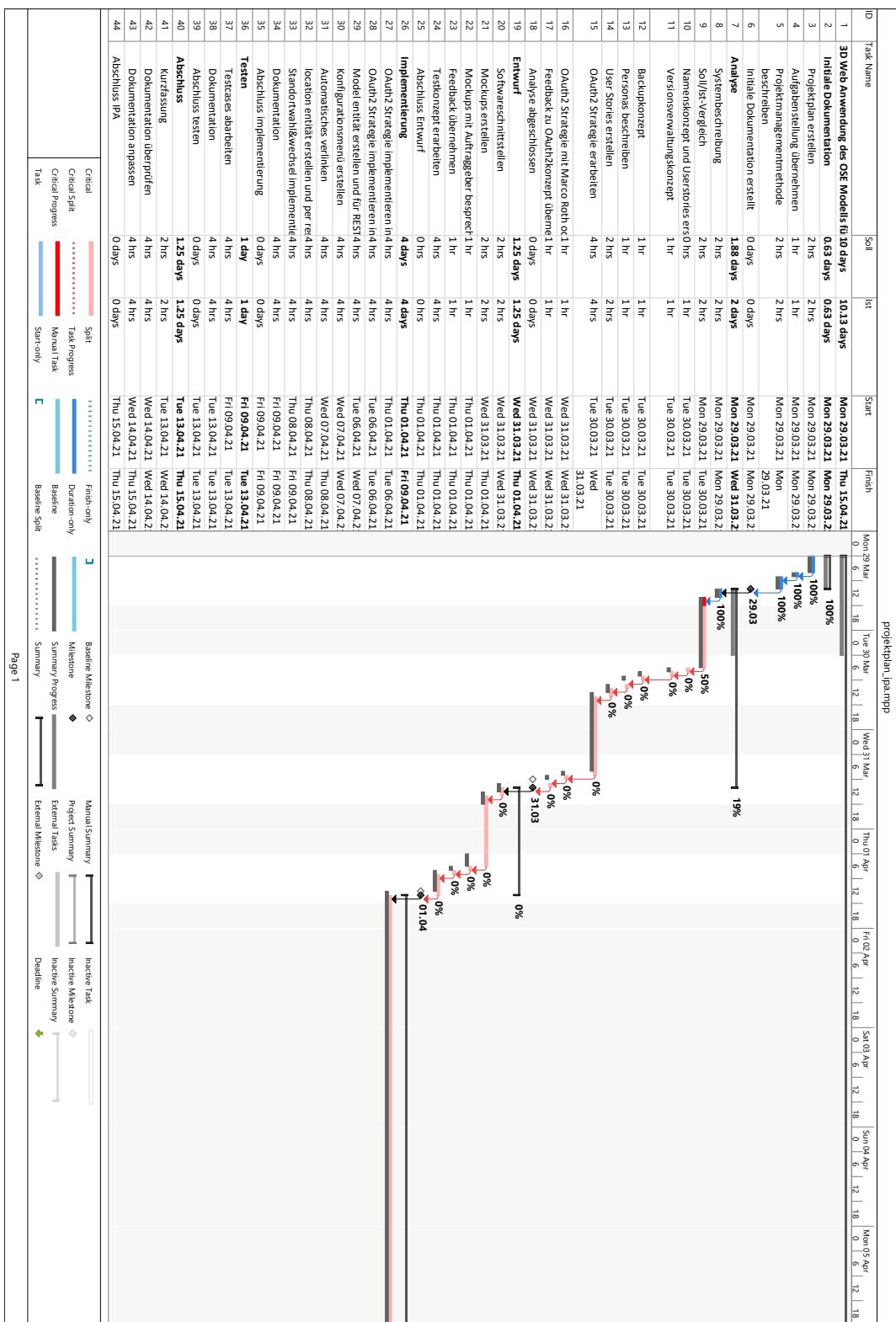
Antrittsgespräch: 23.03.2021 08:30

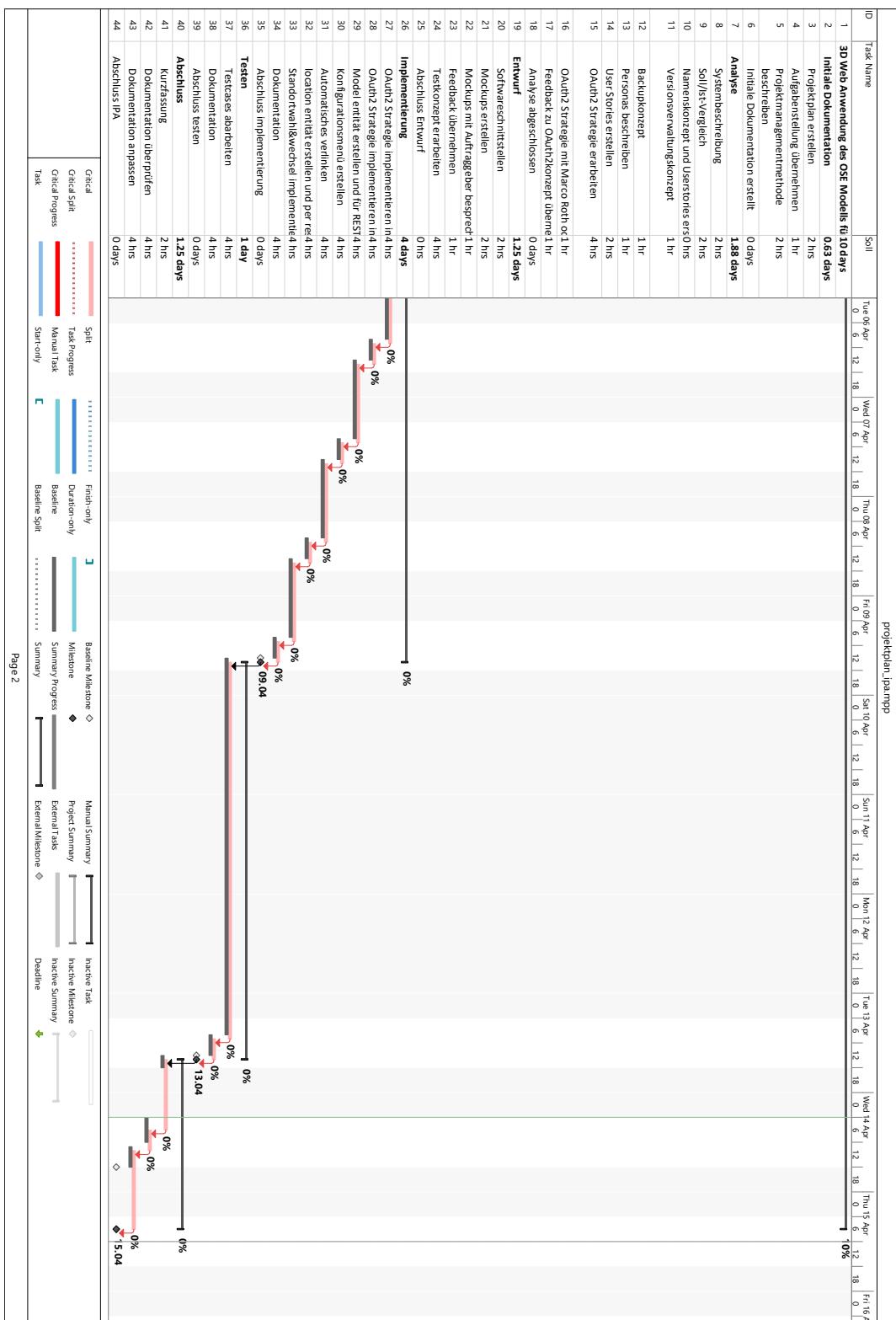
Erster Zwischenbesuch: 01.04.2021 09:00

Zweiter Zwischenbesuch: 01.04.2021 00:00

Endbesuch: 28.04.2021 09:00

1.11. Projektplan





Teil II.

Dokumentation

KAPITEL 2

PROJEKTMANAGEMENT

2.1. Entscheidung

Die Entscheidung der Projektmanagement Methode 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 Kriterien gegeben sind oder noch welche zu einem späterem Zeitpunkt dazu kommen. Die Y-Achse beschreibt, wie klar 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 Schätzungen 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 Projektmanagement Methode anwenden.

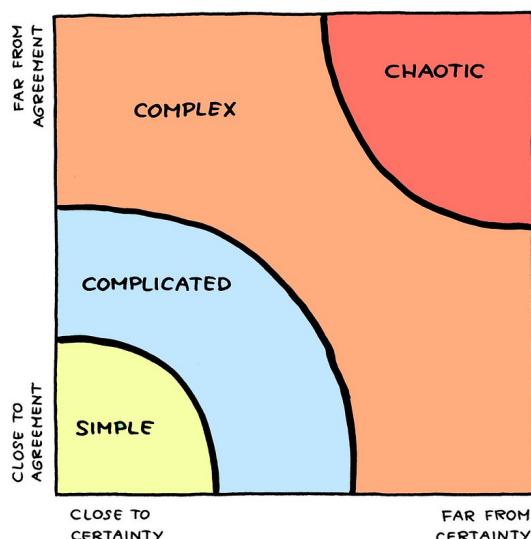


Abbildung 2.1.: Stacey Matrix

- **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 Anforderungen, sollte eine Agile Methode 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

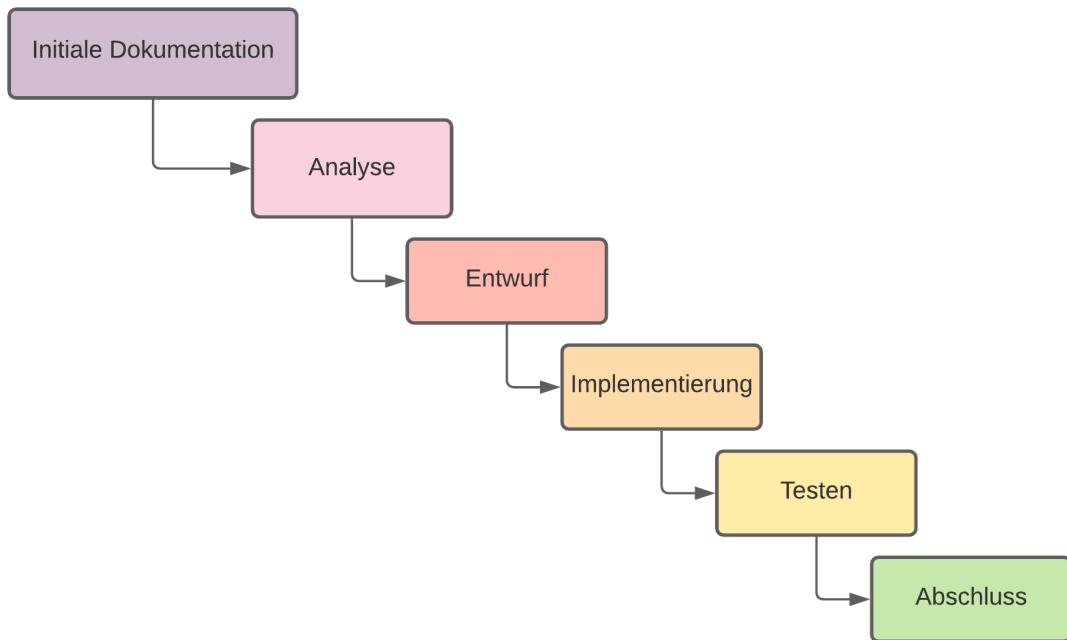


Abbildung 2.2.: Wasserfall Modell

Das Wasserfallmodell arbeitet mit auf sich selber aufbauenden sequentiellen Phasen. Man beginnt erst mit der nächsten Phase, 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 sicher zu gehen, dass alles stimmt und richtig konfiguriert ist. Dannach sollten essentielle Bestandteile, wie der detaillierten Zeitplan, das Arbeitsjournal, der obligatorisch Teil, erstellt werden. Dies habe ich soweit erstellt. Nun fehlt nurnoch 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 Soll/Ist-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 Soll/Ist-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 Loginprozess 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 dieses Feature implementiert wird, 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 deren Aufbau, und so weiter. Das Ergebniss dieser Phase ist ein ausgearbeitetes

UI/UX Design, genaue Pläne wie die Software aufgebaut werden soll und Testpläne.

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 Test wie gewünscht ablaufen. Stimmt das Ergebniss 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

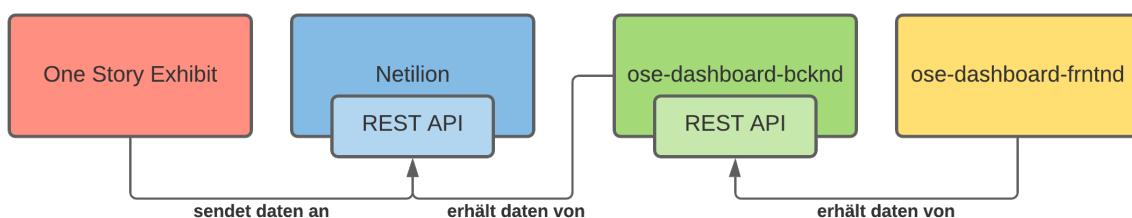


Abbildung 3.1.: Grobe Systemarchitektur

Das System des OSE Dashboard besteht aktuell aus vier essentiellen 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

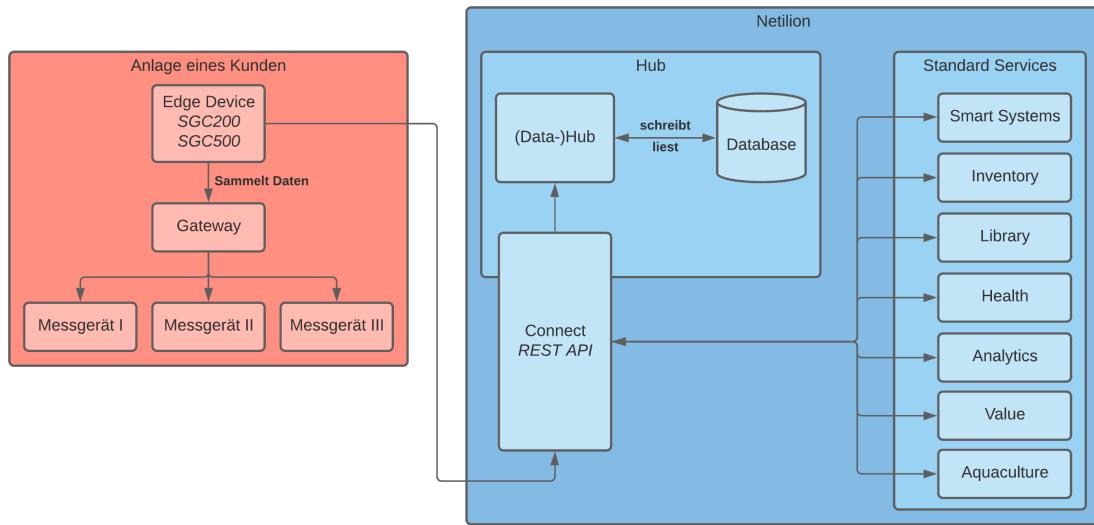


Abbildung 3.2.: Netilion Ökosystem

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 das Gerät 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 eine unseren Web Applikationen öffnen und die verarbeiteten Daten grafisch aufbereitet ansehen.

Hub

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

3.1.4. OSE-Dashboard: Backend

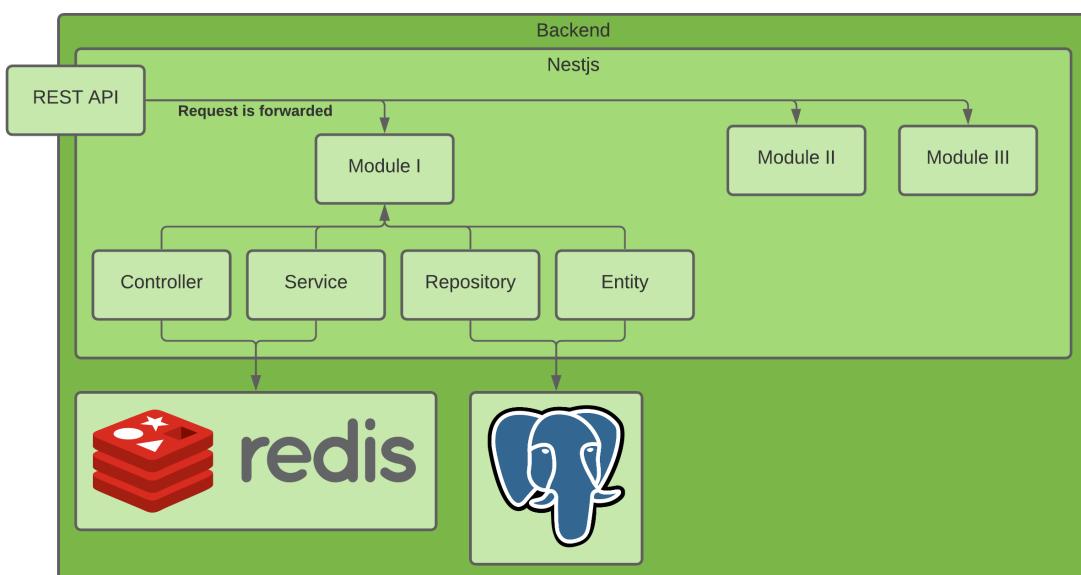


Abbildung 3.3.: OSE-Dashboard Backend

Die Aufgabe des Backends war primär das cachen der Daten. Ohne caching hätte jeder Client des Frontends direkt Anfragen an Netilion gesendet. Auch wenn ein Client nur alle 30 Minuten die Daten aktualisiert, wir wissen trotzdem nicht auf wievielen Clients unser Frontend gerade aktiv ist. Das Abonnement der REST-API besitzt eine maximale Nummer an Anfragen welche gemacht werden können, bevor auf das nächst teurere Abonnement 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 allerdings auch 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 Intervale/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. Mit der Hauptgrund war, dass man mit JavaScript nicht nur ein Frontend erstellen kann, sondern mittlerweile dank Nodejs auch Backends/Servers und sogar Apps fürs Smartphone. Wenn ein Produkt im ganzen Stack nur 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 Nodejs Backendframeworks, konnte mich allerdings nie ganz mit einem anfreunden. Die meisten Frameworks sind unopinionated. Dies gefällt mir im Backend nicht, da nach einer relativ kurzen Zeit ein Durcheinander entsteht, welches man dann selber aufräumen kann. Sobald man dann einen passenden Platzt 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 Boot und Angular, ist extrem opinionated und das meiste kommt out-of-the-box, ohne das man selber viele Konfigurierungen machen muss.

Inspiration

Wie in Java Spring Boot gibt es Controller, Services, Repositories und Entitäten. Diese werden jeweils nur für eine Resource erstellt. Mögliche Resourcen sind zum Beispiel "users" oder "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 verwendet.

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

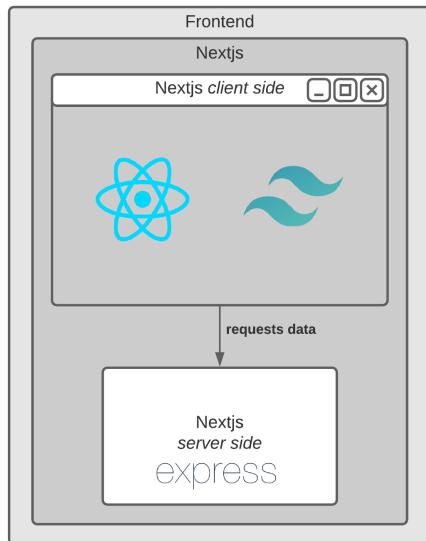


Abbildung 3.4.: OSE-Dashboard Frontend

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 Nextjs, welches kurze Zeit später veröffentlicht wurde. Nextjs ist ein Frontend Framework, welches routing, server side rendering und viele weitere features mit React verbindet. Nextjs 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 Nextjs ausmacht, ist das server side rendering, welches es out-of-the-box anbietet. So 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

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 sogar nicht einmal, dass die Endress+Hauser Gruppe nochmehr von ihnen besass, geschweige den, dass sie intern standartisiert wurde. Das heisst, dass immer die gleichen Messgeräte auf den Modellen sind.

Da es sich nur um ein einziges Model handelte, habe ich die ID's der digitalen Zwillingen 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 credentials verwenden. Dies wurde von Anfang mit Environment Variablen gelöst, da ich leider in der Vergangenheit schon einmal erfahren musste, was es heisst, die Credentials auf GitHub zu pushen. Anfangs wurde dies mit BasicAuth im Server-Side Teil von Nextjs gelöst. 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 Environment Variablen gespeichert oder direkt in den SourceCode geschrieben. Da es nun mehrere Modelle sind, muss unbedingt 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 nun 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 UseCase 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 Login braucht.
- OAuth2 vermindert den administrativen Aufwand
- Die Lösung ist optimaler und schöner

Konfigurationsmenü

Damit der Verantwortliche eines OSE Models Einstellungen vornehmen kann, muss unbedingt 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 erfahrung damit 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

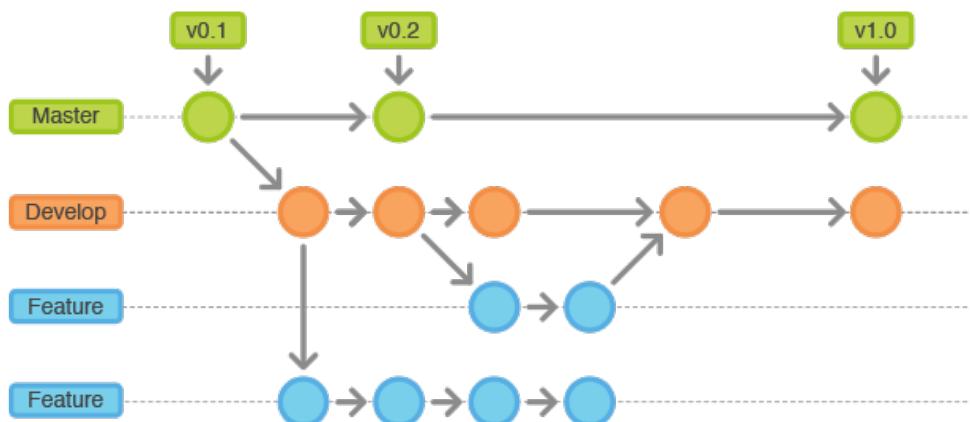


Abbildung 3.5.: Git Flow visualisiert

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 sie zum Master gemerged werden.
- **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 rendern 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 [2] und die externe SSD [3] sind verschlüsselt.

3.6. Personas

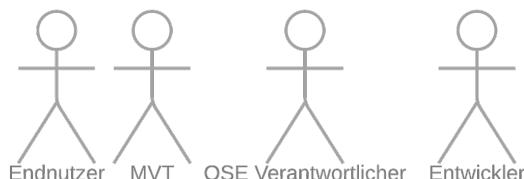


Abbildung 3.6.: Personas

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 allerdings noch einige weitere in Deutschland, Frankreich und so weiter. Für sie ist wichtig, dass sie wenig Mehraufwand haben. Im optimalen Fall sollte Login 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

Teil III.

Anhang

ANHANG A

ABKÜRZUNGSVERZEICHNIS

ANHANG B

GLOSSAR

ANHANG C

ABBILDUNGSVERZEICHNIS

1.1	Diagram der Projektaufbauorganisation	10
2.1	Stacey Matrix Grafik von Jurgen Appello.	15
2.2	Ein von mir mit Lucidchart erstelltes Wasserfallmodell	16
3.1	Eine von mir mit Lucidchart erstellte grobe Systemarchitektur	19
3.2	Diagram Netilion Ökosystem von Jonas Schultheiss	20
3.3	Diagram OSE-Dashboard Backend von Jonas Schultheiss	21
3.4	Diagram OSE-Dashboard Frontend von Jonas Schultheiss	23
3.5	Grafik, welche den Git Flow arbeitsablauf visualisiert	26
3.6	Personas	28