IPA Bericht

Webanwendung Reporting-System

Autor: Florian Leimer

Version / Datum: 1.0 / 22.01.2021

Projekt: Webanwendung Reporting-System

Projektstart: Mittwoch, 09.06.2020

Projektende: Freitag, 22.01.2021

Auftraggeber: Daniel Mosimann

# Dokumentinformationen

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Datum** | **Name** | **Beschreibung** |
| Vorlage | 23.06.2013 | A. Müller | Dokumentvorlage QV2013, Version V1.0 |
| V0.1 | 11.08.2020 | F. Leimer / Lernender | Übernahme der Vorlage und einfügen der Aufgabenstellung. |
| V1.0 | 22.01.2021 | F. Leimer / Lernender | Fertiger Probe-IPA-Bericht |

# Inhaltsverzeichnis

[Dokumentinformationen 2](#_Toc62245651)

[Inhaltsverzeichnis 3](#_Toc62245652)

[Abbildungsverzeichnis 6](#_Toc62245653)

[Tabellenverzeichnis 7](#_Toc62245654)

[1 Einleitung 8](#_Toc62245655)

[1.1 Inhalt und Zweck des Dokuments 8](#_Toc62245656)

[1.2 Abkürzungen und Definitionen 8](#_Toc62245657)

[Teil 1: Ablauf Organisation und Umfeld 9](#_Toc62245658)

[2 Aufgabestellung 9](#_Toc62245659)

[2.1 Ausgangslage 9](#_Toc62245660)

[2.2 Detaillierte Aufgabenstellung 9](#_Toc62245661)

[2.3 Mittel und Methoden 10](#_Toc62245662)

[2.4 Vorkenntnisse 10](#_Toc62245663)

[2.5 Vorarbeiten 10](#_Toc62245664)

[2.6 Neue Lerninhalte 10](#_Toc62245665)

[2.7 Arbeiten in den letzten 6 Monaten 10](#_Toc62245666)

[3 Projektmethodik 11](#_Toc62245667)

[3.1 Projektmethode 11](#_Toc62245668)

[3.2 Szenario 11](#_Toc62245669)

[3.3 Phasen 11](#_Toc62245670)

[3.4 Module 11](#_Toc62245671)

[3.5 Aufgaben 12](#_Toc62245672)

[3.6 Projektorganisation 13](#_Toc62245673)

[4 Zeitplanung 14](#_Toc62245674)

[4.1 Zeitplan 14](#_Toc62245675)

[5 Organisation der IPA 15](#_Toc62245676)

[5.1 Arbeitsplatz 15](#_Toc62245677)

[5.2 Datensicherung 15](#_Toc62245678)

[5.3 Wiederherstellung 15](#_Toc62245679)

[6 Firmenstandards 15](#_Toc62245680)

[7 Arbeitsjournal 16](#_Toc62245681)

[7.1 Woche 1: KW24 2020 16](#_Toc62245682)

[7.2 Woche 2: KW25 2020 16](#_Toc62245683)

[7.3 Woche 3: KW26 2020 17](#_Toc62245684)

[7.4 Woche 4: KW33 2020 17](#_Toc62245685)

[7.5 Woche 5: KW34 2020 18](#_Toc62245686)

[7.6 Woche 6: KW35 2020 18](#_Toc62245687)

[7.7 Woche 7: KW36 2020 18](#_Toc62245688)

[7.8 Woche 8: KW37 2020 19](#_Toc62245689)

[7.9 Woche 9: KW38 2020 19](#_Toc62245690)

[7.10 Woche 10: KW39 2020 19](#_Toc62245691)

[7.11 Woche 11: KW43 2020 20](#_Toc62245692)

[7.12 Woche 12: KW44 2020 20](#_Toc62245693)

[7.13 Woche 13: KW45 2020 21](#_Toc62245694)

[7.14 Woche 14: KW46 2020 21](#_Toc62245695)

[7.15 Woche 15: KW47 2020 21](#_Toc62245696)

[7.16 Woche 16: KW48 2020 22](#_Toc62245697)

[7.17 Woche 17: KW49 2020 22](#_Toc62245698)

[7.18 Woche 18: KW50 2020 22](#_Toc62245699)

[7.19 Woche 19: KW51 2020 23](#_Toc62245700)

[7.20 Woche 20: KW01 2021 23](#_Toc62245701)

[7.21 Woche 21: KW02 2021 24](#_Toc62245702)

[7.22 Woche 22: KW03 2021 24](#_Toc62245703)

[Teil 2: Projektdokumentation 25](#_Toc62245704)

[8 Kurzfassung 25](#_Toc62245705)

[8.1 Kurze Ausgangssituation 25](#_Toc62245706)

[8.2 Umsetzung 25](#_Toc62245707)

[8.3 Ergebnis 25](#_Toc62245708)

[9 Initialisierung 26](#_Toc62245709)

[9.1 Analyse 26](#_Toc62245710)

[9.2 Varianten - Suche 28](#_Toc62245711)

[10 Konzept 29](#_Toc62245712)

[10.1 Systementwurf 29](#_Toc62245713)

[10.2 Testkonzept 30](#_Toc62245714)

[11 Realisierung 32](#_Toc62245715)

[11.1 System realisieren 32](#_Toc62245716)

[11.2 Testprotokoll 37](#_Toc62245717)

[12 Reflexion 41](#_Toc62245718)

[13 Schlusswort 41](#_Toc62245719)

[14 Quellverzeichnis 41](#_Toc62245720)

[15 Glossar 42](#_Toc62245721)

# Abbildungsverzeichnis

[Abbildung 1 Use Case Diagramm 27](#_Toc62245603)

[Abbildung 2 Datenbank-Modell 30](#_Toc62245604)

[Abbildung 3 Angaben für Namespace und Einbinden von Autoload 33](#_Toc62245605)

[Abbildung 4 REST-Klasse 33](#_Toc62245606)

[Abbildung 5 Aufruf Controller und Ausgabe mit REST 34](#_Toc62245607)

[Abbildung 6 Beispiel Methode "createFromArray" 35](#_Toc62245608)

[Abbildung 7 Router-Konfiguration 36](file:///C:\xampp\htdocs\IPA-Testlauf\_Dokumente\IPA-Bericht.docx#_Toc62245609)

# Tabellenverzeichnis

[Tabelle 1 Abkürzungen und Definitionen 8](#_Toc62245612)

[Tabelle 2 Projektmethodik - Aufgaben 12](#_Toc62245613)

[Tabelle 3 Projektrollen 13](#_Toc62245614)

[Tabelle 4 Arbeitsjournal - Woche 1 16](#_Toc62245615)

[Tabelle 5 Arbeitsjournal - Woche 2 16](#_Toc62245616)

[Tabelle 6 Arbeitsjournal - Woche 3 17](#_Toc62245617)

[Tabelle 7 Arbeitsjournal - Woche 4 17](#_Toc62245618)

[Tabelle 8 Arbeitsjournal - Woche 5 18](#_Toc62245619)

[Tabelle 9 Arbeitsjournal - Woche 6 18](#_Toc62245620)

[Tabelle 10 Arbeitsjournal - Woche 7 18](#_Toc62245621)

[Tabelle 11 Arbeitsjournal - Woche 8 19](#_Toc62245622)

[Tabelle 12 Arbeitsjournal - Woche 9 19](#_Toc62245623)

[Tabelle 13 Arbeitsjournal - Woche 10 19](#_Toc62245624)

[Tabelle 14 Arbeitsjournal - Woche 11 20](#_Toc62245625)

[Tabelle 15 Arbeitsjournal - Woche 12 20](#_Toc62245626)

[Tabelle 16 Arbeitsjournal - Woche 13 21](#_Toc62245627)

[Tabelle 17 Arbeitsjournal - Woche 14 21](#_Toc62245628)

[Tabelle 18 Arbeitsjournal - Woche 15 21](#_Toc62245629)

[Tabelle 19 Arbeitsjournal - Woche 16 22](#_Toc62245630)

[Tabelle 20 Arbeitsjournal - Woche 17 22](#_Toc62245631)

[Tabelle 21 Arbeitsjournal - Woche 18 22](#_Toc62245632)

[Tabelle 22 Arbeitsjournal - Woche 19 23](#_Toc62245633)

[Tabelle 23 Arbeitsjournal - Woche 20 23](#_Toc62245634)

[Tabelle 24 Arbeitsjournal - Woche 21 24](#_Toc62245635)

[Tabelle 25 Arbeitsjournal - Woche 22 24](#_Toc62245636)

[Tabelle 26 Entscheidungsmatrix 29](#_Toc62245637)

[Tabelle 27 Konzept Testfall T-001 31](#_Toc62245638)

[Tabelle 28 Konzept Testfall T-002 31](#_Toc62245639)

[Tabelle 29 Konzept Testfall T-003 32](#_Toc62245640)

[Tabelle 30 Konzept Testfall T-004 32](#_Toc62245641)

[Tabelle 31 Konzept Testfall T-005 32](#_Toc62245642)

[Tabelle 32 Konzept Testfall T-006 32](#_Toc62245643)

[Tabelle 33 Testfall T-001 37](#_Toc62245644)

[Tabelle 34 Testfall T-002 38](#_Toc62245645)

[Tabelle 35 Testfall T-003 38](#_Toc62245646)

[Tabelle 36 Testfall T-004 39](#_Toc62245647)

[Tabelle 37 Testfall T-005 39](#_Toc62245648)

[Tabelle 38 Testfall T-006 40](#_Toc62245649)

[Tabelle 39 Glossar 42](#_Toc62245650)

# Einleitung

## Inhalt und Zweck des Dokuments

Dieses Dokument beinhaltet den Bericht zur Test-IPA, welche im 1. Semester des 4. Lehrjahres durchgeführt wird. Zweck davon ist alle wichtigen Informationen bezüglich Vorgabe, Planung und Umsetzung festzuhalten.

## Abkürzungen und Definitionen

| **Abkürzung** | **Bedeutung** |
| --- | --- |
| IPA | Individuelle praktische Arbeit |
| QV | Qualifikationsverfahren |
| HW | Hardware |
| SW | Software |
| DB | Datenbank |
| PHP | Personal Home Page Tools / PHP: Hypertext Preprocessor |
| SQL | Structured Query Language |

Tabelle 1 Abkürzungen und Definitionen

# Teil 1: Ablauf Organisation und Umfeld

# Aufgabestellung

## Ausgangslage

Jedes Dienstleistungsunternehmen muss die für ein Projekt, bzw. Auftrag aufgewendete Zeit rapportieren, um diese korrekt weiterverrechnen zu können. Dazu müssen Kundendaten sowie Projektdaten verwaltet werden können.

Das System muss per Login gesichert werden. Die Logindaten sollen von festgelegten Administratoren über die Webanwendung verwaltet werden können.

## Detaillierte Aufgabenstellung

Erstellung einer Webanwendung zur Verwaltung von Daten eines Dienstleistungsunternehmens.

**Daten je Benutzer:**

* Name
* Initialen
* Passwort
* Aktiv
* Status

**Daten je Kunde:**

* Name
* Kundennummer
* Adresse
* Bemerkungen

**Daten je Projekt:**

* Name
* Kunde (Relation zu Kunden)
* Startdatum
* Status (Umsetzung/Abgeschlossen/Support)
* Volumen
* Projektleiter (Relation zu Benutzern)
* Bemerkungen

**Daten je Report (Zeiterfassung):**

* Datum
* Projekt (Relation zu Projekten)
* Aufgewendete Zeit
* Beschreibung der Tätigkeit
* Erfasser (Aktueller Benutzer)

**Funktionen**

* Login
* Benutzer erstellen und bearbeiten
* Kunden erstellen, suchen und bearbeiten
* Projekte erstellen, suchen und bearbeiten
* Report erstellen und bearbeiten (solange Projekt nicht abgeschlossen ist)

**Rollen:**

* Admin: Alle Funktionen
* Benutzer: Alle Funktionen, ausgenommen Benutzerverwaltung

Die Reports können nur vom Erfasser bearbeitet werden.

## Mittel und Methoden

* PHP7, MySQL, Vue.js, REST und JWT
* Projektmethode Hermes

## Vorkenntnisse

Kenntnisse in PHP, MySQL und Vue.js werden vorausgesetzt.

## Vorarbeiten

## Neue Lerninhalte

* Login mit Benutzerverwaltung in Vue.js
* Verschiedene Benutzerrollen in Vue.js
* Datensätze mit beschränktem Zugriff

## Arbeiten in den letzten 6 Monaten

Projekt JWT in Modul 183.

# Projektmethodik

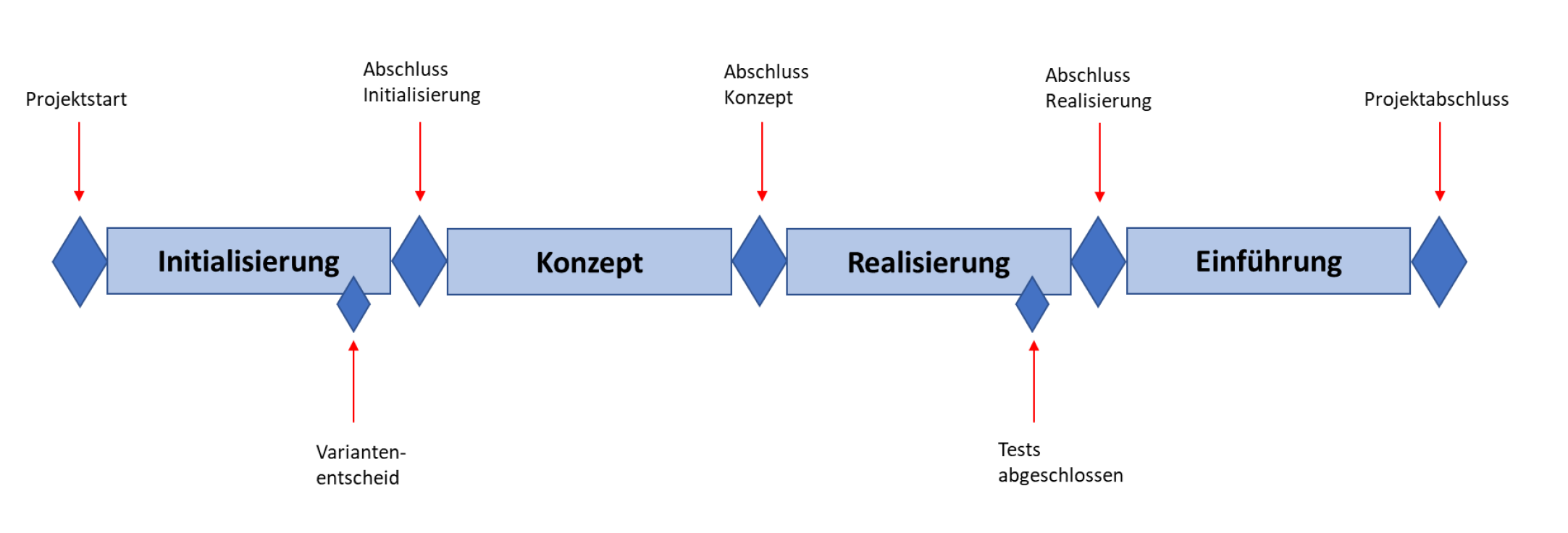
## Projektmethode

Als Projektmethodik verwende ich Hermes. Dies ist eine durch den Bund und die Swisscom entwickelte Projektmethodik. Genauer werde ich Hermes 5.1 IPA verwenden, eine durch den Kanton Bern vereinfachte Version für die IPA.

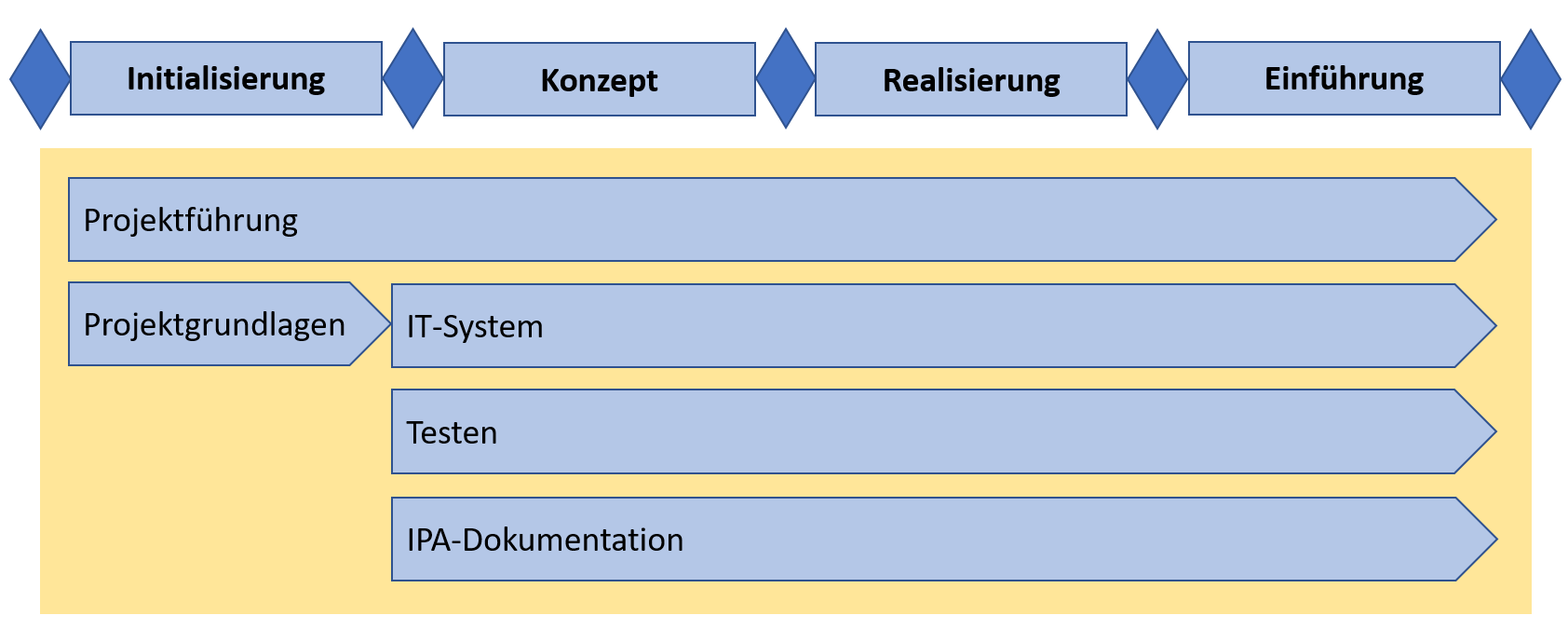
## Szenario

|  |  |
| --- | --- |
| Szenario: | IT-Individualanwendung |

## Phasen



## Module



## Aufgaben

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Initialisierung | Konzept | Realisierung | Einführung |
| Projektführung | Projektauftrag erarbeiten | Projekt führen und kontrollieren Projekt steuern Probleme behandeln und Erfahrungen nutzen Risiken managen Qualitätssicherung führen | | |
| Projekt-grundlagen | Studie erarbeiten   * Situationsanalyse (IST-Situation) * Ziele und Anforderungen verfeinern * Use-Case/Aktivitätsdiagramm * Lösungsvarianten erarbeiten oder Begründung, warum keine Varianten |  |  |  |
| IT-System |  | Systemkonzept erarbeiten   * Produkte vorstellen * Architektur * Struktur der Anwendung * Klassendiagramme | System realisieren   * Konzeptionelle Überlegungen notieren * Code dokumentieren | System aktivieren |
| Testen |  | Testkonzept erarbeiten   * Testziele * Testobjekte * Testarten (manuelle/automatisierte Tests) * Testumgebung, Testinfrastruktur * Testdaten * Testfälle (Testaufgaben) | Testinfrastruktur realisieren   * Test durchführen * Testprotokoll erstellen * Test auswerten |  |
| Dokumentation IPA | Initialisierung dokumentieren | Konzept dokumentieren | Realisierung dokumentieren | Einführung dokumentieren |

Tabelle 2 Projektmethodik - Aufgaben

## Projektorganisation

Die Projektorganisation kann von diesem Muster abweichen. Ggf. anpassen. Muss mit den Angaben auf dem Titelblatt übereinstimmen und zur Projektmethode passen.

### Projektorganisation

**Auftraggeber:**

BBZ Solothurn – Daniel Mosimann

**Projektleiter:**

Florian Leimer

**Fachspezialist**

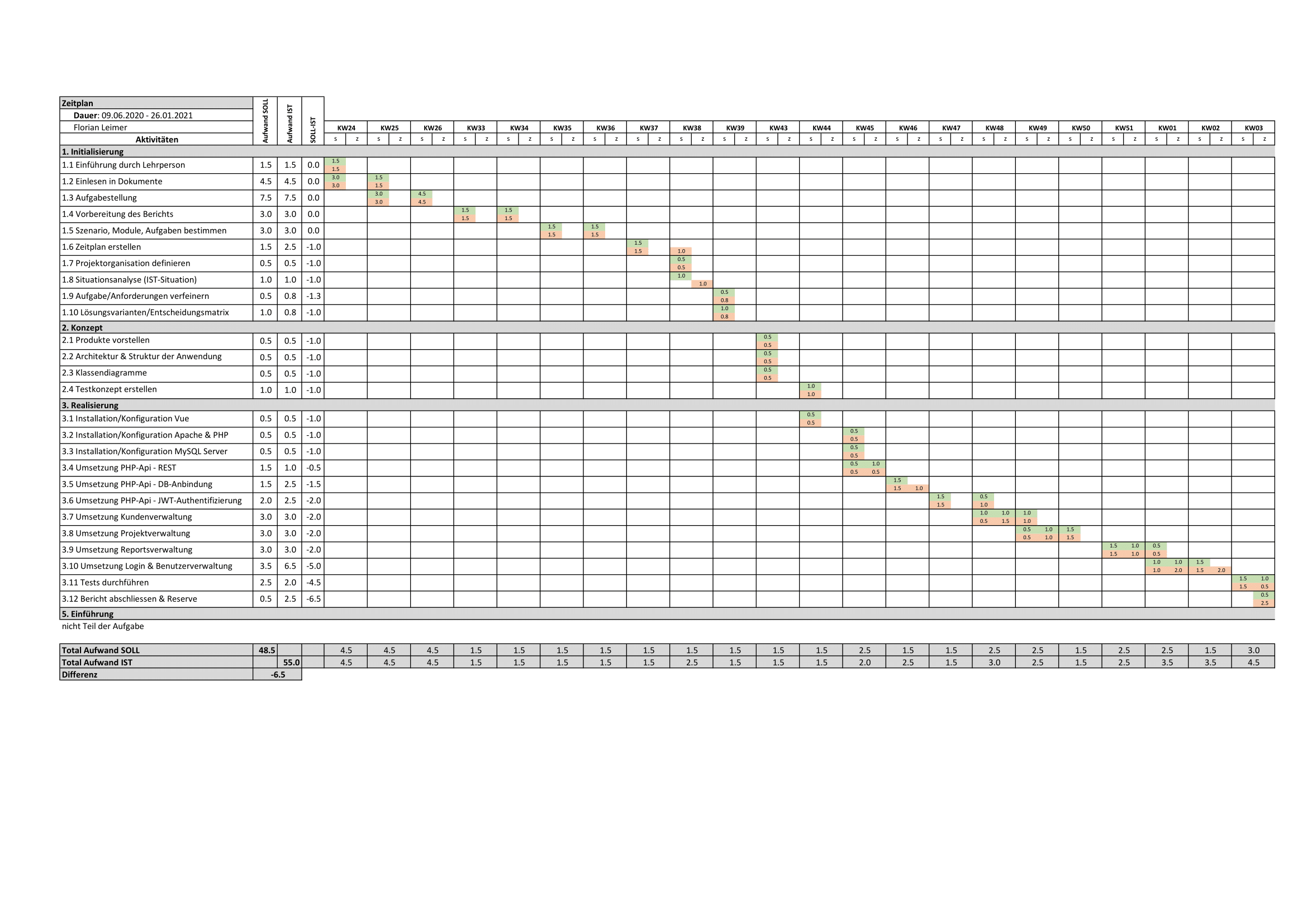
Daniel Mosimann

### Projektrollen

|  |  |
| --- | --- |
| **Auftraggeber:** | BBZ Solothurn -Daniel Mosimann |
| **Projektleiter:** | Florian Leimer |
| **Fachspezialist** | Daniel Mosimann |

Tabelle 3 Projektrollen

# Zeitplanung



# Organisation der IPA

## Arbeitsplatz

Als Arbeitsplatz wird bei der Probe-IPA hauptsächlich das Schulzimmer in der Berufsschule dienen. Der persönliche Laptop wird als Arbeitsgerät genutzt. Für die Dokumentation werden die Office-Programme gebraucht. Als Entwicklungsumgebung wird PHPStorm eingesetzt.

## Datensicherung

Die IPA-Dokumentation sowie alle anderen Dokumente wie auch das Projekt selbst werden via Github in einem Repository gesichert. Dies geschieht immer nach dem Beenden eines Arbeitsblockes.

Mit Github kann auch eine Versionierung gewährleistet werden, wobei jederzeit zu einer vorherigen Version zurückgekehrt werden kann.

## Wiederherstellung

Verlorene Daten oder Versionen können über Github und dessen Versionierung wiederhergestellt werden. Es besteht auch immer eine lokale Kopie der Daten auf dem privaten Laptop.

# Firmenstandards

Firmenstandards sind bei der Probe-IPA keine vorhanden, da diese im Rahmen der Berufsschule durchgeführt wird.

# Arbeitsjournal

## Woche 1: KW24 2020

|  |  |  |  |
| --- | --- | --- | --- |
| **Tätigkeiten** | **Projektphase** | **Zeit SOLL** | **Zeit IST** |
| Einführung durch Herr Mosimann | Initialisierung | 1.5 | 1.5 |
| Einlesen in Dokumente auf OneNote | Initialisierung | 3 | 3 |
| ***Total*** | | ***4.5*** | ***4.5*** |
| **Tagesablauf** | | | |
| Der Tag hat am Morgen mit einer Teams-Besprechung mit Herr Mosimann gestartet, wo alle Dokumente und die Vorgaben grob erklärt wurden. Anschliessend habe ich mir die Dokumente genauer angeschaut. | | | |
| **Wissensbeschaffung** | | | |
| Dokumente auf OneNote von Lehrperson | | | |
| **Hilfestellungen** | | | |
| - | | | |
| **Reflexion** | | | |
| Heute kamen ziemlich viele Infos, welche ich noch nicht alle verstehe. Da werde ich noch Zeit benötigen, mich selber einzulesen. | | | |

Tabelle 4 Arbeitsjournal - Woche 1

## Woche 2: KW25 2020

|  |  |  |  |
| --- | --- | --- | --- |
| **Tätigkeiten** | **Projektphase** | **Zeit SOLL** | **Zeit IST** |
| Einlesen in Dokumente auf OneNote | Initialisierung | 1.5 | 1.5 |
| Ideenfindung für Projekt | Initialisierung | 3 | 3 |
| ***Total*** | | ***4.5*** | ***4.5*** |
| **Tagesablauf** | | | |
| Nachdem ich mir die restlichen Dokumente angeschaut hatte, begann ich mir eine Projektidee zu überlegen und nach Lösungen zu recherchieren. | | | |
| **Wissensbeschaffung** | | | |
| Dokumente auf OneNote von Lehrperson, Internet | | | |
| **Hilfestellungen** | | | |
| - | | | |
| **Reflexion** | | | |
| Leider hatte ich etwas lange keine umsetzbare Idee, wodurch ich viel Zeit verloren habe. | | | |

Tabelle 5 Arbeitsjournal - Woche 2

## Woche 3: KW26 2020

|  |  |  |  |
| --- | --- | --- | --- |
| **Tätigkeiten** | **Projektphase** | **Zeit SOLL** | **Zeit IST** |
| Ideenfindung für Projekt | Initialisierung | 1.5 | 1.5 |
| Schreiben der Aufgabenstellung | Initialisierung | 3 | 3 |
| ***Total*** | | ***4.5*** | ***4.5*** |
| **Tagesablauf** | | | |
| Durch die Woche habe ich eine Idee gefunden, welche umsetzbar sein sollte. Dafür musste ich ein paar Dinge recherchieren. Anschliessend habe ich die Aufgabenstellung geschrieben und diese Herr Mosimann zukommen lassen. | | | |
| **Wissensbeschaffung** | | | |
| Dokumente auf OneNote von Lehrperson, Internet | | | |
| **Hilfestellungen** | | | |
| - | | | |
| **Reflexion** | | | |
| Diese Woche kam ich gut voran und konnte die Aufgabenstellung Herr Mosimann abgeben. | | | |

Tabelle 6 Arbeitsjournal - Woche 3

## Woche 4: KW33 2020

|  |  |  |  |
| --- | --- | --- | --- |
| **Tätigkeiten** | **Projektphase** | **Zeit SOLL** | **Zeit IST** |
| Vorbereitung des IPA-Berichts | Initialisierung | 0.75 | 0.75 |
| Einrichtung des Github-Repository | Initialisierung | 0.25 | 0.25 |
| Einrichtung eines Vue-Templates | Initialisierung | 0.5 | 0.5 |
| ***Total*** | | ***1.5*** | ***1.5*** |
| **Tagesablauf** | | | |
| Die erste Woche nach den Sommerferien habe ich für Vorbereitungen genutzt. Dazu gehören die Vorbereitung des IPA-Berichts, die Einrichtung des Github-Repositorys sowie die Einrichtung des Vue-Templates, welches ich für die Webanwendung benutzen werde. | | | |
| **Wissensbeschaffung** | | | |
| IPA-Bericht Vorlage | | | |
| **Hilfestellungen** | | | |
| - | | | |
| **Reflexion** | | | |
| Bei den Vorbereitungen kam ich relativ gut voran und kann nächstes Mal mit der Dokumentation starten. | | | |

Tabelle 7 Arbeitsjournal - Woche 4

## Woche 5: KW34 2020

|  |  |  |  |
| --- | --- | --- | --- |
| **Tätigkeiten** | **Projektphase** | **Zeit SOLL** | **Zeit IST** |
| Nachführen des Arbeitsjournals | Initialisierung | 0.75 | 0.75 |
| Beginn mit IPA-Bericht (Aufgabenstellung & Projektmethodik) | Initialisierung | 0.75 | 0.75 |
| ***Total*** | | ***1.5*** | ***1.5*** |
| **Tagesablauf** | | | |
| Da ich das Arbeitsjournal bisher nicht geführt habe musste ich dieses heute im vorbereiteten Bericht nachführen. Anschliessend habe ich die Aufgabenstellung in den Bericht übernommen und mit der Projektmethodik begonnen. | | | |
| **Wissensbeschaffung** | | | |
| IPA-Bericht Vorlage, Dokumente zu Hermes auf OneNote | | | |
| **Hilfestellungen** | | | |
| - | | | |
| **Reflexion** | | | |
| Das Nachführen des Arbeitsjournals hat relativ viel Zeit gebraucht. Trotzdem konnte ich noch mit der Projektmethodik beginnen. | | | |

Tabelle 8 Arbeitsjournal - Woche 5

## Woche 6: KW35 2020

|  |  |  |  |
| --- | --- | --- | --- |
| **Tätigkeiten** | **Projektphase** | **Zeit SOLL** | **Zeit IST** |
| Dokumentation der Projektmethodik | Initialisierung | 1.5 | 1.5 |
| ***Total*** | | ***1.5*** | ***1.5*** |
| **Tagesablauf** | | | |
| Nach dem Beginn der Dokumentation der Projektmethodik letzte Woche habe ich diese Woche daran weitergearbeitet. | | | |
| **Wissensbeschaffung** | | | |
| Dokumente zu Hermes auf OneNote- | | | |
| **Hilfestellungen** | | | |
| - | | | |
| **Reflexion** | | | |
| Da ich Hermes momentan noch nicht wirklich durchblicke kam ich schlecht voran. | | | |

Tabelle 9 Arbeitsjournal - Woche 6

## Woche 7: KW36 2020

|  |  |  |  |
| --- | --- | --- | --- |
| **Tätigkeiten** | **Projektphase** | **Zeit SOLL** | **Zeit IST** |
| Dokumentation der Projektmethodik | Initialisierung | 1.5 | 1.5 |
| ***Total*** | | ***1.5*** | ***1.5*** |
| **Tagesablauf** | | | |
| Übernahme der Projektmethodik von der Vorlage. | | | |
| **Wissensbeschaffung** | | | |
| Hermes Beispiel in OneNote | | | |
| **Hilfestellungen** | | | |
| Herr Mosimann durch Beantwortung von Fragen | | | |
| **Reflexion** | | | |
| Herr Mosimann hat uns heute mitgeteilt, dass die Vorlage von Hermes praktisch 1 zu 1 übernommen werden kann. Dadurch war die Dokumentation von der letzten Woche überflüssig. | | | |

Tabelle 10 Arbeitsjournal - Woche 7

## Woche 8: KW37 2020

|  |  |  |  |
| --- | --- | --- | --- |
| **Tätigkeiten** | **Projektphase** | **Zeit SOLL** | **Zeit IST** |
| Beginn mit Zeitplan | Initialisierung | 1.5 | 1.5 |
| ***Total*** | | ***1.5*** | ***1.5*** |
| **Tagesablauf** | | | |
| Heute habe ich mit dem Zeitplan begonnen. | | | |
| **Wissensbeschaffung** | | | |
| Zeitplan-Beispiel auf OneNote | | | |
| **Hilfestellungen** | | | |
| - | | | |
| **Reflexion** | | | |
| Heute kam ich relativ gut voran. Der Zeitplan sollte aber bereits etwas weiter sein. | | | |

Tabelle 11 Arbeitsjournal - Woche 8

## Woche 9: KW38 2020

|  |  |  |  |
| --- | --- | --- | --- |
| **Tätigkeiten** | **Projektphase** | **Zeit SOLL** | **Zeit IST** |
| Umsetzung Zeitplan | Initialisierung | 0 | 1.0 |
| Definierung Projektorganisation | Initialisierung | 0.5 | 0.5 |
| Situationsanalyse | Initialisierung | 1.0 | 1.0 |
| ***Total*** | | ***1.5*** | ***2.5*** |
| **Tagesablauf** | | | |
| Nachdem ich den Zeitplan fertiggestellt hatte, hat Herr Mosimann uns erneut das Hermes-Konzept erläutert. Anschliessend konnte ich noch die Projektorganisation definieren. Zuhause habe ich dann noch die Situationsanalyse durchgeführt. | | | |
| **Wissensbeschaffung** | | | |
| Zeitplan-Beispiel auf OneNote und Vorlage IPA-Bericht | | | |
| **Hilfestellungen** | | | |
| - | | | |
| **Reflexion** | | | |
| Diese Woche kam ich gut voran und konnte dank Arbeit Zuhause wieder mit dem Zeitplan aufschliessen. | | | |

Tabelle 12 Arbeitsjournal - Woche 9

## Woche 10: KW39 2020

|  |  |  |  |
| --- | --- | --- | --- |
| **Tätigkeiten** | **Projektphase** | **Zeit SOLL** | **Zeit IST** |
| Aufgaben/Anforderungen verfeinern | Initialisierung | 0.5 | 0.75 |
| Lösungsvarianten | Initialisierung | 1.0 | 0.75 |
| ***Total*** | | ***1.5*** | ***1.5*** |
| **Tagesablauf** | | | |
| Diese Woche konnte ich die Aufgabe aus der Aufgabenstellung etwas genauer definieren sowie Lösungsvarianten ausarbeiten und mich mittels einer Entscheidungsmatrix für eine entscheiden. | | | |
| **Wissensbeschaffung** | | | |
| Vorlage IPA-Bericht und Aufgabenstellung | | | |
| **Hilfestellungen** | | | |
| - | | | |
| **Reflexion** | | | |
| Trotzdem, dass die Zeiten nicht genau mit denen im Zeitplan übereinstimmten ging es schlussendlich auf. Und endlich konnte ich den Abschnitt der Initialisierung abschliessen. | | | |

Tabelle 13 Arbeitsjournal - Woche 10

## Woche 11: KW43 2020

|  |  |  |  |
| --- | --- | --- | --- |
| **Tätigkeiten** | **Projektphase** | **Zeit SOLL** | **Zeit IST** |
| Systementwurf erstellt | Konzept | 1.5 | 1.5 |
| ***Total*** | | ***1.5*** | ***1.5*** |
| **Tagesablauf** | | | |
| Den Systementwurf zu erstellen ging relativ gut. Ich habe die mehreren Aufgaben im Zeitplan hier als eine zusammengefasst, da dies meiner Meinung nach zusammen gehört. | | | |
| **Wissensbeschaffung** | | | |
| Aufgabenstellung und Hermes-Vorgaben | | | |
| **Hilfestellungen** | | | |
| - | | | |
| **Reflexion** | | | |
| Heute kam ich in der Schule sehr gut voran und konnte alle geplanten Aufgaben abschliessen. | | | |

Tabelle 14 Arbeitsjournal - Woche 11

## Woche 12: KW44 2020

|  |  |  |  |
| --- | --- | --- | --- |
| **Tätigkeiten** | **Projektphase** | **Zeit SOLL** | **Zeit IST** |
| Testkonzept erarbeiten | Konzept | 1.0 | 1.0 |
| Installation Vue | Realisierung | 0.5 | 0.5 |
| ***Total*** | | ***1.5*** | ***1.5*** |
| **Tagesablauf** | | | |
| Nachdem ich das Testkonzept, welches ich unter anderem auf Basis des Systementwurfs und der Aufgabenstellung erstellt habe, festgelegt hatte, ging es bereits an die Realisierung. Diese Woche habe ich aber nur die Installation und Grund-Konfiguration von Vue durchgeführt. | | | |
| **Wissensbeschaffung** | | | |
| Das Testkonzept habe ich auf Basis meines eigenen Systementwurfs erstellt. Für die Installation von Vue habe ich Dokumente aus früheren Modulen verwendet. | | | |
| **Hilfestellungen** | | | |
| - | | | |
| **Reflexion** | | | |
| Ich konnte weiterhin produktiv arbeiten und bin froh nun auch bereits die Phase des Konzepts abgeschlossen zu haben. | | | |

Tabelle 15 Arbeitsjournal - Woche 12

## Woche 13: KW45 2020

|  |  |  |  |
| --- | --- | --- | --- |
| **Tätigkeiten** | **Projektphase** | **Zeit SOLL** | **Zeit IST** |
| Installation Apache & PHP | Realisierung | 0.5 | 0.5 |
| Installation MySQL-Server | Realisierung | 0.5 | 0.5 |
| PHP-Api – REST | Realisierung | 1.5 | 1.0 |
| ***Total*** | | ***2.5*** | ***2.0*** |
| **Tagesablauf** | | | |
| Nach der problemlosen Installation von den Servern mit XAMPP konnte ich bald mit der PHP-API beginnen. Für die REST-Umsetzung habe ich mich an einer in einem früheren Modul erstellten PHP-Applikation orientiert. Dadurch konnte ich etwas Zeit sparen und musste Zuhause weniger daran arbeiten. | | | |
| **Wissensbeschaffung** | | | |
| Aus früheren Modulen und Internetrecherchen. | | | |
| **Hilfestellungen** | | | |
| Die in einem früheren Modul von mir umgesetzte PHP-Applikation hat mir stark geholfen. | | | |
| **Reflexion** | | | |
| XAMPP läuft zu meiner Zufriedenheit grösstenteils ohne Probleme. Ausserdem war ich froh, dass ich etwas weniger Zeit Zuhause für das Projekt aufwenden musste. | | | |

Tabelle 16 Arbeitsjournal - Woche 13

## Woche 14: KW46 2020

|  |  |  |  |
| --- | --- | --- | --- |
| **Tätigkeiten** | **Projektphase** | **Zeit SOLL** | **Zeit IST** |
| PHP-Api – DB-Anbindung | Realisierung | 1.5 | 2.5 |
| ***Total*** | | ***1.5*** | ***2.5*** |
| **Tagesablauf** | | | |
| Nachdem ich letzte Woche bereits den Grundstein der API mit der REST-Implementation umsetzen konnte, setzte ich diese Woche mit der DB-Anbindung um. | | | |
| **Wissensbeschaffung** | | | |
| Dokumentation PDO | | | |
| **Hilfestellungen** | | | |
| - | | | |
| **Reflexion** | | | |
| Aufgrund dem etwas komplizierterem Ansatz mit den Repositories habe ich mehr Zeit gebraucht als gedacht. Trotzdem bin ich froh dies Zuhause noch beendet zu haben. | | | |

Tabelle 17 Arbeitsjournal - Woche 14

## Woche 15: KW47 2020

|  |  |  |  |
| --- | --- | --- | --- |
| **Tätigkeiten** | **Projektphase** | **Zeit SOLL** | **Zeit IST** |
| PHP-Api – JWT-Authentifizierung | Realisierung | 1.5 | 1.5 |
| ***Total*** | | ***1.5*** | ***1.5*** |
| **Tagesablauf** | | | |
| Für die JWT-Authentifizierung habe ich die Library von «RobDWaller» installiert und eine Klasse zur Authentifizierung erstellt. | | | |
| **Wissensbeschaffung** | | | |
| Dokumentation SimpleJWT von RobDWaller | | | |
| **Hilfestellungen** | | | |
| - | | | |
| **Reflexion** | | | |
| Die JWT-Authentifizierung funktioniert mit der Library relativ einfach. Ich kam also gut voran. | | | |

Tabelle 18 Arbeitsjournal - Woche 15

## Woche 16: KW48 2020

|  |  |  |  |
| --- | --- | --- | --- |
| **Tätigkeiten** | **Projektphase** | **Zeit SOLL** | **Zeit IST** |
| PHP-Api – JWT-Authentifizierung | Realisierung | 0.5 | 1.0 |
| Umsetzung Kundenverwaltung | Realisierung | 2.0 | 2.0 |
| ***Total*** | | ***2.5*** | ***3.0*** |
| **Tagesablauf** | | | |
| Die Fertigstellung der JWT-Authentifizierung hat aufgrund der unterschiedlichen Rollen von Benutzern etwas länger gedauert. Anschliessend konnte ich mit der Kundenverwaltung im Frontend beginnen. | | | |
| **Wissensbeschaffung** | | | |
| - | | | |
| **Hilfestellungen** | | | |
| - | | | |
| **Reflexion** | | | |
| Insgesamt lief es die Woche ganz gut, ich habe wieder einiger Dinge Zuhause erledigt. | | | |

Tabelle 19 Arbeitsjournal - Woche 16

## Woche 17: KW49 2020

|  |  |  |  |
| --- | --- | --- | --- |
| **Tätigkeiten** | **Projektphase** | **Zeit SOLL** | **Zeit IST** |
| Umsetzung Kundenverwaltung | Realisierung | 1.0 | 1.0 |
| Umsetzung Projektverwaltung | Realisierung | 1.5 | 1.5 |
| ***Total*** | | ***2.5*** | ***2.5*** |
| **Tagesablauf** | | | |
| Nachdem ich in der Schule die Kundenverwaltung abgeschlossen hatte begann ich noch kurz mit der Projektverwaltung. Zuhause habe ich dann noch etwas weiter an der Projektverwaltung gearbeitet. | | | |
| **Wissensbeschaffung** | | | |
| - | | | |
| **Hilfestellungen** | | | |
| - | | | |
| **Reflexion** | | | |
| Die erste Verwaltung fertigstellen zu können war gut, nun sollte ich mich bei den weiteren etwas daran orientieren können. | | | |

Tabelle 20 Arbeitsjournal - Woche 17

## Woche 18: KW50 2020

|  |  |  |  |
| --- | --- | --- | --- |
| **Tätigkeiten** | **Projektphase** | **Zeit SOLL** | **Zeit IST** |
| Umsetzung Projektverwaltung | Realisierung | 1.5 | 1.5 |
| ***Total*** | | ***1.5*** | ***1.5*** |
| **Tagesablauf** | | | |
| Die in der letzten Woche begonnene Projektverwaltung konnte ich in der Schule abschliessen. | | | |
| **Wissensbeschaffung** | | | |
| - | | | |
| **Hilfestellungen** | | | |
| - | | | |
| **Reflexion** | | | |
| Auch die Projektverwaltung konnte ich ohne grosse Probleme abschliessen, es läuft also sehr gut. | | | |

Tabelle 21 Arbeitsjournal - Woche 18

## Woche 19: KW51 2020

|  |  |  |  |
| --- | --- | --- | --- |
| **Tätigkeiten** | **Projektphase** | **Zeit SOLL** | **Zeit IST** |
| Umsetzung Reportsverwaltung | Realisierung | 2.5 | 2.5 |
| ***Total*** | | ***2.5*** | ***2.5*** |
| **Tagesablauf** | | | |
| Die Umsetzung der Reportsverwaltung, welche ich in der Schule begonnen habe, lief relativ gut. Nach den Ferien muss ich nur noch kleine Dinge erledigen. | | | |
| **Wissensbeschaffung** | | | |
|  | | | |
| **Hilfestellungen** | | | |
|  | | | |
| **Reflexion** | | | |
| Den Zeitplan sollte ich auch bei der Reportsverwaltung einhalten können, nach den Ferien ist dafür nicht mehr viel zu tun. | | | |

Tabelle 22 Arbeitsjournal - Woche 19

## Woche 20: KW01 2021

|  |  |  |  |
| --- | --- | --- | --- |
| **Tätigkeiten** | **Projektphase** | **Zeit SOLL** | **Zeit IST** |
| Umsetzung Reportsverwaltung | Realisierung | 0.5 | 0.5 |
| Umsetzung Login & Benutzerverwaltung | Realisierung | 2.0 | 3.0 |
| ***Total*** | | ***2.5*** | ***3.5*** |
| **Tagesablauf** | | | |
| Die Reportsverwaltung im Frontend sowie im Backend konnte ich ohne Probleme abschliessen. Anschliessend habe ich mit dem Login & der Benutzerverwaltung im Frontend begonnen. Aufgrund der verschiedenen Rollen wird dies relativ kompliziert, weshalb ich auch mehr Zeit als geplant investiert habe. | | | |
| **Wissensbeschaffung** | | | |
| Projektziele und Anforderungen | | | |
| **Hilfestellungen** | | | |
| - | | | |
| **Reflexion** | | | |
| Es lief relativ gut. Für das Login und die Benutzverwaltung, was auch die Autorisierung umfasst, merkte ich aber, dass ich mehr Zeit als geplant brauche. Deshalb habe ich Zuhause etwas mehr Zeit investiert. | | | |

Tabelle 23 Arbeitsjournal - Woche 20

## Woche 21: KW02 2021

|  |  |  |  |
| --- | --- | --- | --- |
| **Tätigkeiten** | **Projektphase** | **Zeit SOLL** | **Zeit IST** |
| Umsetzung Login & Benutzerverwaltung | Realisierung | 1.5 | 3.5 |
| ***Total*** | | ***1.5*** | ***3.5*** |
| **Tagesablauf** | | | |
| Die Autorisierung war, wie bereits letzte Woche beschrieben, einiges aufwendiger als geplant. Aufgrund dessen habe ich Zuhause noch daran gearbeitet und das Ganze fertiggestellt. Die Programmierarbeiten sind somit abgeschlossen. | | | |
| **Wissensbeschaffung** | | | |
| - | | | |
| **Hilfestellungen** | | | |
| - | | | |
| **Reflexion** | | | |
| Ich bin froh die Programmierarbeiten nun wie geplant abgeschlossen zu haben. Nun muss noch die Doku, welche ich nebenbei immer geschrieben habe, fertiggestellt sowie die Tests an der Anwendung durchgeführt werden. | | | |

Tabelle 24 Arbeitsjournal - Woche 21

## Woche 22: KW03 2021

|  |  |  |  |
| --- | --- | --- | --- |
| **Tätigkeiten** | **Projektphase** | **Zeit SOLL** | **Zeit IST** |
| Tests durchführen | Realisierung | 2.5 | 2.0 |
| Bericht abschliessen | Realisierung | 0.5 | 2.5 |
| ***Total*** | | ***3.0*** | ***4.5*** |
| **Tagesablauf** | | | |
| Die Tests konnte ich durchführen, ohne grössere Bugs fixen zu müssen. Den Bericht abzuschliessen hat mir aber viel Zeit gekostet um alle Kriterien zu kontrollieren und den Abschluss zu schreiben. | | | |
| **Wissensbeschaffung** | | | |
| Vorlage IPA-Bericht und Kriterienkatalog | | | |
| **Hilfestellungen** | | | |
| - | | | |
| **Reflexion** | | | |
| Mit den Tests bin ich sehr zufrieden. Dank dem Testing während dem Programmieren hatte ich kaum Bugs und konnte dadurch etwas Zeit gegenüber dem Zeitplan einsparen.  Den Bericht abzuschliessen habe ich aber eindeutig unterschätzt. Alles abzuschliessen und zusammenzustellen hat viel Zeit gebraucht. | | | |

Tabelle 25 Arbeitsjournal - Woche 22

# Teil 2: Projektdokumentation

# Kurzfassung

## Kurze Ausgangssituation

Ein Reporting-System wird aufgrund der Verrechnung der Arbeit an die Kunden von jedem Dienstleistungsunternehmen benötigt. Darin sollen Kunden, Projekte und Reports erfasst werden können. Jedes Projekt ist einem Kunden zugeordnet, an welchen die Reports von diesem Projekt verrechnet werden.

Die gesamte Anwendung muss per Login gesichert sein, um unautorisierten Personen den Zugriff zu verhindern. Benutzer mit der Rolle «Administrator» sollen die Benutzer verwalten können.

## Umsetzung

### Vorbereitungen

Als Vorbereitung habe ich mich in HERMES eingearbeitet und die Dokumentationen dazu studiert. Anschliessend habe ich das Dokument für die Dokumentation dieser Probe-IPA vorbereitet und die ersten Punkte ergänzt. Nach der Beschreibung der Projektmethodik sowie der Organisation habe ich einen Zeitplan für die Umsetzung erstellt.

### Realisierung

Gemäss Zeitplan begann ich mit der Einrichtung aller notwendigen Services wie Apache, PHP, MySQL Datenbank und Vue.js. Nachdem alle Services liefen programmierte ich die PHP-API, welche über REST läuft und eine Authentifizierung per JWT unterstützt. Jede Datenbanktabelle wird in der API durch ein Model repräsentiert und ist über ein Repository an die Datenbank angebunden. Die Aufrufe der API werden über einen Controller empfangen und gesteuert.

Die Verwaltung der verschiedenen Datensätze wurde vorerst ohne Login umgesetzt, was im Nachhinein nicht die beste Lösung war. Bei der Umsetzung des Logins war ein grosser Aufwand damit verbunden im Nachhinein die Berechtigungen auf die verschiedenen Verwaltungen umzusetzen.

Während der Realisierung wurde laufend das Arbeitsjournal geführt sowie die Dokumentation ergänzt.

### Abschluss

Nach der Umsetzung des Projekts wurde die Applikationen durch manuelle Tests überprüft und Fehler behoben. Alle Tests wurden im Testprotokoll (Kapitel 11.2) dokumentiert.

Ebenfalls habe ich die gesamte Dokumentation überarbeitet und wo nötig ergänzt.

## Ergebnis

Mit dem Ergebnis der IPA bin ich zufrieden, auch wenn ich während der Umsetzung viele Punkte entdeckt habe, welche noch umgesetzt werden könnten. Da die Zeit dafür aber nicht gereicht hat, müssen diese ausserhalb der Probe-IPA umgesetzt werden.

# Initialisierung

## Analyse

### Studie; IST-Zustand

Da aktuell keine Form dieser Reports-Verwaltung vorhanden ist welche ersetzt oder erweitert werden soll, ist keine Studie des IST-Zustands möglich. Die Applikation basiert auf einer Idee von mir, ohne ein vorheriges Projekt ersetzen zu wollen.

Da jedes Dienstleistungsunternehmen aber ihre Leistungen und somit die benötigte Arbeitszeit an den Kunden verrechnen muss/will, ist ein Reporting-System aber sicher nötig. Ob dies momentan per Excel, von Hand oder mit einem anderen System erledigt wurde spielt bei der Entwicklung dieser Webapplikation keine Rolle.

### Projektziele

Als Projektziele wurde die Verwaltung von Kunden, Projekten und Reports festgelegt. Ausserdem sollen als Administrator gekennzeichnete Benutzer eine Benutzerverwaltung zur Verfügung haben.

Jeder Benutzer muss sich vor dem Zugriff auf die Applikation authentifizieren und bei jedem Zugriff auf das Frontend sowie auch auf die API muss diese Authentifikation überprüft werden.

Von jedem Datensatz-Typ soll eine Übersicht dieser Datensätze bestehen. Jeder dieser Datensatz kann bearbeitet oder gelöscht werden. Ausserdem kann auch ein neuer Datensatz erstellt und gespeichert werden. Bei jedem Speichern wird eine Validierung der Eingaben durchgeführt, um falsche Eingaben zu verhindern.

### Anforderungen

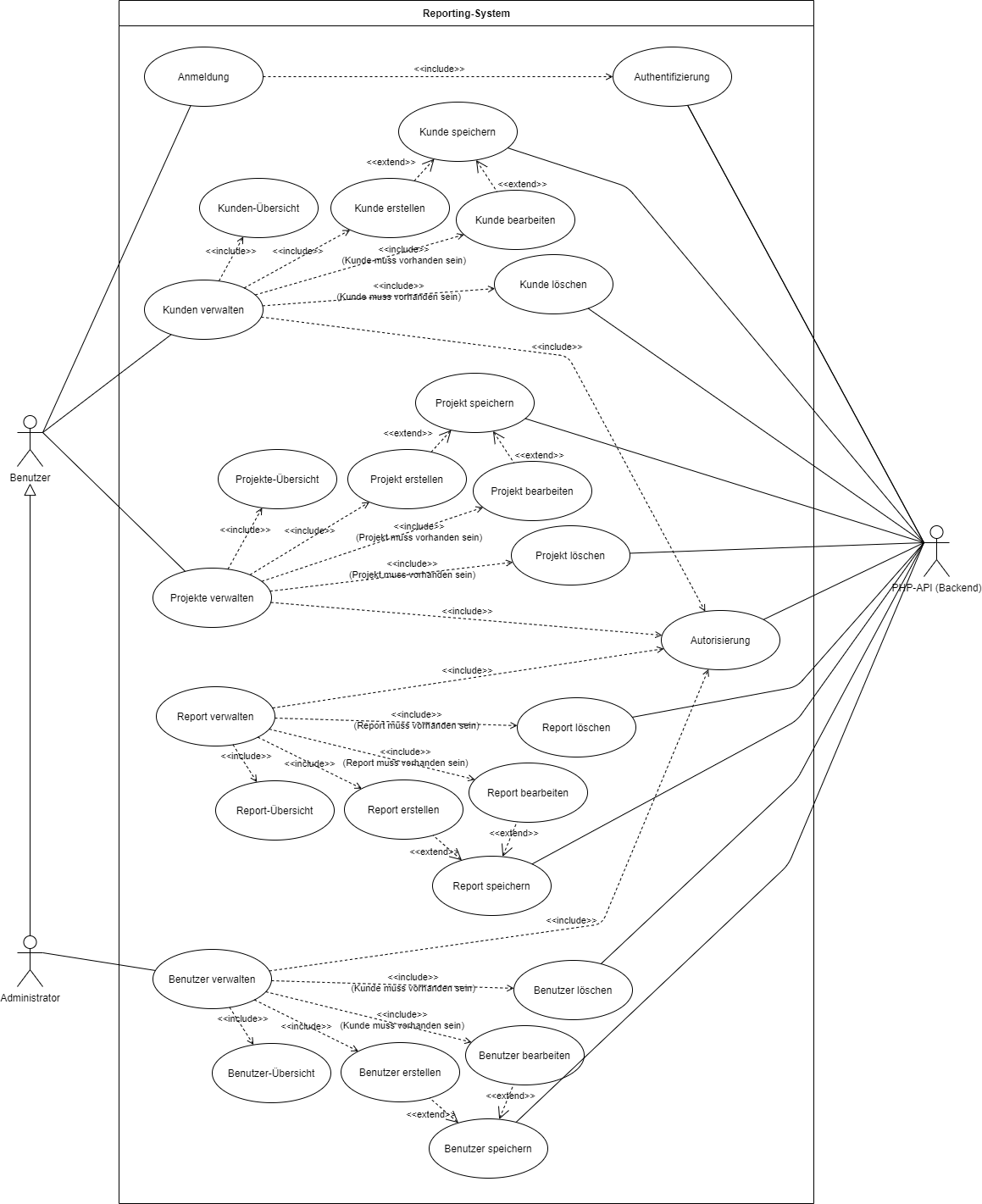


Abbildung 1 Use Case Diagramm

#### Anmeldung & Authentifizierung

Mittels eines Login-Formulars werden Benutzername und Passwort an die API gesendet. Diese sucht diesen Benutzer mittels Benutzername in der DB. Beim Benutzernamen wird nicht zwischen Gross- und Kleinschreibung unterschieden. Falls nichts gefunden werden kann, werden alle Felder im Frontend als ungültig gekennzeichnet. Wenn der Benutzer gefunden wird, die Passwortvalidierung aber fehlschlägt, wird nur das Passwort als ungültig gekennzeichnet. Wenn alle Validierungen erfolgreich sind, sprich ein Benutzer gefunden wird und das Passwort korrekt ist, wird mittels JWT ein Token erstellt. Dazu wird ein in der API gespeicherter Secret-Key verwendet. Dieser wird anschliessend an das Frontend retourniert und dort im LocalStorage gespeichert. Bei erfolgreichem Login wird man weitergeleitet.

#### Autorisierung

Bei jedem Aufruf des Frontends wird das im LocalStorage gespeicherte Token an die API gesendet und validiert. Zurückgegeben wird ein Objekt mit den Attributen ob die Validierung erfolgreich war und ob der eingeloggte Benutzer ein Admin ist oder nicht. Dieses wird für den Aufruf der Seiten im Frontend benutzt.

#### Übersicht Datensätze

Die Übersicht bei allen Typen von Datensätzen funktioniert in etwa gleich. Dabei wird eine Tabelle mit den jeweiligen Attributen dieses Datensatzes mit den Datensätzen von diesem Typ befüllt. Jeder Datensatz hat einen Button zum Bearbeiten sowie einen zum Löschen.

Ausserdem sollen die Datensätze durchsuchbar sein.

#### Datensatz erstellen

Für jeden Typ Datensatz gibt es ein Formular zur Erstellung eines neuen Datensatzes. Darin sollen alle Attribute dieses Datensatzes angegeben werden können. Beim Klick auf Speichern wird die API aufgerufen und der Datensatz wird in der Datenbank gespeichert. Bei ungültigen Eingaben sollen diese im Frontend angezeigt werden und der Datensatz darf nicht gespeichert werden.

#### Datensatz bearbeiten

Die Funktion «Datensatz bearbeiten» funktioniert praktisch gleich wie «Datensatz erstellen». Allerdings wird das Formular mit allen Attributen aus dem zu bearbeitenden Datensatz vorausgefüllt und kann bearbeitet werden. Beim Speichern werden ebenfalls alle Eingaben geprüft und der Datensatz wird bei erfolgreicher Prüfung in der Datenbank gespeichert. Hier muss darauf geachtet werden, dass kein neuer Datensatz erstellt, sondern der bisherige geupdatet wird.

#### Datensatz löschen

Beim Klick auf den Lösch-Button in der Übersicht wird ein Request an die API gesendet. Die API soll anschliessend den Datensatz aus der Datenbank löschen. Im Frontend muss der Datensatz aus der Übersicht entfernt werden.

## Varianten - Suche

Bei der Suche von Datensätzen musste ich mich entscheiden, wie ich dies lösen will. Es gibt die Möglichkeit bei jeder Sucheingabe einen API-Aufruf auszulösen oder die Datensätze direkt per JavaScript zu filtern.

### Variante 1 – Suche per API

Bei der Eingabe eines Suchbegriffes wird eine Anfrage an die API gesendet. Als Parameter muss dabei selbstverständlich der Suchbegriff mitgegeben werden. Die API sucht anschliessend direkt in der Datenbank mittels SQL-Query nach passenden Datensätzen und retourniert diese.

### Variante 2 – Suche per JavaScript

Die Suche von Datensätzen wird ohne API-Aufruf durchgeführt und kann direkt über JavaScript abgewickelt werden. Dabei werden die bereits im Frontend vorhandenen Daten per JavaScript gefiltert und in der Übersicht ausgegeben.

### Entscheidungsmatrix

|  |  |  |  |
| --- | --- | --- | --- |
| Kriterien | Gewichtung | Variante 1 | Variante 2 |
| Ladezeit | 6 | 4 | 6 |
| Anzahl SQL-Queries | 4 | 5 | 10 |
| Benutzerfreundlichkeit | 8 | 7 | 8 |
| Implementationsaufwand | 6 | 6 | 5 |
| Summe |  | 136 | 170 |

Tabelle 26 Entscheidungsmatrix

### Gewählte Variante

Durch die Entscheidungsmatrix bin ich klar auf den Entschluss gekommen die Variante 2 zu wählen.

Die Ladezeit ist durch den Verzicht auf einen API-Aufruf um einiges geringer und die Ergebnisse können somit schneller angezeigt werden. Ausserdem müssen weniger Zugriffe auf die Datenbank durchgeführt werden. Bei einem Live-Update der Suche kann es schnell zu sehr vielen SQL-Queries kommen. Die Benutzerfreundlichkeit sowie den Implementationsaufwand bewerte ich bei beiden fast identisch.

# Konzept

## Systementwurf

### API

Um die Notwendigkeit eines Webhostings zu vermeiden wird die API per XAMPP auf dem localhost laufen. Die API benötigt einen MySQL- sowie einen Apache-Server mit PHP-Compiler.

Ausserdem muss die API per Composer aufgebaut werden, um Librarys einbinden zu können und den Aufruf von Klassen über Namespaces zu gewährleisten. Es werden also alle Dateien automatisch via Composer eingebunden und können somit sehr leicht über den Namespace und den Klassennamen gefunden und aufgerufen werden.

Die Authentifizierung wird mit der Library «ReallySimpleJWT» von RobDWaller umgesetzt (<https://github.com/RobDWaller/ReallySimpleJWT>). Der Secret-Key wie auch andere benötigten Werte werden in der für die Authentifizierung angelegten Klasse als Konstanten gespeichert.

Der Einstiegspunkt der API ist eine «index.php»-Datei, welche von Apache-Servern automatisch als auszuführende Datei gefunden wird. Diese Datei soll den Composer-Autoloader einbinden, die übermittelten Daten an den Controller weitergeben und dessen Rückgabewerte ausgeben.

Im Controller wird als erstes die Authentifizierung durchgeführt, ausgenommen es handelt sich um einen Login-Versuch. Anschliessend wir das Management über den dafür vorgesehenen Parameter bestimmt und aufgerufen. Via REST-Anforderungsmethode wird der Typ der Anfrage bestimmt, beispielsweise ob in diesem Management etwas gespeichert oder gelöscht werden soll.

Mit den Models und den Repositories wird anschliessend die gewünschte Aufgabe durchgeführt. Bei Fehler wird stets mit REST und definierten Fehlercodes Rückmeldung an den Client gegeben. Falls alles reibungslos läuft werden die angeforderten Daten als JSON ausgegeben.

### Frontend

Das Frontend wird basierend auf einem frei im Internet verfügbaren Template von Creative Tim aufgebaut. Die Lizenz von Creative Tim wird dabei eingehalten. Genauer wird das Template «Vue Black Dashboard» verwendet (<https://www.creative-tim.com/product/vue-black-dashboard>).

Der Zugriff auf die API funktioniert über die Library ‘AXIOS’, mit welcher alle möglichen HTTP-Requests durchgeführt werden können. Das Mapping auf die korrekte URL der API funktioniert über eine Config, somit muss die URL bei einer Änderung nicht überall im Code angepasst werden.

### Datenbank

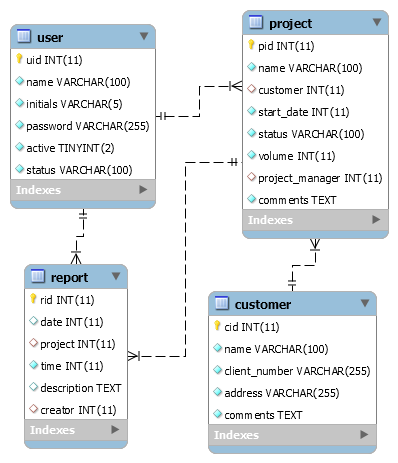


Abbildung 2 Datenbank-Modell

## Testkonzept

### Testziele

Mit den durchgeführten Tests soll eine fehlerfreie Applikation gewährleistet werden. Ausserdem sollen alle in der Aufgabenstellung als benötigt beschrieben Funktionen einwandfrei funktionieren.

Das Ziel sind bei der Programmierung nicht festgestellte Bugs zu beheben und so böse Überraschungen für den Nutzer vorzubeugen.

### Teststrategie

Aufgrund dem verhältnismässig zu hohem Aufwand, welcher automatisierte Tests generieren würden, werden nur manuelle Tests durch mich durchgeführt.

Die Tests werden der Aufgabenstellung entnommen und sollen so eine funktionsfähige Anwendung, die allen Anforderungen entspricht, gewährleisten.

Bei Fehlern wird der Bug behoben und der Test erneut durchgeführt. Dies wird solange wiederholt, bis das Problem behoben ist oder keine Zeit mehr zur Verfügung steht.

### Testrahmen

Um die Tests durchführen zu können muss der MySQL- sowie der Apache-Server mit PHP-Interpreter von XAMPP laufen. Ausserdem muss die API am korrekten Ort gespeichert sein, um über localhost erreichbar zu sein. In der «vue.config.js» muss der korrekte Pfad zu der API hinterlegt sein.

Ausserdem muss der NPM-Server ausgeführt werden und die Vue-Applikation muss über den Webbrowser erreichbar sein. Ebenfalls muss die Datenbank auf dem MySQL-Server mit den dafür bereitgestellten Skripts erstellt worden sein.

### Testinfrastruktur

Alle Tests werden auf einem Windows 10 Rechner durchgeführt. Die installierten und benötigten Programme sind die folgenden:

* XAMPP: Version 7.4.12
* Node.js: Version 12.18.4
* Chrome: Build 87.0.4280.141

Durchgeführt werden alle Tests direkt in der Webapplikation durch mich als User. Die Datenbank wird via SQL-Skript erstellt. Ausserdem wird mein Benutzer manuell der DB hinzugefügt, um ein erstes Login in der Anwendung zu ermöglichen.

### Testfälle

|  |  |
| --- | --- |
| **ID / Bezeichnung** | *T-001 / Login* |
| **Beschreibung** | Das Login mit dem im Voraus manuell in der DB erstellten Benutzer soll getestet werden. Dabei sollen auch Versuche mit einem falschen Benutzernamen und/oder einem falschen Passwort durchgeführt werden, um die Validierung zu testen. |
| **Testvoraussetzung** | In der DB muss ein Benutzer mit bekanntem und gehashtem Passwort vorhanden sein. |
| **Testschritte** | 1. Login-Seite aufrufen. 2. Falsche Benutzernamen eingeben -> Absenden 3. Falsches Passwort eingeben -> Absenden 4. Korrekte Logindaten eingeben -> Absenden |
| **Erwartetes Ergebnis** | Nach Punkt 2 soll ein Validierungsfehler für Benutzername und Passwort angezeigt werden, da der Benutzer nicht gefunden werden konnte.  Nach Punkt 3 soll ein Validierungsfehler für das Passwort angezeigt werden, da das Passwort falsch war.  Nach Punkt 4 soll man eingeloggt und weitergeleitet werden. |

Tabelle 27 Konzept Testfall T-001

|  |  |
| --- | --- |
| **ID / Bezeichnung** | *T-002 / Benutzer erstellen* |
| **Beschreibung** | Es soll ein neuer Benutzer über das Frontend erstellt werden. |
| **Testvoraussetzung** | Um diesen Test auszuführen muss ein Benutzer mit der Rolle «Administrator» eingeloggt sein. |
| **Testschritte** | 1. Benutzer-Verwaltung aufrufen. 2. Formular zur Erstellung eines Benutzers öffnen. 3. Formular leer oder mit falschen Angaben absenden. 4. Formular mit korrekten Angaben absenden. |
| **Erwartetes Ergebnis** | Nach Punkt 3 soll ein Validierungsfehler für die falsch eingegebenen Daten angezeigt werden und der Benutzer darf nicht gespeichert worden sein.  Nach Punkt 4 soll der Benutzer in der DB gespeichert werden und man soll zurück zur Übersicht geleitet werden, wo dieser Benutzer auch gleich angezeigt wird. |

Tabelle 28 Konzept Testfall T-002

|  |  |
| --- | --- |
| **ID / Bezeichnung** | *T-003 / Kunde bearbeiten* |
| **Beschreibung** | Ein bestehender Kunde wird bearbeitet und gespeichert. |
| **Testvoraussetzung** | Der Benutzer muss eingeloggt sein und es muss ein Kunde existieren. |
| **Testschritte** | 1. Kunden-Verwaltung aufrufen. 2. Kunde aus Übersicht wählen und bearbeiten. 3. Ungültige Werte eingeben und absenden. 4. Gültige Werte eingeben und absenden. |
| **Erwartetes Ergebnis** | Nach Punkt 3 soll ein Validierungsfehler für die falsch eingegebenen Daten angezeigt werden und der Benutzer darf nicht gespeichert worden sein.  Nach Punkt 4 soll der Benutzer in der DB aktualisiert werden und man soll zurück zur Übersicht geleitet werden, wo der aktualisierte Benutzer auch gleich angezeigt wird. |

Tabelle 29 Konzept Testfall T-003

|  |  |
| --- | --- |
| **ID / Bezeichnung** | *T-004 / Projekt löschen* |
| **Beschreibung** | Es soll ein Projekt aus der DB über das Frontend gelöscht werden. |
| **Testvoraussetzung** | Der Benutzer muss eingeloggt sein und es muss ein Projekt existieren. |
| **Testschritte** | 1. Projekt-Verwaltung aufrufen. 2. Projekt aus Übersicht wählen und Löschen-Button klicken. |
| **Erwartetes Ergebnis** | Das Projekt soll nach dem Klick auf den Löschen-Button per API aus der Datenbank gelöscht werden. Ausserdem muss es aus der Übersicht entfernt und nicht mehr angezeigt werden. |

Tabelle 30 Konzept Testfall T-004

|  |  |
| --- | --- |
| **ID / Bezeichnung** | *T-005 / Logout* |
| **Beschreibung** | Es soll getestet werden, ob das Logout funktioniert. |
| **Testvoraussetzung** | Der Benutzer muss eingeloggt sein. |
| **Testschritte** | 1. Logout-Button klicken. |
| **Erwartetes Ergebnis** | Alle mit dem Authentisieren verbundenen Infos sollen gelöscht werden. Ebenfalls soll man zum Login weitergeleitet werden. Ein erneuter direkter Aufruf einer Verwaltung muss abgelehnt werden. |

Tabelle 31 Konzept Testfall T-005

|  |  |
| --- | --- |
| **ID / Bezeichnung** | *T-006 / API-Aufruf ohne Token* |
| **Beschreibung** | Es soll die API direkt über den Browser aufgerufen werden. Dabei werden keine Authentisierungs-Infos mitgesendet. |
| **Testvoraussetzung** | Die direkte URL zur API muss bekannt sein und es dürfen keine Authentisierungs-Infos mitgesendet werden. |
| **Testschritte** | 1. API über Browser aufrufen. |
| **Erwartetes Ergebnis** | Es dürfen keine Informationen ausgegeben werden. Es muss also eine Meldung in der Art «Kein Zugriff» ausgegeben werden. |

Tabelle 32 Konzept Testfall T-006

# Realisierung

## System realisieren

### API

Die Umsetzung der Reports-Verwaltung habe ich mit der Umsetzung der API begonnen. Dabei ist mir wichtig eine in Namespaces aufgeteilte Klassen zu verwenden, um einen guten Überblick zu gewährleisten. Die API setze ich in einem separaten Ordner mit dem Namen «api» im Projektverzeichnis, welches im «htdocs»-Ordner von XAMPP liegt, um. Die API muss im «htdocs»-Ordner von XAMPP liegen, weil nur dies das Verzeichnis des Apache-Servers von XAMPP ist und somit nur Dateien in diesem Ordner über localhost aufgerufen und ausgeführt werden können.

Als allererstes habe ich in der API eine index.php Datei angelegt, welche den Startpunkt der API festlegt. Anschliessend habe ich mich bereits um die Composer-Installation gekümmert, um die Klassen per Namespace laden zu können. Dafür habe ich mich an die Anleitung gehalten und den Composer mittels Kommandozeile heruntergeladen. Folgende Commands müssen dafür ausgeführt werden, die Prüfsumme wechselt aber immer wieder:

php -r "copy('https://getcomposer.org/installer', 'composer-setup.php');"

php -r "if (hash\_file('sha384', 'composer-setup.php') === '756890a4488ce9024fc62c56153228907f1545c228516cbf63f885e036d37e9a59d27d63f46af1d4d07ee0f76181c7d3') { echo 'Installer verified'; } else { echo 'Installer corrupt'; unlink('composer-setup.php'); } echo PHP\_EOL;"

php composer-setup.php

php -r "unlink('composer-setup.php');"

Anschliessend kann mit dem Command «php composer.phar init» eine Composer-Installation erstellt werden. Dabei müssen Angaben zu Autor, Paketname und sonstigen Einstellungen gemacht werden. In der Config-Datei vom Composer, der «composer.json», habe ich den Subordner «Classes» als Startpunkt für den Namespace «ProbeIPA\Classes» angegeben. Mit der Ausführung des Befehls «php composer.phar install» werden im Subordner «vendor» automatisch Autoload-Informationen generiert, welche ich in der Datei «index.php» eingebunden habe. Dadurch können alle Dateien im Ordner «Classes» direkt über den Namespace und den Klassennamen aufgerufen werden und es muss nicht jede Datei manuell geladen werden.

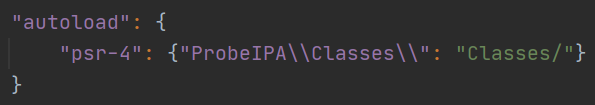
 

Abbildung 3 Angaben für Namespace und Einbinden von Autoload

Danach habe ich auch bereits die REST-Klasse angelegt, welche für das Setzen von HTTP-Headern sowie das Umwandeln von PHP-Objekten (Arrays, Models etc.) zu JSON verantwortlich ist.

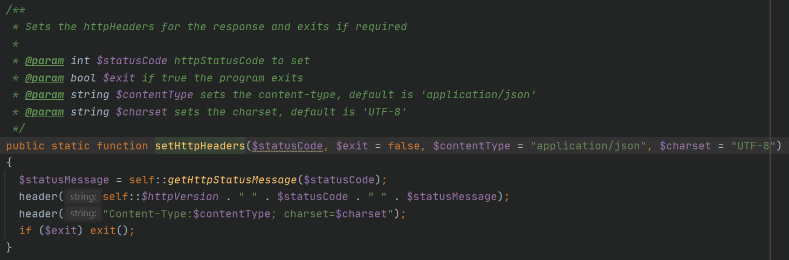
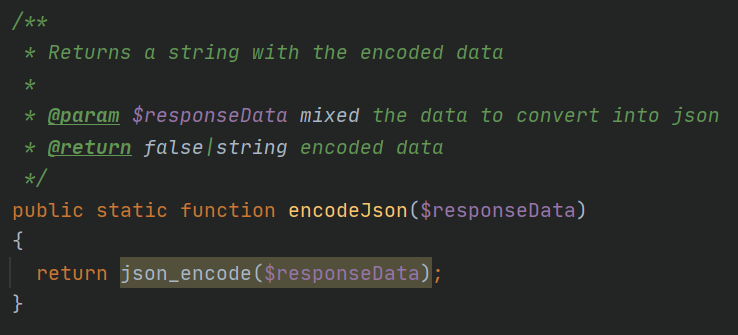
 

Abbildung 4 REST-Klasse

Ebenfalls eine sehr wichtige Komponente der API ist der Controller, welchen ich gleich darauffolgend umgesetzt habe. Der Controller ist wie die REST-Klasse auch im Namespace «ProbeIPA\Classes» und kann dadurch extrem einfach aufgerufen werden. Der Controller ist sozusagen das Management für Aufrufe der API. Er empfängt die Daten, führt den gewünschten Befehl aus und gibt wenn nötig die angeforderten Dinge zurück. Der Aufruf des Controllers und die Ausgabe durch Rest in der «index.php» sieht wie folgend aus:



Abbildung 5 Aufruf Controller und Ausgabe mit REST

Welche Verwaltung im Controller aufgerufen werden soll wird über den Parameter «management» festgelegt. Um diesen jeweils einfacher anzugeben habe ich in der «.htaccess»-Datei Rewrites erstellt. Der Pfad hinter der URL zur API wird nun zum Management umgeschrieben. Beispielsweise wird «/api/customer/» zu «/api?management=customer» umgeschrieben.

Um auf die Datenbank zuzugreifen und eine schön aufgeräumte Verwaltung der Datensätze zuzulassen habe ich mich für Models und Repositories entschieden. Die Repositories basieren auf einem Interface welches die Methoden «save», «insert», «update», «delete», «findAll» und «findById» vorschreibt. Die Methode «save» überprüft, ob das Model bereits in der Datenbank gespeichert ist. Falls schon wird «update» aufgerufen, ansonsten «insert». Die Methode «update» aktualisiert den Datensatz in der DB mit der im Model gespeicherten ID. Bei «insert» wird ein neuer Datensatz erstellt und das Model mit der Autoincrement-ID angereichert. Bei «delete» wird ein Datensatz anhand der ID gelöscht. «findAll» retourniert ein Array mit allen Datensätzen als Models, «findByID» hingegen retourniert den Datensatz mit der übergebenen ID als Model.

Alle Repositories erstellen eine Instanz der Klasse «DB», welche mit PDO funktioniert und durch eigene Methoden etwas leichter zu gebrauchen ist als jedesmal direkt PDO aufzurufen. Die Repositories befinden sich im Namespace «ProbeIPA\Classes\Repositories».

Die Models, von welchen bereits im Bezug auf die Repositories die Rede war, basieren auf den Datenbank-Tabellen. Die Attribute zeigen 1:1 die Spalten in der DB auf. Die Attribute sind alle «protected» und werden über Getter und Setter verwaltet. Ausserdem gibt es noch die Methoden «createFromArray» sowie «validate». Erstere ist «public» und dient zur Erstellung des Models aus Datenbank-Einträgen oder übergebenen Daten beim API-Aufruf. Die Methode «validate» prüft die übergebenen Daten und gibt bei Fehlern mittels REST den Fehlercode «420» zurück sowie die Felder, welche nicht erfolgreich geprüft wurden.

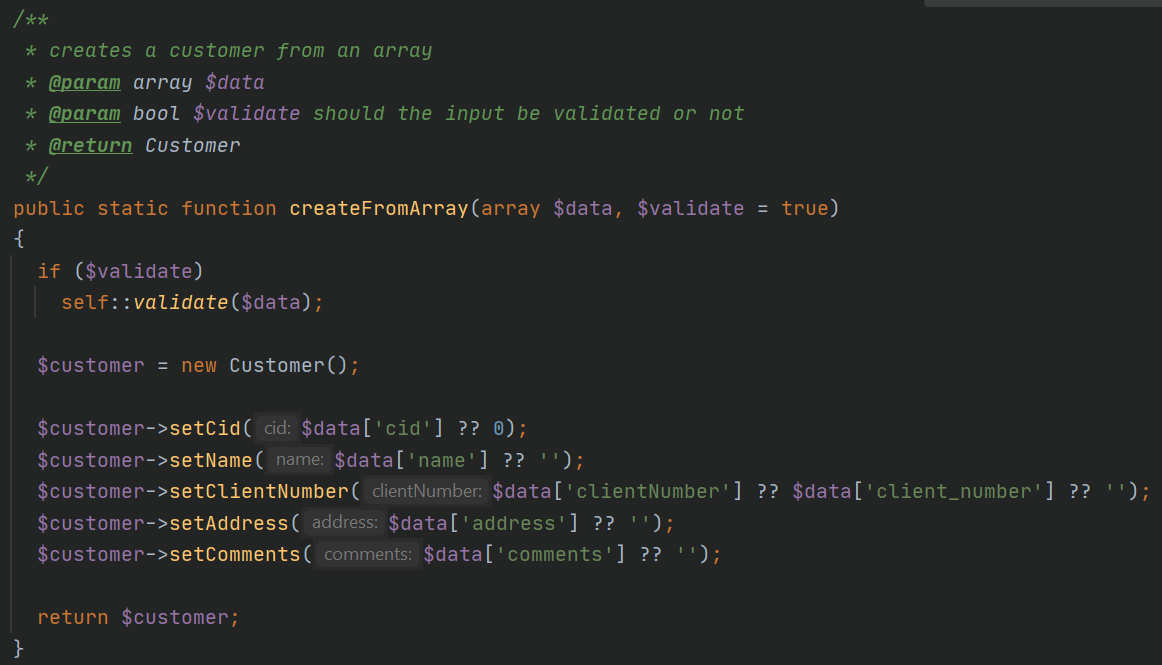


Abbildung 6 Beispiel Methode "createFromArray"

Für die Authentifizierung habe ich die Library «rbdwllr/reallysimplejwt» mittels Composer installiert. Dafür habe ich das Kommando «php composer.phar require rbdwllr/reallysimplejwt» in der Kommandozeile ausgeführt. Anschliessend habe ich die Klasse «Authentication» erstellt, welche als Konstanten den Secret-Key, die Lebensdauer sowie den Austeller beinhaltet. Diese Informationen werden zur Erstellung eines JWT-Tokens gebraucht. Alle Methoden in dieser Klasse sind statisch, es muss also keine Instanz erstellt werden. Die Funktionen der Methoden sind einerseits einen Token zu authentisieren sowie einen Benutzer zu authentifizieren und einen Token auszustellen.

### Frontend

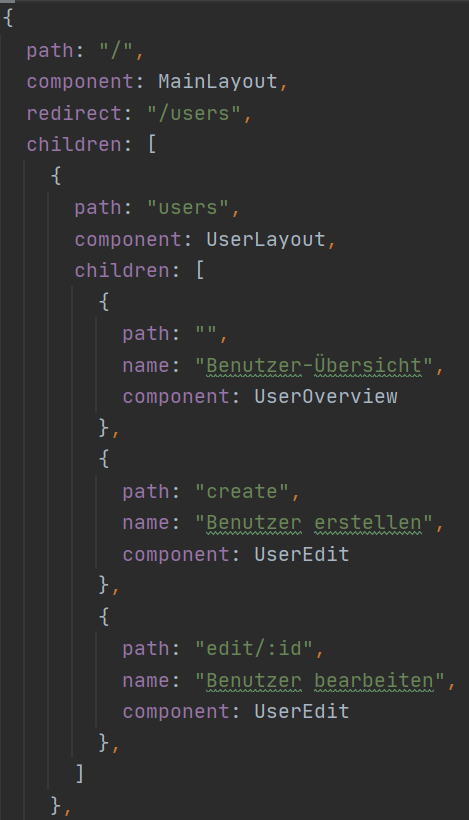
Als allererstes habe ich das bereits heruntergeladene Template von CreativeTim etwas aufgeräumt und nicht benötigte Komponenten gelöscht. Danach habe ich den Router aufgeräumt und mir den Aufbau etwas genauer angeschaut. Nachdem dies getan war startete ich mit der Kundenverwaltung. Jede Verwaltung wird im Router als Kind vom Hauptlayout angelegt, damit Navigation etc. nicht in jeder Datei separat angelegt werden müssen.

Abbildung 7 Router-Konfiguration

Für die Kundenverwaltung habe ich eine Seite für die Übersicht sowie eine für das Bearbeiten und Erstellen von Datensätzen erstellt. Wie die Router-Konfiguration für ein Management aussieht ist auf der rechten Seite abgebildet.

Bei der Übersicht wird eine Tabelle mit Daten aus der API aufgebaut. Konkret werden diese in der «mounted»-Methode geladen und gespeichert. Jeder Datensatz umfasst eine Zeile mit den Attributen sowie einem Bearbeiten- und einem Löschen-Knopf. Beim Klick auf den Löschen-Knopf wird die API aufgerufen und der Datensatz wird aus der DB sowie aus der Tabelle gelöscht. Beim Klick auf den Bearbeiten-Button wird man auf die Seite zum Bearbeiten weitergeleitet und die ID wird in der URL mitgegeben.

Auf der Seite, welche zum Bearbeiten sowie zum Erstellen benutzt wird, wird beim Aufruf geprüft, ob eine ID mitgegeben wurde. Falls ja wird der Datensatz mit dieser ID mithilfe der API geladen und kann bearbeitet werden. Falls nein wird das leere Formular angezeigt und beim Speichern wird ein neuer Kunde erstellt.

Jedes Management im Frontend ist sehr sehr ähnlich aufgebaut. Und trotzdem mussten jeweils einige Dinge angepasst werden, welche mich viel Zeit gekostet haben. Dazu gehören beispielsweise Formatierungen in der Übersicht oder die Eingabefelder im Formular. Ich sehe aber keinen Grund alle Verwaltung hier einzeln zu beschreiben.

Zum Abschluss des Projektes musste noch das Login sowie die Benutzerverwaltung umgesetzt werden. Die Benutzerverwaltung ist wieder sehr ähnlich wie die anderen Verwaltungen aufgebaut. Hier musste aber darauf geachtet werden, dass der eigenen Benutzer nicht bearbeitet werden kann und nur Administratoren darauf Zugriff zu haben.

Für das Login habe ich eine neue Seite mit Anmeldeformular erstellt. Die dort eingegebenen Daten werden an die API gesendet und dort überprüft. Falls erfolgreich wird ein JWT-Token zurückgegeben und im sessionStorage gespeichert. Anschliessend wird man auf das erste Management weitergeleitet. Bei jedem Aufruf einer Seite oder der API wird nun dieses JWT-Token per API überprüft. Nur bei erfolgreicher Überprüfung wird der Zugang gewährt.

Kopfzerbrechen haben mir die verschiedenen Rechte von Benutzern gebracht. Schlussendlich kam ich zur Lösung bei der Antwort der API mitzusenden, ob der Benutzer ein Administrator ist oder nicht. Im JavaScript werden dann die nur für Administratoren erlaubten Seiten explizit ausgewiesen und können so überprüft werden.

## Testprotokoll

### Testfall T-001

|  |  |
| --- | --- |
| **ID / Bezeichnung** | *T-001 / Login* |
| **Beschreibung** | Das Login mit dem im Voraus manuell in der DB erstellten Benutzer soll getestet werden. Dabei sollen auch Versuche mit einem falschen Benutzernamen und/oder einem falschen Passwort durchgeführt werden, um die Validierung zu testen. |
| **Testvoraussetzung** | In der DB muss ein Benutzer mit bekanntem und gehashtem Passwort vorhanden sein. |
| **Testschritte** | 1. Login-Seite aufrufen. 2. Falsche Benutzernamen eingeben -> Absenden 3. Falsches Passwort eingeben -> Absenden 4. Korrekte Logindaten eingeben -> Absenden |
| **Erwartetes Ergebnis** | Nach Punkt 2 soll ein Validierungsfehler für Benutzername und Passwort angezeigt werden, da der Benutzer nicht gefunden werden konnte.  Nach Punkt 3 soll ein Validierungsfehler für das Passwort angezeigt werden, da das Passwort falsch war.  Nach Punkt 4 soll man eingeloggt und weitergeleitet werden. |

| **Tester (Testperson)** | Florian Leimer |
| --- | --- |
| **Datum Testdurchführung** | 19.01.2021 |
| **Testergebnis** | Beim ersten Versuch musste ich feststellen, dass bei einem nicht vorhandenen Benutzername keine Eingabefehler angezeigt wird und man weitergeleitet wird. Nach der Behebung dieses Fehlers lief alles reibungslos. |
| **Fehlerbeschreibung** | Beim Abrufen der Ergebnisse von der Datenbank wird bei keinem Treffer false zurückgegeben. Dies hat beim Umwandeln zum Model einen Fehler gegeben, wodurch alles nicht mehr funktioniert hat. |

Tabelle 33 Testfall T-001

### Testfall T-002

|  |  |
| --- | --- |
| **ID / Bezeichnung** | *T-002 / Benutzer erstellen* |
| **Beschreibung** | Es soll ein neuer Benutzer über das Frontend erstellt werden. |
| **Testvoraussetzung** | Um diesen Test auszuführen muss ein Benutzer mit der Rolle «Administrator» eingeloggt sein. |
| **Testschritte** | 1. Benutzer-Verwaltung aufrufen. 2. Formular zur Erstellung eines Benutzers öffnen. 3. Formular leer oder mit falschen Angaben absenden. 4. Formular mit korrekten Angaben absenden. |
| **Erwartetes Ergebnis** | Nach Punkt 3 soll ein Validierungsfehler für die falsch eingegebenen Daten angezeigt werden und der Benutzer darf nicht gespeichert worden sein.  Nach Punkt 4 soll der Benutzer in der DB gespeichert werden und man soll zurück zur Übersicht geleitet werden, wo dieser Benutzer auch gleich angezeigt wird. |

| **Tester (Testperson)** | Florian Leimer |
| --- | --- |
| **Datum Testdurchführung** | 19.01.2021 |
| **Testergebnis** | Glücklicherweise hat alles sofort funktioniert. Die Felder wurden wie gewünscht validiert und auch korrekt markiert. Ebenfalls wurde der Benutzer bei einer Eingabe ohne Fehler erfolgreich in der Datenbank gespeichert. |
| **Fehlerbeschreibung** | - |

Tabelle 34 Testfall T-002

### Testfall T-003

|  |  |
| --- | --- |
| **ID / Bezeichnung** | *T-003 / Kunde bearbeiten* |
| **Beschreibung** | Ein bestehender Kunde wird bearbeitet und gespeichert. |
| **Testvoraussetzung** | Der Benutzer muss eingeloggt sein und es muss ein Kunde existieren. |
| **Testschritte** | 1. Kunden-Verwaltung aufrufen. 2. Kunde aus Übersicht wählen und bearbeiten. 3. Ungültige Werte eingeben und absenden. 4. Gültige Werte eingeben und absenden. |
| **Erwartetes Ergebnis** | Nach Punkt 3 soll ein Validierungsfehler für die falsch eingegebenen Daten angezeigt werden und der Benutzer darf nicht gespeichert worden sein.  Nach Punkt 4 soll der Benutzer in der DB aktualisiert werden und man soll zurück zur Übersicht geleitet werden, wo der aktualisierte Benutzer auch gleich angezeigt wird. |

| **Tester (Testperson)** | Florian Leimer |
| --- | --- |
| **Datum Testdurchführung** | 19.01.2021 |
| **Testergebnis** | Der Kunde wurde in der Datenbank nicht aktualisiert, sondern als neuer gespeichert. Ansonsten funktioniert alles reibungslos. |
| **Fehlerbeschreibung** | In der API wurde in der «save»-Methode im Repository nicht korrekt überprüft, ob der Benutzer bereits existiert. Die eindeutige ID wurde also nicht korrekt erkannt und der Benutzer wurde deshalb neu gespeichert. |

Tabelle 35 Testfall T-003

### Testfall T-004

|  |  |
| --- | --- |
| **ID / Bezeichnung** | *T-004 / Projekt löschen* |
| **Beschreibung** | Es soll ein Projekt aus der DB über das Frontend gelöscht werden. |
| **Testvoraussetzung** | Der Benutzer muss eingeloggt sein und es muss ein Projekt existieren. |
| **Testschritte** | 1. Projekt-Verwaltung aufrufen. 2. Projekt aus Übersicht wählen und Löschen-Button klicken. |
| **Erwartetes Ergebnis** | Das Projekt soll nach dem Klick auf den Löschen-Button per API aus der Datenbank gelöscht werden. Ausserdem muss es aus der Übersicht entfernt und nicht mehr angezeigt werden. |

| **Tester (Testperson)** | Florian Leimer |
| --- | --- |
| **Datum Testdurchführung** | 19.01.2021 |
| **Testergebnis** | Das Löschen aus der DB funktioniert gut. Allerdings wird trotzdem das Projekt weiterhin im Frontend in der Übersicht angezeigt. |
| **Fehlerbeschreibung** | Das Projekt wurde zwar durch die API korrekt aus der Datenbank gelöscht, allerdings funktionierte das Filtern dieses Datensatzes aus der Ergebnisliste im Frontend noch nicht korrekt. |

Tabelle 36 Testfall T-004

### Testfall T-005

|  |  |
| --- | --- |
| **ID / Bezeichnung** | *T-005 / Logout* |
| **Beschreibung** | Es soll getestet werden, ob das Logout funktioniert. |
| **Testvoraussetzung** | Der Benutzer muss eingeloggt sein. |
| **Testschritte** | 1. Logout-Button klicken. |
| **Erwartetes Ergebnis** | Alle mit dem Authentisieren verbundenen Infos sollen gelöscht werden. Ebenfalls soll man zum Login weitergeleitet werden. Ein erneuter direkter Aufruf einer Verwaltung muss abgelehnt werden. |

| **Tester (Testperson)** | Florian Leimer |
| --- | --- |
| **Datum Testdurchführung** | 20.01.2021 |
| **Testergebnis** | Das Logout funktioniert korrekt. Jeder Aufruf der API oder einer nicht öffentlichen Seite wird danach geblockt. |
| **Fehlerbeschreibung** | - |

Tabelle 37 Testfall T-005

### Testfall T-006

|  |  |
| --- | --- |
| **ID / Bezeichnung** | *T-006 / API-Aufruf ohne Token* |
| **Beschreibung** | Es soll die API direkt über den Browser aufgerufen werden. Dabei werden keine Authentisierungs-Infos mitgesendet. |
| **Testvoraussetzung** | Die direkte URL zur API muss bekannt sein und es dürfen keine Authentisierungs-Infos mitgesendet werden. |
| **Testschritte** | 1. API über Browser aufrufen. |
| **Erwartetes Ergebnis** | Es dürfen keine Informationen ausgegeben werden. Es muss also eine Meldung in der Art «Kein Zugriff» ausgegeben werden. |

| **Tester (Testperson)** | Florian Leimer |
| --- | --- |
| **Datum Testdurchführung** | 20.01.2021 |
| **Testergebnis** | Der API-Aufruf wird nur mit Token autorisiert. Wenn kein korrekter Token mitgesendet wird, wird die Anfrage abgelehnt und es wird ein 403-Fehler ausgegeben. |
| **Fehlerbeschreibung** | - |

Tabelle 38 Testfall T-006

# Reflexion

Die Umsetzung des Projektes lief nicht schlecht. Mit dem Aufbau der API kam ich sehr gut voran und hatte kaum Probleme. Ausserdem bin ich stolz auf meine Lösung mit den Models und Repositories. Diese Lösung funktioniert sehr gut. Das nächste Mal muss ich aber die Anbindung an die Datenbank etwas anders lösen. Momentan erstellt jedes Repository eine eigene Verbindung zur Datenbank.

Im Frontend war ein grosser Fehler zuerst die Verwaltungen und nicht das Login umzusetzen. Bei zukünftigen Projekten muss ich auf jeden Fall zuerst das Login und die Benutzerverwaltung umsetzen, bevor ich mit dem Rest beginne. Der Grund dafür ist, dass die Implementierung im Nachhinein um einiges komplizierter ist, als wenn von Anfang an alles mit Login und Autorisierung aufgebaut wird.

Die Dokumentation lief nicht hervorragend, ich legte etwas wenig Wert darauf und dokumentierte nur das Wichtigste. Am Schluss ist mir dies etwas zum Verhängnis geworden. Deshalb bin ich leider nicht komplett zufrieden mit meiner Dokumentation.

# Schlusswort

Mein persönliches Fazit ist, dass ich die Probe-IPA etwas ernster hätte nehmen sollen. Leider bin ich schlussendlich nicht komplett zufrieden mit meiner Probe-IPA, die Doku hätte ich um einiges besser machen können.

Für die richtige IPA nehme ich mit, dass ich mehr Wert auf die Doku legen muss. Ich musste bei dieser Probe-IPA in der letzten Woche noch sehr viel Zeit in die Dokumentation stecken. Hätte ich von Anfang an etwas besser dokumentiert wäre der Aufwand sicher um einiges geringer gewesen. Ausserdem hatte ich einige Abweichungen vom Zeitplan. In der Probe-IPA muss ich also die Zeit besser planen, denke aber dies sollte mir gelingen, da ich mehr Erfahrung mir Projekten in der Firma habe als mit solchen in der Schule.

# Quellverzeichnis

**Vue.js Template:**

CreativeTim, «Vue Black Dashboard», <https://www.creative-tim.com/product/vue-black-dashboard>, 01.12.2020

**JWT-Library:**

RobDWaller, «ReallySimpleJWT», <https://github.com/RobDWaller/ReallySimpleJWT>, 24.11.2020

# Glossar

|  |  |
| --- | --- |
| **Begriff** | **Bedeutung** |
| Apache | Einer der meistbenutzten Webserver im Internet. |
| API | Eine Programmierschnittstelle, welche durch andere Programme aufgerufen wird. |
| Composer | Anwendungsorientierter Paketmanager für PHP. |
| Frontend | Grafische Anzeige einer Webseite für den Benutzer mithilfe von HTML, CSS und JavaScript. |
| PDO | Abstraktionsebene für den Datenbankzugriff von PHP. |
| REST | REST ist eine Abstraktion der Struktur und des Verhaltens des World Wide Web. |
| PHP | PHP ist eine Skriptsprache mit einer an C und Perl angelehnten Syntax, die hauptsächlich zur Erstellung dynamischer Webseiten oder Webanwendungen verwendet wird. |
| Vue.js | Vue.js ist ein JavaScript-Webframework zur Erstellung von dynamischen Webseiten. |

Tabelle 39 Glossar