



Diplomarbeit

Refundable

effiziente Reise- und Exkursionsverwaltung

Projektleitung & Frontend - responsives Webdesign

Linus Dehner 5BHIT

Backend - REST-Schnittstelle und Infrastruktur

Michael Beier 5BHIT

Frontend - Webapplikation als REST-Client

Ryan Foster 5BHIT

Betreuer: DI Stefan Zakall, BSc

Ausgeführt im Schuljahr 2020/21

Abgabevermerk:

21. April 2021

Übernommen von:

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Ort, Datum

Linus Dehner

Ort, Datum

Michael Beier

Ort, Datum

Ryan Foster

Kurzfassung

Für viele Schülerinnen und Schüler zählen die unzähligen Exkursionen, Sportwochen, Sprachreisen, Skikurse und Projektwochen zu den schönsten Erinnerungen ihrer Schullaufbahn. Momentan sind diese hier am TGM, jedoch noch mit viel Bürokratie verbunden. Für Exkursionen müssen diverse Anträge ausgefüllt und Reisekostenabrechnungen durchgeführt werden. Dies führt zu einer Mehrbelastung der Lehrkräfte. *Refundable* soll den Lehrern eine einfache Verwaltung ihrer Exkursionen ermöglichen, sodass die Mehrbelastung auf ein Minimum reduziert werden kann. Durch automatische Generierung und Verwaltung verschiedener Anträge, samt Daten, übernimmt und vereinfacht *Refundable* die sonst notwendige Bürokratie. Das Ziel von *Refundable* ist es, den Fokus, weg von der Bürokratie, auf die eigentliche Exkursion zu lenken.

Abstract

For many students, the numerous excursions, sporting weeks, language and skiing trips as well as project weeks at the TGM are among the fondest moments of their school careers. Currently, these activities require considerable administrative work. Some of the necessary paperwork includes filling out numerous application forms and submitting travel expense reports, this can put additional unnecessary strain on teachers. *Refundable* aims to provide teachers with a tool to manage their excursions more easily and to reduce the additional workload to a minimum. By automatically generating and managing various applications, the *Refundable* software will eliminate much of the necessary paperwork. The aim of *Refundable* is to shift the focus away from administrative tasks to allow teachers to focus on creating a memorable experience for all participants of a projekt week.

Gender-Erklärung

Zur besseren Lesbarkeit und zur Erhaltung eines beständigen Leseflusses wird in dieser Diplomarbeit die Sprachform des generischen Maskulinums verwendet. Es wird hiermit ausdrücklich darauf hingewiesen, dass etwaige männliche Bezeichnungen in jedem Fall geschlechtsunabhängig zu verstehen sind. Diese Entscheidung wurde nach langer Bedenkzeit unter Betrachtung des Aspekts der Lesbarkeit und des Leseflusses getroffen. Des Weiteren soll sie keinesfalls eine Geschlechterdiskriminierung darstellen.

Inhaltsverzeichnis

1 Vorwort	15
2 Danksagung	17
3 Einleitung	19
4 Projektmanagement	21
4.1 Traditionelles Projektmanagement	22
4.1.1 Analyse und Konzeptionsphase	22
4.1.2 Designphase	22
4.1.3 Entwicklungsphase	22
4.1.4 Testphase	22
4.1.5 Implementierungsphase	23
4.2 Agiles Projektmanagement	23
4.2.1 Scrum	23
4.2.2 Kanban	24
4.3 Fazit	24
5 Studie	25
5.1 Frontend - responsives Webdesign	26
5.1.1 Einleitung	26
5.1.2 HTML, CSS, JS	26
5.1.3 SASS	29
5.1.4 Frameworks	29
5.1.5 Vergleich	32
5.1.6 Fazit	34
5.2 Backend - REST-Schnittstelle und Infrastruktur	35
5.2.1 Einleitung	35
5.2.2 Docker	36
5.2.3 Deployment	37
5.2.4 REST-Schnittstelle	37
5.2.5 Kommunikation und Datenformate	39
5.3 Frontend - Webapplikation als REST-Client	43
5.3.1 Einleitung	43
5.3.2 Entwurfsmuster	43
5.3.3 Umsetzungsmöglichkeiten	45
5.3.4 Aufbereitung der Daten	51

5.3.5	Fazit	51
6	Konzept	53
6.1	Frontend	54
6.1.1	Theoretische Konzeption	54
6.1.2	Visuelle Konzeption	55
6.2	Backend und Infrastruktur	65
6.2.1	Einleitung	65
6.2.2	Infrastruktur	65
6.2.3	Datenmodell	74
6.2.4	REST-Schnittstelle	75
6.2.5	Backend	77
6.3	Datenschnittstelle und Webseitenlogik	81
6.3.1	Webseitenlogik	81
6.3.2	Daten laden	83
6.3.3	Befehle senden	84
7	Implementierung	85
7.1	Frontend	86
7.1.1	Vorbereitung	86
7.1.2	Komponenten der Webseite	87
7.2	Backend und Infrastruktur	120
7.2.1	Einleitung	120
7.2.2	Docker	120
7.2.3	Steuerungsskript	124
7.2.4	MongoDB	129
7.2.5	LDAP	131
7.2.6	Untis	133
7.2.7	Dateierstellung	136
7.2.8	Token Management	137
7.2.9	REST-Schnittstelle	138
7.3	Datenschnittstelle und Webseitenlogik	140
7.3.1	Webseitenlogik	140
7.3.2	Daten laden & Befehle senden	147
7.3.3	Anmelden	149
7.3.4	Abmelden	150
7.3.5	Antrag erstellen	151
7.3.6	Antrag Ansicht	161
7.3.7	Antrag Übersicht	166
7.3.8	Administrator	167
8	Retrospektive	169
8.1	Probleme	169
8.2	Erkenntnisse	169
8.3	Webdesign	170
8.4	Backend und Infrastruktur	171
8.5	Datenschnittstelle und Webseitenlogik	172

9 Conclusio	173
Glossar	175
Akronyme	179
Literaturverzeichnis	181
Abbildungsverzeichnis	185
Tabellenverzeichnis	187
Auflistungsverzeichnis	189
Appendix	193
Abwesenheitsmeldung eines Jahrgangs	194
Abwesenheitsmeldung eines Lehrers	195
Abgeltung für pädagogische Betreuung	196
Reiserechnung	197
Dienstreiseantrag	198
Reiserechnung - Excel	199
Dienstreiseantrag - Excel	200

Kapitel 1

Vorwort

Die Diplomarbeit *Refundable - effiziente Reise- und Exkursionsverwaltung* wurde im Rahmen einer Abschlussarbeit an der höheren technischen Lehranstalt TGM durchgeführt. Die Arbeit wurde von drei Personen verfasst und ist in verschiedene Unterabschnitte aufgeteilt. Die verschiedenen Themenbereiche sind das Erstellen, der responsive Webapplikation, der REST-Schnittstelle, als auch Infrastruktur und der Schnittstelle, zwischen Frontend und Backend.

Das Ziel dieser Diplomarbeit war es sämtliche Anträge, welche intern in der höheren technischen Lehranstalt, dem TGM gestellt werden, zu digitalisieren und durch technische Unterstützung zu automatisieren.

Das Projektteam von *Refundable* wünscht Ihnen viel Freude, beim Lesen dieser Arbeit.

Kapitel 2

Danksagung

Der fünfte Jahrgang, der Maturajahrgang, ist keine einfache Zeit. Er ist für uns Schüler voller Herausforderungen. Seien es die beiden normalen Semester, die es zu absolvieren gilt, die Diplomarbeit oder die Vorbereitung auf die Matura. Danach gilt es bei der Matura selbst noch einmal Alles zu geben und das unter Beweis zu stellen, was man während der mindestens 13 Jahre langen Schullaufbahn erlernt und geübt hat. Zusätzlich zu diesen Herausforderungen erschwert die Coronavirus-Pandemie dieses Jahr den schulischen Alltag massiv. Gerade deswegen wollen wir hiermit jenen ausdrücklich von Herzen danken, die uns in dieser schwierigen Zeit geholfen haben und zur Seite gestanden sind.

Zuerst wollen wir uns bei unserem Diplomarbeitsbetreuer Herrn Professor Zakall und unserem Betreuer aus dem ITP-Unterricht Herrn Professor Dolezal dafür bedanken, dass sie jederzeit ein offenes Ohr für unsere Probleme hatten und uns immer mit Rat und Hilfe bei der Lösung unserer Probleme beigestanden sind. Des Weiteren wollen wir Herrn Professor Borko für die Organisation des Diplomarbeitsseminars und für die immer klaren Antworten auf all unsere unzähligen Fragen bedanken.

Dieser Maturajahrgang war auf Grund der andauernden Pandemie wahrscheinlich einer der Schwierigsten und Aufwendigsten der letzten Jahre. Von Lockdown zu Lockdown wurden soziale Kontakte eingeschränkt, sodass wir mit Freunden und Klassenkollegen nur online reden konnten. Aus unserer Situation heraus wollen wir auch unseren Freunden in der Klasse danken, die uns durch diese schwere Zeit geholfen haben. Wir wollen hier speziell Maximilian Frühmann für die vielen Ratschläge, die er uns gegeben hat, und für das offene Ohr bei etwaigen Problemen danken. Ein weiterer Dank gilt unseren restlichen Freunden, die uns jederzeit aufgemuntert und unterstützt haben. Besonders ist uns hierbei aufgefallen, dass wir Freunde haben, die sich andere nur wünschen können.

Zuletzt wollen wir auch unseren Familien danken, die in diesen turbulenten Zeiten immer für uns da waren und an unserer Seite standen. Wir glauben, dass wir aus der Pandemie mit neuen Fähigkeiten herausgehen werden, und dies nur auf Grund jener Förderung im Leben, die wir von unseren Familien erhalten haben. Speziell jenen, die unseren Abschluss leider nicht mehr miterleben können, wollen wir von ganzem Herzen für die unendliche Inspiration danken. Wenn wir eines mit Sicherheit wissen, dann, dass ihr immer stolz auf uns wart und das auch immer sein werdet.

Kapitel 3

Einleitung

Ob Reiseantrag, Pflegefreistellung, oder Fortbildungsantrag - die Lehrkräfte des *TGMs* werden laufend mit immer mehr administrativem Aufwand belastet. Im Zeitalter der Digitalisierung ist es speziell für eine Schule, wie dem *TGM* mit einer der größten Abteilungen für höhere Informationstechnologie mit exzellenter Reputation, ein wichtiges Anliegen, solche unnötigen Aufwände zu beseitigen und die ständig voranschreitende Digitalisierung nachhaltig zu unterstützen.

Im Rahmen dieses Projektes wird versucht, diesen Schritt zu gehen und mittels Digitalisierung dieser Agenden den Arbeitsaufwand der Lehrkräfte zu minimieren. Dieses Problem zu lösen und den Lehrkräften einen großen Teil der Bürokratie abzunehmen ist hierbei von höchster Priorität.

Diese Diplomarbeit handelt von eben jenen Lösungsversuch. Es wird untersucht, mit welchen Methoden, das Projekt *Refundable*, am besten, effektivsten und kostengünstigsten umgesetzt und durchgeführt werden kann.

Kapitel 4

Projektmanagement

Aufgrund der zeitlichen Befristung und der erheblichen Komplexität von *Refundable* ist es notwendig, eine passende Projektmanagementmethode zu wählen. Zuerst gilt es aber zu klären, was der Begriff Projektmanagement (PM) überhaupt bedeutet.

„Das PM umfasst die Führungsaufgaben, -organisation, -techniken und -mittel zur erfolgreichen Abwicklung eines Projekts. Die DIN 69901 definiert Projektmanagement als Gesamtheit von Führungsaufgaben, -organisation, -techniken und -mittel für die Abwicklung eines Projekts. Allgemeiner definiert das Project Management Institut (PMI) im Project Management Body of Knowledge (PMBOK) PM als Anwendung von Wissen, Fähigkeiten, Methoden und Techniken auf die Vorgänge innerhalb eines Projekts“ [44].

Das Projektteam hat während seiner Zeit am TGM im Rahmen des Unterrichts für *Informationstechnische Projekte (ITP)* vor allem mit der traditionellen Methode, dem Wasserfallmodell, und den zwei agilen Methoden Scrum und Kanban gearbeitet. Daher wird der Fokus nachfolgend auf diese Methoden gelegt.

4.1 Traditionelles Projektmanagement

Das traditionelle Projektmanagement hat einen linearen Ablauf. Das heißt, man notiert sich alle Aufgaben, die man abzuarbeiten hat und arbeitet sie alle der Reihe nach ab. Keine neue Aufgabe wird angefangen, bis diejenige davor abgearbeitet ist. Ein sehr wichtiger Punkt in diesem Managementsystem ist die viele Dokumentation, ohne die das System nicht funktioniert. Verkörpert wird dieses Verfahren durch das Wasserfallmodell:

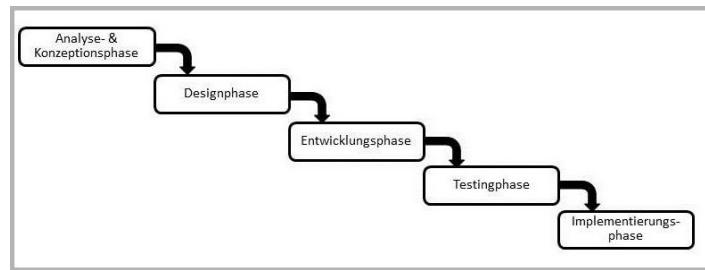


Abbildung 4.1: Illustration des Wasserfallmodells [9]

4.1.1 Analyse und Konzeptionsphase

In dieser Phase wird das Lastenheft und ein Projektstrukturplan erstellt, welche die groben Anforderungen des Projektes verkörpern. Danach wird die Machbarkeit (Grobanforderungen, Kosten, Ertrag, Realisierbarkeit, Umfeldanalyse, Projektorganisation, etc.) des Projektes mittels einer Machbarkeitsstudie analysiert [56]. In der Anforderungsdefinition werden alle möglichen Funktionen definiert, die das Software-Projekt beinhalten soll und kann. Diese werden im Pflichtenheft notiert, auch Fachspezifikationen oder Blueprint genannt. Alljene Dinge, die in der *Analysephase* nicht bedacht werden, können im Laufe des Projektes gravierende Folgen auslösen [9].

4.1.2 Designphase

Die *Designphase* wird streng in Kundenkontakt durchgeführt [9] und ist dafür da, dass ein konkreter Lösungsvorschlag mit den definierten Anforderungen ausgearbeitet wird [56]. Es wird eine detaillierte Anleitung für die Erstellung der Software verfasst, die sich vor allem auf Komponenten wie Schnittstellen, Frameworks oder Bibliotheken bezieht. Als Resultat dieser Phase erhält man ein Entwurfsdokument mit Software-Bauplan, bzw. eine technische Spezifikation und Testplänen, die sich auf einzelne Komponenten beziehen.

4.1.3 Entwicklungsphase

In dieser Phase, werden die in der *Designphase* entwickelten Konzepte und Spezifikationen umgesetzt. Der Projektleiter spielt hier eine tragende Rolle, denn er muss sich bei aufkommenden Fragestellungen sowohl um externe, als auch interne Probleme kümmern und in Kontakt mit dem Auftraggeber bleiben [9].

4.1.4 Testphase

In der *Testphase* wird das Ergebnis der Entwicklungsphase getestet und auf Fehler geprüft [9]. Sollten diese auftreten werden, sie ausgebessert und einem *Re-Test* zugeführt, bis der Kunde das angefragte und funktionierende Endprodukt hat.

4.1.5 Implementierungsphase

In der *Implementierungsphase* findet der *Rollout* statt, welcher mittels einem *Rolloutplan*, der bereits in der *Designphase* erstellt wurde, durchgeführt wird [9]. Sollte es sich bei dem Projekt um Erweiterungen für eine bestehende Applikation handeln, dann sollte der *Rollout* nicht zu *Peakzeiten* stattfinden.

4.2 Agiles Projektmanagement

In agilen Methoden steht vor allem die Funktionalität im Vordergrund und nicht die Dokumentation, wie im traditionellen System [1]. Durch die kurzen Iterationen haben Entwickler und Kunden ein besseres Gefühl für das entstehende Produkt und die Kundenwünsche können besser implementiert werden. Alle agile Methoden sind Vorgehensmodelle, die auf einem Wertekanon, Prinzipien und Praktiken beruhen [1].

„Agiles Projektmanagement bezeichnet Vorgehensweisen, bei denen das Projektteam über hohe Toleranzen bezüglich Qualität, Umfang, Zeit und Kosten verfügt und eine sehr hohe Mitwirkung des Auftraggebers bei der Erstellung des Werks erforderlich ist. Charakteristisch für agiles Projektmanagement ist die Fokussierung auf das zu liefernde Werk und die Akzeptanz durch die Anwender. Hingegen werden geschäftliche Anforderungen, wie z.

B. die Termintreue, Kostentreue oder Erfüllung eines spezifizierten Leistungsumfangs weniger oder nicht berücksichtigt“ [55].

4.2.1 Scrum

Scrum ist ein empirisches Vorgehen, das in sogenannte *Sprints* (ein zeitlich begrenztes Event) eingeteilt ist [1]. Ein *Sprint*, besteht aus dem *Sprint Planning*, in dem der kommende, maximal vier wöchige *Sprint*, geplant und durchgesprochen wird. Genauer gesagt werden Aufgaben von dem *Project Backlog* (der Wunschliste, des Kunden), in den *Sprint Backlog* (die Aufgaben, die das *Development Team* in einem Sprint durchführen wird) verschoben. Jeden Tag im *Sprint* gibt es einen *Daily Scrum*, in dem ein 15 minütiges *Standup-meeting* stattfindet. Dieser Ablauf ist grafisch in Abbildung 4.2 dargestellt.

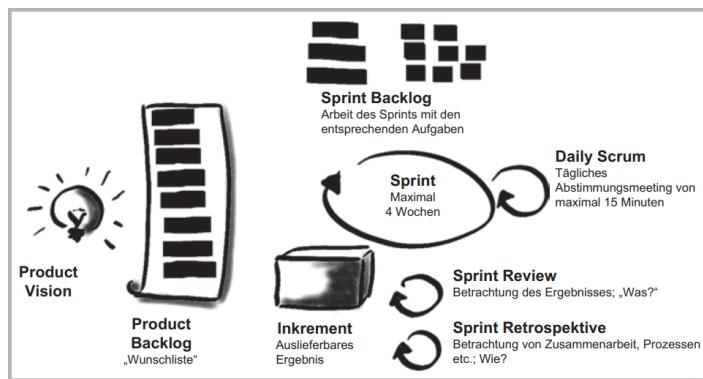


Abbildung 4.2: Illustration Scrum [1]

4.2.1.1 Scrum-Rollen

Product Owner

Der *Product Owner* trägt die Verantwortung des Produktes. Üblicherweise bringt dieser ein ausgeprägtes Branchenwissen mit sich, das den Erfolg und die Profitabilität des Produktes garantieren soll.

Scrum Master

Der *Scrum Master* hilft dem *Development Team*, bei der Lösung, spezieller Herausforderungen. Seine Hauptaufgabe ist es, sich sowohl um die Scrum-Werte und Methoden, als auch um die Prozesse zwischen den verschiedenen Rollen zu kümmern.

Development Team

Das *Development Team* setzt die *Tasks* selbstständig um und erstellt die aus den *Sprints* resultierenden Produktinkremente, die jeweils testbare Einheiten bilden sollten und im Anschluss getestet werden.

4.2.2 Kanban

In *Kanban* gibt es keine bestimmten Rollen, die den Projektmitglieder zugewiesen werden, anders als in *Scrum* [1]. Es gibt ein *Kanban-Board*, welches einen strukturierten Aufbau, von Links nach Rechts beinhaltet. Alle Aufgaben aus dem *Backlog* werden als sogenannte *To do's* ganz Links angezeigt. Des Weiteren gibt es eine Spalte, in der die Aufgaben, die als Nächstes zu behandeln sind gesammelt werden. Dieses Verfahren wird bis zu der Spalte fortgeführt, in welcher die abgeschlossenen *To do's* gesammelt werden. Der Ablauf dieser Spalten ist ähnlich inkrementell, wie bei *Scrum*. Jedoch werden die Arbeitspakete nicht zugeteilt, sondern das *Development Team* zieht sich die *Tickets* selbstständig (auch unter dem Begriff *Pull-Prinzip* bekannt). Siehe Abbildung 4.3 zur Veranschaulichung des *Kanban-Boards*.

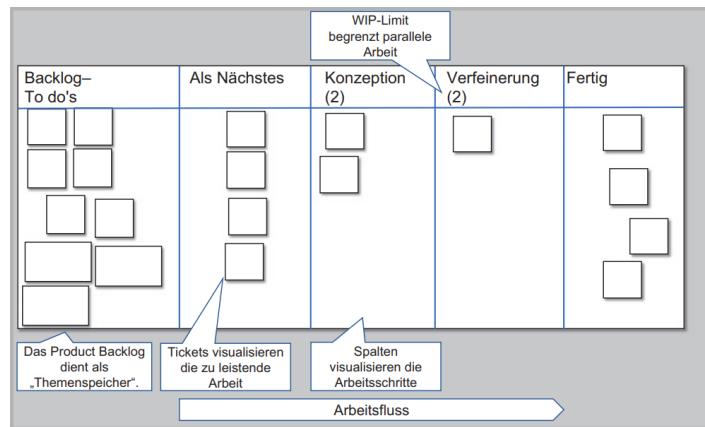


Abbildung 4.3: Illustration Kanban [1]

4.3 Fazit

Angesichts des hohen Dokumentationsaufwandes im traditionellen Projektmanagement, hat sich das Team, wegen des Fokus, auf das Endprodukt für eine agile Methode entschieden. Da *Kanban* im Gegensatz zu *Scrum* nicht disruptiv, sondern evolutionär ist und man daher keine neuen Rollen zuweisen muss, hat sich das Projektteam für *Kanban* entschieden, nicht zuletzt ob des geringen Implementierungsaufwandes. Ein weiterer Entscheidungsgrund war, dass das Projektteam parallel zu ihren schulischen Aktivitäten arbeiten muss und gleichmäßige Sprints nicht garantiert möglich sind. Ebenfalls hat die Integration, des *Kanban-Boards* in *GitHub* die Entscheidung für *Kanban* beeinflusst.

Kapitel 5

Studie

Unsere Studie beschäftigt sich mit aktuell am Markt befindlichen Technologien und vergleicht diese in einem projektspezifischen Kontext. Auf Basis der Analyse treffen wir schließlich im Fazit eine fundierte Entscheidung über die im Projekt zu verwendenden Technologien, wie Programmiersprachen, Frameworks und Werkzeuge, zur Entwicklung, sowie Auslieferung der Software.

5.1 Frontend - responsives Webdesign

5.1.1 Einleitung

In diesem Kapitel werden die Anforderungen des *Frontend-Designs* geschildert. Da es mehrere Möglichkeiten gibt, das *Frontend* zu realisieren, werden hier drei wesentliche Methoden verglichen. Die erste Variante wäre, ganz klassisch *HTML*, *CSS* und *JavaScript* zu verwenden. Die zweite Methode, die zum Vergleich herangezogen wird, verwendet statt *Vanilla CSS* die etwas agilere Sprache *SASS*. Die dritte und auch letzte Methode in diesem Vergleich ist, die Arbeit durch die Verwendung eines *CSS Frameworks* zu vereinfachen. Anschließend werden die drei verschiedenen Methoden gegenübergestellt und die, für unser Projekt *Refundable* am besten geeignete Variante ausgewählt. Zu guter Letzt wird das *Design* im Hinblick auf die Zielgruppe der Lehrer analysiert, die sich möglichst gut und schnell auf der Webseite zurecht finden soll.

5.1.2 HTML, CSS, JS

Der eigentliche Standard *HTML5* wird in der Praxis meist als Überbegriff für *HTML*, *CSS* und *JS* verwendet [58]. In den folgenden Kapiteln wird erklärt, wozu *HTML*, *CSS* und *JS* da sind und welche Funktionalitäten sie bieten.

5.1.2.1 HTML

HTML ist eine Auszeichnungssprache, sie steht für „*Hypertext Markup Language*“ und wurde 1989 von dem britischen Informatiker Tim Burners-Lee veröffentlicht.

„*Hypertext* bezeichnet die Möglichkeit, Texte mit Hilfe von *Hyperlinks*, oder kurz *Links*, miteinander zu verbinden“ [4].

Dies heißt, dass man mittels *Hypertexts* (*Links*) auf der Seite beliebig zwischen Sektionen hin und her springen kann. Auszeichnungssprachen werden im Fachjargon auch als *Markup Language* (*ML*) bezeichnet [32]. *Markup Languages* werden in zwei verschiedene Gruppen aufgeteilt, zum einen *Procedural Markup Languages* (*PML*), das sind jene Auszeichnungssprachen, die für die Verarbeitung von Daten optimiert sind. Zum anderen *Descriptive Markup Languages* (*DML*), diese sind für die logische Strukturierung von Daten da.

Bekannte Beispiele hierfür sind:

- PML
 - PDF
 - TeX
- DML
 - HTML
 - SVG

Auflistung 5.1: PML/DML Beispiele

HTML wird hauptsächlich verwendet, um Texte, Grafiken und *Hyperlinks* (*Links*) darzustellen [58, 4]. Die Bearbeitung von *HTML-Dokumenten* ist relativ einfach und unkompliziert, da es eine rein textbasierte

Sprache ist und mit jedem Texteditor bearbeitet werden kann.

Allerdings ist *HTML* nicht mit einer Programmiersprache zu verwechseln, da nur *Tags* und keine Befehle oder Anweisungen verwendet werden. Solche *Tags* können wie folgt aussehen:

```
1 <tagname> Inhalt</tagname>
2 <einzeltag attribut="123">
```

Auflistung 5.2: *HTML Tags*

Das grundlegende Gerüst von *HTML* besteht aus einer Deklaration von *HTML*, einem *html*-, *head*- und *body*-Tag. In den *html*-Tag kommt ein *title*-Tag, in diesem wird der Titel der Webseite angegeben und ein *meta*-Tag, in diesem werden Meta-Informationen angegeben. In den *Body*-Tag kommen wiederum *Tags*, die den Inhalt der Seite darstellt.

```
1 <!DOCTYPE html>
2 <html lang="de">
3   <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width,
6       ↵ initial-scale=1.0">
7     <title>Kurzes BSP</title>
8   </head>
9   <body>
10    <h1>Überschrift</h1>
11    <button>Drück mich!</button>
12  </body>
13 </html>
```

Auflistung 5.3: Kurzes *HTML* Beispiel

Wenn man die Datei nun im *Browser* öffnet, sieht dies wie folgt aus:



Abbildung 5.1: Beispiel einer *HTML* Seite mit einer Überschrift und einem Button

Da *HTML* nur für die Grundstruktur einer Webseite gedacht ist, also zum Beschreiben der Struktur und des Inhalts, ist das *Design* noch nicht sonderlich ansprechend. Um dies zu verändern, wird *CSS* benötigt.

5.1.2.2 CSS

Cascading Stylesheets, oder kurz *CSS*, ist für den *Style*, also für das Aussehen der Webseite verantwortlich. Da *HTML* anfangs nur im Printbereich verbreitet wurde, war es nicht notwendig, die Seiten zu gestalten [4, 58]. Das Internet bekam aber einen immer stärker werdenden Einfluss und daher auch eine höhere Bekanntheit. Deswegen wurde *HTML* mit der Formatierungssprache *CSS* ergänzt.

Mittels *CSS* sind unter anderem folgende Dinge möglich:

- Den Hintergrund verändern
- Die Schrift verändern
- Die Webseite automatisch an die Bildschirmgröße anpassen
- Den Formfaktor von Elementen verändern
 - Größe
 - Rand
 - Farbe
 - Form
 - Schatten
 - Hover-Effekt

Auflistung 5.4: Beispiele Verwendung von CSS

Man kann CSS auf verschiedene Arten in *HTML* benützen. Als Beispiel werden wir eine Überschrift in die Mitte der Webseite setzen, die Schriftgröße auf „40pt“ stellen und die Schriftart auf „sans-serif“ ändern. Die Umsetzung kann auf mehrere Arten erfolgen. Beispielsweise fügt man einem Element ein *Style-Attribut* hinzu und ändert direkt im Element das Aussehen. Hierbei ist darauf zu achten, dass nur das Element, in dem man diese Änderungen vornimmt, verändert wird:

```
1 <h1 style="text-align: center; font-size: 40pt; font-family:
2   ↵ sans-serif;">Ich bin eine tolle Überschrift</h1>
```

Auflistung 5.5: Beispiele Verwendung von Inline-CSS

Man kann auch im „head“ einen *style-Bereich* eröffnen und dort das Aussehen verändern. Dabei ist darauf zu achten, dass man den *Style* von allen Elementen mit dem Tag, den man ausgewählt hat, verändert. Um dies zu verhindern, kann man Elementen auch eine *ID* (für einzelne Elemente verwendbar) oder eine *CLASS* (für mehrere Elemente verwendbar) hinzufügen, sowie diese im *CSS-Code* auswählen und verändern:

```
1 <head>
2   <style>
3     //Für das ganze Element
4     h1 {
5       text-align: center;
6       font-size: 40pt;
7       font-family: sans-serif;
8     }
9
10    //Für IDs
11    #ueberschrift1 {
12      text-align: center;
13      font-size: 40pt;
14      font-family: sans-serif;
15    }
```

```

16
17      //Für Klassen
18      .ueberschriftenGruppe {
19          text-align: center;
20          font-size: 40pt;
21          font-family: sans-serif;
22      }
23      </style>
24  </head>

```

Auflistung 5.6: Beispiel CSS

Ebenfalls kann man ein *CSS-File*, welches den Inhalt des obigen *style-Tags* hat, im *head* als externes File einbinden:

```

1  <head>
2      <link rel="stylesheet" href="file.css" type="text/css">
3  </head>

```

Auflistung 5.7: CSS Verlinkung

Wie man erkennen kann, ist dem ganzen kein Ende gesetzt und mit viel Aufwand kann man alles Denkbare verändern. Um den Aufwand jedoch gering zu halten, sind SASS und *CSS-Frameworks* da, welche in Unterabschnitt 5.1.3 und Unterabschnitt 5.1.4 erklärt werden.

5.1.2.3 Java Script

JavaScript wird verwendet, um Elementen Funktionen zu geben. Zum Beispiel, dass wenn man auf einen *Button* drückt, ein Fenster *aufpoppt*, ein neues Element hinzugefügt wird, oder ein Element im Nachhinein verändert wird. Da *JavaScript* eine Skriptsprache ist, kann man auch eigene Funktionen schreiben und Variablen verwenden. Für das *Designen* braucht man *JS* nur, wenn man mit *CSS* oder *SASS* arbeitet. Wird ein *Framework* verwendet, hat dieses meist ein *JavaScript-Framework* inkludiert und man muss kein *JavaScript* mehr verwenden.

5.1.3 SASS

Syntactically Awesome Style Sheets oder kurz *SASS* ist eine Erweiterung von *CSS* [19]. Es fügt *CSS* ein paar Funktionalitäten von *Java Script* hinzu. *SASS* wird aber nicht wie *CSS* direkt in das *HTML-File* eingebunden und kann auch nicht direkt in ein Element geschrieben werden. Das *.sass* File muss erst kompiliert werden. Anschließend wird ein *.css* File generiert, welches man im *HTML-Code* einbinden kann. Man kann unter anderem Funktionen erstellen, um Elemente zu verändern. Zusätzlich kann man Variablen kreieren und so zum Beispiel eine Primärfarbe festlegen, wodurch man sich nicht immer den *HEX-Code* von einer bestimmten Farbe merken muss. Ebenfalls kann man durch die Variablen die Farbe von mehreren Elementen auf einmal ändern. Dadurch kann man zum Beispiel *Light-* und *Darktheme* in die Webseite einbauen.

5.1.4 Frameworks

Frameworks für *HTML*, *CSS* und *JS* beinhalten vorgestaltete Komponenten, welche mittels Klassen, einem *HTML* Element hinzugefügt werden. Da *Frameworks* das Arbeiten an einer Webseite wesentlich einfacher

machen und wir dieses Hilfsmittel auch benutzen werden, müssen folgende Fragen beantwortet werden, um das optimale Framework für dieses Projekt auszuwählen:

- **Welche CSS-Frameworks unterstützen explizit die User-Experience und Usability?**
- **Welche Vor- und Nachteile bringen diese Frameworks mit sich?**

Auflistung 5.8: Fragestellungen - Frameworks

Zum Vergleich werden vier sehr verbreitete und geschätzte *Frameworks* herangezogen:

5.1.4.1 Bootstrap

Bootstrap ist ein *Framework*, welches sowohl am bekanntesten ist, als auch am meisten verwendet wird [29, 30]. Das von *Twitter* entwickelte *Framework* hat in der aktuellen Version 4.5 viele verschiedene Komponenten, die optimal für erfahrene Entwickler, aber auch für Anfänger geeignet sind. In der Dokumentation von *Bootstrap* gibt es einige *Templates* und sehr viel gut dokumentierten Beispiel Code[29]. *Bootstrap* kann über einen *Paketmanager*, ein *CDN*, oder als heruntergeladene Datei zur *HTML-Datei* hinzugefügt werden [25]. Durch diesen großen Verwendungsgrad gibt es viele Erweiterungen und zahlreiche Plattformen auf denen man sich informieren kann, falls man Probleme bei der Entwicklung der Webseite hat [30]. *Bootstrap* bietet, mit seinem überschaubaren *Grid-System*, eine gute Umsetzung für die Responsivität der Seite, damit sie auf allen möglichen Endgeräten perfekt aussieht.

Pro / Contra

Pro

- *HTML/CSS/JS Framework*
- Umfangreich
- Mittels *SASS* veränderbar
- Zahlreiche Erweiterungen
- Viele Erkundungsmöglichkeiten
- Viele Einbindungsmöglichkeiten
- Arbeitet gut mit *VueJS* zusammen
- Responsiv

Auflistung 5.9: Bootstrap Pro

Contra

- Viele Webseiten sehen gleich aus
- Komplexer zu erlernen
- Kein schönes Standarddesign

Auflistung 5.10: Bootstrap Contra

5.1.4.2 Materialize

Materialize ist, wie *Bootstrap*, ein intuitives *HTML, CSS und JS Framework* [42]. Es ist *Bootstrap* sogar sehr ähnlich, aber simpler aufgebaut. Dadurch hat es auch nicht so viele verschiedene Komponenten und nicht so viel Beispiel Code wie *Bootstrap*. *Materialize* legt vor allem Wert auf das *Design*, welches von *Google's Material Design* abstammt. Ein weiterer großer Punkt ist *Usability* und *User Experience*, welche unter anderem Hand in Hand mit dem *Design* gehen. Um die *User Experience* möglichst hoch zu halten, arbeitet *Materialize* viel mit Animationen. *Materialize* kann ebenfalls über ein *CDN*, *Paketmanager* oder als Datei in das Projekt eingebunden werden. Falls man das *Design* verändern will, stellt *Materialize* ebenfalls noch eine *SASS* Version zur Verfügung [11].

Pro / Contra

Pro

- HTML/CSS/JS Framework
- Mittels SASS veränderbar
- Ansehnliches Standard-Design
- Viele Animationen
- Viele Einbindungsmöglichkeiten
- Einfach zu verstehen
- Responsiv

Auflistung 5.11: Materialize Pro

Contra

- Einige Probleme mit VueJS
- Weniger Erkundungsmöglichkeiten
- Beispielcode ist oft unvollständig

Auflistung 5.12: Materialize Contra

5.1.4.3 ZURB Foundation

Foundation wird, wie *Bootstrap* und *Materialize*, als intuitives *Web-Framework* bezeichnet [47]. Das von *Zurb* entwickelte *Framework* beinhaltet viele verschiedenen Komponenten, die für mobile Endgeräte optimiert sind. Mit dem *Framework* können Webseiten schnell und effizient gestaltet werden. Darum ist es auch in zwei verschiedene Kategorien geteilt, in das *Framework* für das Web und in das *Framework*, welches explizit für *HTML-Mails* gedacht ist. In der *Complete-Version* sind alle Komponenten enthalten. In der *Essential-Version* sind nur die essentiellen Komponenten, wie zum Beispiel *Buttons*, enthalten. Man kann sich seine *ZURB-Datei* auch mit allen Komponenten, die man benötigt, selbst konfigurieren. Auch eine *SASS* Version ist verfügbar, falls man nur das Aussehen der Komponenten verändern möchte. Natürlich ist auch eine *CDN* Einbindung möglich, die für die schnellste Ladegeschwindigkeit optimiert ist.

Pro / Contra**Pro**

- *HTML/CSS/JS Framework*
- Mittels SASS veränderbar
- Auf Bedürfnisse anpassbar
- Ansehnliches Standard-*Design*
- Viele Einbindungsmöglichkeiten
- Einfach zu verstehen
- Responsiv

Auflistung 5.13: ZURB Foundation Pro

Contra

- Keine persönliche Erfahrung
- Relativ junges Framework
- Nicht so viele Erkundungsmöglichkeiten
- Kein ansprechendes Standarddesign

Auflistung 5.14: ZURB Foundation Contra

5.1.5 Vergleich

Der Vergleich der *Frameworks* setzt sich aus dem Umfang, den Erweiterungen, dem Aussehen, der Leichtigkeit im Bezug auf das Erlernen des *Frameworks*, dem Umfang der Dokumentation, der Kompatibilität mit *VueJS* und unserer Erfahrung, mit dem jeweiligen *Framework* zusammen.

Die Punkte (0-9) setzen sich immer in der Relation mit den anderen *Frameworks* zusammen, wobei 9 die Beste, also maximale Punktzahl ist. Die Basis 0 ist in diesem Vergleich mit den Funktionen und Aufwand von *Vanilla CSS* gleichzusetzen. Alle dafür benötigten Informationen wurden von der jeweiligen Dokumentationsseite bezogen.

Tabelle 5.1: Vergleich zwischen *Bootstrap*, *Materialize* und *Foundation*

<u>Kriterien</u>	<u>Bootstrap</u>	<u>Materialize</u>	<u>Foundation</u>
Umfang	9	9	7
Erweiterungen	9	3	7
Aussehen	6	9	5
Leichtigkeit	6	9	8
Dokumentation	9	6	9
Kompatibilität	9	5	7
Erfahrung	5	9	0
Gesamt	53	50	43

5.1.5.1 Umfang

Die Punktevergabe des Umfangs setzt sich zusammen aus der Anzahl der Komponenten und Hilfsklassen. *Bootstrap* und *Materialize* haben in etwa die selbe Anzahl (53 & 55) und bekommen daher die Maximalpunktzahl [25, 11, 16]. *Foundation* hat hingegen deutlich weniger Komponenten (41) als die anderen zwei *Frameworks* und hat daher nur 7 Punkte bekommen.

5.1.5.2 Erweiterungen

Bootstrap hat durch das langjährige Bestehen und durch die große Reichweite unzähliger Erweiterungen, die Schwächen, wie zum Beispiel das Standarddesign wieder auszumerzen. Deswegen hat *Bootstrap* hier auch die volle Punkteanzahl bekommen [29]. *Materialize* hat nur kostenpflichtige, offizielle Plugins, deswegen hat dieses *Framework* nur 3 Punkte bekommen. *Foundation* hingegen, hat wiederum eine ausreichende Auswahl an Plugins, die sogar in der Dokumentation verlinkt sind [16].

5.1.5.3 Aussehen

Bei dem Aussehen des *Standard-Designs* gewinnt klar und deutlich *Materialize*, da es die ansprechendste Darstellung mit einem schönen *Design* bietet [11]. *Bootstrap* und *Foundation* legen mehr Wert auf die Funktionalität des *Frameworks* und weniger auf das Aussehen [25, 16]. Dies merkt man daran, dass die Elemente optisch nur minimal von den *HTML*-Standardelementen abweichen. *Materialize* hat hingegen ein komplett neues und modernes *Design* aufgezogen.

5.1.5.4 Leichtigkeit

Aus eigener Erfahrung kann das Team sagen, dass *Materialize* wirklich einfach zu erlernen ist [11]. Bei einem groben Einlernen in *Foundation* hat das Team festgestellt, dass es dem Aufbau von *Materialize* sehr ähnelt. *Bootstrap* hingegen sah auf den ersten Blick sehr kompliziert aus und war am Anfang sehr schwer zu verstehen. Nach ein paar Stunden Einlernen war schon ein sehr guter *Workflow* vorhanden.

5.1.5.5 Dokumentation

Auf den ersten Blick erscheint die Dokumentation von *Materialize* als sehr gut, jedoch ist der Beispielcode zu einzelnen Komponenten in manchen Fällen unvollständig und man muss sich mit mühsamer Recherche nach Lösungen erkunden [11]. *Bootstrap* und *Foundation* sind sehr gut dokumentiert, haben viele Code-Beispiele und sind in manchen Fällen sogar mit Code-Pen Beispielen bestückt [25, 16].

5.1.5.6 Kompatibilität

Wie in Abschnitt 5.3 erklärt wird, werden wir als *JS Framework VueJS* verwenden. Kompatibel sind alle *Frameworks*, jedoch ist *Bootstrap* für die Zusammenarbeit mit *Vue* optimiert [25]. Da das Projektteam in einem früheren Projekt bereits *Materialize* mit *Vue* benutzt hat, ist bekannt, dass in Verbindung der beiden *Frameworks* Probleme auftreten können. *Foundation* ist nicht speziell dafür optimiert mit *Vue* zusammenzuarbeiten. Die Funktionalität ist trotzdem vorhanden. Jedoch wurde *Foundation* bis jetzt von keinem Mitglied der Projektes in Verbindung mit *Vue* verwendet.

5.1.5.7 Erfahrung

Das *Frontend-Team* hat bis jetzt hauptsächlich Erfahrung mit *Materialize* gemacht. *Bootstrap* wurde in der Schule kurz angeschnitten, aber im privaten Umfeld etwas vertieft. Mit *Foundation* hingegen wurde noch keine Erfahrung gesammelt.

5.1.6 Fazit

Nach dem Vergleich, der in dieser Arbeit durchgeführt wurde, ist es leicht, die optimale Auswahl der Umsetzung für *Refundable* zu treffen. Eine Entwicklung ohne ein *Framework* kommt auf Basis der Analyse nicht in Frage, da der Aufwand für die Umsetzung wesentlich höher wäre. Wir haben uns schließlich für *Bootstrap* entschieden, da es im Vergleich eindeutig am besten abschneidet und auch am Besten mit *VueJS* harmoniert.

5.2 Backend - REST-Schnittstelle und Infrastruktur

5.2.1 Einleitung

Das *Backend* besteht aus mehreren Komponenten. Einerseits soll eine gewisse System-Infrastruktur aufgebaut werden, um das *Webinterface* und die *REST*-Schnittstelle bereitzustellen. Andererseits muss die Anwendung selbst entwickelt werden. Diese besteht aus mehreren Teilen. Darunter fällt die *REST*-Schnittstelle, inklusive der implementierten *Endpoints*, Schnittstellen zu diversen Diensten, wie dem *TGM-LDAP Server*, zur Datenbank und zu *WebUntis*, aber auch die allgemeine Funktionalität der Anwendung, unter anderem das Erstellen von *PDF*-Dateien. Für die Verbindung zu den Schnittstellen sowie zur Implementierung der geforderten Funktionalität werden diverse *Third Party Packages* genutzt werden. Die folgende Grafik gibt einen Überblick hinter der Infrastruktur und den verwendeten Diensten:

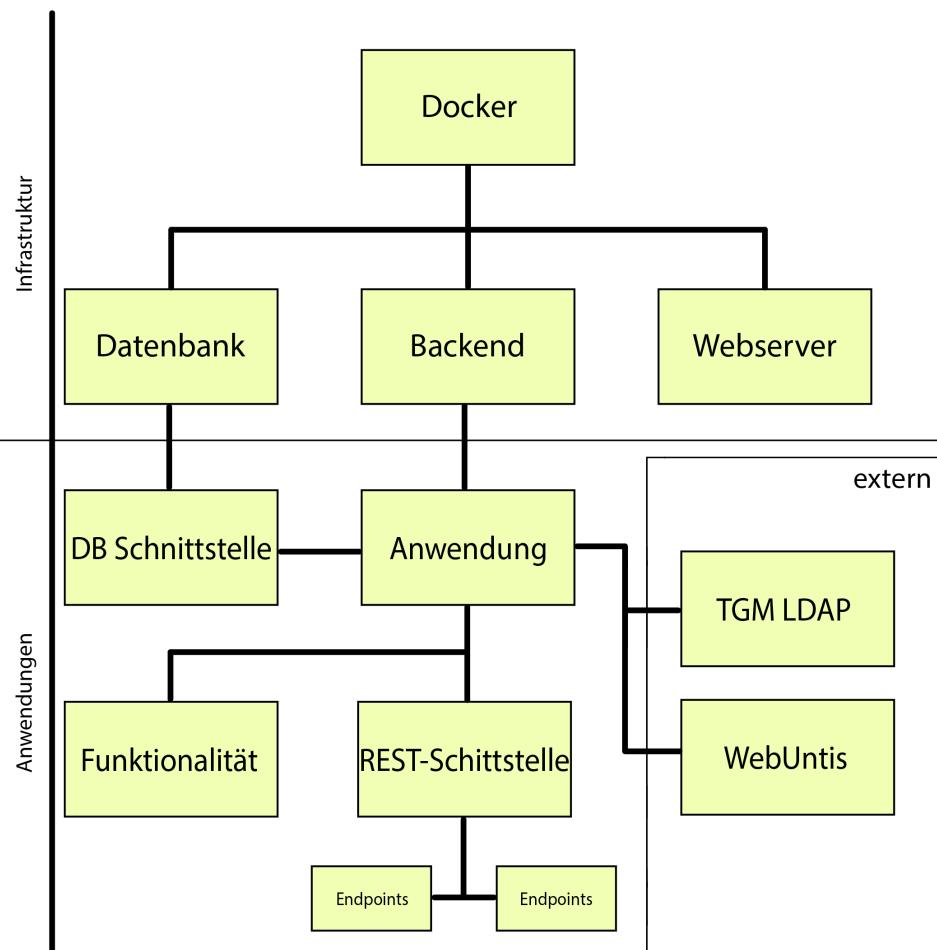


Abbildung 5.2: Übersicht über die verschiedenen Komponenten der Infrastruktur und der Anwendung

In den folgenden Kapiteln werden die aktuell verfügbaren Technologien, welche für die Umsetzung im *Backendbereich* in Frage kommen, beschrieben und gegebenenfalls - sofern mehrere sinnvolle Kandidaten vorhanden sind - auch verglichen. Auf einen detaillierten Vergleich jeglicher verwendbaren *Third Party Packages*, welche dazu genutzt werden könnten, um die Funktionalität im Backend zu implementieren, wird auf Grund der extrem hohen Anzahl im Sinne der Übersichtlichkeit verzichtet. Unter die benutzten Bibliotheken, fallen jedenfalls Module wie „*maroto*“ (zum Erstellen von *PDF*-Dateien [33]), „*excelize*“ (zum Erstellen von *Excel*-Dateien [14]) und „*jwt-go*“ (zum Erstellen von *JSON Web Token* [28])

5.2.2 Docker

Um die Infrastruktur des Projektes einfach aufbauen zu können, wird *Docker* genutzt. Da es sich hier um eine komplex strukturierte Infrastruktur handelt, wird zusätzlich das Werkzeug *Docker Compose* genutzt. Mit *Docker Compose* kann eine Infrastruktur aufgebaut werden, die der Abbildung 5.2 entspricht. Für diese sind folgende *Container* vorgesehen, die in den nächsten Kapiteln noch im Detail beschrieben werden.

5.2.2.1 Datenbank

Um die Daten, die durch *Refundable* erhoben und generiert werden, zu speichern, wird eine Datenbank (DB) benötigt. Auf Grund der Daten, welche sich durch unterschiedliche Datenstrukturen auszeichnen, ist der Einsatz einer relationale Datenbank nicht sinnvoll. Stattdessen empfiehlt sich die Verwendung einer nicht-relationale Datenbank (auch *NoSQL* Datenbank). Standardmäßig wird zwischen 4 verschiedenen Typen von *NoSQL* Datenbanken unterschieden, welche jeweils nur in ihrem eigenen *Use Cases* sinnvoll anwendbar sind [52]:

- *Key-Value* Datenbank
- spaltenorientierte Datenbank
- graphenorientierte Datenbank
- dokumentenorientierte Datenbank

Auflistung 5.15: *NoSQL* Datenbank-Typen

Bei *Key-Value* Datenbanken wird einem Schlüssel ein Wert hinterlegt. Dieser Wert ist dann jederzeit über den Schlüssel in der Datenbank abrufbar. Für unseren *Use Case* ist dieses System nicht sinnvoll anzuwenden, da die von uns benutzten Daten hierfür zu komplex im Aufbau sind.

Bei spaltenorientierten Datenbanken werden Daten vorrangig über ihre Spalten (statt wie bei relationalen DBs in Zeilen) analysiert. Dies ermöglicht die einfache Umsetzung statistischer Methoden auf Basis der Spalten. Da jedoch wieder eine Tabelle als Grundstruktur vorliegt, ist dieser Typ von Datenbank nicht sinnvoll anwendbar für *Refundable*.

Bei graphenorientierten Daten wird die Beziehung zwischen einzelnen Elementen hervorgehoben. Daten werden hier in Knoten gespeichert, welche zu anderen verbunden werden können. Die primären Elemente sind hierbei die Beziehungen, anstatt der Daten selbst. Da die Daten von *Refundable* nicht über starke Beziehungen charakterisiert sind, ist auch dieser Datenbank-Typ nicht sinnvoll zu benutzen.

Zuletzt bei dokumentenorientierten Datenbanken unterliegt jeder Datensatz in einem eigenen Dokument, welches in *JSON*, *YAML*, *XML* oder ähnlichen Datenformaten gespeichert wird. Dadurch ist auch eine jeweils von einander unabhängige Datenstruktur möglich. Auf Grund der Flexibilität bei Datenstrukturen ist eine dokumentenorientierte Datenbank eindeutig sinnvoll zu verwenden.

Als Datenbankmanagementsystem (DBMS) kommt bei dieser Auswahl einige Software in Frage. Die am meisten verbreitete Software hier ist *MongoDB* und *CouchDB* [34]. Wo *MongoDB* auf strenge Konsistenz setzt, setzt *CouchDB* auf hohe Verfügbarkeit. Da in unserem Projekt Konsistenz wichtiger ist als Verfügbarkeit wird *MongoDB* in einem Container als Datenbankmanagementsystem verwendet.

5.2.2.2 Backend-Container

Ebenfalls wird ein *Container*, also eine Umgebung, in dem das *Backend* laufen kann, erstellt. Dieser wird direkt zu den anderen *Containern* hinzugefügt, damit dieser über ein *Docker*-Netzwerk mit den anderen *Containern* kommunizieren kann.

Um diesen *Container* zu realisieren wird als Basis ein „*golang*“-Image genutzt [21]. Dieses stellt eine sehr sparsame Linux-Instanz dar, welche mit einer „*golang*“-Umgebung ausgestattet ist. Um diesen *Container* noch entsprechend anzupassen, wird ein entsprechendes *Dockerfile* über ein *Dockerfile* gebaut.

5.2.2.3 Webserver

Als *Webserver* wird ein *Apache2 Server* genutzt [22]. Der Service stellt hierbei das *Webinterface (Frontend)* im Internet zur Verfügung. Dieser *Container* ruft automatisch das *Frontend* auf und kopiert es in seine Umgebung. Ebenfalls muss der *Container* Zugriff auf Zertifikaten bekommen, um einen sicheren Zugriff über *HTTPS* gewährleisten zu können.

5.2.3 Deployment

Das *Deployment* soll automatisch geschehen. Um dies einfach zu ermöglichen, werden die oben zuvor beschriebenen *Docker-Container*, *GitHub* und ein Skript, welches die Schritte ausführt, genutzt. Das Skript ist hier der Hauptbaustein, welcher den Vorgang startet und steuert. Zusätzlich zum Installationsvorgang, soll das Skript auch die weitere Steuerung der Software, also starten, stoppen, updaten, cleanen und deinstallieren, beinhalten.

Das Skript liegt in einem eigenem *Install-Repository*, in welchem sonst keine weiteren Dateien liegen. Dadurch kann es einfach geklont und direkt installiert werden; die restlichen Installations-Schritte werden automatisch erledigt.

Da das *Deployment* auf einer *Linux*-Maschine ermöglicht werden soll, wird *Bash* als Standard-Skriptsprache benutzt. Zur Verteilung des Scripts wird, wie erwähnt, *Git* mit dem *Online Repository-Hosting Service GitHub* verwendet.

5.2.4 REST-Schnittstelle

Bei einer *REST*-Schnittstelle handelt es sich um einen bestimmten Aufbau einer Softwareschnittstelle bei verteilten Systemen [40]. Das hierbei angewandte Prinzip nennt sich „*Representational State Transfer*“ (kurz *REST*). Es zeichnet sich durch folgende Eigenschaften aus:

- **Client-Server**, wobei es um eine strikte Trennung zwischen dem *Client* (dem *REST-Client*) und dem *Server* (der *REST*-Schnittstelle, als *Webservice*) geht
- **Stateless**, wobei es um die Zustandslosigkeit des *Servers* geht. Das heißt, dass der Server sich keinerlei Zustände der *Clients* merkt.
- **Caching**: Der *Server* speichert die *Responses* zwischen. Dadurch kann die Latenzzeit minimiert werden, da Daten öfters zurückgegeben werden, anstatt sie jedes Mal erneut berechnen zu müssen.

- **Einheitliche Schnittstelle** bedeutet, dass die Schnittstelle ein einheitliches Datenformat verwendet, um via *HTTP* über *CRUD*-Methoden (*Create, Read, Update, Delete*) zu kommunizieren.
- **Layer-System** bedeutet, dass die Schnittstelle, als *Webservice* so designt wird, dass auch weitere Schichten, wie *Gateways* und *Proxies*, transparent dazwischen aufgebaut werden können.
- **Code-on-demand** (optional): Hierbei besteht die Möglichkeit ausführbaren *Code* an die *Clients* zu schicken, sodass diese den dann ausführen können.

Auflistung 5.16: Prinzipien des *Representation State Transfer*

5.2.4.1 Java und Spring

Java ist eine der bekanntesten Programmiersprachen. Sie zeichnet sich durch Plattform-Unabhängigkeit aus [26]. Um dies zu erreichen, wird *Bytecode* von einem eigenem Programm, der *Java Virtual Machine*, interpretiert. *Java* arbeitet objektorientiert und ist statisch typisiert. Dies bedeutet, dass bereits vor dem Ausführen die Datentypen definiert sind. *Java* unterstützt auch *Multithreading*. Um einfach eine *REST*-Schnittstelle in *Java* bauen zu können, wird *Spring Boot* genutzt [49]. Die Verwendung von *Spring* ermöglicht *Webapps*, *Tasks* oder *Microservices* einfacher umzusetzen. Prinzipiell arbeitet *Spring* asynchron und flexibel, sodass es einfach zu skalieren ist.

5.2.4.2 Python und Flask

Um mit *Python* eine *REST*-Schnittstelle zu realisieren muss auf das *Package (Framework)* *Python Flask* zurückgegriffen werden [15]. *Flask* wird hierbei dazu genutzt, um den *Webservice* zu bauen. *Python* und *Java* unterscheiden sich prinzipiell sehr. *Python* ist im Gegensatz zu *Java* dynamisch typisiert, dies bedeutet, dass eine Variablen Deklaration nicht notwendig ist und der Datentyp einer Variable erst zur Laufzeit klar ist [38]. Anders als *Java* setzt *Python* auf *Third-Party Packages*, welche sehr einfach importiert werden können. Durch die große *Community Pythons* ist ein Großteil der *Tools*, die man benötigt, meist schon vorprogrammiert und kann einfach importiert werden. Zusätzlich ist der Syntax (speziell was Zeichensetzung anbelangt) einfacher zu verstehen, als jener von *Java*.

5.2.4.3 Golang

Golang (kurz *Go*) ist eine von *Google* entwickelte und publizierte Programmiersprache [12]. Sie wurde aus der Unzufriedenheit über *Java*, *C++* und *Python* heraus entwickelt, welche am häufigsten bei *Google* eingesetzt wurden. All diese Programmiersprachen haben Nachteile in *Googles Business Case*, deswegen wurde *Go* speziell für skalierbare Netzwerkdienste und *Cloud Computing* entwickelt. Aus diesem Grund besitzt *Go* auch eine native Möglichkeit für den einfachen Aufbau von *REST*-Schnittstellen. Da bei der Entwicklung von *Go* speziell aus den Fehlern in Performance und Sprachdesign aus anderen Sprachen gelernt wurde, verbindet *Go* die Vorteile der anderen Sprachen. Darunter fallen die starke und statische Typisierung, Objektorientierung, *Pointer* und eine verbesserte *Compiler-Effizienz*.

5.2.4.4 Vergleich

Diese drei Sprachen werden nun in den Aspekten der Performance, dem Sprachdesign, der Komplexität des Aufbaus einer REST-Schnittstelle, das Vorhandensein von *Frameworks*, die Erfahrung des Teams und eine vorhandene ausführliche Dokumentation analysiert und verglichen. Hierbei wird auf einer Punkteskala von 0 - 9 bewertet.

Tabelle 5.2: Vergleich zwischen Java, Python und Golang

Kriterien	Gewichtung	Java und Spring		Python und Flask		Golang	
		Punkte	Wertung	Punkte	Wertung	Punkte	Wertung
Performance	15%	6	0,9	3	0,45	9	1,35
Sprachdesign	25%	6	1,5	4	1	8	2
Aufbau einer REST-Schnittstelle	5%	7	0,35	8	0,40	9	0,45
Vorhandensein von Frameworks	15%	7	1,05	9	1,35	9	1,35
Erfahrung des Teams	20%	4	0,8	3	0,6	6	1,2
Dokumentation	20%	8	1,6	7	1,4	9	1,8
Summe	100%		6,2		5,2		8,15

Daraus und aus der durchgeföhrten Recherche, lässt sich schließen, dass *Golang* sich speziell bei den Punkten Performance und Sprachdesign durchsetzen kann. Ansonsten schneiden die verschiedenen Sprachen inklusive der teilweise benötigten *Frameworks* großteils ähnlich hoch ab. Zusammenfassend eignet sich die Verwendung von *Golang* am Besten als Programmiersprache für unser Projekt, da sie im numerischen Vergleich mit der höchsten Punktzahl abschnitt, aber auch durch die speziellen Hintergründe ihrer Entwicklung genau zu den Voraussetzungen passt.

5.2.5 Kommunikation und Datenformate

Wie bereits erwähnt, zeichnen sich REST-Schnittstellen unter anderem dadurch aus, dass sie ein einheitliches Datenformat voraussetzen. Aus diesem Grund stellen sich die Fragen:

Welche Datenformate eignen sich für einen konsistenten und performanten Datenaustausch zwischen Datenbank, REST-Schnittstelle und *Client*?

Welche Vor- und Nachteile bringen diese im Hinblick auf Performance, Softwarewartung und -evolution?

Um diese Fragen zu beantworten werden in den nächsten Kapiteln entsprechende Datenformate vorgestellt, beschrieben und analysiert.

5.2.5.1 JSON

JSON (JavaScript Object Notation) ist ein Textformat, welches für die Serialisierung von Daten genutzt werden kann [8]. Es umfasst 6 Datentypen, davon 4 Primitive und 2 Strukturen:

- *Strings*
- Zahlen
- *Booleans*
- *Null*
- *Arrays*
- Objekte

Auflistung 5.17: JSON - Datentypen

Wie so eine Serialisierung aussieht, wird in folgendem Beispiel ersichtlich:

```

1   {
2     "classes": [
3       {
4         "name": "1AHIT",
5         "lastYear": false,
6         "students": 35,
7         "class-rep": "Michaela Musterfrau"
8       },
9       {
10        "name": "5B HIT",
11        "lastYear": true,
12        "students": 25,
13        "class-rep": "Maximilian Frühmann"
14      }
15    ],
16    "date": "2021-09-01"
17  }

```

Auflistung 5.18: JSON Beispiel

Das in Auflistung 5.8 stehende Beispiel stellt ein Objekt mit zwei Feldern dar. Das erste Feld ist ein „*classes*“-Array, welches verschiedene Schulklassen beinhaltet. Diese Schulklassen haben jeweils einen *String* als Namen, einen *Boolean* als Feld, das angibt, ob es sich um eine Abschlussklasse handelt, die Anzahl der Schüler als Zahl und den Namen des Klassensprechers. Nachdem die Definition des *Arrays* abgeschlossen ist, wird noch ein weiteres Feld im obersten Objekt angegeben, welches das aktuelle Datum als *String* beinhaltet.

Der *JSON*-Standard umfasst genaue Regeln, wie mit den Datentypen, speziell mit *Strings* und Zahlen, umzugehen ist [8]. Über einen *JSON*-Generator kann ein *JSON*-Text auf Basis eines Dateninputs generiert werden. Wenn die Daten aus dem *JSON*-Textformat wieder extrahiert werden sollen, spricht

man vom „*parsen*“ von Daten.

Jede moderne Programmiersprache hat einen eigenen *JSON*-Parser implementiert oder es ist möglich einen über ein *Third Party Packages* zu laden.

5.2.5.2 XML

XML (Extensible Markup Language) ist ein strukturiertes textbasiertes Datenformat [3]. Es handelt sich hierbei um eine *Markup-Language*. Als Syntax von *XML* werden *Tags* verwendet. Ein <Tag> ist ein Anfangs-Tag und ein </Tag> ein End-Tag. Ebenfalls können Attribute in einem Anfangs-Tag definiert werden (<Tag attribut="wert"). *XML*-Dateien brauchen im Gegensatz zu *JSON*-Texten ein Wurzelement.

```
1  <school>
2      <classes>
3          <class lastYear="false" students="35" class-rep="null">1AHIT</class>
4          <class lastYear="true" students="25" class-rep="null">5BHT</class>
5      </classes>
6      <date>2021-09-01</date>
7  </school>
```

Auflistung 5.19: *XML* Beispiel

Das Beispiel definiert ein Wurzelement für die Datenstruktur namens „*school*“. Dieses hat ein Kind-element „*classes*“. In diesem sind einzelne „*class*“-Tags mit den nötigen Attributen und dem Namen der Klasse innerhalb der Tags. Dies wird als Alternative für die in *XML* nicht vorhandenen *Arrays* genutzt. Zuletzt wird wieder ein untergeordnetes „*date*“-Element mit dem aktuellem Datum als Inhalt hinzugefügt.

Das *XML*-Dateiformat kann entweder gültig oder wohlgeformt sein [3]. Dies beschreibt die Einhaltung des Syntax und der Regeln des *XML Standards*.

Des Weiteren besteht die Möglichkeit die Struktur von *XML*-Dateien anhand der *XML-Schema-Sprache* oder durch die *Document Type Definition* zu beschreiben. *XML* Dateien brauchen aus diesen Gründen unbedingt einen *Parser* für die Umwandlung.

5.2.5.3 CSV

CSV (*Comma-Separated Values*) ist ein Textformat, mit welchem man Tabellen textbasiert darstellen kann [46]. Hierbei werden Spalten durch "," (Beistriche) getrennt und Zeilen durch einen Zeilenumbruch (*CRLF*; *LF*). Die erste Zeile repräsentiert die Namen der Spalten. Eine Beschreibung einer Tabelle mit CSV sieht wie folgt aus:

```
1 Class,lastYear,students,class-rep  
2 1AHIT,false,35,null  
3 5BHIT,true,25,null
```

Auflistung 5.20: CSV Beispiel

Im Beispiel werden die beiden Klassen als zwei Datensätze in einer Tabelle aufgelistet. Auf Grund der Struktur von CSV ist es nur möglich den *Array*-Teil des Objektes darzustellen. Ansonsten ist der CSV-Standard recht simpel gehalten. Der letzte Punkt auf den der Standard hierbei eingeht ist, dass Zeichen um die Daten gesetzt werden können [46]. Sollten, diese nicht geschlossen werden kann es jedoch zu unvorhersehbaren Verhalten kommen.

5.2.5.4 Vergleich

Ein Vergleich zwischen den Datenformaten *JSON*, *XML* und CSV ist auf dem ersten Blick nicht zielführend. Wie aus den einzelnen Beschreibungen der Technologien oben entnommen werden kann, handelt es sich beim CSV-Format um Tabellen. Bereits bei der Wahl der dokumentenorientierten Datenbank *MongoDB* war der Hintergedanke relationale Datenbanken und ihrer Speicherstruktur, Tabellen, weitestmöglich zu vermeiden.

Wenn man sich nun auch das CSV Format und speziell das Beispiel anschaut, merkt man, dass die Umwandlung des ursprünglichen Objektes nur indirekt möglich war und als Auflistung der Klassen endete. Das Resultat ist, dass Objekte zu CSV-Tabellen mit vielen Spalten und einer Zeile werden und *Arrays* zu Tabellen mit wenigen Spalten und vielen Zeilen. Eine Darstellung beider Datentypen in einer Tabelle ist nicht möglich. Dies disqualifiziert CSV für den Einsatz als Datenformat im Projekt.

Vergleicht man nun *JSON* und *XML* so scheinen diese erst ähnlich. Jedoch besteht bei *XML* keine Möglichkeit *Arrays* zu definieren. Des Weiteren wird *XML* nicht nur als Datenformat zur Kommunikation genutzt, sondern hat noch weitere Nutzen als *Markup Language*. Demnach ist *XML* sehr überladen, speziell deswegen weil es einen eigenen speziellen *XML-Parser* braucht.

Auch *JSON* braucht einen sogenannten *Parser*, jedoch handelt es sich hierbei nur um das Einlesen des *JSON*-Textes und dem Generieren des Objekts. Eine Validierung wie in *XML* bleibt aus. Zuletzt ist *JSON* nicht nur schneller, sondern auch effizienter was den Verbrauch an *CPU*-Leistung und an Arbeitsspeicher angeht [36].

Aus diesen Gründen wird *JSON* als Datenformat zur Kommunikation zwischen den einzelnen Komponenten *Refundables* genutzt werden.

5.3 Frontend - Webapplikation als REST-Client

5.3.1 Einleitung

Das *Backend* muss mit dem *Frontend* verbunden werden. Es gibt unterschiedliche Möglichkeiten dies zu realisieren. Beim Realisieren muss darauf geachtet werden, dass eine Struktur vorhanden ist. Es werden zwei verschiedene Entwurfsmuster betrachtet und verglichen. Umgesetzt wird dann ein Entwurfsmuster mithilfe von *JavaScript*. Hier kann ein *JavaScript-Framework* zum Einsatz kommen. Dazu werden hier verschiedene *JavaScript-Frameworks* miteinander verglichen. Für die Verarbeitung der Daten ist es wichtig, Datenformate festzulegen. Die Aufbereitung der Elemente für das *Frontend* mit den Daten des *Backends* wird ebenfalls untersucht.

Die Fragestellung der Studie ist erstens, welche Unterschiede es bei der Datenrepräsentation und -manipulation zwischen den Entwurfsmustern *Model-View-ViewModel* (*MVVM*) und *Model-View-Controller* (*MVC*) gibt und zweitens, mit welchen Werkzeugen bzw. *JavaScript-Frameworks* die jeweiligen Methoden zugriffsperformant umgesetzt werden können.

5.3.2 Entwurfsmuster

Dieser Teil des Projektes wird in Verwendung eines Entwurfsmusters umgesetzt. Zwei *MV** Entwurfsmuster werden hierbei in Betracht gezogen: zum einen *MVVM* und zum anderen *MVC*. Es kommen diese zwei Entwurfsmuster in Frage, da *MVC* ein sehr bekanntes Entwurfsmuster ist und *MVVM* eine neuere und spezifischere Variante von *MVC* ist [50].

5.3.2.1 MVC

MVC ist ein Entwurfsmuster, mit dem eine Software in die drei Teile (*Model*, *View* und *Controller*) geteilt wird [31]. Das *Model* beinhaltet alle Daten der Software und auch alle Funktionen, die mit den Daten interagieren oder mit ihnen rechnen. Die *View* ist der Teil der Software, den Benutzer sehen und mit dem sie interagieren. Dieser Teil der Software beinhaltet keine wichtigen Daten oder Funktionen, welche Daten bearbeiten. Die *View* hört nur auf Benutzereingaben und zeigt die bereitgestellten Daten an. Der *Controller* ist der Teil der Software, der *Model* und *View* verbindet. Im *Controller* wird auf die Benutzereingaben reagiert und es werden die gewünschten Funktionen aus dem *Model* aufgerufen.

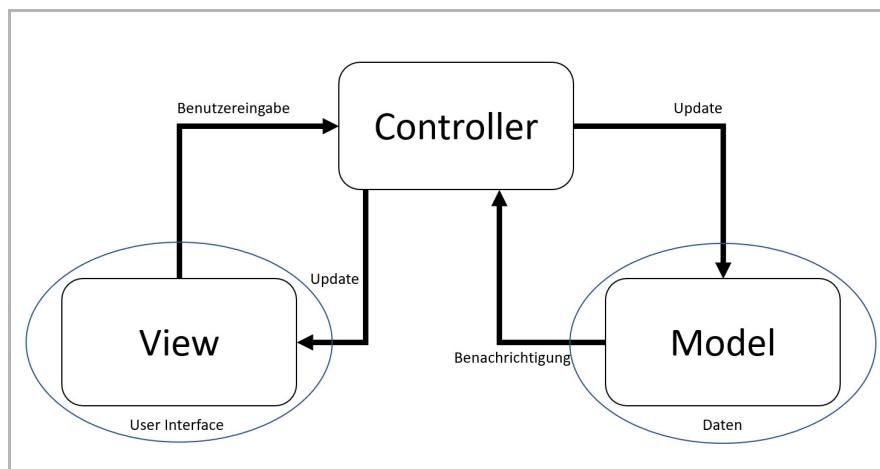


Abbildung 5.3: Übersicht über die Komponenten des *MVC-Patterns* und ihre Zusammenhänge

5.3.2.2 MVVM

MVVM oder auch *MVVC* ist ein Entwurfsmuster mit dem eine Software in drei Teile geteilt wird [50]. Jedoch wird bei *MVVM* die Software in *Model*, *View* und *ViewModel* aufgeteilt. Das *Model* beinhaltet, wie im konventionellen *MVC-Pattern*, alle wichtigen Daten und Funktionen. Die *View* ist, wie beim *MVC-Pattern*, der Teil der Software, mit dem der Benutzer interagiert. Das *ViewModel* ist ein Bindeglied zwischen *Model* und *View* [50]. Dabei stellt das *ViewModel* der *View* Funktionen zur Verfügung. Diese können auch Daten verändern bzw. mit Daten rechnen. Das *Model* kann über das *ViewModel* auch mit der *View* direkt interagieren. *MVVM* sieht nicht vor, dass ein *Controller* verwendet wird. Diese Funktion übernimmt zu einem gewissen Teil das *ViewModel* bzw. auch die *View* und das *Model*.

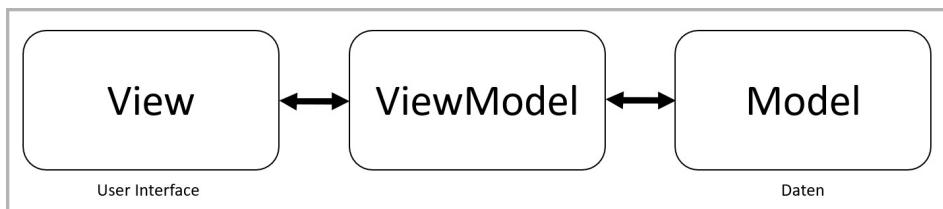


Abbildung 5.4: Übersicht über die Komponenten des *MVVM-Patterns* und ihre Zusammenhänge

5.3.2.3 Vergleich

Diese zwei Entwurfsmuster sind zwar sehr ähnlich, jedoch gibt es wichtige Unterschiede zwischen ihnen.

MVC ist ein Entwurfsmuster, welches auf einem niedrigen Level überall im Einsatz ist - z.B. bei Tastatureingaben [31]. Dieses Entwurfsmuster ist zwar schon lange verfügbar, jedoch kann es bei *Webapplikationen* zu Problemen führen.

Bei der Entwicklung einer *Webapplikation* ist es nicht einfach, das *Model* von dem *Controller* zu trennen. *MVVM* auf der anderen Seite zielt explizit auf *HTML5* ab [50]. Somit ist es bei webbasierten Applikationen zu bevorzugen.

5.3.3 Umsetzungsmöglichkeiten

Dieser Teil des Projekts wird mittels *JavaScript* umgesetzt. Hierbei kommen unterschiedliche *JavaScript-Frameworks* in Frage. Hier werden die verbreitetsten *JavaScript-Frameworks* vorgestellt und dann verglichen. *Angular* wird nicht in Betracht gezogen, da dies ein *Framework* für professionelle Projekte und demnach nicht für vergleichsweise kleine Projekte verwendbar ist, wegen dem Aufwand das *Framework* zu erlernen [2].

5.3.3.1 VueJS

VueJS ist ein progressives *JavaScript-Framework*. Dies bedeutet, dass *VueJS* in bereits bestehende Webseiten bzw. in webbasierte Software implementiert werden kann.

VueJS kann aber auch von Anfang an verwendet werden, wobei man hier mit den Bibliotheken von *VueJS* skalieren kann [54]. *VueJS* kann Teile einer Webseite in Komponenten aufteilen, um diese mehrmals zu verwenden, falls dies notwendig ist. Jeder dieser Komponenten besitzt seine eigene *HTML*-, *CSS*- und *JavaScript* Datei, die gebraucht werden, um diese Komponente zu rendern.

Um einzelnen Elementen *VueJS*-Funktionalität zu geben, kann auf diese Elemente folgender *Code* angewendet werden [54]:

```

1   <!DOCTYPE html>
2   <html lang="de">
3     <head>
4       <meta charset="UTF-8">
5       <meta name="viewport" content="width=device-width, initial-scale=1.0">
6       <title>Kurzes VueJS-Beispiel</title>
7     </head>
8     <body>
9       <h1>Überschrift</h1>
10      <div id="app">
11        <!-- Der Text, welcher weiter unten Definiert wird, wird hier
12          eingefügt -->
13        <!-- Ändert sich die Variable, dann ändert sich auch der Text auf
14          der Webseite -->
15        <p> {{ text }} </p>
16      </div>
17      <script src="https://unpkg.com/vue"></script>
18      <script>
19        const app = new Vue({
20          el: '#app',
21          data: {
22            text: 'Hier steht Text!'
23          }
24        })
25      </script>
    </body>
  </html>
```

Auflistung 5.21: Beispiel für *VueJS* Webseite

5.3.3.2 React

React ist eine *JavaScript*-Bibliothek, mit der man Benutzeroberflächen entwickeln kann [43]. *React* hat, wie *VueJS*, Komponenten. Dies bedeutet, dass man einzelne Elemente mehrfach verwenden kann, falls man dies benötigt.

React wurde von Facebook entwickelt und wurde 2013 als *Open-Source*-Lösung veröffentlicht.

Folgender Code beschreibt eine Beispiel-*React*-Webseite [43]:

```
1  <!DOCTYPE html>
2  <html lang="de">
3      <head>
4          <meta charset="UTF-8">
5          <meta name="viewport" content="width=device-width, initial-scale=1.0">
6          <title>Kurzes React-Beispiel</title>
7      </head>
8      <body>
9          <h1>Überschrift</h1>
10         <div id="likebuttoncontainer"></div>
11
12         <!-- Load React. -->
13         <!-- Note: when deploying, replace "development.js" with
14             "production.min.js". -->
15         <script src="https://unpkg.com/react@17/umd/react.development.js"
16             crossorigin></script>
17         <script src="https://unpkg.com/react-dom@17/umd/react-dom.development.js"
18             crossorigin></script>
19         <!-- Load our React component. -->
20         <script src="likebutton.js"></script>
21     </body>
22 </html>
```

Auflistung 5.22: Beispiel für *React* Webseite

In der letzten Zeile des *body*-Tags, wird auf *likebutton.js* referenziert. Dies ist eine Komponente und muss noch erstellt werden [43]:

```
1  'use strict';
2
3  const e = React.createElement;
4
5  class LikeButton extends React.Component {
6      constructor(props) {
7          super(props);
8          this.state = { liked: false };
9      }
10
11     render() {
12         if (this.state.liked) {
13             return 'You liked this.';
14         }
15
16         return e(
17             'button',
18             { onClick: () => this.setState({ liked: true }) },
19             'Like'
20         );
21     }
22 }
23
24 const domContainer = document.querySelector('#likebuttoncontainer');
25 ReactDOM.render(e(LikeButton), domContainer);
```

Auflistung 5.23: Beispiel für eine *React*-Komponente

5.3.3.3 Ohne Framework

Die Schnittstelle zwischen *Backend* und *Frontend* kann auch ohne ein *Framework* umgesetzt werden. Dies ist bei kleinen und leichten Applikationen von Vorteil, da keine umständlichen *JavaScript-Frameworks* aufgesetzt und richtig implementiert werden müssen. Ohne *Framework* implementiert man eine *JavaScript-Datei* in eine Webseite, um die gewünschte Funktionalität einzufügen. Mit folgendem *Code* kann eine Implementierung einer *JavaScript-Datei* umgesetzt werden:

```
1  <!DOCTYPE html>
2  <html lang="de">
3      <head>
4          <meta charset="UTF-8">
5          <meta name="viewport" content="width=device-width, initial-scale=1.0">
6          <title>Kurzes Plain-Beispiel</title>
7      </head>
8      <body>
9          <div id="inhalt">
10             <h1>Überschrift</h1>
11         </div>
12         <!-- Die Implementierung der JavaScript-Datei -->
13         <script src="javascript.js"></script>
14     </body>
15 </html>
```

Auflistung 5.24: Beispiel für Webseite ohne *Framework*

Es kann auch direkt in eine *HTML-Datei* *JavaScript* in ein *<script></script>* geschrieben werden.

5.3.3.4 Vergleich

Verglichen werden diese unterschiedlichen Umsetzungsmöglichkeiten an Gestaltung, Entwicklungszeit, Performanz und Lines of Code (LoC).

Die Webseite, die für den Vergleich umgesetzt werden soll, wird mit den unterschiedlichen Umsetzungsmöglichkeiten umgesetzt.



Abbildung 5.5: Die Beispielwebseite, die mit den Umsetzungsmöglichkeiten umgesetzt werden soll.

Folgende Tabelle zeigt die unterschiedlichen Werte bei der Umsetzung der Beispielwebseite:

Tabelle 5.3: Vergleich von *JavaScript-Frameworks*

Kriterium	Maximale Punkte	VueJS	React	Ohne Framework
Gestaltung	-	✓	✓	✓
Entwicklungszeit	9	8	6	9
Performanz	9	7	4	9
LoC	9	9	8	2
Gesamtpunkte	27	24	18	20

Kriterien:

Gestaltung

Die Gestaltung fällt in die Wertung, da hier angegeben wird, ob die Beispiel-Webseite so umgesetzt werden kann oder ob die Umsetzung in dieser Gestaltung nicht möglich ist.

Entwicklungszeit

VueJS: 40min

React: 55min

Ohne Framework: 20min

Die Entwicklungszeit zwischen den zwei *Frameworks* ist sehr ähnlich. Jedoch ist zu beachten, dass bei dem *VueJS-Framework* bereits mit Komponenten gearbeitet worden ist und dadurch die Entwicklungszeit höher ist als notwendig. Ohne *Framework* war die Entwicklungszeit am kürzesten, da nichts installiert, importiert oder aufgesetzt werden musste.

Performanz

VueJS: 112ms

React: 175ms

Ohne Framework: 21ms

Ohne Framework ist die Performanz sehr gut, da im Hintergrund keine *Frameworks* geladen werden müssen. Das JavaScript wird direkt ausgeführt. Bei *VueJS* und *React* werden viele Bibliotheken im Hintergrund geladen, wodurch die Performanz sinkt. Obwohl Komponenten bei der Umsetzung mittels *VueJS* verwendet worden sind, ist *VueJS* performanter als *React*.

Lines of Code

Die beiden *Frameworks* haben besonders gut abgeschnitten, da für die Umsetzung mit *VueJS* nur wenige *Codezeilen* geschrieben werden mussten. Durch die Funktionalität von *VueJS* können viele *Codezeilen* eingespart werden. *React* ist in dem Aspekt für das Projekt geeignet, da auch sehr wenige *Codezeilen* geschrieben werden mussten. Jedoch musste bei *React* deutlich mehr entwickelt werden, wodurch es nicht die vollen Punkte erreicht. Ohne *Framework* konnten weniger Punkte erreicht werden, da die Umsetzung im Vergleich nicht nur unübersichtlicher und komplizierter ist, sondern auch damit zu rechnen ist, dass die Umsetzung bei größeren Beispielen zu großen Problemen führen kann.

5.3.4 Aufbereitung der Daten

Die Daten, welche von der *REST*-Schnittstelle gelesen werden, sind im *JSON*-Format. Dies wird genauer im Unterabschnitt 5.2.5 erklärt. Um die Daten von der *REST*-Schnittstelle zu bekommen, wird *Axios* verwendet. Mit dieser Bibliothek kann eine *HTTP*-Anfrage gesendet und damit die Daten von der *REST*-Schnittstelle ausgelesen werden. Im folgenden *Code* wird eine Beispiel-Anfrage gesendet:

```

1  <html>
2      <head>
3          <!-- Import von Axios --&gt;
4          &lt;script src="https://unpkg.com/axios/dist/axios.min.js"&gt;&lt;/script&gt;
5          &lt;meta charset="UTF-8"&gt;
6          &lt;title&gt;Axios-Anfrage&lt;/title&gt;
7      &lt;/head&gt;
8      &lt;body&gt;
9          &lt;script&gt;
10             // Anfrage an den Server um die Daten zu bekommen
11             axios.get("http://localhost:3000/data").then(response =&gt; {
12                 // Die Daten, die zurückkommen werden in der Konsole ausgegeben
13                 console.log(response.data);
14             })
15         &lt;/script&gt;
16     &lt;/body&gt;
17 &lt;/html&gt;</pre>

```

Auflistung 5.25: Umsetzung einer *HTTP*-Anfrage mittels *Axios*

Die Daten, welche von der Anfrage zurückkommen, können dann verwendet werden, um *HTML*-Elemente zu erstellen oder zu verändern, um das *Frontend* dynamisch mit den Daten zu füllen.

Nachdem nicht alle Daten immer benötigt werden, kann bei den Anfragen die Endung verändert werden, um spezifische Daten abzufragen. Dies muss natürlich von der *REST*-Schnittstelle unterstützt werden. Eine Beispielanfrage für solch eine spezifische Anfrage ist wie folgt:

```

1  // Hier werden z.B.: die Daten von dem Benutzer 1234 abgefragt.
2  axios.get("http://localhost:3000/user/1234 infos").then(response => {
3      // Die Daten, die zurückkommen werden in der Konsole ausgegeben
4      console.log(response.data);
5  })
```

Auflistung 5.26: Umsetzung einer *Axios* Anfrage unter Verwendung eines genauen Pfades

Die praktische Durchführung ist, wie bereits im vorherigen Kapitel besprochen, bei den Umsetzungsmöglichkeiten unterschiedlich.

5.3.5 Fazit

Nach dieser Arbeit ist die Entscheidung für eine Umsetzungsmöglichkeit einfach. *MVVM* ist genau auf webbasierte Anwendungen zugeschnitten und bei *VueJS* kann durch die vordefinierten Komponenten viel Zeit und Arbeit gespart werden. *VueJS* hat im Vergleich zu *React* den Vorteil, dass hier bereits mehr Erfahrung in der Umsetzung besteht, woraus folgt, dass *VueJS* die effizienteste Variante für die Umsetzung ist.

Kapitel 6

Konzept

6.1 Frontend

In diesem Kapitel wird die Erstellung des Konzepts für das *Frontend* erläutert. Dabei müssen unter anderem Faktoren, wie die Zielgruppe, unsere Lehrerschaft, beachtet werden. Im folgenden Kapitel werden die theoretischen Hintergründe des Konzeptes dargestellt und erörtert. Im Anschluss werden die ersten Entwürfe mittels *Mockups* erstellt.

6.1.1 Theoretische Konzeption

Da die Zielgruppe „Lehrer“ nicht gleichaltrig ist, sondern die darin enthaltenen Personen ein Alter von 25 bis 65 Jahren haben, gibt es keine Literatur zur „perfekten Usability für Lehrpersonal“. Demnach richten wir uns nach den Büchern „Don't Make Me Think: A Common Sense Approach to Web Usability“ von Steve Krug und „101 UX Principles“ von Will Grant. Um eine gute *Usability* für die Lehrer des TGMs zu gewährleisten, werden wir den Prozess der Entwicklung in engem Kontakt zu den Lehrern durchführen und das Produkt auch mit Technik nicht versierten Anwendern testen.

6.1.2 Visuelle Konzeption

Login-Seite

Wie man auf der Abbildung 6.1 erkennen kann, ist die *Login-Seite* sehr schlicht gehalten. Neben dem Schriftzug „Refundable“ soll das Formular zum Einloggen dargestellt sein. Da die Plattform nur für das Personal des *TGMs* zugänglich sein soll, wird bei der Eingabe der E-Mail Adresse automatisch eine „@tgm.ac.at“ Endung angefügt. Wird der „Passwort vergessen?“ Link betätigt, soll automatisch auf die Seite des *TGMs* verwiesen werden, da die Daten der Lehrer über *LDAP* abgerufen werden. Bei der Eingabe von validen Daten und dem Betätigen des „Anmelden“ Buttons wird der Benutzer auf die *Startseite* weitergeleitet.

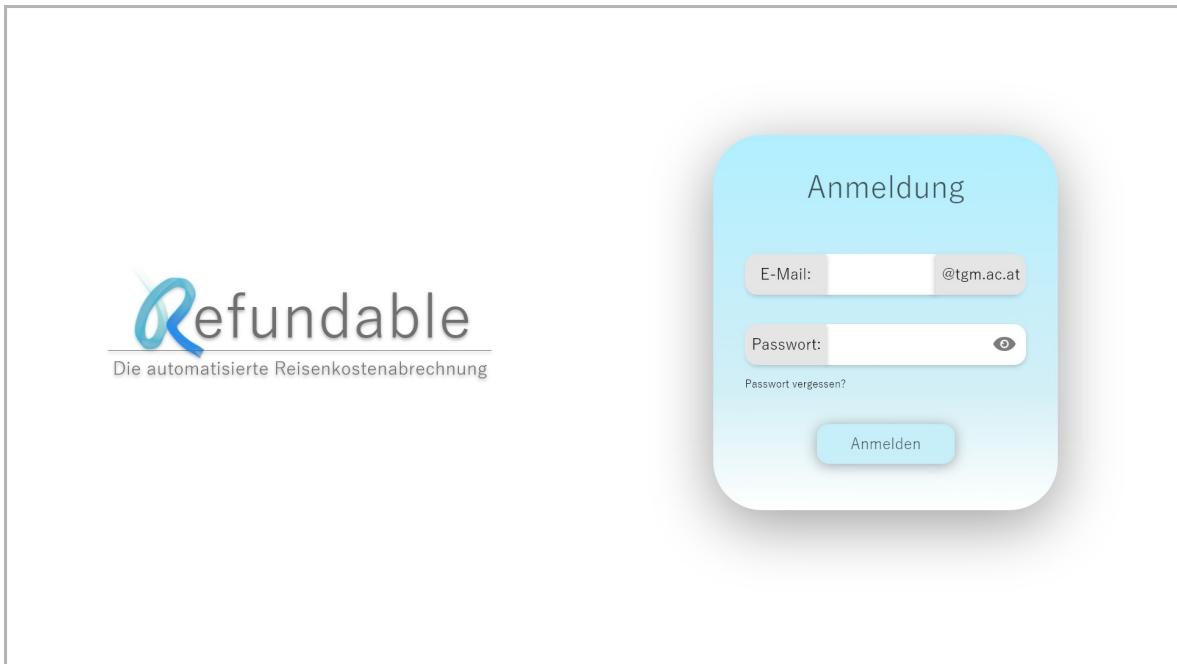


Abbildung 6.1: Das Mockup der Login Seite

Zur Veranschaulichung der oben genannten Prozesse ist in Abbildung 6.2 ein beispielhaftes *Use-Case Diagramm* zu sehen:

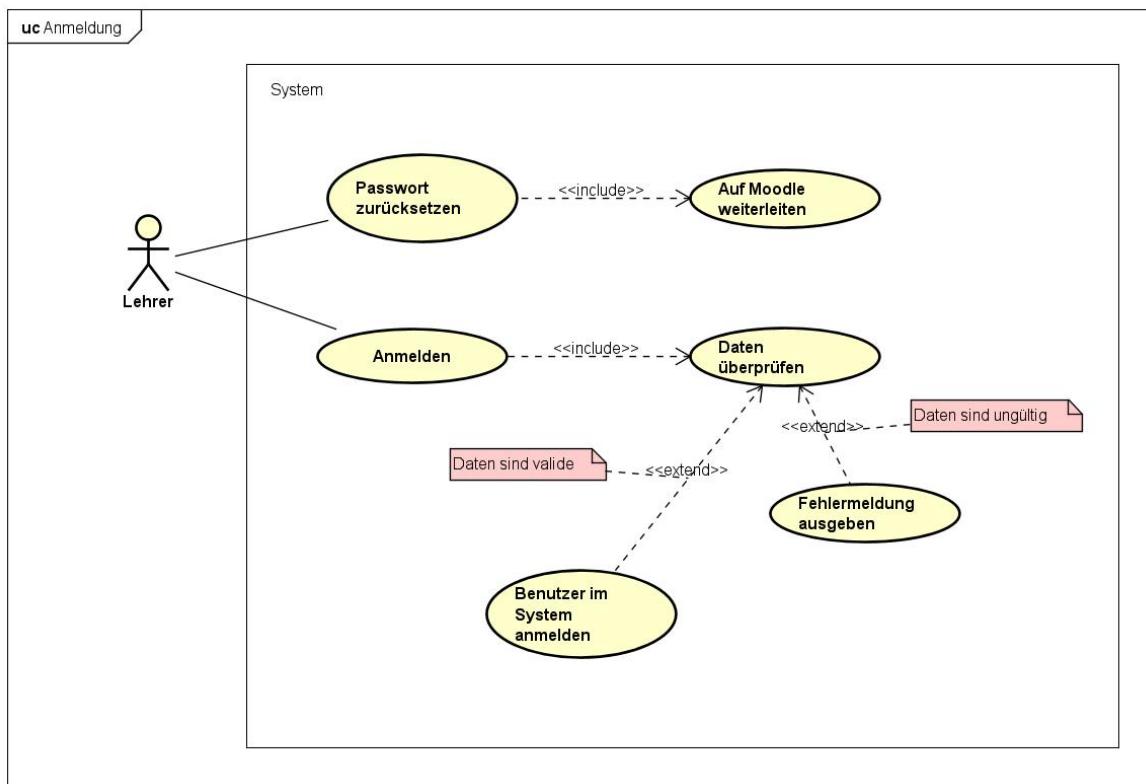


Abbildung 6.2: Das Use-Case Diagramm der Login Seite

Startseite

Die *Startseite* soll möglichst einfach gestaltet sein, damit selbst Lehrer, die keinen Bezug zur Technik haben, sich auf den ersten Blick auskennen. Links auf der Abbildung 6.3 kann man entweder einen neuen Antrag erstellen, sich alle Anträge oder nur alle aktiven Anträge anzeigen lassen. Auf der rechten Seite soll man Neuigkeiten zu seinen aktiven Anträgen sehen können.

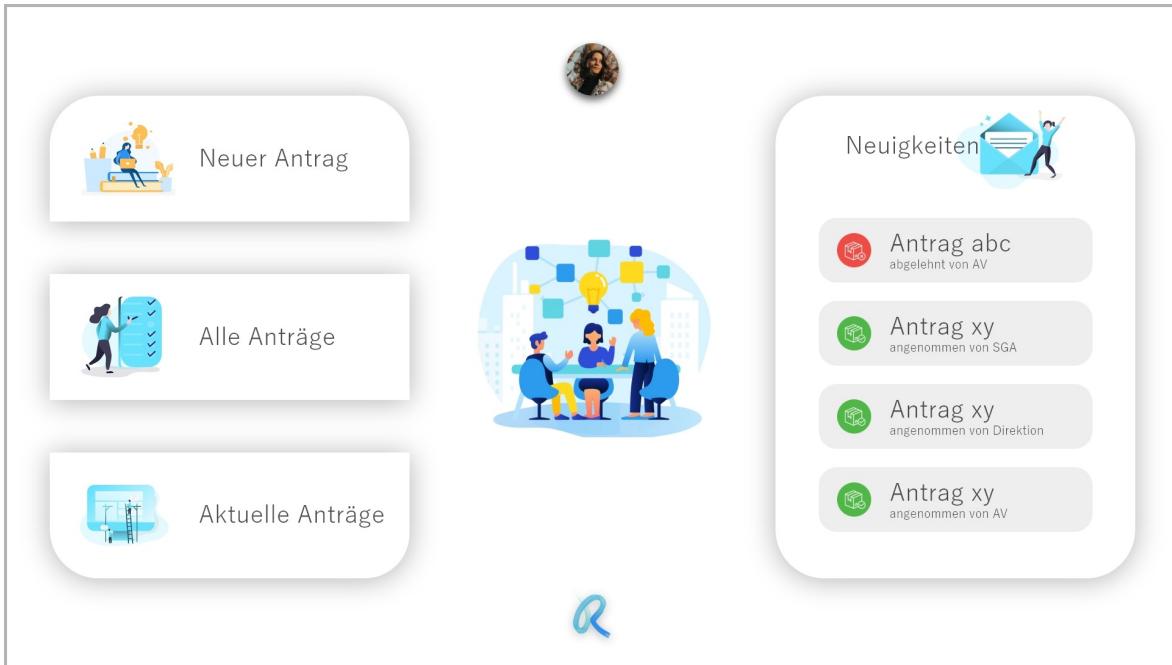


Abbildung 6.3: Das Mockup der Startseite, nachdem man sich eingeloggt hat

Zur Veranschaulichung der oben genannten Prozesse ist in Abbildung 6.4 ein beispielhaftes *Use-Case Diagramm* zu sehen:

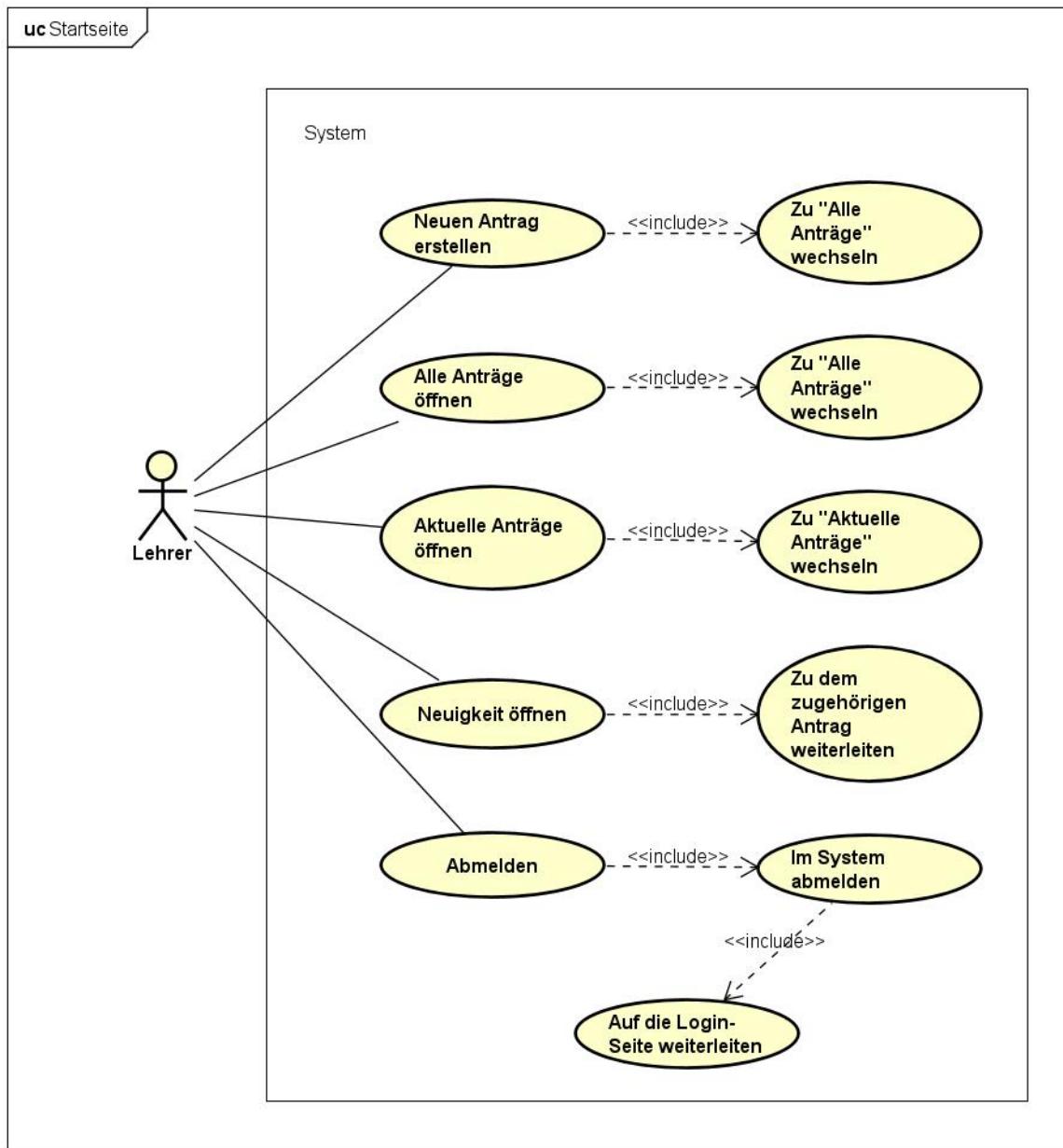


Abbildung 6.4: Das Use-Case Diagramm der Startseite

Neuer Antrag

Nachdem man auf den Button zum Erstellen eines *neuen Antrages* gedrückt hat, soll man zu einer Auswahl kommen, in der man auswählen kann, welche Art von Antrag man erstellen möchte (Abbildung 6.5). Danach sollen je nach Antragsart, wie man auf der Abbildung 6.6 sieht, Eingabefelder erscheinen, in denen die benötigten Daten zu dem Antrag eingetragen werden sollen.

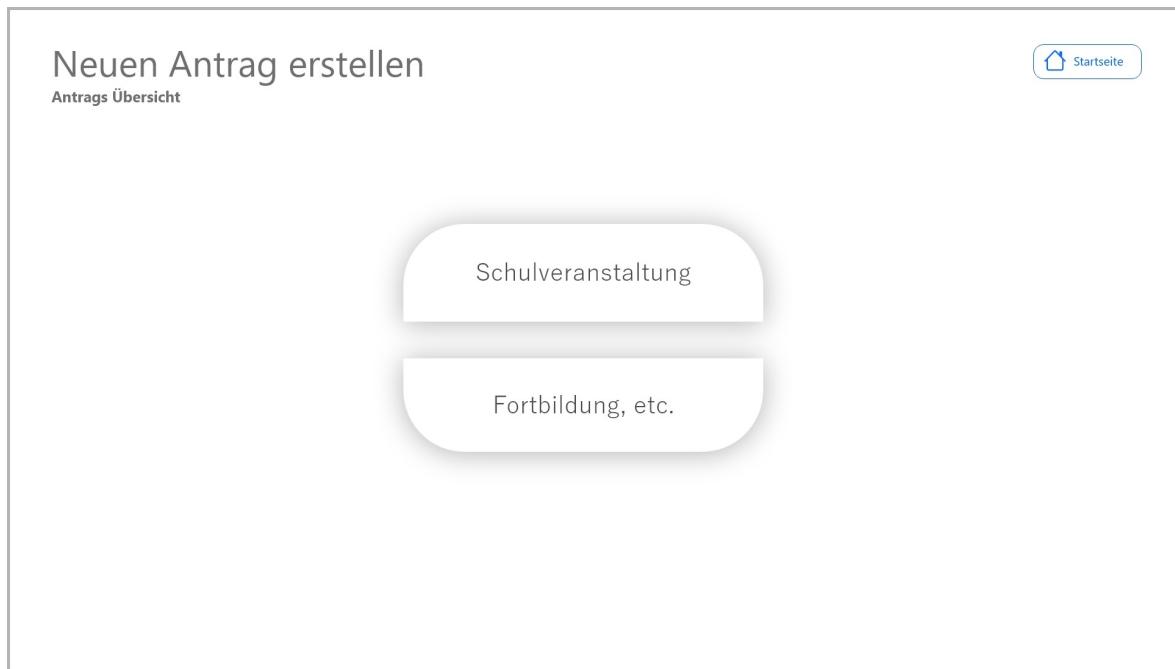


Abbildung 6.5: Das Mockup der Seite, auf der man eine Antragsart auswählen kann

The mockup shows a form for creating a new school event. At the top left is the title 'Neuen Antrag erstellen'. At the top right is a 'Startseite' button. Below the title is a link 'Antrags Übersicht / Schulveranstaltung - Allg. Infos'. The form consists of several input fields:

Bezeichnung	Sommersportwoche
Startdatum	<input type="button"/> Datum auswählen
Startzeit	<input type="button"/> Zeit auswählen
Enddatum	<input type="button"/> Datum auswählen
Endzeit	<input type="button"/> Zeit auswählen
Begleitpersonen	<input type="button"/> zaks <input type="button"/> dold
Klassen	<input type="button"/> 5bhit

At the bottom of the form are three small black dots.

Abbildung 6.6: Das Mockup zum erstellen einer neuen Schulveranstaltung

Zur Veranschaulichung der oben genannten Prozesse ist in Abbildung 6.7 ein beispielhaftes *Use-Case Diagramm* zu sehen:

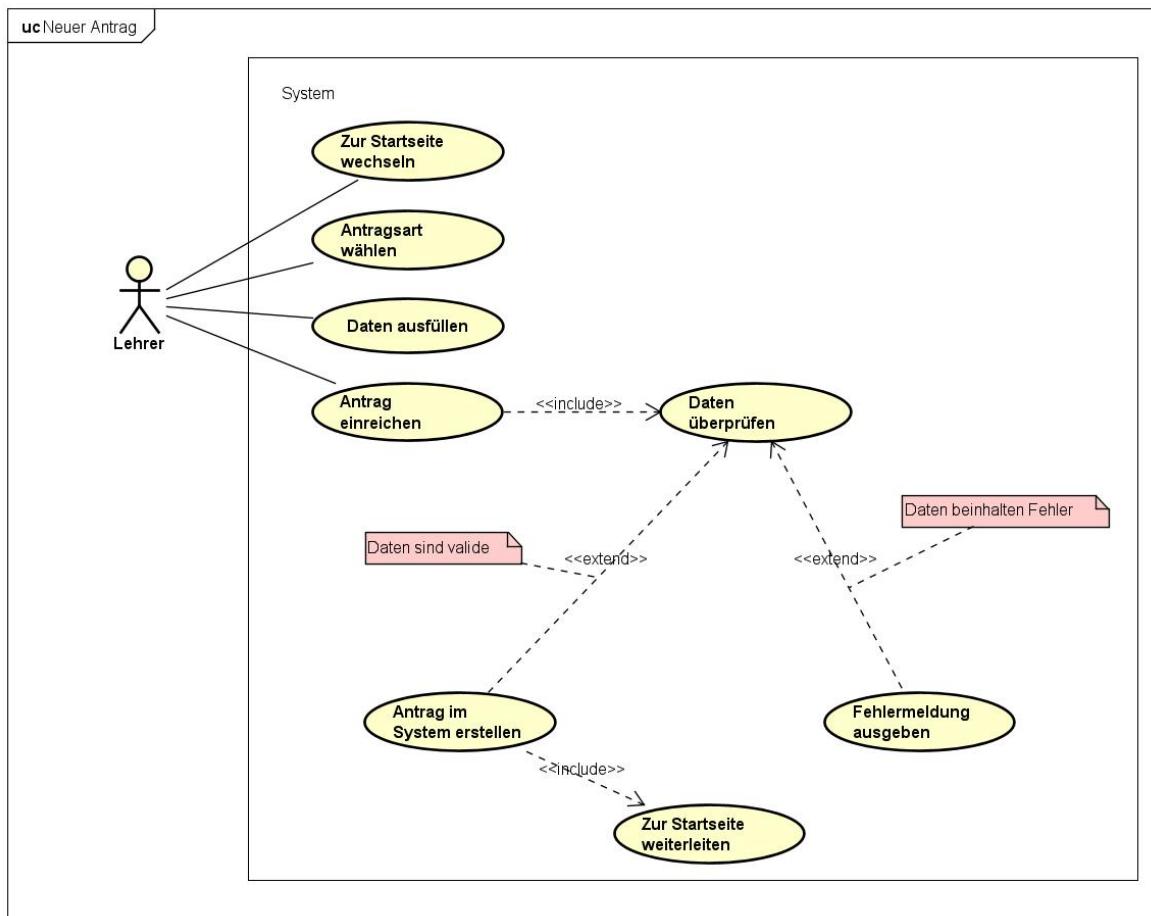


Abbildung 6.7: Das *Use-Case Diagramm* vom erstellen eines neuen Antrages

Ansicht alle Anträge

Auf der *alle Anträge* Seite, sieht man alle Anträge, mit denen man in Verbindung steht. Akzeptierte, abgelehnte Anträge und jene, die noch in Bearbeitung sind, werden hier angezeigt und können über das Menü auf der rechten Seite geöffnet werden (Abbildung 6.8).

Alle Anträge				
Begriff eingeben	Suchen			
Titel	Leiter	Einreichdatum	Status	Aktionen
Sommersportwoche	Stefan Zakall	01.01.2020	In Bearbeitung	<button>Öffnen</button>
Sportwoche	Dominik Dolezal	02.02.2020	Abgelehnt	<button>Öffnen</button>
Sportwoche	Dominik Dolezal	02.02.2019	Akzeptiert	<button>Öffnen</button>

Abbildung 6.8: Das Mockup, welches alle Anträge einer gewissen Person veranschaulicht

Zur Veranschaulichung der oben genannten Prozesse ist in Abbildung 6.9 ein beispielhaftes *Use-Case* Diagramm zu sehen:

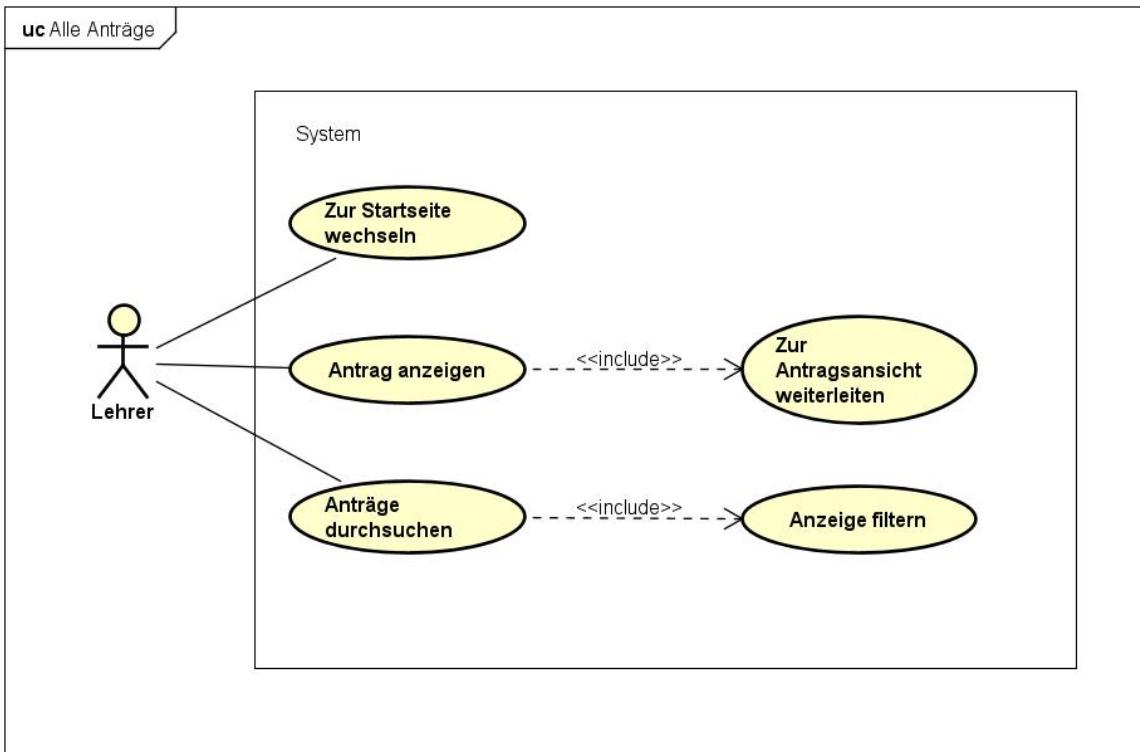


Abbildung 6.9: Das *Use-Case* Diagramm für die „Alle Anträge“ Seite

Ansicht aktive Anträge

Auf der *aktive Anträge* Seite sieht man alle Anträge, die aktiv sind (also in Bearbeitung oder vorerst abgelehnt) und mit denen man in Verbindung steht. Diese können über das *Menü* auf der rechten Seite geöffnet werden (Abbildung 6.10).

Aktive Anträge				
Begriff eingeben			Suchen	
Titel	Leiter	Einreichdatum	Status	Aktionen
Sommersportwoche	Stefan Zakall	01.01.2020	In Bearbeitung	Öffnen
Sportwoche	Dominik Dolezal	02.02.2020	Abgelehnt	Öffnen

Abbildung 6.10: Das Mockup, welches alle aktiven Anträge einer gewissen Person veranschaulicht

Zur Veranschaulichung der oben genannten Prozesse ist in Abbildung 6.11 ein beispielhaftes *Use-Case* Diagramm zu sehen:

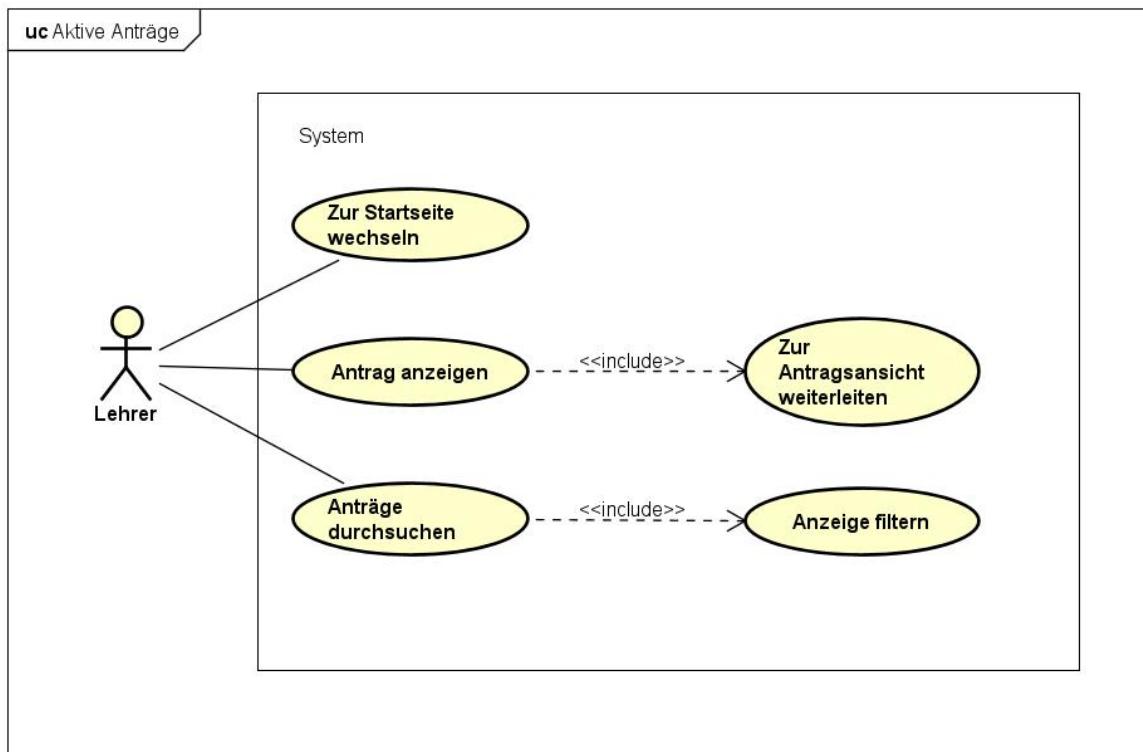


Abbildung 6.11: Das *Use-Case* Diagramm für die „Aktive Anträge“ Seite

Administrator Ansicht

Auf der *Administrator* Seite können bestimmte Nutzer (wie AV, Rechnungsstelle, etc.) Anträge verwalten. Wie man auf der Abbildung 6.12 sehen kann, werden dem Benutzer einige wichtige Daten in einer durchsuchbaren und filtrierbaren *Tabelle* angezeigt. Um den Antrag anzunehmen oder abzulehnen, kann der „Öffnen“ *Button* auf der rechten Seite der *Tabelle* betätigt werden.

Aktive Anträge					
Titel	Antragssteller	Einreichdatum	Art	Status	Aktionen
Sommersportwoche	Stefan Zakall	01.01.2020	Schulveranstaltung	Akzeptierungsphase	<button>Öffnen</button>
Sportwoche	Dominik Dolezal	02.02.2020	Krankschreibung	Rechnungsphase	<button>Öffnen</button>
Skilehrer Seminar	Stefan Zakall	01.01.2020	Fortbildung	Akzeptierungsphase	<button>Öffnen</button>

Abbildung 6.12: Das Mockup, der Admin Ansicht, zum akzeptieren und ablehnen von Anträgen

Zur Veranschaulichung der oben genannten Prozesse ist in Abbildung 6.13 ein beispielhaftes *Use-Case* Diagramm zu sehen:

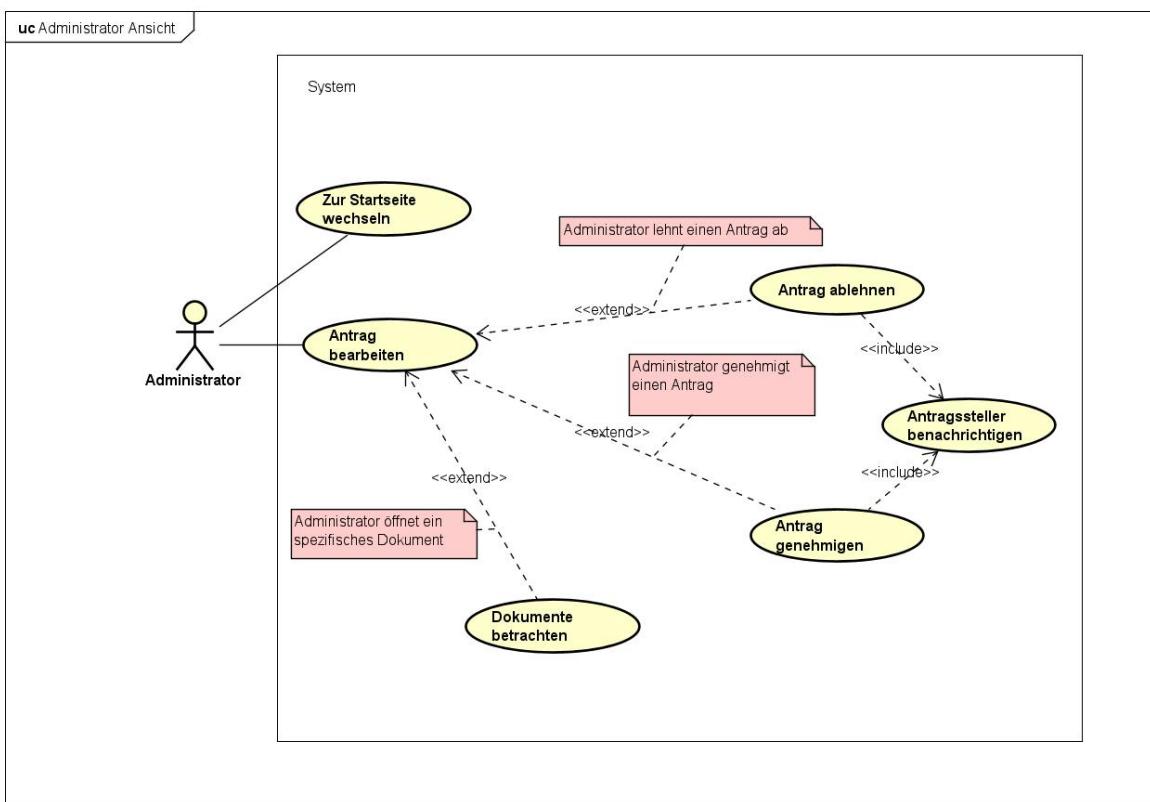


Abbildung 6.13: Das *Use-Case* Diagramm für die Administrator Ansicht

Antragsansicht

Hat der Nutzer einen bestimmten Antrag oder eine Neuigkeit zu einem Antrag ausgewählt, wird er auf die passende *Antragsseite* weitergeleitet (Abbildung 6.14). Auf dieser Seite ist es möglich, vorhandene Daten zu bearbeiten. Außerdem kann der Nutzer *PDF-Dateien* zu allen vorhandenen Anträgen generieren.



Abbildung 6.14: Das Mockup, welches einen eingereichten Antrag veranschaulicht

Zur Veranschaulichung der oben genannten Prozesse ist in Abbildung 6.15 ein beispielhaftes *Use-Case* Diagramm zu sehen:

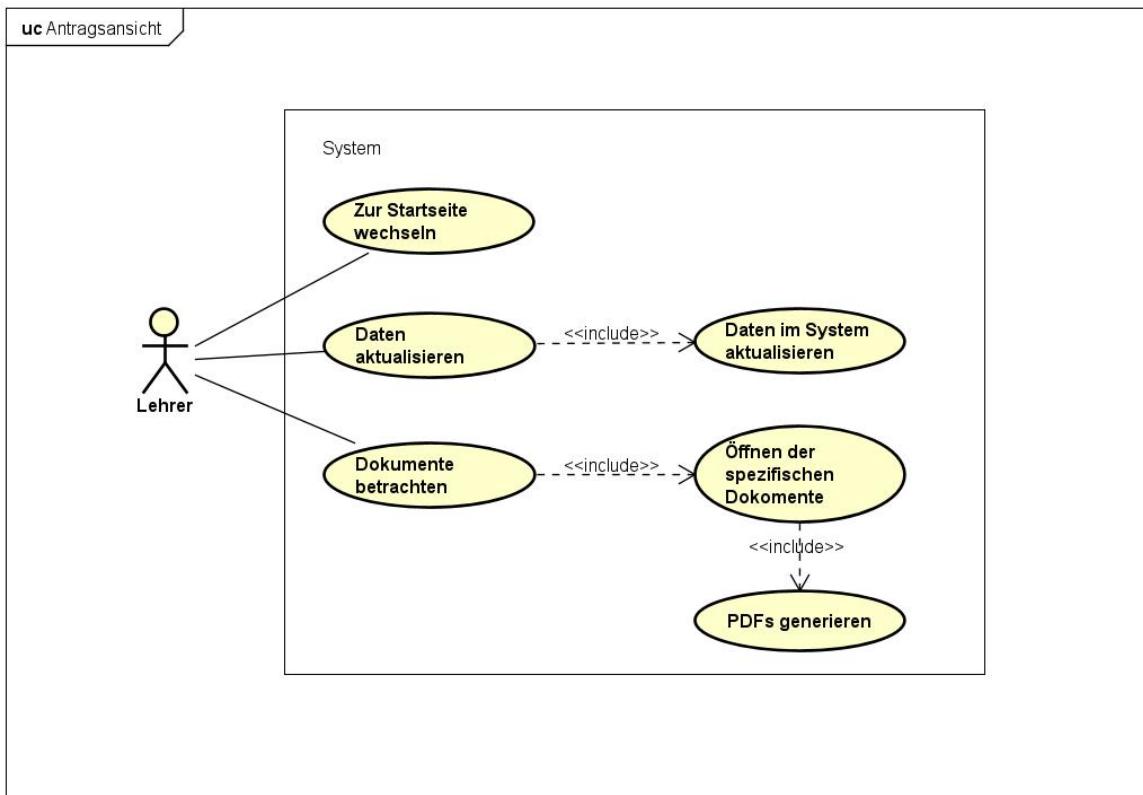


Abbildung 6.15: Das *Use-Case* Diagramm für die Antragsansicht

6.2 Backend und Infrastruktur

6.2.1 Einleitung

Der Anspruch an die Infrastruktur und an das *Backend* besteht darin, die Software als ganzes zu steuern und zu erhalten. Die Infrastruktur muss einfach aufzubauen und zu starten sein. Die folgenden Kapiteln erläutern nach der generellen Analyse von Lösungsmöglichkeiten im Abschnitt 5.2 nun einen ausgearbeiteten Plan, wie die später folgende Implementierung aussehen soll. Hierfür werden Graphiken oder Tabelle verwendet. Sie sollen ausreichend Informationen über Abläufe oder verschiedene Funktionalitäten vorgeben, um diese entsprechend daraufhin zu implementieren. Des Weiteren sollen die genormten Diagrammarten dazu dienen das komplexe *Backend* und die Infrastruktur einfach und übersichtlich darzustellen, um speziell die Schnittstellenherausforderungen verständlich zu machen.

6.2.2 Infrastruktur

Die Infrastruktur enthält jegliche Dienste und Anwendungen, welche von der Software in der Projektumgebung benötigt werden. Um diese Dienste einfach mit aufbauen zu können wird eine Virtualisierung über *Container* mittels *Docker* umgesetzt. Diese virtuellen Instanzen, verbunden in einem *Docker-Compose Stack*, werden über ein eigenes Skript gesteuert.

6.2.2.1 Docker Stack

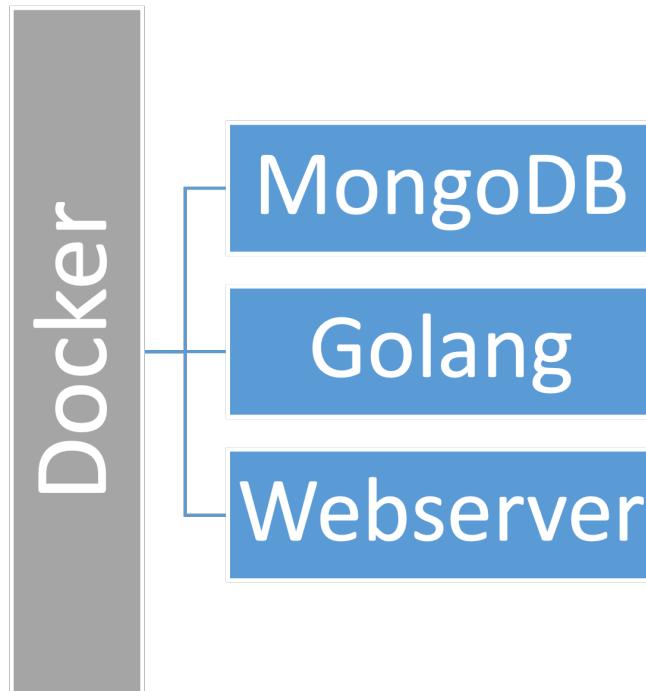


Abbildung 6.16: Übersicht über die Container des Docker-Compose Stack

Diese in *Docker Container* befindlichen Dienste und Anwendungen werden benötigt um die Software funktionsfähig bereitzustellen. Der automatische Aufbau dieser *Container*-Infrastruktur geschieht auf Basis einer *Docker-Compose* Konfiguration automatisch.

Docker Container stellen eigene Umgebungen dar, welche nach dem Stoppen oftmals vollständig zerstört und gelöscht werden. Die Daten, auf die die Software und auch die anderen Anwendungen zugreifen müssen, sind jedoch über die Lebenszeit eines *Docker Containers* hinaus zu persistieren. Um dies garantieren zu können, werden *Volumes* in die *Container* eingehängt, sodass die Daten prinzipiell außerhalb der *Container* gespeichert werden und nur von außen in die Umgebungen eingehängt werden. Dies ist bei dem *MongoDB Container* erforderlich um die Daten, welche in der Datenbank gespeichert werden, permanent zu speichern. Ebenso ist es erforderlich die generierten Dateien des *Golang-Backends* zu persistieren, um auf diese später wieder zugreifen zu können. Auch werden im *Backend* diverse gleichbleibende Passphrasen, beispielsweise zum Signieren von Tokens, auf die gleiche Art und Weise gespeichert.

Die Zugangsdaten für den Nutzeraccount, welche beim Installieren abgefragt werden (siehe Abbildung 6.17), der Datenbank werden über *Dockers Secrets-Management* sowohl in den Datenbank-, als auch in den *Golang-Container* eingehängt, damit sich dieser zur Datenbank verbinden kann. In der Datenbank müssen initial neben den Datenbanknutzern auch die Datenbankinstanz selbst angelegt werden. Dies geschieht über ein eigenes zum Start ausführendes Skript.

Das *Frontend* selbst besteht aus zwei Ebenen. Zu Beginn wird das *Frontend* selbst kompiliert und installiert. Hierfür gibt es einen eigenen *Container*, der hierfür eine Umgebung bietet. Danach wird der eigentliche *Webserver-Container* gestartet und die zuvor kompilierten Dateien werden in den neuen *Container* kopiert. Ebenfalls wird eine angelegte Konfigurationsdatei, die den *Webserver* Dienst weitergehend konfiguriert, in das entsprechende Verzeichnis hinzugefügt.

6.2.2.2 Steuerungsskript

Das Steuerungsskript ermöglicht die genaue und einfache Steuerung der Infrastruktur. Hierfür implementiert das Steuerungsskript folgende Submethoden mit entsprechenden Abläufen und Funktionalitäten:

Installationsvorgang:

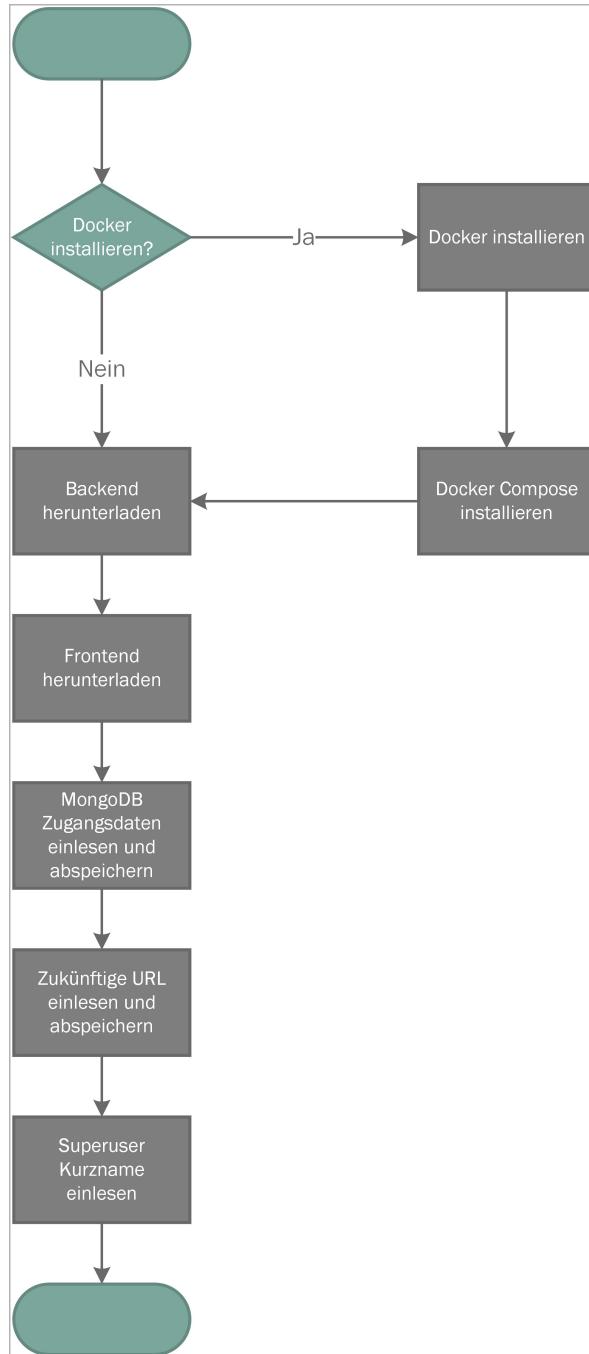


Abbildung 6.17: Flussdiagramm über den Installationsvorgang

Die Idee hinter der Etablierung eines Installationsvorganges soll die Einfachheit der Steuerung der Software hervorheben. Das *Install-Repository* beinhaltet hierbei lediglich eine grundlegende Ordnerstruktur und das Steuerungsskript selbst. Durch den Installationsvorgang werden die restlichen benötigten Daten automatisch heruntergeladen und installiert.

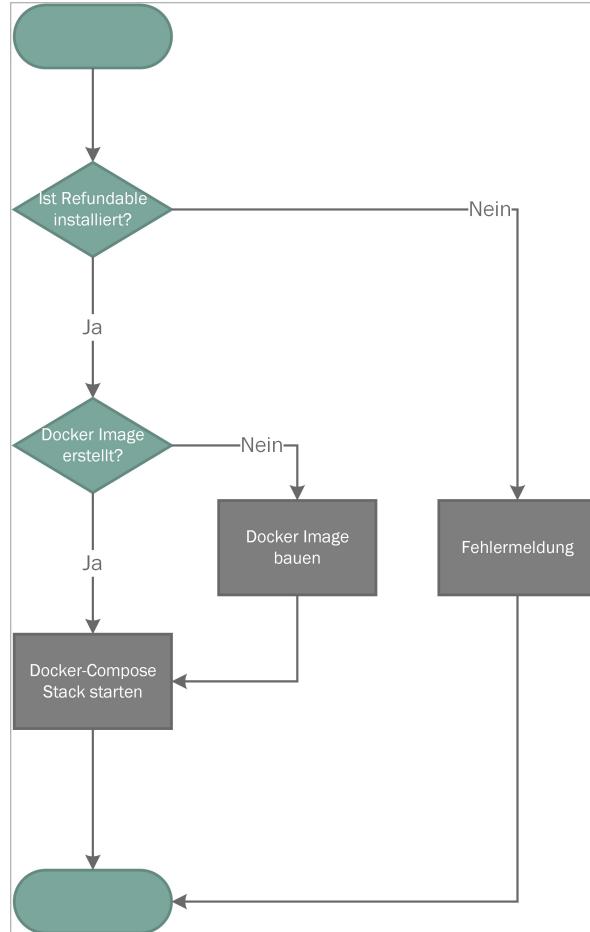
Startvorgang:

Abbildung 6.18: Flussdiagramm über den Startvorgang

Der Startvorgang baut die *Docker-Images*, sofern diese noch nicht vorhanden sind, grundlegend auf Basis der installierten Dateien auf und startet daraufhin den gesamten Stack auf einmal. Voraussetzung hierfür ist, dass der Installationsvorgang bereits abgeschlossen ist und die heruntergeladenen Dateien vorhanden sind.

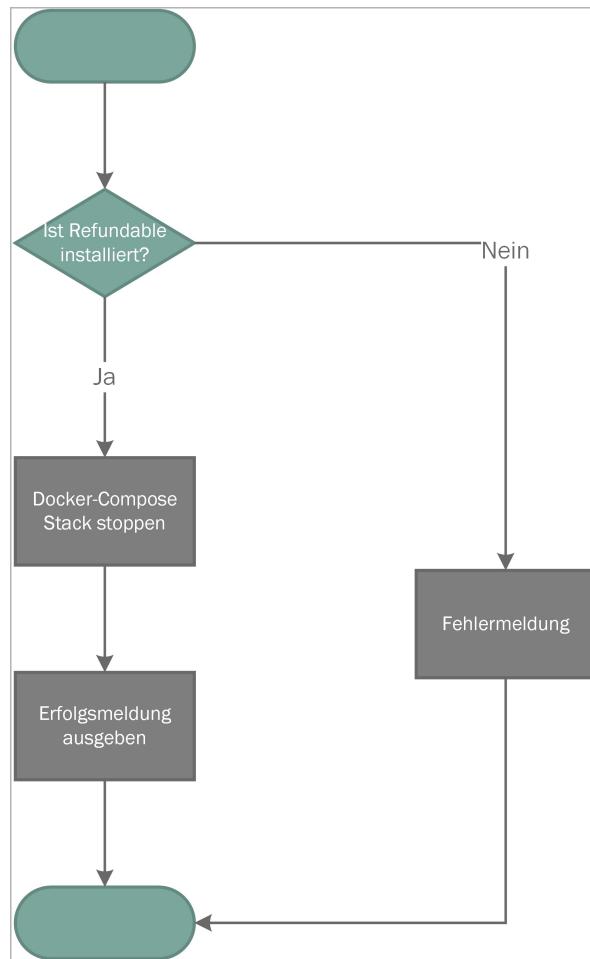
Stoppvorgang:

Abbildung 6.19: Flussdiagramm über den Stoppvorgang

Der Stoppvorgang beendet die laufenden *Docker-Container* und entfernt diese aus der *Docker* Umgebung. Dies führt dazu, dass garantiert wird, dass die sich wieder aufbauende Umgebung tatsächlich neu ist, und nicht nur eine alte Instanz wiederverwendet wird.

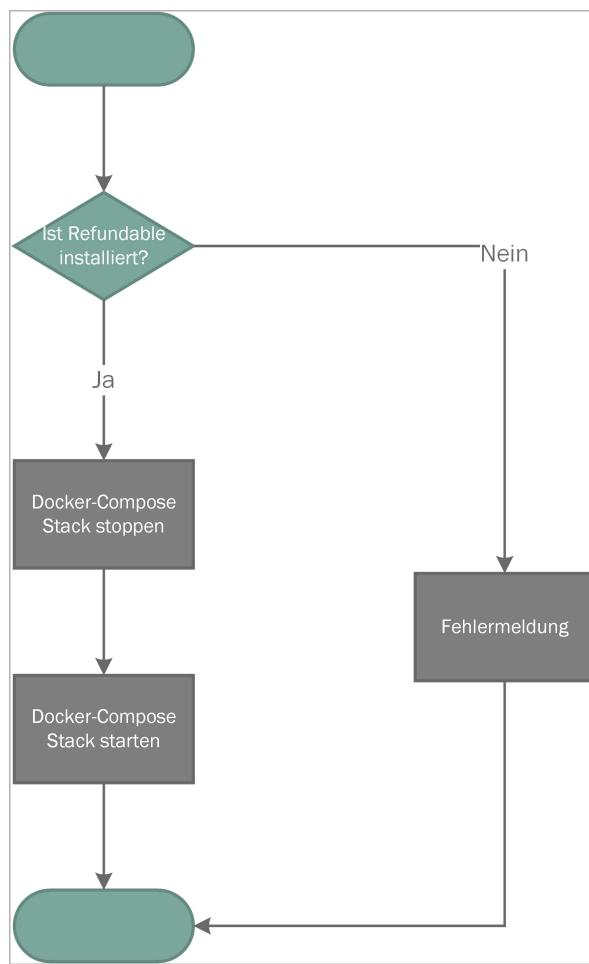
Neustartvorgang:

Abbildung 6.20: Flussdiagramm über den Neustartvorgang

Der Neustartvorgang kombiniert den Start- und Stoppvorgang und führt diese beiden auf einmal aus. Er wird hauptsächlich dafür genutzt, um neue Konfigurationen, welche nur beim Start des Systems eingelesen werden, von der Software übernehmen zu lassen.

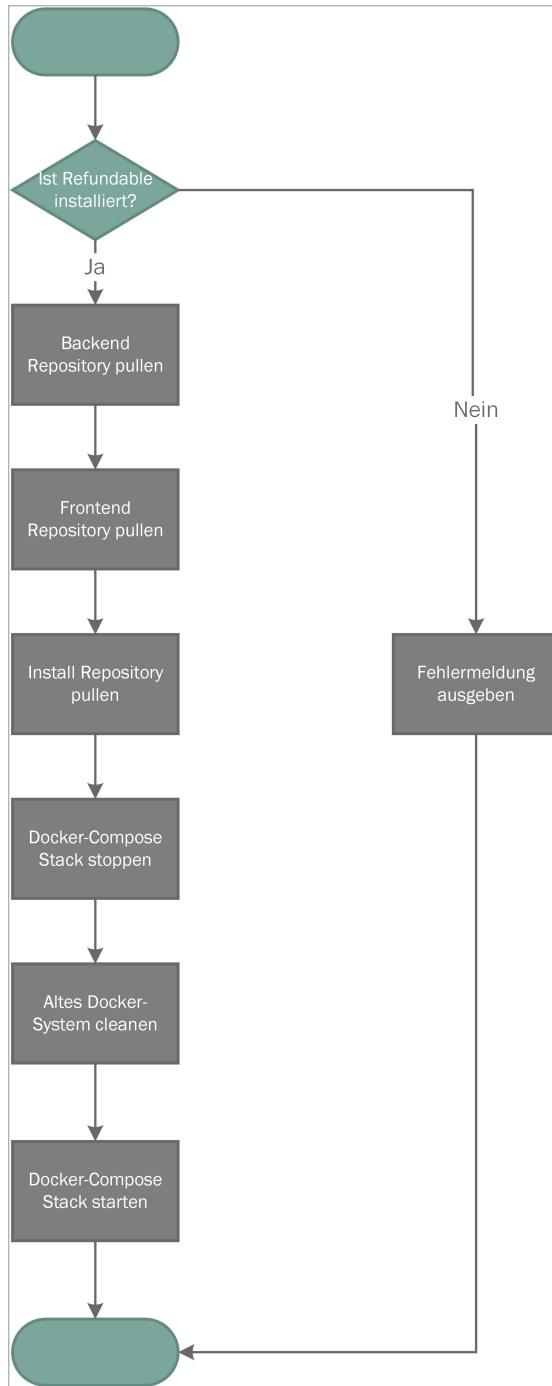
Aktualisierungsvorgang:

Abbildung 6.21: Flussdiagramm über den Aktualisierungsvorgang

Der Aktualisierungsvorgang ist dem Installationsvorgang sehr ähnlich. Da es sich bei den installierten Dateien um solche aus *Git-Repositories* handelt, können diese auch einfach aktualisiert werden. Daraufhin wird nur noch ein Neustart des Systems durchgeführt.

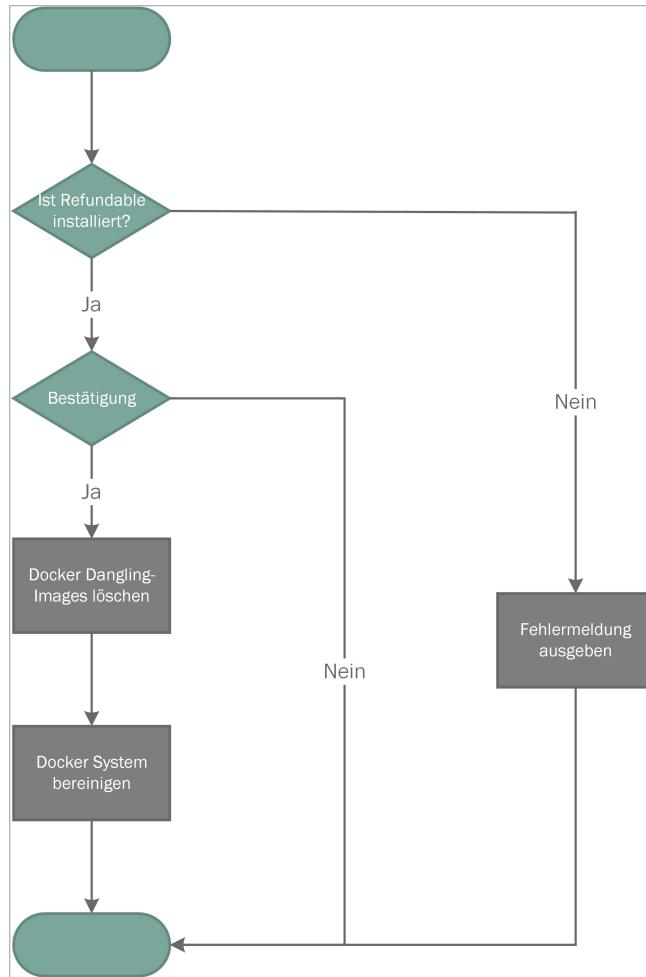
Bereinigungsvorgang:

Abbildung 6.22: Flussdiagramm über den Bereinigungsvorgang

Der Bereinigungsvorgang nutzt die *Docker* Funktionen zum Löschen von nicht mehr benutzten *Docker Images* (*Dangling Images*). Nachdem werden die nicht mehr benutzten Ressourcen über die entsprechende *Docker*-Funktion aus dem System und der *Docker* Umgebung entfernt.

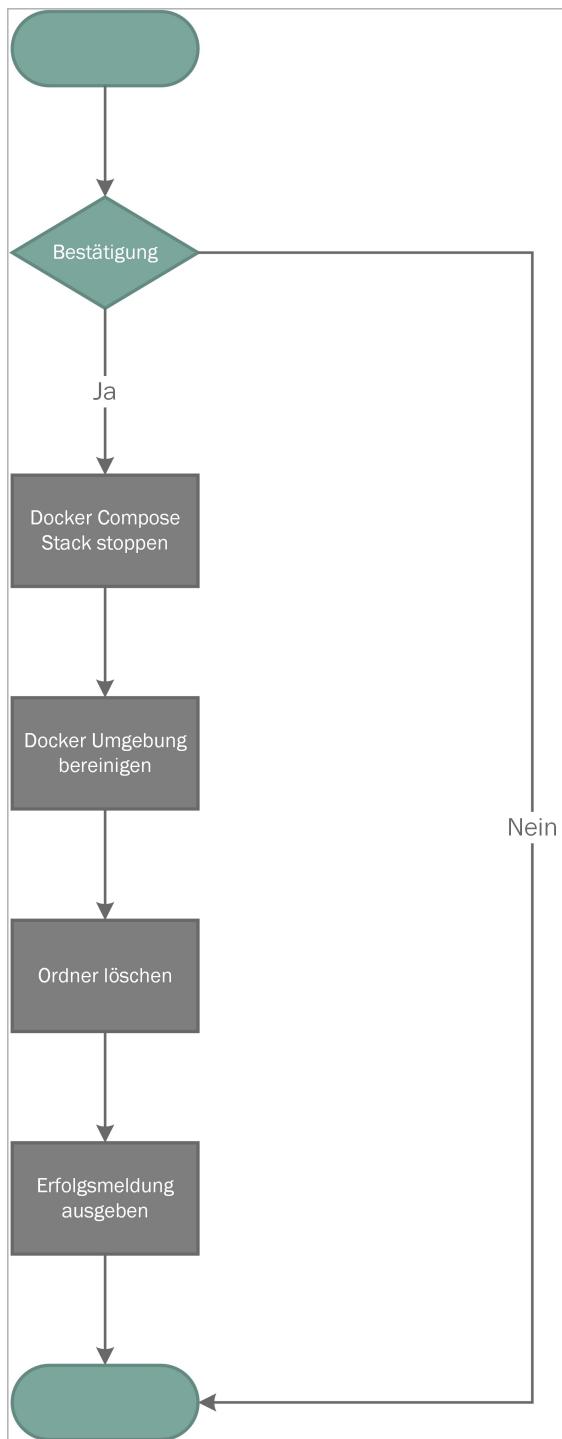
Deinstallation:

Abbildung 6.23: Flussdiagramm über die Deinstallation

Der Vorgang hinter der Deinstallation löscht jegliche Information der Software aus der *Docker* Umgebung und auch alle Dateien, welche mit den Diensten assoziiert sind.

6.2.3 Datenmodell

Auf Basis der zu erstellenden Anträge und der allgemeinen Nutzerverwaltung wurde folgendes Datenmodell entwickelt. Darin wurde speziell darauf geachtet, dass jegliche Information auch wirklich dort zu finden ist, wo sie auch auf einem papierenen Antrag zu finden wäre.

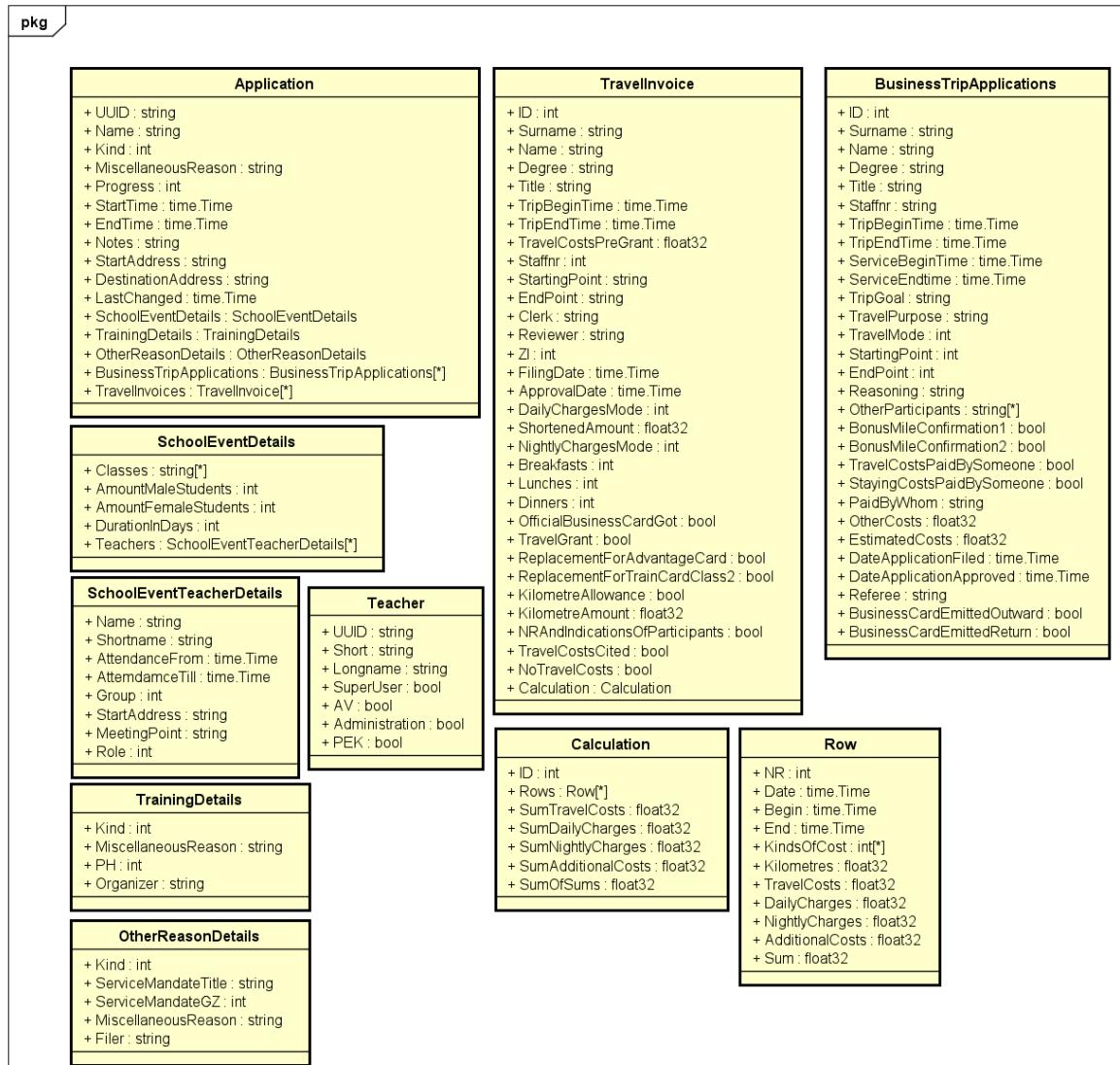


Abbildung 6.24: Das zentrale Datenmodell beinhaltet jegliche Daten, die von der Software benötigt werden.

Insgesamt gibt es zehn verschiedene Datenstrukturen, wobei hierbei neun die eigentlichen Anträge inklusive Subinformationen widerspiegeln und einer die Lehrkraft-Datenstruktur, welche das Schlüssellement der Nutzerverwaltung darstellt.

6.2.4 REST-Schnittstelle

erstellt. Diese sind in sich abgeschlossene Vorgänge, welche sich jeweils immer auf eine Aufgabe beschränken. Im folgenden Kapitel sind alle geplanten *Endpoints* beschrieben.

6.2.4.1 Endpoints

Die folgende Tabelle beschreibt die geplanten *Endpoints*, wobei sie jeweils mit der Protokollmethode (*GET*, *POST*, *PUT* oder *DELETE*), dem Pfad zum *Endpoint*, einer kurzen Beschreibung, Rückgaben, Eingaben und dem Erfordernis eines Tokens beschrieben werden.

Tabelle 6.1: Übersicht *REST-Endpoints*: Teil 1

Methode	Endpoint	Beschreibung	Rückgabe (Erfolg)	Eingabe
POST	/login	Loggt eine Lehrkraft ein	Tokenpaar	Benutzerinformationen (Nutzernname Passwort)
POST	/logout	Loggt eine Lehrkraft aus	Erfolg	-
GET	/login/refresh	Erneuert einen Login	Tokenpaar	Refresh-Token (als Token)
GET	/getTeacherByShort	Gibt Informationen zu einer Lehrkraft zurück	Lehrkraft	Abkürzung einer Lehrkraft
GET	/getTeacher	Gibt Informationen zu einer Lehrkraft zurück	Lehrkraft	UUID einer Lehrkraft
GET	/setTeacher Permissions	Setzt die Berechtigungen einer Lehrkraft	Erfolg	Lehrkraft-Berechtigungen
GET	/getActive Applications	Gibt alle aktiven Anträge (einer Lehrkraft) zurück	Liste an Anträgen	optional: Name einer Lehrkraft
GET	/getAllApplications	Gibt alle Anträge zurück	Liste an Anträgen	optional: Name einer Lehrkraft
GET	/getNews	Gibt die letzt veränderten Anträge zurück	Liste an Anträgen	-
GET	/getAdmin Application	Gibt alle Anträge zurück, die von einem Admin anzuschauen sind	Liste an Anträgen	-

Tabelle 6.2: Übersicht REST-Endpoints: Teil 2

Methode	Endpoint	Beschreibung	Rückgabe (Erfolg)	Eingabe
GET	/getApplication	Gibt Informationen zu einem Antrag zurück	Antrag	UUID des Antrages
POST	/createApplication	Erstellt einen Antrag	Erfolg	Antragsdaten
PUT	/updateApplication	Aktualisiert einen Antrag	Erfolg	Antragsdaten
DELETE	/deleteApplication	Löscht einen Antrag	Erfolg	Antragsdaten
GET	/getAbsenceForm ForClasses	Erstellt das Abwesenheitsformular von Klassen	PDF	UUID des Antrages, optional: Liste an Klassen
GET	/getAbsenceForm ForTeacher	Erstellt das Abwesenheitsformular einer Lehrkraft	PDF	UUID des Antrages, Name der Lehrkraft
GET	/getCompensation ForEducational SupportForm	Erstellt das Formular für pädagogische Betreuung	PDF	UUID des Antrages
GET	/getTravel InvoiceForm	Erstellt ein Reiserechnungsformular	PDF	UUID des Antrages, Name der Lehrkraft, ID der Reiserechnung
GET	/getBusinessTrip ApplicationForm	Erstellt ein Dienstantragsformular	PDF	UUID des Antrages, Name der Lehrkraft, ID des Dienstreiseantrags
GET	/getTravel InvoiceExcel	Erstellt eine Reiserechnung als Excel-Datei	Excel	UUID des Antrages, Name der Lehrkraft, ID der Reiserechnung
GET	/getBusinessTrip ApplicationExcel	Erstellt einen Dienstantrag als Excel-Datei	Excel	UUID des Antrages, Name der Lehrkraft, ID des Dienstreiseantrags
POST	/saveBillingReceipt	Speichert einen Beleg als PDF ab	-	PDF

6.2.4.2 Token System

Die Implementierung eines nicht-persistenten Token Systems, um die Kommunikation mit der REST-Schnittstelle weiter zu schützen, ist wenn es sich um sensible Daten handelt, sehr wichtig. Ansonsten könnte einfach jeder die *Endpoints* der Schnittstelle ansprechen. Durch die Ausgabe von sogenannten „Access Tokens“ und „Refresh Tokens“ werden diese jedoch abgesichert. Diese müssen bei jeder Anfrage nach dem Login mit übergeben werden (im *Headerfeld „Authorization“*), wodurch sich der Absender als eingeloggte Lehrkraft verifiziert. Anfragen ohne Tokens werden in Folge automatisch abgewiesen.

Diese Tokens sind verschlüsselte Zeichenketten, welche nur durch das *Backend* entschlüsselbar sind, da nur das System selbst den Schlüssel hierfür kennt. Im Token selbst werden Informationen über die Sitzung und den Nutzer gespeichert, sodass er nur durch die Angabe des Tokens verifiziert werden kann.

Das „Access Token“ ist genau 15 Minuten lang gültig und erlaubt solange direkten Zugriff auf die Schnittstelle. Sobald dieses abgelaufen ist, kann mit dem „Refresh Token“, welches 7 Tage lang gültig ist, ein neues Tokenpaar bei der Schnittstelle generiert werden. Da es sich um zwei komplett neue Tokens handelt, verlängert sich hierbei auch die Gültigkeit des Logins. Sollte keines der beiden Token mehr gültig sein, so muss ein neues Paar durch einen neuen Loginvorgang erstellt werden.

6.2.5 Backend

Um jene in Unterunterabschnitt 6.2.4.1 beschriebenen *Endpoints* auch mit Funktionalität ausstatten zu können, müssen entsprechende Methoden im *Backend* implementiert werden. Hierzu gehören hauptsächlich die Schnittstellen zu folgenden Diensten und die Implementierungen von Hauptfunktionen des Systems. Diese sind in den folgenden Abschnitten beschrieben.

6.2.5.1 Untis

Die offizielle *Untis*-Schnittstelle ermöglicht dem System die Abfrage des aktuellen Stunden- und Supplier-plans. Dadurch können diese Informationen auch auf den etwaigen Formularen direkt genutzt werden. Um mit dieser Schnittstelle einfach kommunizieren zu können, ist folgender Aufbau und Implementierung eines *REST-Clients* im *Backend* geplant. Mit Hilfe diesen kann das System einfach die Sitzung, welche schon durch die *REST*-Schnittstelle existiert, weiter nutzen und somit möglichst effizient arbeiten.

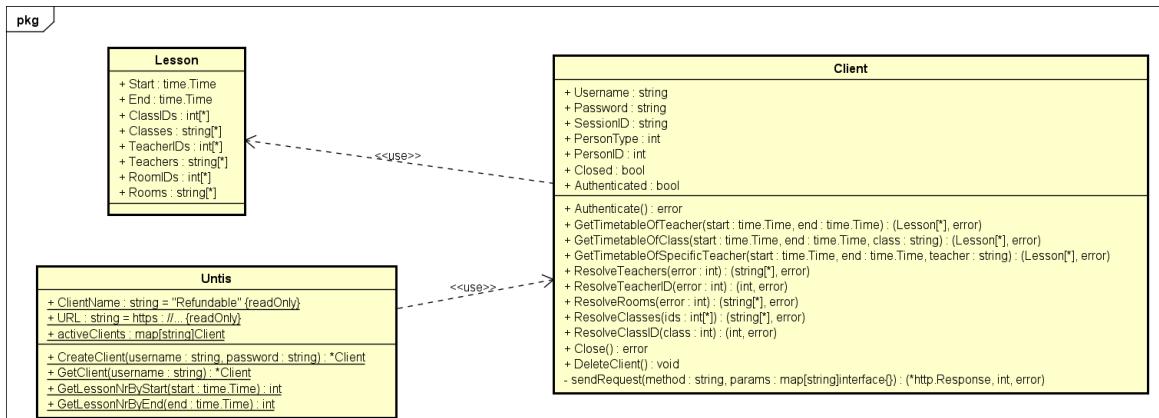


Abbildung 6.25: UML-Klassendiagramm des *Untis API-Client* für das Abfragen von Stundenplaninformationen.

6.2.5.2 LDAP

LDAP ist ein Zugriffsprotokoll um auf ein *Active Directory* zugreifen zu können. In diesem wird im *TGM* weitere Informationen zu den Lehrkräften und Schülern gespeichert. Um diese Daten einsehen zu können, muss man sich bei diesem Dienst anmelden, was mit den jeweils eigenen *TGM-Account* Zugangsdaten möglich ist. Dadurch ist die Anmeldung mit den *TGM-Nutzerdaten* zu realisieren. Des Weiteren ist die Abfrage weiterer benötigter Daten hierdurch möglich. Der Ablauf einer solchen Interaktion über *LDAP* wird in folgendem Flussdiagramm veranschaulicht.

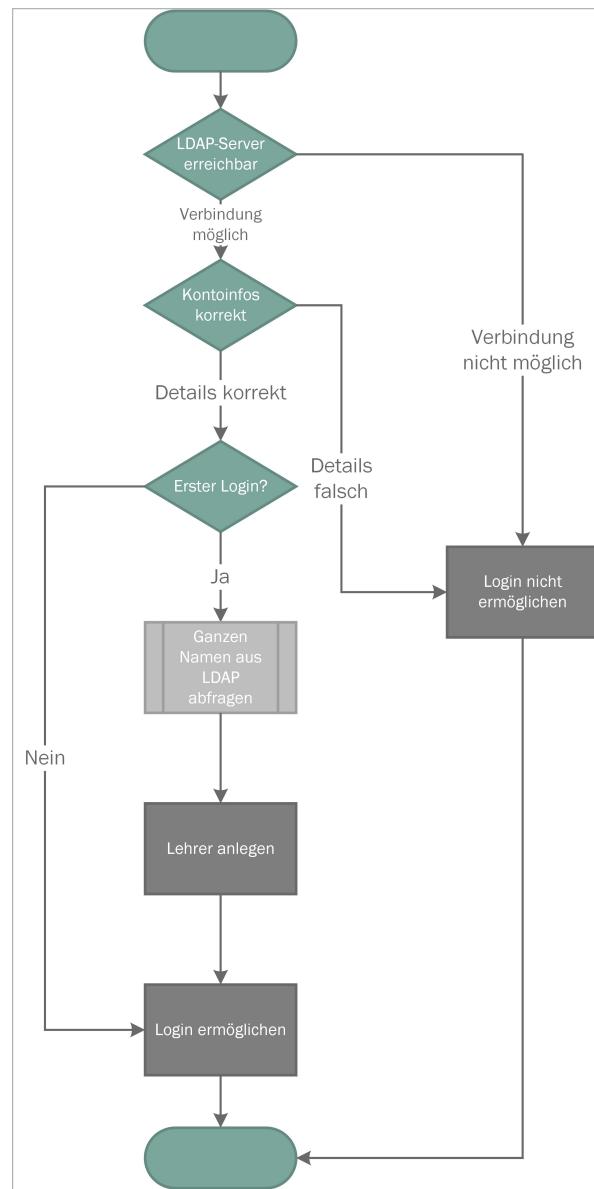


Abbildung 6.26: Authentifizierung über den *TGM-eigenen LDAP Server*

Um jenen in Abbildung 6.26 abgebildeten Ablauf einfach implementieren zu können, bietet sich folgender Aufbau eines *LDAP-Clients* an. Dieser kann hierbei die Zugangsdaten durch eine einfache Anmeldung beim Dienst verifizieren und auch die benötigten Daten, wie den kompletten Namen der Lehrkraft, beim Dienst abfragen.

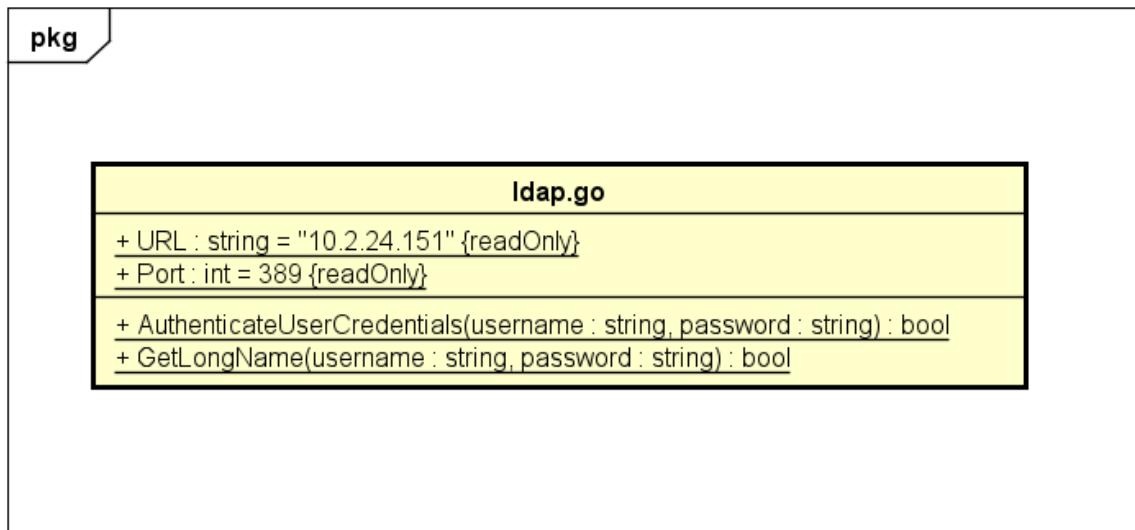


Abbildung 6.27: UML-Klassendiagramm des *LDAP Packages* für die Authentifizierung von Lehrkräften

6.2.5.3 Dateierstellung

Die Datei- und Formularerstellung ist die Hauptfunktion der Software. Das Generieren folgender Formulare wird unterstützt. Bei „Reiserechnung“ und „Dienstreiseantrag“ werden auf Grund des festgelegten *Designs* auch eine *Excel* Variante angeboten, in der dieses umgesetzt ist. Das Format steht hierbei für Hoch- oder Querformat, wobei *Portrait* das Hochformat und *Landscape* das Querformat ist.

Tabelle 6.3: Übersicht über die verschiedenen Dateien, die durch das *Backend* erstellt werden.

Formular	Dateityp	Format
Abwesenheitsmeldung eines Jahrganges	PDF	Portrait
Abwesenheitsmeldung eines Lehrers	PDF	Portrait
Abgeltung für pädagogische Betreuung	PDF	Portrait
Reiserechnung	PDF	Landscape
Dienstreiseantrag	PDF	Portrait
Reiserechnung	Excel	Landscape
Dienstreiseantrag	Excel	Portrait

Alle eigens generierten *PDFs* sind jedenfalls mit einer Kopfzeile ausgestattet. In dieser ist in der linken Ecke das *TGM*-Logo zu sehen, in der Mitte steht der Name des Formulars (siehe Tabelle 6.3) und rechts ein *QR-Code*, welcher eine *URL* beinhaltet, die den zugehörigen Antrag direkt im *Frontend* öffnet.

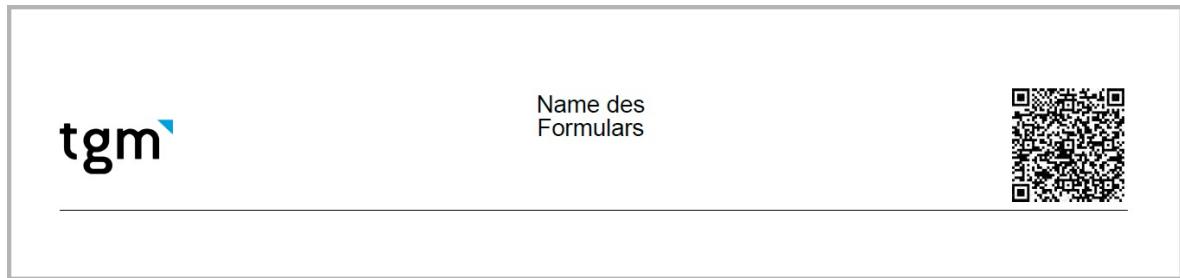


Abbildung 6.28: Beispiel einer Kopfleiste in den vom *Backend* generierten *PDF*-Dateien

6.3 Datenschnittstelle und Webseitenlogik

Die Schnittstelle zwischen *Frontend* und *Backend* wird mittels *JavaScript-Framework* umgesetzt. Wie in Unterabschnitt 5.3.5 beschrieben, wird hierfür *VueJS* verwendet. *VueJS* bietet viele passende Funktionen und *Features*, um eine solche Webseite umzusetzen.

6.3.1 Webseitenlogik

Die Webseite wird mittels *Vue*-Komponenten aufgebaut werden. Eine Seite besteht aus einer Hauptkomponente und möglicherweise noch zusätzlichen Nebenkomponenten. Die Struktur der einzelnen Komponenten und *Links* der Webseite soll wie folgt aussehen:

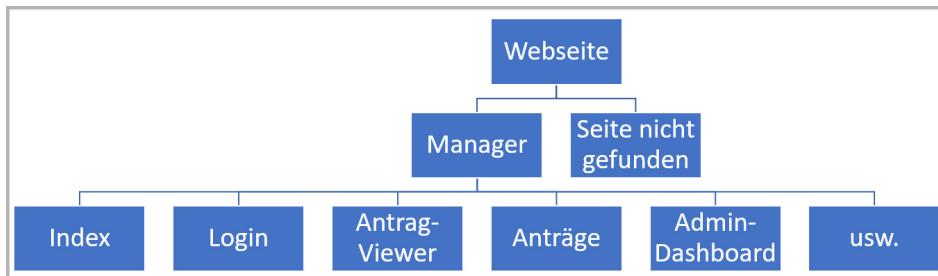


Abbildung 6.29: Übersicht der Hierarchie der Webseite

6.3.1.1 Seite nicht gefunden

Wird ein *Link* eingegeben, welcher nicht auf die Seite selbst oder eine definierte Unterseite verweist, wird eine Seite geladen, bei der der Benutzer darauf hingewiesen wird, dass kein korrekten *Link* eingegeben wurde.



Abbildung 6.30: Seite nicht gefunden

6.3.1.2 Manager

Der *Manager* ist der Hauptbestandteil der Webseite. Er beinhaltet alle einzelnen Seiten und zeigt immer die gewünschte Seite an. In dem *Manager* werden auch *Cookie*-Einstellungen und Rechte gespeichert. Fast alle Komponenten werden dem *Manager* untergeordnet. Der *Manager* ist auch dafür verantwortlich, Daten zwischen Komponenten zu übermitteln.

6.3.1.3 Komponenten

Einzelne Komponenten laden die Daten eigenständig aus dem *Backend*. Dies ist eine Maßnahme, um den *Manager* nicht mit Funktionen zu überladen und damit nicht bei jedem Aufruf Daten vom *Manager* an die einzelnen Komponenten gesendet werden müssen.

Im Fall, dass eine Komponente Daten an eine weitere Komponente sendet, wird diese Kommunikation vom *Manager* übernommen.

6.3.1.4 Antrag-Viewer

Es soll eine weitere Seite vorhanden sein, auf die man durch einen *Link* gelangen kann. Auf dieser Seite soll es möglich sein, einen Antrag per *ID* anzeigen zu lassen. Da dieser separate *Link* technisch auf der selben Ebene wie der *Manager* ist, wird dafür gesorgt, dass dieser *Link* dem *Manager* nahtlos untergeordnet ist. Der *Antrag-Viewer* ist nur verwendbar, sofern der Benutzer angemeldet ist und auch die Berechtigung hat, diesen Antrag zu betrachten.

Wird in den Parametern des *Links* bereits spezifiziert, welcher Antrag geöffnet werden soll, wird dem Benutzer nach der Anmeldung dieser Antrag angezeigt. Es können nur jene Anträge angezeigt werden, zu denen der Benutzer die erforderlichen Berechtigungen besitzt. Besitzt der Benutzer nicht die erforderlichen Berechtigungen, wird eine Fehlermeldung ausgegeben.

6.3.1.5 Navigation

Da fast alle Komponenten dem *Manager* untergeordnet sind, ist es auch die Verantwortung des *Managers* die Navigation zu steuern. Dazu gibt es eine zentrale Funktion im *Manager*, welche die derzeit geladene Seite verändern kann. Des Weiteren werden die benötigten Informationen geladen bzw. erstellt, welche die Komponenten benötigen.

Die Funktion zum Verändern der derzeitigen Anzeige wird von den untergeordneten Komponenten aufgerufen. Hinzu kommt, dass mit dem Aufruf der Funktion auch spezifiziert wird, welche Seite geladen werden soll. Spezielle Seiten, wie z.B: der *Antrag-Viewer*, benötigen zusätzlichen Informationen, welche über diese Funktion mitgegeben werden.

6.3.1.6 Features

Es sollen auch Funktionen implementiert werden, welche das Benutzen der Webseite erleichtern.

Es soll möglich sein, bei erneutem Laden der Webseite auf die zuletzt geöffnete Seite zu gelangen. Beim Beenden des *Browsers* speichert die Webseite die aktuelle Seite und lädt diese bei erneutem Aufrufen der Webseite automatisch.

Es ist durch die *Browser-Pfeile* auch möglich zwischen den Seiten zu navigieren. Dies ist wichtig, da durch versehentliches wechseln der Komponente ein Benutzer möglichst schnell wieder auf die vorherige Seite gelangen soll. Durch die Implementierung der *Browser-Pfeile* ist es möglich intuitiv auf die vorherige Seite zu wechseln. Andere Funktionen, welche die selbe Funktionalität haben, werden durch diese Implementierung auch unterstützt.

6.3.2 Daten laden

Auf den Seiten, wo es benötigt wird, werden über Funktionen des *Frameworks* dynamisch Komponenten erstellt und geladen. Der Prozess Daten aus dem *Backend* zu laden wird in folgender Grafik beschrieben:

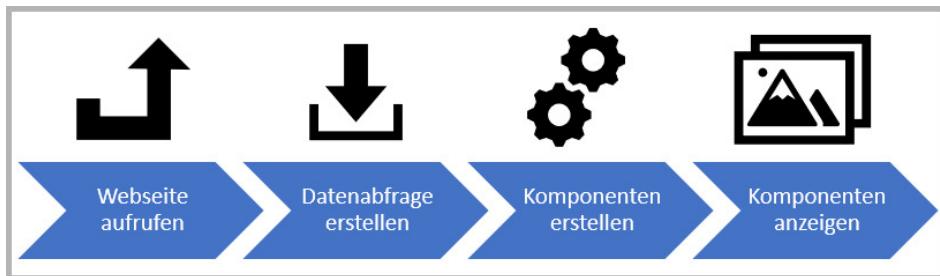


Abbildung 6.31: Übersicht über den Prozess, welcher Daten aus dem *Backend* lädt

6.3.2.1 Webseite aufrufen

Die Webseite wird aufgerufen und der Benutzer bekommt die Startseite angezeigt.

6.3.2.2 Datenabfrage erstellen und Daten erhalten

Auf der Startseite der Webseite werden die neuesten Nachrichten angezeigt, welche für den Lehrer relevant sind. Hierfür müssen aus dem *Backend* Daten geladen werden. Dies erfolgt über eine Abfrage, welche aus dem *Frontend* gesendet wird, Daten ausliest und diese dann verwendet, um Komponenten zu erstellen.

6.3.2.3 Komponenten erstellen

Die geladenen Daten aus dem *Backend* werden mittels Einbindung der Variablen in einzelne Komponenten eingebunden.

Seiten, auf denen viele gleiche Komponenten sind, werden mittels Schleifen erstellt. Diese sorgen dafür, dass nicht zu viele Komponenten auf der Seite geladen sind, wenn diese nicht gebraucht werden. Ein Beispiel hierfür sind die neuen Nachrichten, welche in folgender Grafik gezeigt werden:

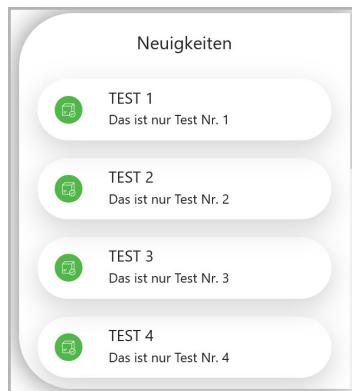


Abbildung 6.32: Grafik der neuen Nachrichten

6.3.2.4 Komponenten anzeigen

Die bereits erstellten Komponenten werden über die Einbindung in der Webseite angezeigt. Dies kann durch einfache Implementierung bei einzelnen Komponenten erreicht werden.

Durch die Schleife können auch die Komponenten angezeigt werden. Diese ordnet alle Komponenten hintereinander und zeigt diese an.

Bei den Nachrichten kann man sich jede Nachricht als eigene Komponente vorstellen, welche jeweils erstellt und danach angezeigt wird.

6.3.3 Befehle senden

Die Webseite besitzt auch einige Aktionen, die von Benutzern ausgeführt werden können. Diese müssen je nach Aktion eine neue Seite aufrufen oder einen Befehl an das *Backend* senden.

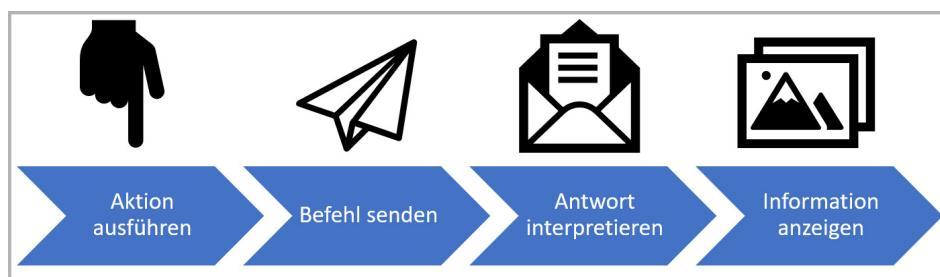


Abbildung 6.33: Übersicht über den Prozess, welcher Befehle an das *Backend* sendet

6.3.3.1 Aktion ausführen

Eine Aktion wird ausgeführt, wenn z.B. auf eine Komponente gedrückt wird. Dies ruft eine Funktion auf, bei der eine Anfrage erstellt wird.

6.3.3.2 Befehl senden

Eine Anfrage beinhaltet alle wichtigen Informationen des Befehls für das *Backend*. Sie wird anschließend an das *Backend* gesendet, um dort abgearbeitet zu werden.

6.3.3.3 Antwort interpretieren

Sobald das *Backend* den Befehl in der Anfrage ausgeführt hat, wird eine Antwort an das *Frontend* gesendet und soll dort interpretiert werden. Die Interpretation der Antwort soll durch den Status der Antwort geschehen. Der Status der Antwort ist ein *Code*, welcher aussagt, ob der Befehl funktioniert hat oder fehlgeschlagen ist. Sollte der Befehl fehlgeschlagen, so deutet der *Code* auf den genauen Fehlergrund hin.

6.3.3.4 Information anzeigen

Tritt ein Fehler auf, soll dies dem Benutzer offensichtlich mitgeteilt werden. Sollte kein Fehler auftreten, soll das dem Benutzer je nach Befehl auf unterschiedliche Weise mitgeteilt werden.

Um weiterhin das Beispiel zu nutzen: Wenn eine Nachricht gelöscht wird, wird dem Benutzer mitgeteilt, dass die Nachricht gelöscht worden ist, indem diese verschwindet.

Kapitel 7

Implementierung

7.1 Frontend

In diesem Kapitel wird die Implementierung des *Frontends* von *Refundable* erläutert. Dabei wird als Erstes die Vorbereitung im Unterabschnitt 7.1.1 erklärt. Danach werden die einzelnen Komponenten im Unterabschnitt 7.1.2 der Webseite aufgezeigt und vorgeführt.

7.1.1 Vorbereitung

Wie im Abschnitt 5.1 und Abschnitt 5.3 festgelegt, wird im Frontend *Bootstrap* in Verbindung mit *VueJS* verwendet. Zu aller erst muss daher *VueJS* auf dem Computer, auf welchem das Projekt erstellt wird, installiert werden. Dies geschieht auf *Windows* über den *Node Packet Manager (NPM)* [24]. Der Befehl um *VueJS* zu installieren, sieht wie folgt aus:

```
1 npm install -g @vue/cli
```

Nachdem *VueJS* installiert ist, kann man in das gewünschte Verzeichnis wechseln und über

```
1 vue create refundable
```

ein neues *Vue* Projekt erstellen, das in diesem Fall *Refundable* heißt [6]. Um die Funktionalität von *Bootstrap* und *VueJS* optimal auszunutzen, wird *Bootstrap* direkt mit dem *Node Packet Manager* zu *VueJS* installiert [17]:

```
1 npm install vue bootstrap-vue
```

Um in den Komponenten *Bootstrap* und dessen vordefinierte *Icons* zu verwenden, muss man in „main.js“ folgendes importieren:

```
1 import { BootstrapVue, BootstrapVueIcons } from "bootstrap-vue";
2 import "./plugins/bootstrap-vue";
3
4 Vue.use(BootstrapVue);
5 Vue.use(BootstrapVueIcons);
```

Auflistung 7.1: Benötigte Befehle um *Bootstrap* im *VueJS* Projekt einzubinden

Alle Teile der Webseite sind in sogenannten Komponenten verpackt, die je nach Benutzereingaben dynamisch gewechselt werden.

7.1.2 Komponenten der Webseite

7.1.2.1 Generell

Die Struktur der Komponenten ist so aufgebaut, dass die Seite sowohl für Desktop- als auch mobile Endgeräte optimiert ist. Dazu wird das *Grid-System* von *Bootstrap* verwendet, welches seinen Platz im *template-Tag*, der *VueJS* Komponente findet. Dieses besteht aus einem *Container*, einer Reihe (*Row*) und einer Spalte (*Column*). Da es am Anfang das Problem gab, dass nicht der ganze Bildschirm ausgenutzt wird, wurden folgende Klassen im CSS-Dokument erstellt:

```
1   .template-main-container {  
2     height: 100vh;  
3     width: 100vw;  
4     margin: 0;  
5     padding: 0;  
6   }  
7  
8   .template-main-row {  
9     height: 100vh;  
10    width: 100vw;  
11    margin: 0;  
12    padding: 0;  
13  }
```

Auflistung 7.2: CSS Klassen für die Haupt- Contianer und Reihen

Diese dienen dazu, dass der *Hauptcontainer* und die Hauptreihe der Webseite 100% der Höhe (100 vertikale Einheiten), sowie 100% der Breite (100 horizontale Einheiten) einnehmen. Außerdem wird in den Klassen definiert, dass die Elemente keinen Abstand zum Rand der Seite haben sollen.

7.1.2.2 Login-Seite

Die *Login-Seite* sollte sehr schlicht gehalten werden und sofort verstanden werden können. Da das Tool *Refundable* nur von Lehrern des TGMs verwendet werden soll, wurde in der *Login-Maske* die E-Mail-Endung „@tgm.ac.at“ vordefiniert. Außerdem soll dem Aufrüfer der Webseite sofort klar sein, worum es sich bei dieser Webseite handelt. Es wurde daher der Projektname *Refundable* und unser Slogan auf der Seite platziert. Des weiteren war aufgrund der DSGVO eine Meldung zu realisieren, die den Benutzer darüber informiert, dass *Cookies* verwendet werden. Falls diese Meldung nicht akzeptiert wird und man sich dennoch anmelden möchte, wird eine Fehlermeldung dargestellt. Diese Meldung wird jedoch dynamisch mit *VueJS* erstellt und nicht direkt mittels *Bootstrap*. In der Umsetzung sieht die *Login-Seite* wie folgt aus (Abbildung 7.1):

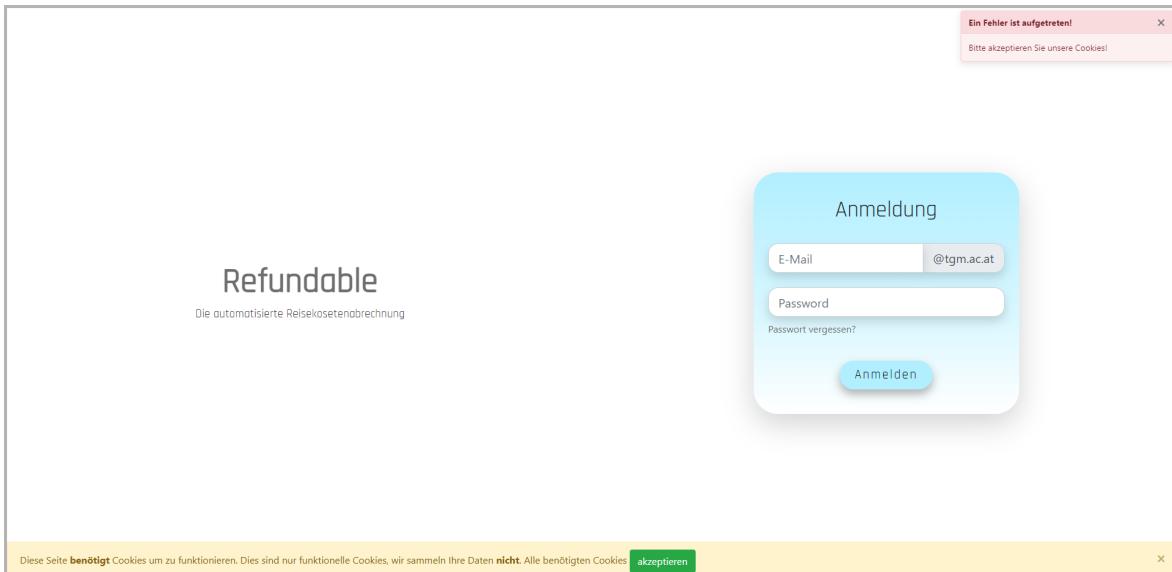


Abbildung 7.1: Ein Bild der Login-Seite

Login-Maske

Die *Login-Maske* besteht aus einem eigenen *Container*, welcher ermöglicht, die Maske vollkommen responsiv aufzubauen. Die *Eingabefelder* wurden mit einer *input-groupe* gelöst (Code Z. 12). Um einen Wert optisch in einem *Eingabefeld* vorzugeben, hat *Bootstrap* ein Element namens *b-input-groupe-text* (Code Z. 14), welches vor oder nach dem Feld hinzugefügt werden kann. Da die *Login-Daten*, der *TGM Server* verwendet werden, haben wir einen Link eingebaut, falls man sein Passwort vergessen hat. Betätigt man diesen, wird man auf die entsprechende Seite des *TGMs* weitergeleitet.

```

1      <!-- Anmeldungsformular -->
2      <b-col cols="12" md="6">
3          <b-container>
4              <b-row align-h="center">
5                  <b-col cols="12" sm="10" md="10" lg="8" xl="6">
6                      <b-container id="login-wrap" class="shadow-lg">
7                          <b-row id="r-login" align-h="center">
8                              <h2 id="lheading2">Anmeldung</h2>
9                          </b-row>
10                         <b-row align-h="center" id="r-email">
11                             <b-col cols="12">
12                                 <!-- Input des Benutzernamens / der Email -->
13                                 <b-input-group size="lg">
14                                     <b-input-group-text
15                                         id="tgm-addon"
16                                         class="shadow"
17                                         slot="append"
18                                         ><span>@tgm.ac.at</span></b-input-group-text
19                                     >
20                                     <b-form-input
21                                         id="email"
22                                         class="shadow login-inputs"
23                                         v-model="email"
24                                         type="text"
25                                         placeholder="E-Mail"
26                                         ></b-form-input>
27                                     </b-input-group>
28                             </b-col>
29                         </b-row>
30                         <b-row id="r-password">
31                             <b-col cols="12">
32                                 <!-- Input des Passworts -->
33                                 <b-form-input
34                                     id="password"
35                                     class="shadow login-inputs"
36                                     v-model="password"
37                                     type="password"
38                                     placeholder="Password"
39                                     size="lg"
40                                     ></b-form-input>
41                             </b-col>
42                         </b-row>
43                         <b-row id="r-forgotten">
44                             <b-col cols="12">
45                                 <!-- Passwort vergessen Weiterleitung auf Moodle -->
```

```

46          <a
47              id="forgotten-password"
48              href="https://elearning.tgm.ac.at/login/forgot_"
49                  ↪ password.php"
50          >Passwort vergessen?</a
51      >
52  </b-col>
53 </b-row>
54 <b-row align-h="center" id="r-login-btn">
55     <!-- Login Button -->
56     <b-button size="lg" id="login-btn" v-on:click="login">
57         Anmelden</b-button
58     >
59 </b-row>
60 </b-container>
61 </b-col>
62 </b-row>
63 </b-container>
64 </b-col>

```

Auflistung 7.3: HTML Code der Login-Maske

Da einige Elemente unseren Ansprüchen optisch nicht genügt haben, wurden eigene Änderungen im CSS-Code hinzugefügt. Dabei wurde vor allem die Gestaltung, der *Eingabefelder* und des *Buttons* verändert, aber auch jene des *Containers*, welcher die Maske beinhaltet. Es wurde der Farbverlauf des Logos in der Maske eingebaut und die Ecken der Elemente etwas abgerundet.

```

1  /* -- Login -- */
2
3 #login-wrap {
4     background: rgb(255, 255, 255);
5     background: -moz-linear-gradient(
6         0deg,
7         rgba(255, 255, 255, 1) 0%,
8         rgba(220, 248, 255, 1) 38%,
9         rgba(177, 239, 255, 1) 100%
10    );
11    background: -webkit-linear-gradient(
12        0deg,
13        rgba(255, 255, 255, 1) 0%,
14        rgba(220, 248, 255, 1) 38%,
15        rgba(177, 239, 255, 1) 100%
16    );
17    background: linear-gradient(
18        0deg,
19        rgba(255, 255, 255, 1) 0%,
20        rgba(220, 248, 255, 1) 38%,
21        rgba(177, 239, 255, 1) 100%
22    );
23    filter: progid: DXImageTransform.Microsoft.gradient(startColorstr="#ffffff",
24        ↪ endColorstr="#b1efff", GradientType=1);
25    padding: 1.5rem;

```

```
25     border-radius: 2.5rem;
26 }
27
28 #email {
29     border-radius: 1rem 0 0 1rem;
30 }
31
32 #tgm-addon {
33     border-radius: 0 1rem 1rem 0;
34 }
35
36 .input-group-append {
37     border-radius: 0 1rem 1rem 0;
38 }
39
40 .input-group-text {
41     border-radius: 0 1rem 1rem 0;
42 }
43
44 #password {
45     border-radius: 1rem 1rem 1rem 1rem;
46 }
47
48 #login-btn {
49     font-family: "Rajdhani", sans-serif;
50     padding: 0.5rem 1.5rem 0.5rem 1.5rem;
51     font-size: 1.3rem;
52     letter-spacing: 2.5px;
53     font-weight: 500;
54     color: #000;
55     background-color: rgba(177, 239, 255, 1);
56     border: none;
57     border-radius: 45px;
58     box-shadow: 0px 8px 15px rgba(0, 0, 0, 0.3);
59     transition: all 0.3s ease 0s;
60     cursor: pointer;
61     outline: none;
62 }
63
64 #login-btn:hover {
65     background-color: #2ec0e5;
66     box-shadow: 0px 15px 20px rgba(46, 229, 157, 0);
67     color: #fff;
68     transform: translateY(3px);
69 }
```

Auflistung 7.4: CSS Code der Login-Maske

Cookie-Meldung

Die Meldung, dass *Cookies* benutzt werden, sollte schlicht und einfach sein. Es wurden die Anforderungen an die Meldung gestellt, dass sie den Nutzer darüber informieren soll, dass ausschließlich funktionelle und notwendige *Cookies* verwendet werden. Für diese Meldung hat sich das vordefinierte *Alert-Element* von *Bootstrap* perfekt geeignet:

```
1  <!-- Bestätigung, dass diese Seite Cookies benutzt -->
2  <b-alert
3      v-model="showBottom"
4      class="position-fixed fixed-bottom m-0 rounded-0"
5      style="z-index: 2000;" 
6      variant="warning"
7      dismissible
8  >
9      Diese Seite <b>benötigt</b> Cookies um zu funktionieren. Dies sind nur
10     funktionelle Cookies, wir sammeln Ihre Daten <b>nicht</b>. Alle benötigten
11     Cookies
12     <!-- Button zum akzeptieren -->
13     <b-button variant="success" @click="validateCookies()">
14         >akzeptieren
15     </b-button>
16 </b-alert>
```

Auflistung 7.5: HTML Code Cookie akzeptieren

7.1.2.3 Startseite

Die *Startseite* sollte so intuitiv und übersichtlich wie möglich sein. Deswegen wurde ein „Drei Spalten Design“ umgesetzt. In der ersten Spalte befinden sich alle wichtigen Funktionen: drei *Buttons*, zum Erstellen von neuen Anträgen, um alle *aktiven Anträge* anzuzeigen und um *alle Anträge* zu betrachten, die jemals gestellt wurden. In der 2. Spalte befindet sich das Logo, eine Illustration für das *Dashboard* und ein *Button* zum Ausloggen. Falls man sich mit einem Administratorkonto anmeldet, wird hier ebenfalls ein *Button* zum Wechseln in die *Administratoransicht* angezeigt. In der dritten Spalte werden Neuigkeiten zu den Anträgen des Benutzers angezeigt. Klickt man auf eine Neuigkeit, wird man zu dem jeweiligen Antrag weitergeleitet. Abbildung 7.2 zeigt die Umsetzung der *Startseite*.

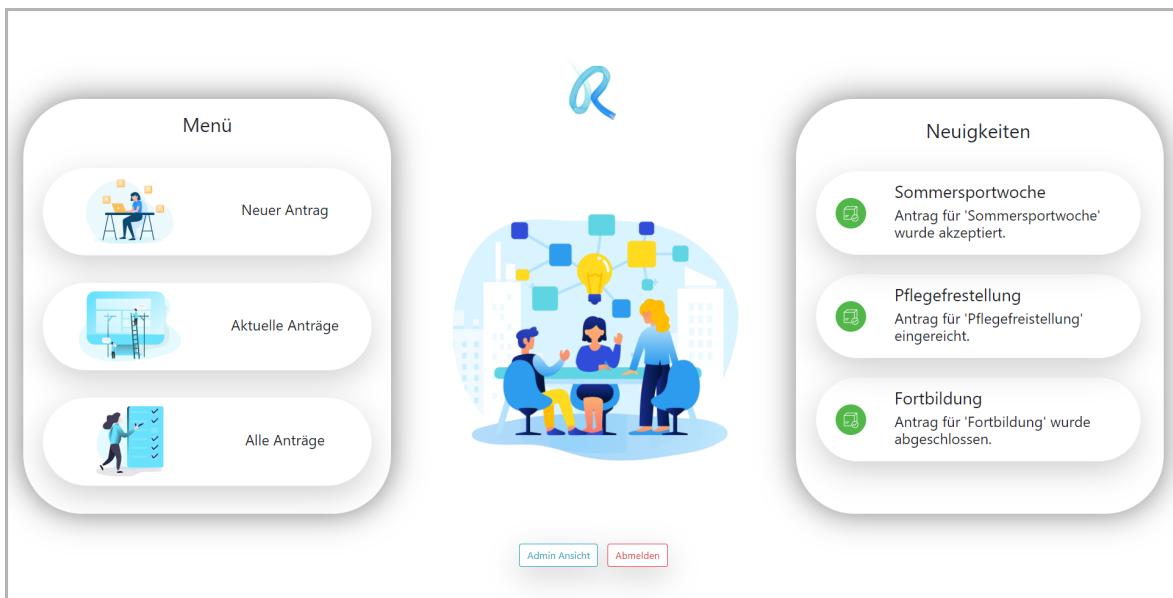


Abbildung 7.2: Ein Bild der endgültigen Startseite

Da dieses System auf mobilen Geräten nicht sonderlich übersichtlich ist, wurde diese hierfür etwas abgeändert. Die geänderte *Startseite* ist sehr schlicht gehalten, und macht die Bedienung auf *Smartphones* um einiges einfacher. Die mobile Version der *Startseite* ist in Abbildung 7.3 dargestellt:

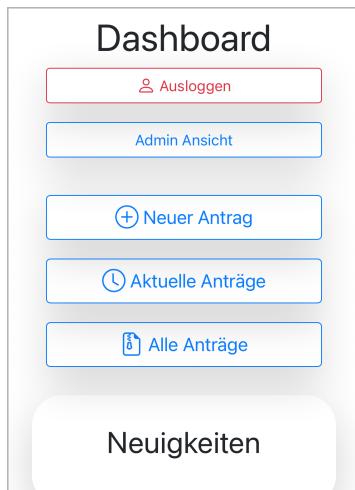


Abbildung 7.3: Ein Bild der endgültigen Startseite (Mobil)

Menü-Element Desktop

Der *Container* mit den Funktionselementen hat, wie beim *Login*, einen Schatten, der ihn vom Hintergrund abhebt. In diesem *Container* befinden sich die Menübezeichnung und die klickbaren Elemente, die auch aus einem *Container* bestehen, sowie eine responsive Illustration, die zu der jeweiligen Funktion passt, welche mit einem *Icon* versehen ist.

```

1  <b-col id="dash-main-cont" class="d-none d-md-block" cols="12" md="4">
2      <b-container>
3          <b-row
4              id="dash-row"
5                  align-v="center"
6                  align-h="center"
7                  class="shadow-xl"
8          >
9              <b-col cols="12">
10                 <center><h2 id="menu-h2">Menü</h2></center>
11             </b-col>
12             <!-- Custom Button für einen neuen Antrag -->
13             <b-col
14                 id="new"
15                 cols="12"
16                 class="shadow-lg dash-elem"
17                 v-on:click="newApplication"
18             >
19                 <b-container style="height: 100%">
20                     <b-row align-h="center" align-v="center" style="height: 100%">
21                         <b-col cols="6" class="ill-wrapper d-none d-md-block">
22                             <!-- Illustration neuer Antrag -->
23                             <b-img
24                                 id="all-ill"
25                                 center
26                                 src="@/assets/new.svg"
27                                 alt="Illustration für alle Anträge"
28                         ></b-img>

```

```

29          </b-col>
30          <b-col cols="12" md="6">
31              <h2 id="new-h2" class="dh">Neuer Antrag</h2>
32          </b-col>
33      </b-row>
34  </b-container>
35 </b-col>
36
37  ...
38      </b-row>
39  </b-container>
40 </b-col>
```

Auflistung 7.6: HTML-Code-Teile des Startseitenmenüs

Optisch wurde vor allem die Höhe des umschließenden *Containers* und dessen umliegende Abstände verändert. Wie man auf der Abbildung 7.2 sehen kann, wurden auch die Ecken abgerundet. Nicht ersichtlich ist, dass sich die Funktionselemente optisch absenken, der Schatten entfernt und die Farbe verändert wird, wenn man mit der Maus, die sich zu einer zeigenden Hand verändert, über das Element fährt.

```

1 #dash-main-cont {
2     height: 70vh;
3 }
4
5 #dash-row {
6     height: 70vh;
7     padding-left: 2rem;
8     padding-right: 2rem;
9     padding-bottom: 1.5rem;
10 }
11
12 .dash-elem {
13     background-color: rgb(255, 255, 255);
14     height: 22%;
15     transition: all 0.3s ease 0s;
16     cursor: pointer;
17     outline: none;
18 }
19
20 .dash-elem:hover {
21     background-color: #8ff2ff;
22     box-shadow: 0px 0px 50px rgba(0, 0, 0, 0.144);
23     color: rgb(141, 141, 141);
24     transform: translateY(3px);
25 }
26
27 #dash-row {
28     border-radius: 6rem 6rem 6rem 6rem;
29 }
```

Auflistung 7.7: Teile des CSS Codes, des Startseitenmenüs

Menü-Element Mobil

Wie bereits erwähnt, ist die *mobile Ansicht* sehr schlicht gehalten, damit sie nicht überladen wirkt und sich vor allem auch jene Lehrer, die sich nicht viel mit Technik beschäftigen, auskennen. Daher wird auf der *mobilen Ansicht* nur das Nötigste angezeigt, siehe Abbildung 7.3. Alle *Buttons* werden nun vertikal angeordnet und mit einem passenden *Icon* bestückt. Die *Newelemente* unterscheiden sich kaum von denen der Desktop-Version. Die Elemente sind lediglich mit etwas weniger Details bestückt und nicht in einem *scrollbaren Container*, da man auf der mobilen Version sonst doppelt scrollen müsste.

```

1  <!-- DASH MOBILE -->
2  <b-col class="d-block d-md-none" cols="12">
3      <!-- Column -->
4      <b-container fluid>
5          <b-row align-v="center" align-h="center">
6              <b-col cols="12">
7                  <center><h1 style="margin-top:10px;">Dashboard</h1></center>
8              </b-col>
9              <b-col cols="12">
10                 <!-- Logout Button -->
11                 <b-button
12                     variant="outline-danger"
13                     class="shadow-lg"
14                     v-on:click="logout"
15                     style="margin-top:0px; margin-bottom:20px; width:100%">
16                     >
17                         <b-icon icon="person" aria-hidden="true"></b-icon> Ausloggen
18                         <!-- Icon -->
19                     </b-button>
20                 </b-col>
21                 <b-col cols="12">
22                     <!-- Neuer Antrag Button --><b-button
23                         size="lg"
24                         variant="outline-primary"
25                         v-on:click="newApplication"
26                         class="shadow-lg"
27                         style="margin-bottom:20px; width:100%">
28                         >
29                             <b-icon icon="plus-circle" aria-hidden="true"></b-icon> Neuer
30                             Antrag
31                         </b-button></b-col>
32             >

```

Auflistung 7.8: Teile des HTML-Codes der mobilen Startseite

Neuigkeiten-Element

Die *Neuigkeiten-Elemente* sind mit einer Illustration bestückt, die den momentanen Status des jeweiligen Antrags beschreiben soll. Außerdem wird ein Titel der Neuigkeit mitgegeben, die zum Beispiel „Antrag X abgelehnt“ heißen könnte. Etwas mehr Details werden in einem Beschreibungsfeld gegeben, das aufgrund der Übersichtlichkeit nur auf Desktopgeräten sichtbar ist.

```
1  <b-row align-v="center">
2    <b-col cols="2">
3      <b-container style="height: 100%">
4        <b-row align-v="center" align-h="center" style="height: 100%">
5          <!-- Illustration Angenommen/Abgelehnt -->
6          
12       </b-row>
13     </b-container>
14   </b-col>
15   <b-col cols="10">
16     <b-container>
17       <b-row align-h="center" align-v="center">
18         <b-col cols="12">
19           <!-- Titel der Neuigkeit -->
20           <h3 class="news-elem-heading">{{ snews.title }}</h3>
21         </b-col>
22       </b-row>
23       <b-row align-h="center" class="d-none d-md-block">
24         <b-col cols="12">
25           <!-- Beschreibung der Neuigkeit -->
26           <h4 class="news-elem-heading">{{ snews.description }}</h4>
27         </b-col>
28       </b-row>
29     </b-container>
30   </b-col>
31 </b-row>
```

Auflistung 7.9: HTML Code, des Neuigkeiten Elementes

7.1.2.4 Neuer Antrag

Auf diese Seite gelangt der Nutzer entweder, durch die Betätigung des *Neuer Antrag Buttons* auf der *Startseite* oder über die Verknüpfung auf der *Alle Anträge* oder *Aktive Anträge* Ansicht. Da es nicht möglich ist, ein einheitliches Formular zu kreieren, wurde vor dem Formular eine Auswahlmöglichkeit der Antragsart eingebaut.

Auswahl

Die von uns gewählte Auswahlmöglichkeit ist sehr einfach und intuitiv gestaltet. Die drei verschiedenen Hauptarten von Anträgen sind klar und deutlich zu unterscheiden. Optisch sind die *Buttons* ähnlich, wie die der *Hauptseite* strukturiert. Damit zieht sich ein einheitliches *Design* durch die Webseite.

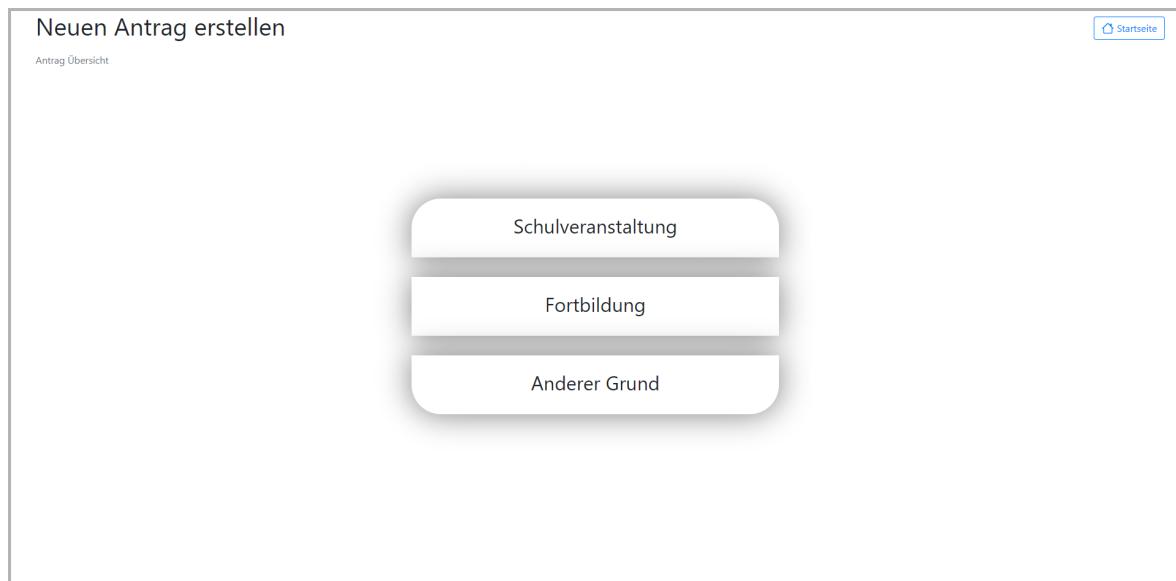


Abbildung 7.4: Ein Bild der endgültigen Seite, um einen neuen Antrag auszuwählen

Die *Buttons* sind in eigenen *Containern* dargestellt, um sie optimal der Größe des Endgerätes anzupassen. In dem *Container*, der sich mit einem Schatten vom Hintergrund abhebt, ist eine eindeutige Bezeichnung der Funktion des jeweiligen Elementes. Fährt man mit der Maus über einen *Button*, wird dieser, wie auf der *Startseite*, in seiner Farbe verändert.

```
1 <b-container id="school-button" class="na-elem shadow-xl">
2   <!-- Custom Button zum erstellen eines Schulantrages -->
3   <b-row
4     class="na-elem-sr"
5     align-v="center"
6     v-on:click="school()">
7     <b-col cols="12">
8       <h2 class="na-elem-h">Schulveranstaltung</h2>
9     </b-col>
10   </b-row>
11 </b-container>
```

Auflistung 7.10: HTML Code, einer Auswahlmöglichkeit

Schulveranstaltung

Der Antrag für eine neue Schulveranstaltung ist der komplexeste Antrag mit den meisten *Eingabefeldern*. Außerdem gehören zu dem Antrag für eine Schulveranstaltung auch die Anträge der Begleitpersonen. Um die Eingabe von Uhrzeiten und Daten einfacher zu gestalten, wurden die Elemente *b-form-datepicker* und *b-form-timepicker* verwendet. Dies sind standardmäßige Elemente, die von *BootstrapVue* vordefiniert sind. Um eine korrekte Eingabe sicherzustellen, wurden alle *Eingabefelder* mit einem Titel und einer Zusatzinformation ausgestattet.

Neuen Antrag erstellen

Antrag Übersicht / Schulveranstaltung - Allg. Infos

Bezeichnung	<input type="text"/>
	Geben Sie die Bezeichnung der Schulveranstaltung ein.
Startdatum	<input type="text"/> Datum auswählen
	Geben Sie das Startdatum der Schulveranstaltung ein.
Startzeit	<input type="text"/> Zeit auswählen
	Geben Sie die Startzeit der Schulveranstaltung ein.
Enddatum	<input type="text"/> Datum auswählen
	Geben Sie das Enddatum der Schulveranstaltung ein.
Endzeit	<input type="text"/> Zeit auswählen
	Geben Sie die Endzeit der Schulveranstaltung ein.
Startadresse	<input type="text"/> Wexstraße 19-23, 1200 Wien, Österreich
	Geben Sie die genaue Startadresse der Schulveranstaltung ein.
Zieladresse	<input type="text"/> Straße & Nr., Postleitzahl & Ort, Land
	Geben Sie die genaue Zieladresse der Schulveranstaltung ein.
Begleitpersonen	<input type="text"/> Einträge durch Leerzeichen trennen
	Geben Sie die Kürzel der Begleitpersonen im Format "mmuster" für z.B.: Max Muster ein.
Jahrgänge	<input type="text"/> Einträge durch Leerzeichen trennen
	Geben Sie die Kürzel der Klassen ein z.B.: "SBHIT"
Anzahl Schüler	<input type="text"/> 0
	Geben Sie die anzahl der Schüler ein.
Anzahl Schülerinnen	<input type="text"/> 0
	Geben Sie die anzahl der Schülerinnen ein.
Anmerkungen	<input type="text"/> Anmerkungen
	Geben Sie zusätzliche Anmerkungen an.

Abbildung 7.5: Ein Bild der Schulantrags Seite (Teil 1)

Anzahl Schülerinnen	<input type="text"/> 0
	Geben Sie die anzahl der Schülerinnen ein.
Anmerkungen	<input type="text"/> Anmerkungen
	Geben Sie zusätzliche Anmerkungen an.
weiter	

Abbildung 7.6: Ein Bild der Schulantrags Seite (Teil 2)

Alle *Eingabefelder* sind in sogenannte *b-form-groups* eingefügt. Diese Elemente bieten die Möglichkeit, einen Titel hinzuzufügen, der in unserem Fall auf der linken Seite angebracht ist, und eine Beschreibung, die unterhalb des *Eingabefelds* positioniert ist. Diese Elemente sind ebenfalls responsiv. In der *form-group*

wird das jeweilige Element eingefügt, welches man eigentlich einbauen will. Diese Elemente können unter anderem normale *Eingabefelder* sein (*Checkboxen*, *Radiobuttons*, *Timepicker*, *Datepicker*, etc.).

```

1   <b-form-group
2     id="bez"
3     label-cols-sm="4"
4     label-cols-lg="3"
5     content-cols-sm
6     content-cols-lg="7"
7     description="Geben Sie die Bezeichnung der Schulveranstaltung ein."
8     label="Bezeichnung"
9     label-for="bezeichnung"
10    >
11    <b-form-input
12      id="bezeichnung"
13      v-model="data.Name"
14      :readonly="readonly"
15      @input="updateData"
16    ></b-form-input>
17  </b-form-group>
```

Auflistung 7.11: Beispiel für eine *Eingabegruppe*

Timepicker

Das *Timepicker* Element kann ganz einfach, wie im oberen Absatz beschrieben, in eine *Eingabegruppe* eingefügt werden. Da das Element von *Bootstrap* vordefiniert ist, muss man lediglich die üblichen Werte, wie *ID* und Platzhalter, setzen. Das Einzige, was beachtet werden sollte, ist, dass man die richtige Zeitzone (Auflistung 7.12 *locale*) wählt.

```

1   <b-form-timepicker
2     id="stz"
3     v-model="startTime"
4     :readonly="readonly"
5     @input="updateTime"
6     locale="de"
7     placeholder="Zeit auswählen"
8   ></b-form-timepicker>
```

Auflistung 7.12: *Timepicker*

Datepicker

Der *Datepicker* ist noch einfacher zu bedienen, als der *Timepicker*. Hier sind nur die üblichen Parameter zu setzen.

```

1   <b-form-datepicker
2     id="end"
3     v-model="endDate"
4     :readonly="readonly"
5     @input="updateTime"
6     class="mb-2"
7     placeholder="Datum auswählen"
8   ></b-form-datepicker>

```

Auflistung 7.13: Datepicker

Tags

Die Eingabe der Klassen und Begleitlehrer wurde durch die Verwendung von *Tags* etwas vereinfacht. Der Nutzer kann Einträge, verschiedener Klassen einfach mit Leerzeichen trennen. Um einzelne Klassen oder Begleitlehrer wieder zu löschen, muss der Ersteller nur auf das „X“ drücken. Die Umsetzung ist in Abbildung 7.7 zu sehen.

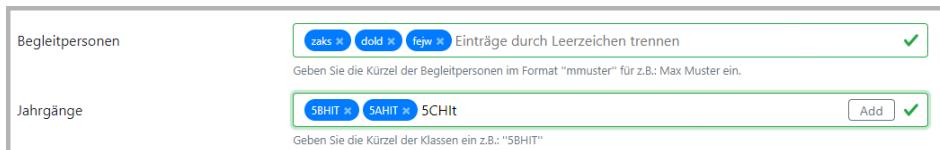


Abbildung 7.7: Ein Bild des Tag Elements

Im Code sieht die Umsetzung wie folgt aus:

```

1  <!-- Bezeichnungen der Klassen -->
2  <b-form-group
3    id="klassen"
4    label-cols-sm="4"
5    label-cols-lg="3"
6    content-cols-sm
7    content-cols-lg="7"
8    description="Geben Sie die Kürzel der Klassen ein z.B.: ''5BHIT'''"
9    label="Jahrgänge"
10   label-for="kl"
11 >
12   <b-form-tags
13     id="kl"
14     input-id="tags-pills"
15     v-model="data.class"
16     :state="Class"
17     v-on:input="checkClass"
18     tag-variant="primary"
19     tag-pills

```

```

20      separator=" "
21      placeholder="Einträge durch Leerzeichen trennen"
22    ></b-form-tags>
23    <b-form-invalid-feedback id="kl-feedback">
24      Keine Jahrgänge angegeben!
25    </b-form-invalid-feedback>
26  </b-form-group>

```

Auflistung 7.14: Tags

Begleitformular

Aus den, im Formular angegebenen Daten, wird automatisch ein weiteres Formular generiert. Dies geschieht, nachdem der *Weiter-Button* betätigt wurde und alle Daten als valide anerkannt wurden. Das Formular beinhaltet für jede Begleitperson die geforderten *Eingabefelder* (Abbildung 7.8). Die Struktur dieser Elemente ist exakt die selbe, wie jene der Elemente, die in Abbildung 7.5 dargestellt wurden.

Abbildung 7.8: Ein Bild des Begleitformulars

Fortbildung

Das Fortbildungsformular ist für Seminare, Tagungen, Lehrgänge und sonstige Events dieser Art gedacht. Um die Formulare einheitlich zu halten, wurde auch hier wieder die selbe Struktur der *Eingabefelder* verwendet. Falls man die sonstige Art auswählt, fügt sich automatisch eine zusätzliche Eingabe hinzu, in der man das Anliegen genauer spezifizieren kann.

Neuen Antrag erstellen

Antrag Übersicht / Fortbildung

Titel	<input type="text"/>
Startdatum	<input type="button" value="Datum auswählen"/> Geben Sie das Startdatum der Fortbildung ein.
Startzeit	<input type="button" value="Zeit auswählen"/> Geben Sie die Startzeit der Fortbildung ein.
Enddatum	<input type="button" value="Datum auswählen"/> Geben Sie das Enddatum der Fortbildung ein.
Endzeit	<input type="button" value="Zeit auswählen"/> Geben Sie die Endzeit der Fortbildung ein.
PH-Zahl	<input type="text"/> Geben Sie Ihre PH-Zahl ein.
Veranstalter	<input type="text"/> Geben Sie den Namen des Veranstalters ein.
Startadresse	<input type="text"/> Geben Sie die Startadresse ein.
Fortbildungsadresse	<input type="text"/> Geben Sie die Adresse der Fortbildung ein.
Art	<input type="radio"/> Seminar <input type="radio"/> Tagung <input type="radio"/> Lehrgang <input type="radio"/> Sonstiges Wählen Sie die Art der Fortbildung.

Abbildung 7.9: Ein Bild des Fortbildungsantrags

Sonstige Anträge

Zu den sonstigen Anträgen gehören Anliegen, wie Pflegefreistellungen, Dienstaufträge, Arzttermine und sonstige Freistellungen dieser Arten. Wählt man einen Dienstauftrag als Art aus, erscheinen automatisch noch *Eingabefelder*, die das Eingeben der GZ und des Dienstauftragtitels ermöglichen.

Neuen Antrag erstellen

Antrag Übersicht / Anderer Grund

Startdatum	<input type="button" value="Datum auswählen"/> Geben Sie das Startdatum der Fortbildung ein.
Startzeit	<input type="button" value="Zeit auswählen"/> Geben Sie die Startzeit der Fortbildung ein.
Enddatum	<input type="button" value="Datum auswählen"/> Geben Sie das Enddatum der Fortbildung ein.
Endzeit	<input type="button" value="Zeit auswählen"/> Geben Sie die Endzeit der Fortbildung ein.
Startadresse	<input type="text"/> Geben Sie die Startadresse ein.
Zieladresse	<input type="text"/> Geben Sie die Zieladresse des Antrags ein.
Gründe	<input type="radio"/> Pflegefreistellung <input checked="" type="radio"/> Dienstauftrag <input type="radio"/> Arzttermin <input type="radio"/> Sonstiges Wählen Sie Ihren Grund aus.
Titel	<input type="text"/> Geben Sie den Titel des Dienstauftrages ein.
GZ	<input type="text"/> Geben Sie die GZ des Dienstauftrages ein.

Abbildung 7.10: Ein Bild des Dienstauftragsformulars

Reiseantrag

Der Reiseantrag wird automatisch bei den Formularen der Anträge, bei denen eine Reise nötig ist angehängt. Ist keine Reise für einen bestimmten Antrag (zum Beispiel ein Arztbesuch) vorgesehen, dann wird das Formular nicht angehängt und es müssen keine Daten eingegeben werden. Die Struktur der Elemente ist ident zu denen, die bereits besprochen wurden.

The screenshot displays a travel application form with the following fields:

- Akademischer Titel:** Prof. (dropdown menu)
- Akademischer Grad:** B. Sc. (dropdown menu)
- Personalnummer:** 12345 (text input)
- Fortbewegungsart:** Amtl. Businesskarte 2. Kl. (radio button selected)
- Begründung:** Aus Gesundheitsgründen (text area)
- Ausgangspunkt:** Dienststelle (radio button)
- Endpunkt:** Wohnung* (radio button selected)
- Begründung:** Weil es näher ist (text area)
- Bonus-Meilen:**
 - Ich bestätige, dass ich anlässlich von Dienstreisen im Rahmen personenbezogener Bonusprogramme erworbene Prämien nicht privat in Anspruch nehme (checkbox selected)
 - Für die Dienstreise verwende ich auf meinem Meilenkonto gutgeschriebene, dientlich erworbene Meilen (checkbox selected)
- Reisekosten - andere Stellen:** Nein (radio button selected)
- Aufenthaltskosten - andere Stellen:** Nein (radio button selected)
- Sonstige Kosten:** 10 (text input)
- Geschätzte Kosten:** 20 (text input)
- Businesskarte:** Hinfahrt (radio button selected)

Abbildung 7.11: Ein Bild des Reiseantrages

7.1.2.5 Reisekostenabrechnung

Das Formular der Reisekostenabrechnung erfordert von den Lehrern die Eingabe vieler verschiedener Werte. Durch die Unterstützung der Webseite ist das Eingeben der Daten wesentlich einfacher, als auf dem Papier, da gewisse Felder, je nach Eingabe des Benutzers, automatisch ausgegraut werden. Wenn zum Beispiel die *Tabelle* in Abbildung 7.12 betrachtet wird, kann man nur in jene Felder schreiben, die man mit den *Checkboxen* angeklickt hat. Außerdem können direkt digital Belege an die Reisekostenabrechnung angehängt werden. Zudem wird automatisch die Zeitspanne des jeweiligen Antrages berechnet und entsprechend viele Zeilen in der *Tabelle* erzeugt.

The screenshot shows a web-based travel expense reporting form. At the top left, there's a section for 'Zusätzliche Daten' (Additional Data) containing several checkboxes for travel-related documents like business cards and kilometer allowances. Below this is a 'Beförderungszuschuss' (Transport Allowance) field with a value of 0. The next section is 'Eigener PKW km' (Own Car km) with a value of 18. Under 'Tagesgebühr' (Daily Rate), there are three radio button options: 'Tagesgebühr nach Tarif I', 'Tagesgebühr nach Tarif II', and 'Tagesgebühr gemäß §17 zu kürzen'. The third option is selected. The 'Zu kürzende Tagesgebühr' (Deductible daily rate) field has a value of 100. In the 'Nächtigungsgebühr' (Accommodation Rate) section, there are three radio button options: 'Nächtigungsgebühr Nachweis', 'Nächtigungen ohne Nachweis', and 'Keinen Anspruch auf Nächtigungsgebühr'. The first option is selected. The 'Frühstücke Anzahl' (Breakfast count) field has a value of 4. The 'Mittagessen Anzahl' (Lunch count) field has a value of 3. The 'Abendessen Anzahl' (Dinner count) field has a value of 2. The 'Belege' (Receipts) section includes a 'Browse' button and a note to upload files via click or drag-and-drop. At the bottom, there's a summary table:

Laufnummer	Tag	Beginn	Ende	Art des Gebührenanspruches	Gesamtkilometer	Reisekosten	Tagesgebühr	Nächtigungsgebühr	Sonstige Nebenkosten	Summe
1	12.4.2021	(09:00) 19:00	<input type="checkbox"/> Reisegebühr <input checked="" type="checkbox"/> Tagesgebühr <input type="checkbox"/> Nächtigungsgebühr <input type="checkbox"/> Zusätzliche Kosten	5	5	5	5	5	15

Abbildung 7.12: Ein Bild der Reisekostenabrechnung

In Abbildung 7.12 wurden Elemente verwendet, die uns im Gegensatz zu normalen Eingaben nicht bekannt sind. Dabei handelt es sich einerseits um *Tabellen* und andererseits um die *Dateiauswahl*.

Dateiauswahl

Wie auch bei den anderen Elementen der Formulare, wie zum Beispiel bei dem *Timepicker*, hat *Bootstrap* ein vorgefertigtes Element für die *Dateiauswahl*.

```

1      <!-- Belege Input -->
2      <b-form-group
3          id="belg"
4              label-cols-sm="4"
5              label-cols-lg="3"
6              content-cols-sm
7              content-cols-lg="7"
8              description="Fügen Sie Belege per Klick oder Drag and Drop hinzu"
9              label="Belege"
10             label-for="bel"
11         >
12         <b-form-file
13             multiple
14             id="bel"
15             v-model="invoices"
16             v-on:input="convert"
17             :disabled="readonly"
18         >
19             <template slot="file-name" slot-scope="{ names }">
20                 <b-badge variant="dark">{{ names[0] }}</b-badge>
21                 <b-badge v-if="names.length > 1" variant="dark" class="ml-1">
22                     + {{ names.length - 1 }} More files
23                 </b-badge>
24             </template>
25         </b-form-file>
26     </b-form-group>
```

Auflistung 7.15: Datei auswählen Code

Wie auch andere Elemente, wird das Element zum Hochladen von Dateien in eine *Eingabegruppe*, mit Titel und Beschreibung eingesetzt. Nach dem Einsetzen des Elementes, kann sofort per *Klick bzw. Drag and Drop* eine oder mehrere Dateien hinzugefügt werden.

Tabelle

Um eine *Tabelle* zu erstellen, kann das Element *b-table* von *Bootstrap* benutzt werden. In diesem kann man direkt definieren, wie groß die *Tabelle* sein soll, was auf mobilen Geräten passieren soll, ob die *Tabelle* gestreift ist oder nicht und noch weitere Parameter. In diesem Element, wurden die verschiedenen Spalten eingebaut. Dies wurde wie folgt umgesetzt:

```
1   <b-table
2       striped
3       :items="data.items"
4       :fields="fields"
5       stacked="md"
6       show-empty
7       small
8   >
9
10 ...
11
12     <template #cell(start)="data">
13         <b-form-timepicker
14             style="min-width: 100px;"
15             id="begin"
16             locale="de"
17             placeholder="Zeit"
18             v-model="data.item.start"
19             :readonly="readonly"
20             v-on:input="update()"
21         ></b-form-timepicker>
22     </template>
23
24 ...
25
26 </b-table>
```

Auflistung 7.16: Beispielcode Tabelle

Die Spalten werden mittels *Templates* erzeugt. Danach können variabel Werte in jede Spalte, bzw. in jeder Zeile eingefügt werden. In diesem Fall befindet sich in der Zelle ein *Timepicker*, welcher zum Auswählen der Startzeit dient.

7.1.2.6 Ansicht aktive Anträge

Um den Lehrern das Betrachten von *aktiven Anträgen* einfacher zu machen, wurde eine zusätzliche Seite erstellt, die über das *Menü* auf der *Startseite* erreichbar ist. Hier werden *alle aktiven Anträge* (ein Antrag gilt als *aktiv*, von der Antragsstellung, bis zur Rechnungsphase) verzeichnet. Der Benutzer sieht mittels Farben auf den ersten Blick, welchen Status der jeweilige Antrag hat, Rot: Antrag wurde abgelehnt, Gelb: Antrag wird bearbeitet, Grün: Antrag wurde angenommen. Außerdem werden in der *Tabelle* der Titel des Antrages, der Leiter, das Einreichdatum und der genaue Status (In Bearbeitung, Rechnungsphase, ...) angezeigt. Für den Fall, dass der Benutzer viele Anträge auf einmal hat, wurde eine Funktion zum Suchen von Anträgen im oberen Teil der Seite eingebaut. Ebenfalls ist es möglich, jede Spalte auf- und absteigend zu sortieren. Falls der *Button* namens *Antrag betrachten* betätigt wurde, kommt der Nutzer auf die *Antrags Ansicht*.

Abbildung 7.13: Ein Bild der aktiven Anträge Seite

Um die oben genannten Daten in einer geordneten Liste zu sehen, kann der Lehrer auf *schnelle Information* klicken. Dieser Vorgang öffnet folgendes Fenster:



Abbildung 7.14: Ein Bild der Detail-Ansicht der aktiven Anträge Seite

Suchfunktion

Die Erstellung der *Suchfunktion* wurde wieder essentiell von den *BootstrapVue* Komponenten unterstützt. Es mussten eine *Eingabegruppe* erstellt werden, die ein *Eingabefeld* mit dem Typ *search* hat. Um eine Eingabe schneller zu löschen wurde noch ein *Button* hinzugefügt. Dieser erscheint, sobald man etwas eintippt. Durch die gute Integration von *Bootstrap* in *VueJS* war die Implementierung der *Suchfunktion* für die *Tabelle* reibungslos. Die Vergabe der Funktionalität der *Suchfunktion* wurde im *Backend/Frontend* Teil umgesetzt.

```

1      <!-- Such Element -->
2      <b-row align-h="center" style="margin-top: 1rem; margin-bottom: 2rem">
3          <b-col cols="12" md="6">
4              <b-form-group
5                  label="Filter"
6                  label-for="filter-input"
7                  label-cols-sm="3"
8                  label-align-sm="right"
9                  label-size="sm"
10                 class="mb-0"
11             >
12             <b-input-group size="sm">
13                 <b-form-input
14                     id="filter-input"
15                     v-model="filter"
16                     type="search"
17                     placeholder="Begriff suchen"
18                 ></b-form-input>
19
20                 <b-input-group-append>
21                     <b-button :disabled="!filter" @click="filter = ''">
22                         Löschen</b-button>
23                     >
24                 </b-input-group-append>
25             </b-input-group>
26         </b-form-group>
27     </b-col>
28 </b-row>
```

Auflistung 7.17: Anträge Suchfunktion Code

Filterbare Tabelle

Die Aktivierung der *filterbaren Tabelle* ist mittels *BootstrapVue* möglich. Dazu wurden folgende drei Festlegungen im *b-table* Element hinzugefügt:

```

1   <b-table
2     ...
3     :sort-by.sync="sortBy"
4     :sort-desc.sync="sortDesc"
5     :sort-direction="sortDirection"
6     ...
7   >
8   ...
9   </b-table>
```

Auflistung 7.18: Anträge filtern Code

Pop-Up Fenster

Das *Pop-Up Fenster* wurde mittels einem *Modal* von *Bootstrap* verwirklicht. Dieses kann über einen zugewiesenen *Button* geöffnet werden und beinhaltet, die spezifischen Daten des jeweiligen Antrags.

```

1   <!-- Info modal -->
2   <b-modal
3     :id="infoModal.id"
4     :title="infoModal.title"
5     ok-only
6     @hide="resetInfoModal"
7   >
8     <pre>{{ infoModal.content }}</pre>
9   </b-modal>
```

Auflistung 7.19: Pop-Up Fenster Code

7.1.2.7 Ansicht alle Anträge

Diese Seite ist sehr ähnlich zu der *aktive Anträge* Seite. Der Unterschied ist, dass der Nutzer hier auch bereits abgeschlossene Anträge sieht. Es wurde ebenfalls die *Suchfunktion* eingebaut, die hier eine wesentlich wichtigere Rolle spielt, als bei der Seite der *aktiven Anträge*. Auch das Filtern der einzelnen Spalten ist wieder möglich.

Alle Anträge						+ Neuer Antrag	Startseite
Titel	Leiter	Einreichdatum	Aktiv	Status	Aktionen		
Sommersportwoche	Stefan Zakall	14-06-2021	Offen	In Bearbeitung	Schnelle Information	Antrag Betrachten	
Pflegefreistellung	Stefan Zakall	14-04-2021	Offen	Abgeschlossen	Schnelle Information	Antrag Betrachten	

Abbildung 7.15: Ein Bild der alle Anträge Seite

7.1.2.8 Antragsansicht

Die *Antragsansicht* bietet den Nutzern alle möglichen Informationen zu ihrem Antrag, die sie brauchen. Um den Nutzern einen besseren Überblick darüber zu geben, in welcher Phase sich ihr Antrag befindet, wurde eine *Fortschrittsanzeige* als erstes Element auf der Seite eingebaut. Des Weiteren findet der Benutzer eine aufklappbare Liste auf der Seite, in der je, nach Status des Antrages, die Daten angezeigt und/oder editiert werden können (Abbildung 7.16 und Abbildung 7.17). Außerdem bietet diese Seite die Funktion, eine generierte *PDF-Datei* von jedem Antrag, bzw. bei einigen Anträgen auch eine *Excel-Datei*, zu öffnen. Falls der Nutzer Daten geändert hat, kann er diese Änderung mit einem *Button*, der sich am Schluss der Webseite befindet speichern. Ebenfalls ist es möglich, einen Antrag als abgeschlossen zu markieren. Hierbei öffnet sich aber eine Warnmeldung, in der man das Schließen des Antrages nochmal bestätigen muss (Abbildung 7.18). Des Weiteren wurde für den Nutzer eine Funktion eingebaut, um einen Antrag zu löschen, wie in Abbildung 7.19 dargestellt.

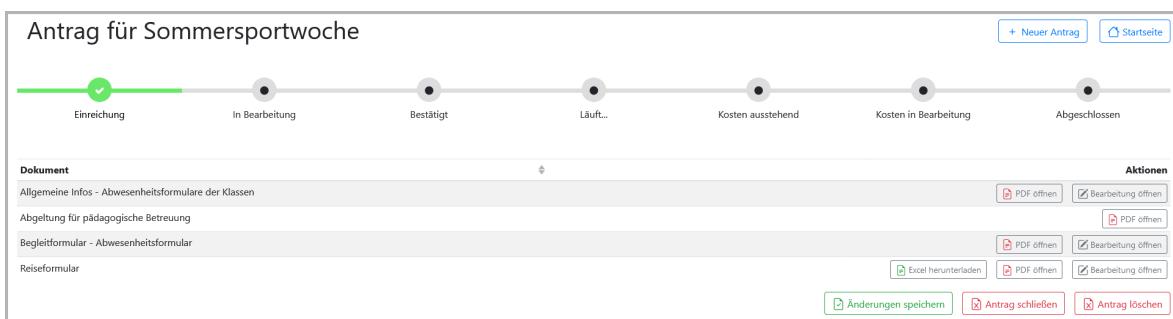


Abbildung 7.16: Ein Bild der Antragsansicht

Antrag für Sommersportwoche

Dokument	Aktionen
Fortbildung - Abwesenheitsformular	<input type="button" value="PDF öffnen"/> <input type="button" value="Bearbeitung schließen"/>
Titel Geben Sie den Titel der Fortbildung ein.	
Startdatum Geben Sie das Startdatum der Fortbildung ein.	<input type="button" value="Montag, 12. April 2021"/>
Startzeit Geben Sie die Startzeit der Fortbildung ein.	<input type="button" value="18:54"/>
Enddatum Geben Sie das Enddatum der Fortbildung ein.	<input type="button" value="Freitag, 16. April 2021"/>
Endzeit Geben Sie die Endzeit der Fortbildung ein.	<input type="button" value="18:54"/>
PH-Zahl Geben Sie Ihre PH-Zahl ein.	<input type="button" value="12"/>
Veranstalter Geben Sie den Namen des Veranstalters ein.	<input type="button" value="PRIA"/>
Art Wählen Sie die Art der Fortbildung.	<input type="radio"/> Seminar <input type="radio"/> Tagung <input type="radio"/> Lehrgang <input checked="" type="radio"/> Sonstiges
Anmerkungen Geben Sie zusätzliche Anmerkungen an.	<input type="button" value="Sommersportwoche ist cool"/>
Reiseformular	<input type="button" value="Excel herunterladen"/> <input type="button" value="PDF öffnen"/> <input type="button" value="Bearbeitung öffnen"/>
Reiserechnung	<input type="button" value="Excel herunterladen"/> <input type="button" value="PDF öffnen"/> <input type="button" value="Bearbeitung öffnen"/>
	<input type="button" value="Änderungen speichern"/> <input type="button" value="Antrag schließen"/> <input type="button" value="Antrag löschen"/>

Abbildung 7.17: Ein Bild der Antragsansicht mit geöffneter Liste

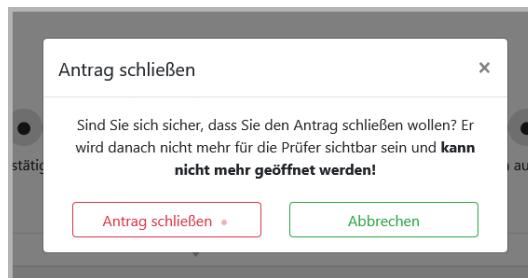


Abbildung 7.18: Ein Bild der Antrag schließen Warnung



Abbildung 7.19: Ein Bild der Antrag löschen Warnung

Fortschritletsanzeige

Ein Schritt der *Fortschritletsanzeige* besteht aus einem *div*, in dem sich ein *span* befindet. Das *div* ist ein „Strich“ und in dem *span* befindet sich ein *Icon*, welches den Zustand beschreibt. Je nach Status des Fortschritts wird die Farbe mittels CSS-Klassen geändert.

```

1   <div
2       :class="{
3           active: progress >= 1
4       }"
5       class="step"
6   >
7       <span class="icon">
8           <i
9               :class="{
10                  'fa-check': progress >= 1,
11                  'fa-circle': progress < 1 && progress > 0,
12                  'fa-exclamation': progress == 0
13              }"
14             class="fa"
15         ></i>
16     </span>
17     <span class="text text-truncate d-none d-md-block">
18         >Einreichung</span>
19     >
20 </div>
```

Auflistung 7.20: Codeausschnitt - Fortschrittsanzeige

```

1 .track .step.fail:before {
2     background: #ff3e3e;
3 }
4
5 .track .step.fail .icon {
6     background: #ff3e3e;
7     color: #fff;
8 }
9
10 .track .step.active .icon {
11     background: #69e769;
12     color: #fff;
13 }
14
15 .track .step.active:before {
16     background: #69e769;
17 }
```

Auflistung 7.21: CSS - Fortschrittsanzeige Farbklassen

Dokumente anzeigen

Die verschiedenen Dokumente, die zu einem gewissen Antrag gehören, werden in einer *Tabelle* gelistet. Falls der Nutzer auf den *Bearbeitung öffnen - Button* drückt, wird der *Tabelle* eine weitere Reihe hinzugefügt, die alle Informationen zu dem jeweiligen Antrag beinhaltet. Mittels *VueJS* wurden die bestehenden Formulare (hier als Beispiel ein Schulveranstaltungsantrag) in der *Tabelle* eingebunden und die *Eingabefelder* über die Datenschnittstelle (Abschnitt 7.3) mit Daten befüllt.

```
1   <template #row-details="row">
2     <b-card>
3       <!-- Schulveranstaltung -->
4       <SchoolGeneral
5         v-bind:readonly="sgreadonly"
6         v-bind:data="app"
7         v-bind:token="token"
8         v-bind:url="url"
9         v-on:update="updateSG"
10        v-if="isLeader && row.item.title == 'Allgemeine Infos'"
11      />
12    </b-card>
13  </template>
```

Auflistung 7.22: Dokumente anzeigen Code

Funktionen

Wird der *Antrag schließen - Button* betätigt, erscheint eine Sicherheitsmeldung. Diese wird mit einem *Modal* erstellt, welches einen Text beinhaltet, der den Nutzer vor den Folgen seiner Tat informiert. Hier erscheinen zwei weitere *Buttons*: ein grüner *Button*, um den Vorgang abzubrechen und ein roter *Button*, der einen rot blinkenden *Spinner* beinhaltet.

```

1  <!-- Sicherheitshinweis Antrag schließen -->
2  <b-modal ref="close-modal" hide-footer title="Antrag schließen">
3      <b-container fluid>
4          <b-row>
5              <b-col cols="12">
6                  <div class="d-block text-center">
7                      <p>
8                          Sind Sie sich sicher, dass Sie den Antrag schließen wollen? Er
9                          wird danach nicht mehr für die Prüfer sichtbar sein und
10                         <b>kann nicht mehr geöffnet werden!</b>
11                     </p>
12                 </div>
13             </b-col></b-row>
14         >
15         <b-row>
16             <b-col cols="6">
17                 <!-- Antrag schließen bestätigung -->
18                 <b-button class="mt-2" variant="outline-danger" block @click="delAn">
19                     Antrag schließen <b-spinner small type="grow"></b-spinner>
20                 </b-button>
21             </b-col>
22             <b-col cols="6">
23                 <!-- Abbrechen Button --><b-button
24                     class="mt-2"
25                     variant="outline-success"
26                     block
27                     @click="hideClose"
28                     >Abbrechen</b-button>
29             </b-col>
30         >
31     </b-row>
32     </b-container>
33 </b-modal>
```

Auflistung 7.23: Antrag schließen Sicherheitshinweis Code

7.1.2.9 Administrator Ansicht

Die *Administratoransicht* ist, wie die *alle Anträge* oder *aktive Anträge* Ansicht der Nutzer mit der Struktur einer *Tabelle* aufgebaut und verfügt sowohl über eine Such- als auch eine *Filterfunktion*. Es gibt aber auch eine Zusatzfunktion, damit man direkt in der Übersicht, einen oder mehrere Anträge auswählen kann und diese direkt als *PDFs* exportieren kann. Außerdem sehen die Administratoren auf den ersten Blick wesentlich mehr Informationen als normale Nutzer. Es wird der Titel des Antrages angezeigt, die Art, das Enreichdatum, für wann der Antrag ist, den Status (je nach Status sehen gewisse Rollen die Anträge) und den Antragssteller in Kurzschreibweise. Auch hier ist es wieder möglich, die *Antragsansicht* zu öffnen.

Alle Anträge							
		Filter		Löschen			
Ausgewählt	Titel	Art	Einreich Datum	Beginn	Status	Antragsteller	Aktionen
	Sommersportwoche	Schulveranstaltung	2021-01-1	2021-31-12	EINREICHUNG	Stefan Zekall	Antrag betrachten
	Pflegefreistellung	Sonstiger Antrag	2021-01-1	2021-31-12	ABGESCHLOSSEN	Stefan Zekall	Antrag betrachten
	Fortbildung	Fortbildung	2021-01-1	2021-31-12	ABGELEHNT	Stefan Zekall	Antrag betrachten

Abbildung 7.20: Ein Bild der Admin Ansicht

Die farbig hinterlegten Elemente heißen *badges* und können wie folgt erstellt werden:

```

1   <b-badge variant="danger">Rote Badge</b-badge>
2   <b-badge variant="success">Grüne Badge</b-badge>
3   <b-badge>Graue Badge</b-badge>
```

Auflistung 7.24: Badge Beispielcode

Administrator Antragsansicht

Die *Administrator Antragsansicht* sieht gleich aus, wie die Ansicht der normalen Benutzer - mit einem Unterschied. Die Administratoren haben nicht die Funktionen *Speichern* und *Antrag schließen*, sondern *Antrag annehmen* und *Antrag ablehnen*. Falls der Antrag abgelehnt wird, erscheint wieder eine Sicherheitsmeldung. Hier muss der Administrator eine Begründung für die Ablehnung angeben.

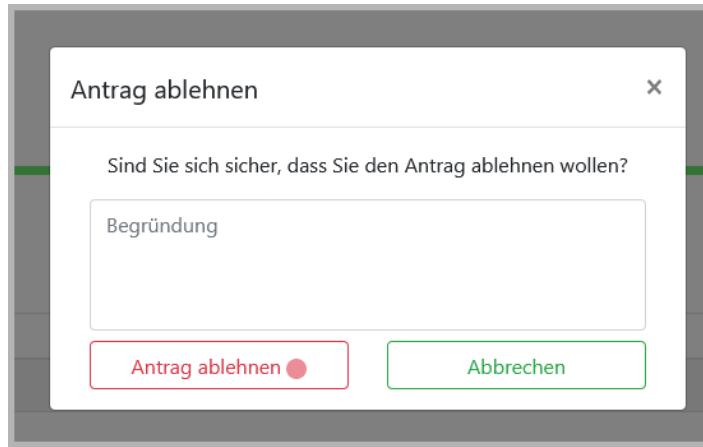


Abbildung 7.21: Ein Bild der Antrag ablehnen Meldung

Im Code wird dieses *Eingabefeld* im *Modal* wie folgt erstellt:

```

1  <!-- Antrag Ablehnen Warnhinweis -->
2  <b-modal ref="close-modal" hide-footer title="Antrag ablehnen">
3      <b-container fluid>
4          <b-row>
5              ><b-col cols="12">
6                  <div class="d-block text-center">
7                      <p>
8                          Sind Sie sich sicher, dass Sie den Antrag ablehnen wollen?
9                      </p>
10                     <!-- Begründung der Ablehnung -->
11                     <b-form-textarea
12                         id="decline-reason"
13                         placeholder="Begründung"
14                         rows="3"
15                         no-resize
16                     ></b-form-textarea>
17                 </div> </b-col>
18             ></b-row>
19             <b-row>
20                 <b-col cols="6">
21                     <!-- Antrag ablehnen Bestätigung -->
22                     <b-button class="mt-2" variant="outline-danger" block @click="delAn">
23                         Antrag ablehnen <b-spinner small type="grow"></b-spinner>
24                     </b-button>
25                 </b-col>
26                 <b-col cols="6">
27                     <!-- Abbrechen Button -->
28                     <b-button

```

```

29         class="mt-2"
30         variant="outline-success"
31         block
32         @click="hideClose"
33         >Abbrechen</b-button
34     ></b-col
35     >
36   </b-row>
37 </b-container>
38 </b-modal>

```

Auflistung 7.25: Antrag ablehnen Code

7.1.2.10 Seite nicht gefunden

Falls der momentane Nutzer von *Refundable* eine ungültige URL aufrufen möchte, wird dieser sofort informiert, dass er etwas falsch gemacht hat. Des Weiteren wird dem Nutzer direkt eine Lösung vorschlagen, wie er dieses Problem beheben kann. Um dem Nutzer zu visualisieren, dass der „Website-Button“ die richtige Wahl ist, haben wir ihn in Grün hervorgehoben. Außerdem wird dem Aufrufer der Seite mit schimmernden grauen Boxen, die einen „Lade-Zustand“ visualisieren, gezeigt, dass die Seite nicht richtig lädt.

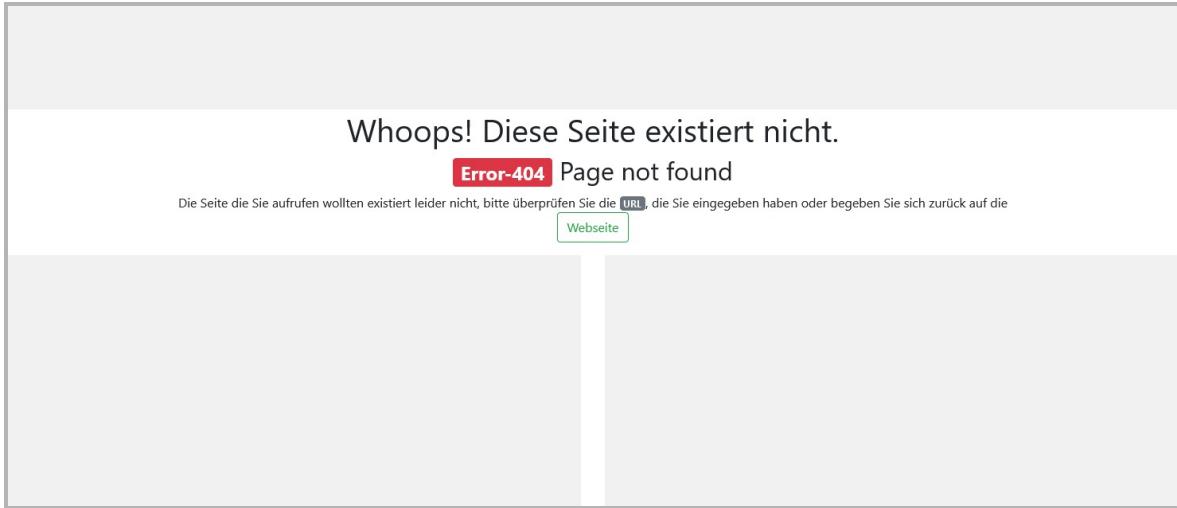


Abbildung 7.22: Ein Bild der nicht gefunden Seite

Die Elemente, welche den Lade-Zustand verkörpern sollen, heißen *b-skeleton-img* und werden wie folgt erstellt:

```
1 <b-skeleton-img animation="fade"></b-skeleton-img>
```

Auflistung 7.26: Skeleton Beispielcode

7.1.2.11 Antrag suchen

Um den Administratoren während ihren Tätigkeiten ein wenig Arbeit abzunehmen, wurde eine Möglichkeit eingebaut, einen Antrag direkt über eine Seite zu suchen. Dazu muss der Administrator die Unterseite „/viewer“ aufrufen und kann, wie man in Abbildung 7.23 sehen kann, die Antrags ID suchen und so direkt zu dem gewünschten Antrag gelangen.

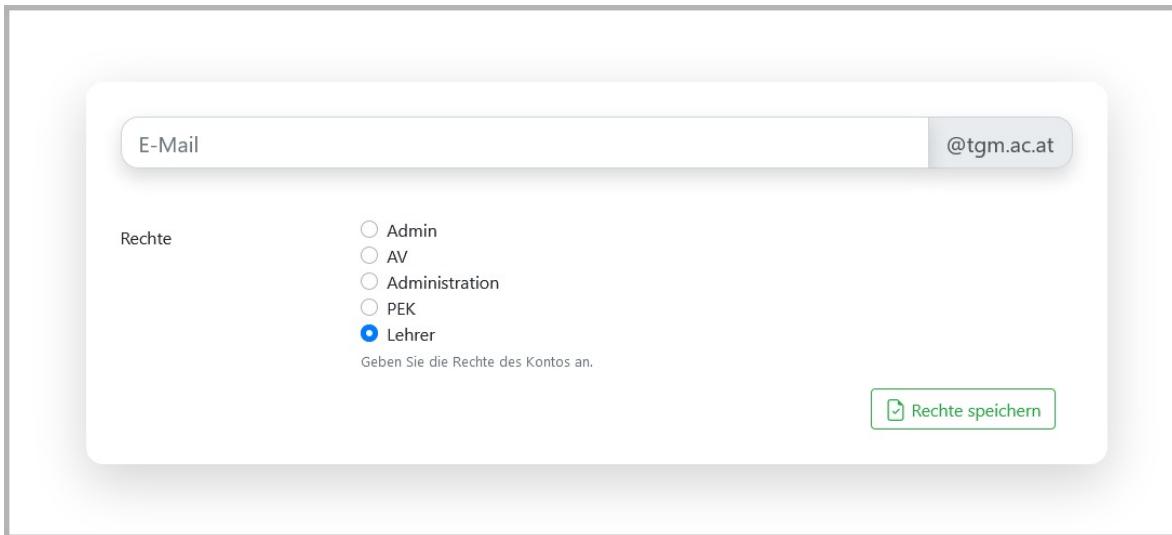


Antrags ID
Geben Sie die ID des gewünschten Antrags ein.
Suchen

Abbildung 7.23: Ein Bild der Antrag suchen Seite

7.1.2.12 Rechte vergeben

Während der Entwicklung ist uns aufgefallen, dass eine Funktion fehlt, einen bestimmten Benutzer mit gewissen Rechten auszustatten. Daher ist es dem *Superuser* möglich, über das *Admindashboard* Rechte zu vergeben. Dazu muss er nur die E-Mail Adresse des in Frage kommenden Nutzers eingeben und mittels *Radio-Buttons* die gewünschten Rechte auswählen. Auch hier wurden wieder die bekannten Elemente der restlichen Webseite verwendet.



E-Mail @tgm.ac.at

Rechte

Admin
 AV
 Administration
 PEK
 Lehrer

Geben Sie die Rechte des Kontos an.

Rechte speichern

Abbildung 7.24: Ein Bild der Rechte vergeben Seite

7.2 Backend und Infrastruktur

7.2.1 Einleitung

Wie im Konzept (in Abschnitt 6.2) zuvor ausführlich beschrieben, soll eine den Plänen entsprechende automatisch aufbaubare Infrastruktur erstellt und ein dem System angepasstes *Backend* programmiert werden. Hierfür wurde bereits in der Studie (in Unterunterabschnitt 5.2.4.3) *Golang* als zu benutzende Sprache ausgewählt.

7.2.2 Docker

In den folgenden Kapiteln wird die genaue Umsetzung der einzelnen *Docker-Images*, jeweils definiert in *Dockerfiles*, des *Docker-Compose Stack* und des Aufbaus der Volumen, Umgebungsvariablen und *Secrets*, beschrieben.

7.2.2.1 MongoDB

Das *Dockerfile* in der Auflistung 7.27 beschreibt die Datenbank, eine *MongoDB* Instanz, die einen eigenen *Container* erhält. Das aus dem *Dockerfile* resultierende *Image* basiert auf der neuesten Version des offiziellen „*mongo*“-*Images*, welches die eigentliche *MongoDB*-Instanz beinhaltet. Des Weiteren wird das *Init Skript* hinzugefügt, sodass dieses beim Erstellen des *Containers* immer ausgeführt wird. Zuletzt wird noch der *Port* 27017, der Standard *Port* der *MongoDB* Anwendung, nach außen hin freigegeben.

```
1 FROM mongo:latest
2
3 ADD ./init-db.sh /docker-entrypoint-initdb.d/
4
5 EXPOSE 27017
```

Auflistung 7.27: *MongoDB Dockerfile*

Das *Bash Skript* in der Auflistung 7.28 sorgt für die Erstellung eines Nutzers als Administrator, einer Datenbank und eines weiteren Benutzers mit einfacher Lese- und Schreibberechtigung. Dieser Vorgang ist nötig, damit die Datenbank direkt verwendbar ist, und nicht erst eine Datenbank von der Seite des Programmes aus erstellt werden muss. Der weitere User stellt ein *Best-Practice* da, indem man den Nutzer, mit welchen man jegliche Daten schreibt, nicht alle Berechtigungen zuschreibt, sondern lediglich jene, die er wirklich braucht.

```

1  #!/bin/bash
2
3 mongo -- "$MONGO_DATABASE" << EOF
4     var rootUser = `$(cat "$MONGO_INITDB_ROOT_USERNAME_FILE")`;
5     var rootPassword = `$(cat "$MONGO_INITDB_ROOT_PASSWORD_FILE")`;
6     var admin = db.getSiblingDB("$MONGO_INITDB_DATABASE");
7     admin.auth(rootUser, rootPassword);
8
9     var user = `$(cat "$MONGO_USERNAME_FILE")`;
10    var passwd = `$(cat "$MONGO_PASSWORD_FILE")`;
11    db.createUser({user: user, pwd: passwd, roles: ["readWrite"]});
12
13 EOF

```

Auflistung 7.28: MongoDB Initskript

Die Auflistung 7.29 zeigt wie das eigens definierte *MongoDB Image* weiterverwendet wird, speziell wie die verschiedenen Nutzerinformationen in den *Container* weitergegeben werden. Dies geschieht über *Secrets* [45]. Hierbei handelt es sich um einfache Dateien, welche sicher in den *Container* eingehängt und dann darin ausgelesen werden können. In diesem Fall werden die Dateien durch den Installationsvorgang des Systems erstellt.

```

1 mongo:
2   build: images/mongo/
3   container_name: 'mongo-database'
4   restart: always
5   ports:
6     - '27017:27017'
7   volumes:
8     - './volumes/mongo/:/data/db/'
9   environment:
10    MONGO_INITDB_ROOT_USERNAME_FILE: '/run/secrets/mongodb_root_username'
11    MONGO_INITDB_ROOT_PASSWORD_FILE: '/run/secrets/mongodb_root_password'
12    MONGO_INITDB_DATABASE: 'admin'
13    MONGO_USERNAME_FILE: '/run/secrets/mongodb_username'
14    MONGO_PASSWORD_FILE: '/run/secrets/mongodb_password'
15    MONGO_DATABASE: 'refundable'
16   secrets:
17     - mongodb_root_username
18     - mongodb_root_password
19     - mongodb_username
20     - mongodb_password

```

Auflistung 7.29: Docker-Compose File: MongoDB-Container

7.2.2.2 Backend

Im *Golang-Backend*, welches in der Auflistung 7.30 beschrieben wird, läuft nicht nur das *Backend* selbst, sondern es wird auch kompiliert. Dies bringt den Vorteil mit sich, dass es automatisch für das von *Docker* bereitgestellte System kompiliert wird und es keine Auslieferung von bereits kompilierten Dateien braucht. Es wird hier ein neues Image gebaut, woraufhin in den *Go-Path* die Quellcodedateien hineinkopiert werden. Danach werden alle benötigten Bibliotheken heruntergeladen und das dann vollständige Projekt kompiliert. Nachdem der *Port* 8080 freigegeben wurde, wird das fertig kompilierte Programm gestartet.

```

1  FROM golang:1.14
2
3  WORKDIR /go/src/huginn
4  COPY . .
5
6  RUN go get -d -v ./...
7  RUN go install -v ./...
8
9  EXPOSE 8080
10
11 CMD ["huginn"]

```

Auflistung 7.30: *Backend Dockerfile*

Dieses *Docker-Image* ist ein Teil des *Docker-Stacks* und somit ist es auch in der *Docker-Compose* Konfiguration vorhanden (Auflistung 7.31). Hier werden verschiedene Volumen eingehängt um Dateien, wie die generierten Formulare, oder *Secrets* zum Signieren von Tokens, permanent zu speichern. Außerdem werden wieder *Secrets*, die bereits im *MongoDB Container*, die Nutzerinformationen des Datenbanknutzers gespeichert hatten, hinzugefügt [45]. Dies ist darauf zurückzuführen, dass diese gleich durch das *Backend* wiederverwendet werden, um sich mit dem Nutzer einloggen zu können.

```

1  rest:
2      build: src/huginn/
3      container_name: 'rest-api-backend'
4      ports:
5          - '8080:8080'
6      volumes:
7          - './volumes/secrets/:/vol/secrets/'
8          - './volumes/files/:/vol/files/'
9      environment:
10         MONGO_USERNAME_FILE: '/run/secrets/mongodb_username'
11         MONGO_PASSWORD_FILE: '/run/secrets/mongodb_password'
12         MONGO_DATABASE: 'refundable'
13      secrets:
14          - mongodb_username
15          - mongodb_password

```

Auflistung 7.31: *Docker-Compose File: Backend-Container*

7.2.2.3 Frontend

Zuletzt wird noch das *Frontend* in Auflistung 7.32 definiert. Hierbei wird ein ähnlicher Zugang wie mit dem *Backend* gewählt, sodass auch das *Frontend* erst in einem *Docker-Container* kompiliert wird. Die Herausforderung besteht darin, dass eine eigene *Node.js* Umgebung hierfür benötigt wird. Aus diesem Grund baut das *Image* grundsätzlich eine solche Umgebung auf und ladet darin die entsprechend benötigten Abhängigkeiten, wie Bibliotheken und *Frameworks*, herunter. Die Quelldateien werden anschließend kompiliert. Das fertig kompilierte *Frontend* wird in einen anderen *Docker-Container* verschoben, welcher den eigentlichen *Webserver* umfasst, der das *Frontend* dann bereit stellt. Der *nginx-Webserver* wird, nachdem er den *Port* 80 freigegeben hat, formell gestartet.

```

1  FROM node:lts-alpine AS vue-builder
2  WORKDIR /app
3  COPY package*.json .
4  RUN npm install
5  COPY . .
6  RUN npm run build
7
8  FROM nginx:stable-alpine as production
9  COPY --from=vue-builder /app/dist /usr/share/nginx/html
10 COPY history.conf /etc/nginx/conf.d/history.conf
11 EXPOSE 80
12 CMD ["nginx", "-g", "daemon off;"]

```

Auflistung 7.32: *Frontend Dockerfile*

Auch dieses *Image* muss in der *Docker-Konfiguration* hinzugefügt werden. Dies wird durch die in Auflistung 7.33 aufgeführten Konfigurationen erreicht. Hierbei werden lediglich die entsprechenden *Ports* festgelegt. Weitere Informationen, wie beispielsweise *Secrets*, werden vom *Webserver* benötigt.

```

1  web:
2    build: src/web/
3    container_name: 'frontend-nginx-webapp'
4    ports:
5      - '80:80'
6      - '443:443'

```

Auflistung 7.33: *Docker-Compose File: Web-Container*

7.2.2.4 Secrets

Nachdem die verschiedenen *Images*, aus denen der *Docker-Compose Stack* bestehen soll, konfiguriert und in die zentrale *Docker-Compose* Datei hinzugefügt wurden, müssen zuletzt noch die benutzten *Secrets* registriert werden [45]. Der *Code* ist in Auflistung 7.34 ersichtlich. Der Name des *Secrets* wird einer Datei zugeordnet. Diese Datei wird, wie bereits erwähnt, in den entsprechenden *Containern* eingehängt. Hierdurch lassen sich die bei der Installation eingelesenen Daten einfach in die *Container* weitergeben.

```

1 secrets:
2     mongodb_root_username:
3         file: config/mongodb_root_username
4     mongodb_root_password:
5         file: config/mongodb_root_password
6     mongodb_username:
7         file: config/mongodb_username
8     mongodb_password:
9         file: config/mongodb_password

```

Auflistung 7.34: *Docker-Compose File: Secrets*

7.2.3 Steuerungsskript

7.2.3.1 Docker Installation

Bei der *Docker* Installation, welche in Auflistung 7.35 ersichtlich ist, wird *Docker* und *Docker-Compose* während des Installationsvorganges installiert, sofern dies so angegeben ist. Hierfür existiert der Parameter „-d“. Ist dieser vorhanden, so wird das offizielle *Docker-Installer* Skript heruntergeladen und ausgeführt. Danach wird noch *Docker-Compose* installiert. Dies geschieht durch das Herunterladen des eigentlichen Programms und der anschließenden Rechtezuteilung zur Ausführung dessen.

```

1 if [ "$1" = "-d" ]; then
2     printf "Downloading Docker.\n"
3     curl -fsSL get.docker.com -o docker-install.sh > /dev/null
4     printf "Download complete. Now installing. This may take a while.\n"
5     sudo sh docker-install.sh
6     rm docker-install.sh
7     printf "\n-----\n"
8     printf "Downloading Docker-Compose.\n"
9     sudo curl -L "https://github.com/docker/compose/releases/download/1.27.4/docker_"
10    ↪ _compose-$(uname -s)-$(uname -m)" -o
11    ↪ /usr/local/bin/docker-compose
12    printf "Download complete.\n"
13    sudo chmod +x /usr/local/bin/docker-compose
14    printf "Docker and Docker Compose installed successfully.\n"
15    printf "\n-----\n"
16 fi

```

Auflistung 7.35: *Docker Installation - Steuerungsskript*

7.2.3.2 Installation

In der Auflistung 7.36 ist der Installationsvorgang dargestellt. Der Kommentar signalisiert, wo die *Docker*-Installation eigentlich in der Methode stehen würde. Der Installationsvorgang gliedert sich in zwei Teile. Im ersten werden alle *Git-Repositories* automatisch geklont. Dadurch wurden nun alle Quellcodedateien, welche nach dem Starten in etwaigen *Docker-Containern* enden würden, heruntergeladen. Zusätzlich wird der Nutzer nun die Nutzerinformationen des Datenbankadministrators und des Datenbanknutzers abgefragt. Diese Informationen werden in die entsprechenden Dateien gespeichert, sodass sie danach von *Docker* als *Secrets* genutzt werden können. Zuletzt wird sowohl im *Backend* als auch im *Frontend* die Adresse benötigt unter welcher der *Server* erreichbar ist. Diese *URL* wird jeweils in eine dafür vorgesehene Datei im *Backend* und *Frontend* gespeichert, sodass diese auslesbar sind.

```

1 sub_install() {
2     # Docker Installation
3     mkdir src/
4     printf "Downloading the Backend (huginn).\n"
5     git clone --quiet https://github.com/refundable-tgm/huginn.git src/huginn >
6         /dev/null
7     printf "Download complete\nDownloading the Frontend (web).\n"
8     git clone --quiet https://github.com/refundable-tgm/web.git src/web > /dev/null
9     printf "Download complete.\n"
10    printf "\n-----\n\n"
11    read -rp "Choose a MongoDB Admin Username: " adminusername
12    read -srp "Choose a MongoDB Admin Password: " adminpassword
13    printf "\n"
14    read -rp "Choose a MongoDB Username: " username
15    read -srp "Choose a MongoDB User Password: " userpassword
16    echo "$adminusername" > config/mongodb_root_username
17    echo "$adminpassword" > config/mongodb_root_password
18    echo "$username" > config/mongodb_username
19    echo "$userpassword" > config/mongodb_password
20    printf "\n"
21    read -rp "Input the URL (without port, this is will be 8080 by default):" url
22    printf '{\n\t"url": "%s"\n}' "$url" > src/web/src/data.json
23    echo "$url" > volumes/files/.url
24    printf "\n"
25    read -rp "Input the name of the initial super user" superuser
26    echo "$superuser" > volumes/files/.url
27    printf "\nInstallation completed.\n"
}
```

Auflistung 7.36: Installationsvorgang - Steuerungsskript

7.2.3.3 Startvorgang

Der Startvorgang ist sehr einfach aufgebaut. Er ist in der Auflistung 7.37 ersichtlich. Zudem wird hier geprüft, ob das System installiert ist. Ist dies der Fall, so wird die Infrastruktur über den entsprechenden *Docker-Compose* Befehl hochgefahren.

```

1  sub_start() {
2      check_installed
3      printf "Refundable is starting.\n"
4      docker-compose up --build -d > /dev/null
5      printf "Refundable started.\n"
6  }

```

Auflistung 7.37: Startvorgang - Steuerungsskript

7.2.3.4 Stoppvorgang

Der in der Auflistung 7.38 aufgeführte Stoppvorgang ist, analog zum Startvorgang, einfach aufgebaut. Er führt nach der Überprüfung, ob das System installiert ist, lediglich den Befehl zum Stoppen des *Docker-Compose Stacks* aus.

```

1  sub_stop() {
2      check_installed
3      printf "Refundable is stopping.\n"
4      docker-compose down > /dev/null
5      printf "Refundable stopped.\n"
6  }

```

Auflistung 7.38: Stoppvorgang - Steuerungsskript

7.2.3.5 Neustart

Der in der Auflistung 7.39 aufgeführte Neustart ist ein Zusammenschluss aus dem Startvorgang und dem Stoppvorgang. Daraus resultiert ein inhärenter Vorgang, welcher durch den bereits beschriebenen einfachen Aufbau der beiden Abläufe nicht komplex ist.

```

1  sub_restart() {
2      check_installed
3      sub_start
4      sub_stop
5  }

```

Auflistung 7.39: Neustartvorgang - Steuerungsskript

7.2.3.6 Aktualisierung

Die in der Auflistung 7.40 gezeigte Aktualisierungsfunktion basiert auf dem Prinzip des *Git-Repositories*. Dadurch, dass während Installation *Git-Repositories* geklont werden, können diese nun einfach neue Erweiterungen der Software herunterladen. Diese Operation wird in allen drei betroffenen Ordner (*Backend*, *Frontend* und Infrastruktur) ausgeführt, um die neuen Änderungen herunterzuladen. Danach wird der *Docker-Compose Stack* gestoppt und alle veralteten Komponenten werden aus dem System entfernt. Zuletzt werden auf Basis der heruntergeladenen Änderungen alle Komponenten erneut aufgebaut und gestartet.

```

1  sub_update() {
2      check_installed
3      printf "Backend (huginn) will be updated, if available.\n"
4      cd src/huginn
5      git pull --quiet origin master > /dev/null
6      printf "Process completed.\n-----\n\n"
7      printf "Frontend (web) will be updated, if available.\n"
8      cd ../web
9      git pull --quiet origin master > /dev/null
10     printf "Process completed.\n-----\n\n"
11     printf "Infrastructure will be updated, if available.\n"
12     cd ../..
13     git pull --quiet origin master > /dev/null
14     printf "Process completed.\n-----\n\n"
15     sub_stop
16     printf "\n-----\n\n"
17     clean
18     printf "\n-----\n\n"
19     sub_start
20     printf "Update process completed.\n"
21 }
```

Auflistung 7.40: Aktualisierungsvorgang - Steuerungsskript

7.2.3.7 Bereinigungsvorgang

Der Bereinigungsvorgang, dargestellt in der Auflistung 7.41, bereinigt das System von allen alten, nicht mehr genutzten Dateien. Hierfür wird eine Bestätigung benötigt, damit die Operation wirklich durchgeführt wird, da es sich hierbei um das Löschen von Daten handelt.

```

1  sub_clean() {
2      check_installed
3      read -rp "All dangling components will be removed. Continue? [y|N] " y
4      printf "\n"
5      case $y in
6          "Y" | "y" )
7          clean
8          ;;
9          *)           printf "Cleaning aborted.\n"
10         exit 0
11         ;;
12     esac
13 }
14 }
```

Auflistung 7.41: Bereinigungsvorgang - Steuerungsskript

7.2.3.8 Deinstallation

Die in der Auflistung 7.42 dargestellte Deinstallation stoppt das System und fährt die Infrastruktur herunter. Danach werden alle verbleibenden Komponenten aus der Systemumgebung entfernt. Zusätzlich werden die heruntergeladenen Dateien allesamt gelöscht. Da es sich hier ebenfalls um Operation handelt, die Daten löscht, benötigt diese ebenfalls eine Bestätigung.

```

1 sub_purge() {
2     read -rp "Refundable will be removed completely. Continue? [y|N] " y
3     printf "\n"
4     case $y in
5         "Y" | "y")
6         sub_stop
7         clean
8         rm -rf config images volumes docker-compose.yml README.md refundable.sh
9             ↳ src .git docker-install.sh
10            printf "Refundable was removed.\n"
11            ;;
12        *)
13            printf "Aborted.\n"
14            exit 0
15            ;;
16    esac
17 }
```

Auflistung 7.42: Deinstallation - Steuerungsskript

7.2.3.9 Subcommands

Die *Subcommands* dieses Steuerungsskripts werden namentlich auf die jeweiligen Methoden übertragen [48]. Dieser Vorgang ist in der Auflistung 7.43 ersichtlich. Durch das Nutzen dieser Methodik können weitere Befehle sehr einfach hinzufügt werden.

```

1 sub=$1
2 case $sub in
3     "" | "-h" | "--help" | "help")
4     sub_help
5     ;;
6     *)
7     shift
8     sub_"${sub}" "$@"
9     if [ $? = 127 ]; then
10        echo "Error: '$sub' is not a known subcommand." >&2
11        sub_help
12        exit 1
13    fi
14    ;;
15 esac
```

Auflistung 7.43: Subcommand parsing - Steuerungsskript [48]

7.2.4 MongoDB

Die Schnittstelle zwischen der *MongoDB* Instanz und dem *Backend* ist wichtig für das System. In dieser Datenbank werden alle Daten gespeichert, die für das System von Relevanz sind. Für diese Datenstrukturen gibt es verschiedene Methoden, um mit den Daten mittels der vier Hauptfunktionen, Erstellen, Lesen, Aktualisieren und Löschen, auf der Datenbank zu interagieren und arbeiten zu können. Die Schnittstelle zwischen Programm und Datenbank wurde hierbei mittels dem offiziellen *Golang-MongoDB*-Treiber umgesetzt [35]. Auf Basis dessen sind die folgenden Methoden implementiert.

Methodenname	Beschreibung	Parameter	Rückgabe
Connect()	Öffnet die Verbindung zur DB	—	ok
Close()	Schließt die Verbindung zur DB	—	ok
CreateApplication()	Erstellt einen neuen Antrag	Application	ok
GetApplication()	Liest einen bestimmten Antrag aus	uuid	Application
GetAllApplications()	Liest alle Anträge aus	—	[]Application
GetActiveApplications()	Liest alle aktiven Anträge aus	—	[]Application
UpdateApplication()	Aktualisiert die Daten eines Antrags	uuid, Application	ok
DeleteApplication()	Löscht eine Antrags	uuid	ok
CreateTeacher()	Erstellt eine neue Lehrkraft	Teacher	ok
GetTeacherByShort()	Gibt die Daten der Lehrkraft zurück	username	Teacher
DoesTeacherExistByShort()	Überprüft ob die Lehrkraft existiert	username	bool
GetTeacherByUUID()	Gibt die Daten der Lehrkraft zurück	uuid	Teacher
UpdateTeacher()	Aktualisiert die Daten einer Lehrkraft	uuid, Teacher	ok
DeleteTeacher()	Löscht eine Lehrkraft aus der DB	uuid	ok
resolveURI()	Erstellt die URI zur Datenbank	—	URI
getInitUserName()	Liest den initialen Usernamen aus	—	username

Tabelle 7.1: Methoden der *MongoDB* Schnittstelle

Hierbei fungiert der offizielle Treiber als Mittel zur Kommunikation zwischen der Programmiersprache und der Datenbank-Instanz. Ein Beispiel kann wie folgt umgesetzt werden:

```
1 func (m MongoDBConnector) GetApplication(uuid string) (application
2     Application) {
3     collection := m.client.Database(m.database).Collection(ApplicationCollection)
4     if err := collection.FindOne(m.context, bson.M{"uuid":
5         uuid}).Decode(&application); err != nil {
6         log.Println(err)
7         return
8     }
9     return application
10 }
```

Auflistung 7.44: *MongoDB* Treiber Beispiel: *GetApplication* Methode

Die abgebildete Methode sucht nach einem bestimmten Antrag, welcher durch seine *UUID* identifiziert wird. Hierfür wird das Verzeichnis, die Kollektion, geöffnet, in der alle Anträge gespeichert sind [35]. Danach wird nach einem Antrag gesucht, wobei in diesem das Feld der *UUID* gleich dem Wert der gesuchten *UUID* sein muss. Das gefundene Objekt wird in eine Variable gespeichert und zurückgegeben.

7.2.5 LDAP

Neben der *MongoDB* muss eine Verbindung zum *TGM*-eigenen *Active Directory* über *LDAP* hergestellt werden. Der Grund dafür ist, dass das System keine eigenen Accounts mit Passwörtern speichert, sondern jene des *TGMs* direkt übernehmen soll. Des Weiteren kann über diesen Dienst der komplette Name der Lehrkraft herausgefunden werden. Die Authentifizierung eines Nutzers gestaltet sich wie folgt:

```

1 func AuthenticateUserCredentials(username, password string) bool {
2     cred := username + "@tgm.ac.at"
3     l, err := ldap.Dial("tcp", fmt.Sprintf("%s:%d", URL, Port))
4     if err != nil {
5         return false
6     }
7     err = l.Bind(cred, password)
8     if err != nil {
9         return false
10    }
11    mongo := db.MongoDatabaseConnector{}
12    defer mongo.Close()
13    if !mongo.Connect() {
14        return false
15    }
16    longname, err := GetLongName(username, password)
17    if err != nil {
18        return false
19    }
20    if !mongo.DoesTeacherExistsByShort(username) {
21        mongo.CreateTeacher(db.Teacher{
22            UUID:          uuid.NewString(),
23            Short:         username,
24            Longname:      longname,
25            SuperUser:     false,
26            AV:            false,
27            Administration: false,
28            PEK:           false,
29        })
30    }
31    return true
32 }
```

Auflistung 7.45: LDAP-Schnittstelle: Authentifizierungsvorgang [20]

Das *Backend* verbindet sich mit dem *TGM*-eigene *LDAP-Server*. Daraufhin werden die Nutzerdaten abgeglichen, um den Nutzer anzumelden. Schlägt diese Aktion fehl, sind die Anmeldedaten falsch. Sind die Informationen richtig wird überprüft ob es der erste Login der Lehrkraft ist. Ist dies der Fall, wird eine neue Lehrkraft in der systemeigenen Datenbank angelegt. Hierfür wird der ganze Name der Lehrkraft benötigt. Dieser wird über folgende Methode ausgelesen:

```

1  func GetLongName(username, password string) (string, error) {
2      cred := username + "@tgm.ac.at"
3      l, err := ldap.Dial("tcp", fmt.Sprintf("%s:%d", URL, Port))
4      if err != nil {
5          return "", err
6      }
7      err = l.Bind(cred, password)
8      if err != nil {
9          return "", err
10     }
11     search := ldap.NewSearchRequest("DC=tgm,DC=ac,DC=at",
12         ldap.ScopeWholeSubtree,
13         ldap.NeverDerefAliases,
14         0,
15         0,
16         false,
17         fmt.Sprintf("(&(mailNickname=%s)", username),
18         []string{"dn"},
19         nil,
20     )
21     res, err := l.Search(search)
22     if err != nil {
23         return "", err
24     }
25     if len(res.Entries) != 1 {
26         return "", fmt.Errorf("user does not exist or too many entries returned")
27     }
28     userdn := res.Entries[0]
29     cn := strings.Split(userdn.DN, ",") [0]
30     name := strings.Split(cn, "=")[1]
31     return name, nil
32 }
```

Auflistung 7.46: LDAP-Schnittstelle: Namensabfrage [20]

Die Abfragemethode ist ähnlich der Authentifizierungsmethode aufgebaut. Beide beginnen mit der Anmeldung beim Dienst, wobei dieser in diesem Fall gelingen muss. Daraufhin wird eine Suche nach dem eigenen Anmeldenamen gestartet, welche einen Pfad zurückliefert [20]. Dieser Pfad kann zerlegt werden, um auf den eigentlichen Namen zugreifen zu können.

7.2.6 Untis

Die *Untis API* wird verwendet, um auf den Stundenplan des *TGMs* zugreifen zu können [57]. Um diese effizient zu nutzen, wurde ein *Client*-Format implementiert. Diese speichert automatisch alle aktiven Sitzungen und ermöglicht ein effektives An- und Abmelden. Als Basis wurde eine *Client* Datenstruktur erstellt, welche in einer Liste gespeichert wird, wodurch die *Clients* dem jeweiligen Nutzer zuordenbar sind.

```

1 const URL = "https://neillo.webuntis.com/WebUntis/jsonrpc.do?school=tgm"
2 var activeClients map[string]Client
3 type Client struct {
4     Username string
5     Password string
6     SessionID string
7     PersonType int
8     PersonID int
9     Closed bool
10    Authenticated bool
11 }

```

Auflistung 7.47: Grundstruktur des *Untis Clients* Teil 1

Diese *Clients* werden zu Beginn der Sitzung erstellt. Sie werden automatisch in die Liste der aktiven *Clients* eingetragen. Ist eine Abfrage von *Untis*-Daten nötig, so kann der *Client* wieder aufgerufen werden, indem der *Code* ihn über die Liste auswählt. Bei der Abmeldung wird der aktive *Client* eines Nutzers wieder abgemeldet und aus der Liste ausgetragen. Diese Vorgänge werden durch folgenden *Code* implementiert:

```

1 func CreateClient(username, password string) *Client {
2     client := Client{
3         Username:     username,
4         Password:    password,
5         SessionID:   "",
6         PersonType:  -1,
7         PersonID:    -1,
8         Closed:      false,
9         Authenticated: false,
10    }
11    activeClients[username] = client
12    return &client
13 }
14 func GetClient(username string) *Client {
15     client := activeClients[username]
16     return &client
17 }
18 func (client Client) DeleteClient() {
19     delete(activeClients, client.Username)
20 }

```

Auflistung 7.48: Grundstruktur des *Untis Clients* Teil 2

Diese Struktur wird genutzt, um Sitzungen bei der *Untis API* starten zu können. Hierfür werden *HTTP POST*-Anfragen gesendet [57]. Diese Anfragen haben jeweils einen Methodennamen und Parameter. Um

Anfragen einfach absenden zu können, wurde folgende Funktion implementiert, welche den Methoden-
namen und die Parameter richtig umwandelt und die Anfrage abschickt. Die Antwort darauf wird dabei
automatisch zurückgegeben.

```
1 func (client Client) sendRequest(method string, params map[string]interface{}) {
2     ← (*http.Response, int, error) {
3         id := rand.Intn(math.MaxInt64)
4         body, _ := json.Marshal(map[string]interface{}{
5             "id":      id,
6             "method":   method,
7             "params":  params,
8             "jsonrpc": "2.0",
9         })
10        req, err := http.NewRequest("POST", URL, bytes.NewBuffer(body))
11        if err != nil {
12            return nil, -1, err
13        }
14        req.Header.Set("JSESSIONID", client.SessionID)
15        reqClient := http.Client{}
16        resp, err := reqClient.Do(req)
17        return resp, id, err
}
```

Auflistung 7.49: Hilffunktion für Anfragen an die *Untis* Schnittstelle [57]

Die folgenden Methoden werden unterstützt, um einfache Abfragen im beschriebenen *Client*-Format
durchführen zu können:

Tabelle 7.2: Weitere Methoden der *Untis API* mit den entsprechenden Endpoints und einer Beschreibung [57]

Funktion	Endpunkt Units	Beschreibung
Authenticate()	authenticate	Authentifiziert den Client und startet die Session
GetTimeTableOfTeacher()	getTimetable	Liefert den Stundenplan des angemeldeten Lehrers
GetTimetableOfClass()	getTimetable	Liefert den Stundenplan einer Klasse zurück
GetTimetableOfSpecificTeacher()	getTimetable	Liefert den Stundenplan eines bestimmten Lehrers
ResolveTeachers()	getTeachers	Ordnet IDs die entsprechenden Lehrkräfte zu
ResolveTeacherID()	getTeachers	Ordnet einer Lehrkraft die entsprechende ID zu
ResolveRooms()	getRooms	Ordnet IDs die entsprechenden Räume zu
ResolveClasses()	getKlassen	Ordnet IDs die entsprechenden Klassen zu
ResolveClassID()	getKlassen	Ordnet einer Klasse die entsprechende ID zu
Close()	logout	Beendet die Session

7.2.7 Dateierstellung

Die Dateierstellung funktioniert über die Bibliothek „*maroto*“ für das Erstellen von *PDFs* und über die Bibliothek „*excelize*“ für das Bearbeiten der *Excel-Templates* [7] [33] [14]. Des Weiteren wird auch „*pdfcpu*“ für das Zusammenfügen von *PDFs* verwendet [41]. Das folgende Beispiel zeigt die Erstellung eines einfachen *PDFs* über „*maroto*“:

```
1 m := pdf.NewMaroto(consts.Portrait, consts.A4)
2 m.SetPageMargins(10, 15, 10)
3 m.SetDefaultFontFamily(consts.Helvetica)
4 m.Line(3.0)
5 m.Row(50, func() {
6     m.Col(12, func() {
7         m.QrCode("https://youtu.be/dQw4w9WgXcQ", props.Rect{
8             Percent: 100,
9             Center: true,
10            })
11        })
12    })
13 m.Row(20, func() {
14     m.Col(12, func() {
15         m.Text("YouTube-Video", props.Rect{
16             Align: consts.Center,
17             Style: consts.Bold,
18            })
19        })
20    })
21 m.Line(3.0)
```

Auflistung 7.50: *Maroto* Beispiel zum Erstellen eines *PDFs* [7]

Dieses *Code*-Beispiel würde ein *PDF* erstellen, welches einen großen zentrierten *QR-Code* zeigt und darunter den Text „*YouTube-Video*“ [33]. Diese zwei Elemente werden durch eine horizontale Linie oben und unten begleitet.

7.2.8 Token Management

Das Token Management bietet Sicherheit, da es den Zugang zu der REST-Schnittstelle beschränkt [28] [53]. Zugriff auf die meisten Endpunkte ist nur noch durch eine Verifizierung mit einem bei der Anmeldung erhaltenen Token möglich. Die Tokens werden durch *Responses* an den *Client* geschickt. Das Token System basiert auf der Bibliothek „jwt-go“ und enthält folgende Methoden:

Tabelle 7.3: Implementierte Methoden im Token Management

Methodenname	Beschreibung	Parameter	Rückgabe
InitTokenManager()	Startet den Token Manager	—	—
CreateToken()	Erstellt ein Tokenpaar	username	Token
SaveToken()	Sichert ein Token in der lokalen Map	username, Token	—
ExtractToken()	Extrahiert das Token aus dem Request	Request	string
VerifyToken()	Verifiziert den Ursprung des angegebene Token	Request	jwt.Token
TokenValid()	Überprüft ob das angegebene Token valide ist	Request	bool
ExtractTokenMeta()	Extrahiert die Meta Informationen aus dem Token	Request	Token
readAccessSecret()	Liest das Access Secret aus oder generiert es	—	—
readRefreshSecret()	Liest das Refresh Secret aus oder generiert es	—	—
ttlCheck()	Asynchroner Thread zur Entfernung ungültiger Token	—	—

Das folgende Beispiel nutzt die in der Tabelle 7.3 beschriebenen Methoden, um jene generelle Beschränkung der Anfragen durchzusetzen, die jegliche Anfrage ohne Token direkt ignoriert:

```

1 func AuthWall() gin.HandlerFunc {
2     return func(con *gin.Context) {
3         ok, err := TokenValid(con.Request)
4         if !ok && err != nil {
5             con.JSON(http.StatusUnauthorized, Error{err.Error()})
6             con.Abort()
7             return
8         }
9         con.Next()
10    }
11 }
```

Auflistung 7.51: Token System Beispiel [53]

Die *AuthWall* fungiert wie eine Art Mauer zwischen der *URL* zum *Endpoint* und dem eigentlichen *Endpoint* und lässt ausschließlich Anfragen durch, welche ein valides Token besitzen [53]. Dadurch werden die *Endpoints* nicht durch unautorisierte Anfragen angesprochen und somit ist die Ressourcenlast niedriger.

7.2.9 REST-Schnittstelle

Die REST-Schnittstelle basiert auf *Endpoints*. Diese werden von der Bibliothek „gin-gonic“ verwaltet und nach dem Start zur Verfügung gestellt [18]. Um einen *Endpoint* mit „gin-gonic“ hinzuzufügen wird wie folgt vorgegangen:

```

1  router := gin.Default()
2  gin.SetMode(getMode())
3
4  api := router.Group("/api")
5  {
6      api.POST("/login", Login)
7      api.POST("/logout", Logout)
8      api.GET("/getAllApplications", GetAllApplications)
9      api.GET("/getNews", GetNews)
10 }
11 log.Fatal(router.Run(": " + strconv.Itoa(Port)))

```

Auflistung 7.52: REST router - Endpoints zu gin-gonic hinzufügen

7.2.9.1 Endpoints

Eine Liste aller implementierten *Endpoints* ist in Tabelle 6.1 und in Tabelle 6.2 ersichtlich. Ein Beispiel für einen implementierten *Endpoint* sieht wie folgt aus:

```

1  func Login(con *gin.Context) {
2      u := User{}
3      if err := con.ShouldBindJSON(&u); err != nil {
4          con.JSON(http.StatusUnprocessableEntity, Error{"invalid request structure
5              ↪ provided"})
6          return
7      }
8      if !ldap.AuthenticateUserCredentials(u.Username, u.Password) {
9          con.JSON(http.StatusUnauthorized, Error{"this credentials do not resolve
10             ↪ into an authorized login"})
11         return
12     }
13     token, err := CreateToken(u.Username)
14     if err != nil {
15         con.JSON(http.StatusUnprocessableEntity, Error{err.Error()})
16         return
17     }
18     SaveToken(u.Username, token)
19     untilis.CreateClient(u.Username, u.Password)
20     out := TokenPair{
21         AccessToken: token.AccessToken,
22         RefreshToken: token.RefreshToken,
23     }
24     con.JSON(http.StatusOK, out)
25 }

```

Auflistung 7.53: Implementierung Login-Endpoint

Zuerst werden Nutzerinformationen aus der Anfrage extrahiert. Diese Informationen werden über die LDAP-Schnittstelle verifiziert. Sollte dies erfolgreich sein, wird ein Tokenpaar erstellt und zurückgesendet. Bei einem Fehler bei einem vorherigen Schritt wird eine Fehlermeldung gesendet.

7.2.9.2 Swagger

„Swagger“ ist eine Dokumentationsmöglichkeit von REST-Schnittstellen [51]. Die einzelnen *Endpoints* wurden dokumentiert. Auf Basis dessen kann *Swagger* automatisch eine komplette API-Dokumentation erstellen. Diese ist als eigener *Endpoint* der REST-Schnittstelle gestaltet und abrufbar. Damit die automatisch generierten Dateien von der REST-Schnittstelle ausgegeben werden können, müssen sie wie folgt importiert werden:

```

1 import(
2     // import to make swagger docs accessible
3     _ "github.com/refundable-tgm/huginn/docs"
4     ginSwagger "github.com/swaggo/gin-swagger"    // gin swagger middleware
5     "github.com/swaggo/gin-swagger/swaggerFiles" // swagger files
6 )

```

Auflistung 7.54: REST-Schnittstelle: *Swagger* Imports [5]

Um die importierte Dokumentation zu veröffentlichen, wird eine Route zu dem entsprechenden Daten gelegt. Es wurde ebenso eine Weiterleitung vom Wurzelpfad aus eingerichtet, sodass, wenn kein *Endpoint* aufgerufen wird, man die Dokumentation angezeigt bekommt.

```

1 router.GET("/swagger/*any", ginSwagger.WrapHandler(swaggerFiles.Handler))
2 router.GET("/", func(context *gin.Context) {
3     context.Redirect(http.StatusMovedPermanently, "swagger/index.html")
4 })

```

Auflistung 7.55: REST-Schnittstelle: *Swagger* Routen

7.2.9.3 Hauptmethode

Der REST-Service wird über die *Main*-Methode im „package main“ gestartet. Diese Methode ist jener Einstiegspunkt, welcher in einer Go Applikation immer gestartet wird. Die *Main*-Funktion dieses Systems Falle ist in Auflistung 7.56 dargestellt.

```

1 package main
2
3 import (
4     "github.com/refundable-tgm/huginn/rest"
5 )
6
7 // main function starting the rest service
8 func main() {
9     rest.StartService()
10 }

```

Auflistung 7.56: Hauptmethode des *Backends*

7.3 Datenschnittstelle und Webseitenlogik

7.3.1 Webseitenlogik

Wie in Unterabschnitt 6.3.1 beschrieben, wird die Webseitenlogik größtenteils von einem *Manager* übernommen.

7.3.1.1 Navigation

Die Unterseiten werden in den *Manager* geladen und je nach Gebrauch angezeigt. Dafür gibt es eine Variable im *Manager*, welche dafür sorgt, dass nur eine Komponente auf einmal angezeigt werden kann. Die Komponente wird gewechselt, indem die folgende Funktion aufgerufen wird:

```
1   changeComponent(  
2     component,  
3     back = true,  
4     application = null,  
5     escortsdata = null  
6   ) {  
7     switch (component) {  
8       case "Login":  
9         this.change("Login", back);  
10        this.deleteCookie();  
11        break;  
12  
13       case "Index":  
14         this.change("Index", back);  
15        break;  
16  
17       case "ApplicationView":  
18         this.loadApplication(application);  
19         this.change("ApplicationView", back);  
20        break;  
21  
22       case "Escorts":  
23         this.loadEscortsData(escortsdata);  
24         this.change("Escorts", back, false);  
25        break;  
26  
27       // [Weitere Unterseiten]  
28     }  
29 }
```

Auflistung 7.57: Funktion zum Ändern der angezeigten Komponente

Diese Funktion besteht hauptsächlich aus einer Verzweigung, die entscheidet, welche Seite angezeigt werden soll. Des Weiteren gibt es bei manchen Komponenten den Zusatz einer *load*-Funktion, welche Daten für die jeweilige Komponente vorbereitet bzw. berechnet.

Die referenzierte *change*-Funktion übernimmt das tatsächliche Ändern der Variable:

```

1  change(page, back = true, cookie = true) {
2      this.currentComponent = page;      // Es wird die angezeigte Seite verändert
3      window.scrollTo(0, 0);          // Es wird zum Anfang der Seite gegangen
4      if (back) {
5          if (window.history.state !== page) {
6              window.history.pushState(page, null);    // Es wird die übergebene
7                  → Seite in die History des Browsers geschrieben
8          }
9      if (cookie) {
10         this.setCookie(page);        // Es wird der Cookie gesetzt
11     }

```

Auflistung 7.58: Funktion für das Management der *Cookies* und *History* des *Browsers*

Hier werden nicht nur die angezeigte Seite verändert, sondern auch je nach Parameter die *Cookies* sowie die *History* des *Browsers* gesetzt.

Die beschriebenen Funktionen müssen von den Komponenten aufgerufen werden können. Deswegen hört der *Manager* auf Signale der Unterseiten, falls diese die Seite geändert werden soll. Dies wird mittels *VueJS* erledigt, da das *Framework* hierfür bereits Funktionen implementiert hat:

```

1  v-on:change-component="changeComponent"
2  // Es wird auf das Signal change-component gehört und die Funktion
3  → changeComponent aufgerufen

```

Auflistung 7.59: Event Handling mittels *v-on*

Die *Codezeile* wird beim Aufruf der Unterseite im *Manager* hinzugefügt.

Die beschriebenen Signale werden von den Unterseiten mittels folgendem *Code* bzw. folgender Funktion gesendet:

```

1  changeComponent(component, back, application, escortsdata) {
2      this.$emit("change-component", component, back, application, escortsdata);
3

```

Auflistung 7.60: Ein Signal an die übergeordnete Komponente senden

Hier wird durch *this.\$emit()* ein Befehl an die übergeordnete Seite gesendet.

7.3.1.2 Komponenten

Die Komponenten der Unterseiten werden, wie im vorherigen Kapitel beschrieben, in den *Manager* eingebunden. Dies wird mit dem Importieren, dem Laden und der Implementierung der Komponente umgesetzt. Das Importieren erfolgt mit folgendem *Code*:

```
1  <script>
2    import ApplicationView from "@/components/ApplicationView.vue";
3    // [Weitere Imports]
4
5    export default {
6      components: {
7        ApplicationView
8        // [Weitere Komponenten]
9      }
10    }
11  </script>
```

Auflistung 7.61: Umsetzung des Importieren und Laden einer Komponente im *Manager*

In dem *Code* werden von "@/components/" die einzelnen Komponenten geladen. Aufgrund der Verwendung von '@' wird das Angeben des vollen Pfades nicht mehr benötigt, da dies auf den relativen Pfade der Softwarestruktur zeigt.

Die Komponente muss noch im *Manager* angezeigt werden. Dies wird im *Template* des *Managers* mit folgendem *Code* umgesetzt:

```

1  <template>
2      <div class="home">
3          <Login
4              v-if="currentComponent == 'Login'"
5              v-on:change-component="changeComponent"
6              v-bind:url="url"
7              v-bind:token="token"
8              v-bind:forward="forward"
9              v-on:requestAnswer="useCookie"
10             v-bind:cookieset="cookies"
11             v-on:login="login"
12         />
13         <Index
14             v-if="currentComponent == 'Index'"
15             v-on:change-component="changeComponent"
16             v-bind:url="url"
17             v-bind:token="token"
18             v-bind:admin="admin"
19             v-bind:user="user"
20         />
21         <ApplicationView
22             v-if="currentComponent == 'ApplicationView'"
23             v-on:change-component="changeComponent"
24             v-bind:url="url"
25             v-bind:token="token"
26             v-bind:appid="appid"
27             v-bind:user="user"
28         />
29         <!-- Weitere Komponenten -->
30     </div>
31 </template>
```

Auflistung 7.62: Anzeigen der Komponenten im *Manager*

Im *Code* wird nicht nur die Komponente angezeigt, sondern auch Daten an die Komponenten gebunden. *v-if*-Attribute sorgen dafür, dass das Rendern nur unter der angegebenen Bedingung erfolgt. Die Komponente wird nur angezeigt, wenn die Variable *currentComponent* den Namen einer Komponente annimmt. *v-bind*-Attribute binden Daten von der übergeordneten Komponente in die eingebundene Komponente ein. Bei dem Beispiel *v-bind:cookieset="cookies"* wird an die *Login*-Komponente ein Attribut *cookieset* mit dem Wert von der Variable *cookies* mitgegeben. Diese Attribute können mit folgendem *Code* aufgefangen und in diesen eingebundenen Komponenten verwendet werden:

```
1  props: ["url", "forward", "cookieset"]
```

Auflistung 7.63: Auffangen übergebener Variablen in einer untergeordneten Komponente

In dem *Code* werden die übergebenen Variablen, welche in der übergeordneten Komponente definiert worden sind, aufgefangen. Diese können als normale Variablen verwendet werden.

7.3.1.3 Seite nicht gefunden

Wie in Unterabschnitt 6.3.1.1 beschrieben wird, wird der Benutzer bei Eingabe eines falschen *Links* auf eine Seite weitergeleitet. Dazu wird der *Vue-Router* so eingestellt, dass jeder *Link*, außer dem Hauptlink zur eigentlichen Webseite, auf die beschriebene Seite weitergeleitet wird. Die Routen des Routers sind wie folgt aufgebaut:

```

1 // Imports für die Komponenten
2 import Vue from "vue";
3 import VueRouter from "vue-router";
4 import Home from "../views/Home.vue";
5 import PageNotFound from "../components/PageNotFound.vue";
6
7 Vue.use(VueRouter);           // Sagt dem Projekt, dass es den Router verwenden soll
8
9 const routes = [
10   {
11     path: "/",                // Der Pfad, der von dem Benutzer eingegeben wird
12     name: "Home",             // Der Name des Pfades
13     component: Home,          // Die Komponente, welche angezeigt werden soll
14     props: { pathing: "base" } // Variablen, welche an die Komponente gesendet
15     // werden sollen
16   },
17   {
18     path: "/viewer",
19     name: "Viewer",
20     component: Home,
21     props: route => ({ query: route.query.uuid }) // Die ID des Antrags, welcher
22     // betrachtet werden möchte
23   },
24   {
25     path: "*",
26     name: "NotFound",
27     component: PageNotFound
28   }
29 ];
30 const router = new VueRouter({
31   mode: "history",
32   routes
33 });
34 export default router;

```

Auflistung 7.64: Router, welcher die Pfade der Software verwaltet

Wird ein *Link* eingegeben, welcher nicht "/" oder "/viewer" entspricht, so wird die *PageNotFound*-Komponente geladen und somit dem Benutzer mitteilt, dass der eingegebene *Link* nicht funktioniert.

7.3.1.4 Antrag-Viewer

Der *Antrag-Viewer* wird über den *Link* "/viewer" aufgerufen. Dies führt dazu, dass der Benutzer auf eine Seite weitergeleitet wird, auf der er die *ID* eines Antrags eingeben kann.

Abbildung 7.25: Die *Antrag-Viewer* Seite der Webseite

Sollte der Benutzer nicht angemeldet sein, so wird er zuerst auf die *Login*-Seite weitergeleitet, bevor er die *ID* des Antrags eingeben kann. Sollte der eingegebene *Link* bereits den Parameter *uuid* besitzen, wird der Benutzer auf die Antragsansichtseite weitergeleitet, sofern dieser angemeldet ist. Ansonsten wird der Benutzer zuerst auf die *Login*-Seite weitergeleitet.

```
1  {
2      // Die Route des Antrag-Viewers
3      path: "/viewer",
4      name: "Viewer",
5      component: Home,
6      props: route => ({ query: route.query.uuid }) // Der Parameter in der URL
7  },
```

Auflistung 7.65: Route für den *Antrag-Viewer* mit Parametern

Es wird die *Home*-Komponente (der *Manager*) aufgerufen, da dieser das Verarbeiten der *URL* und das Weiterleiten auf die *Login*- und *Antragsansicht* übernimmt.

7.3.1.5 Features

Die *Features* in Unterunterabschnitt 6.3.1.6 wurden folgendermaßen implementiert.

Zuletzt besuchte Seite der Webseite

Beim Laden der Webseite wird mithilfe von *Cookies* auf die zuletzt besuchte Seite navigiert. Dazu wird bei jeder Navigation, wie in Unterunterabschnitt 7.3.1.1 beschrieben, der *Cookie* gesetzt. Dieser wird beim Laden der Webseite ausgelesen und es wird, falls der *Cookie* gesetzt ist, auf die gespeicherte Seite navigiert.

```

1   if (this.checkCookie()) {           // Abfrage, ob ein Cookie vorhanden ist
2       this.useCookie(true);         // Sofern ein Cookie vorhanden ist, hat der
3       ↵ Benutzer bereits die Cookies für diese Webseite akzeptiert
4       var c = this.getCookie();     // Die Informationen des Cookies werden
5       ↵ ausgelesen
6       if (c == this.generateState(window.history.state)) {    // Falls die
7           ↵ zuletzt besuchte Seite die gleiche ist, wie die aktuelle Seite, so wird
8           ↵ kein neuer Eintrag in der History generiert
9           this.changeComponent(c, false);
10      } else {
11          this.changeComponent(c);
12      }
13  } else {
14      // Falls kein Cookie gesetzt ist, wird der Benutzer auf die Login-Seite
15      ↵ weitergeleitet
16      this.changeComponent("Login");
17  }

```

Auflistung 7.66: Navigation auf die zuletzt besuchte Seite

Browser-Pfeile

Die *Browser-Pfeile* navigieren durch die *History* des *Browsers*. Da die Webseite ein nur auf einer einzelnen Seite geladen ist, muss das Schreiben in die *History* selbst übernommen werden. Dies wird durch den folgenden *Code* umgesetzt:

```

1 window.addEventListener("popstate", e => {           // Es wird ein Listener auf
2     ↵ die Webseite gesetzt
3     if (this.generateState(e.state) === "Escorts") {    // Dies ist eine extra
4         ↵ Abfrage, welche verhindert auf die Escorts-Seite zu gelangen.
5         this.changeComponent("School", false);
6     } else {
7         this.changeComponent(this.generateState(e.state), false);    // Es wird auf
8         ↵ die Unterseite navigiert.
9     }
10 });

```

Auflistung 7.67: Unterstützung der *Browser-Pfeile*

Das Hinzufügen der einzelnen *Histories* für die jeweiligen Unterseiten wird in Unterunterabschnitt 7.3.1.1 beschrieben.

7.3.2 Daten laden & Befehle senden

Das Laden sowie das senden von Befehle wird mittels *Axios* umgesetzt. Es gibt vier Arten der Anfragen per *Axios*, welche verwendet werden: *Get* (Anfrage von Daten), *put* (Aktualisieren von Daten), *post* (Erstellen von Daten) und *delete* (Löschen von Daten). Mit folgender Struktur können Daten von dem *Backend*, bzw. der *REST*-Schnittstelle geladen werden:

```
1  axios.get(this.url + "/getNews", {
2      headers: {
3          Authorization: "Basic " + this.token
4      }
5  }).then(response => {
6      switch(response.status) {
7          case 200:      // Die Anfrage hat funktioniert
8              this.news = this.cutNews(response.data); // Hier werden die
9                  ↪ Neuigkeiten aus der Anfrage in die Startseite geladen
10             break;
11         case 401:      // Die Anfrage hat einen ''Unauthorized'' Fehler geworfen
12             // Hier wird auf den Fehlercode 401 reagiert
13             break;
14         default:
15             // Hier wird dem Benutzer angezeigt, dass ein Fehler aufgetreten ist
16             break;
17     }
18 });

});
```

Auflistung 7.68: Laden der Neuigkeiten auf der Startseite

Unter der Bedingung, dass die erste Anfrage den *Fehlercode 401* zurückgibt, wird eine Anfrage an das *Backend* gesendet, um die Sitzung des Nutzers zu aktualisieren. Dies wird mit folgendem *Code* umgesetzt:

```
1     axios.post(this.url + "/login/refresh", {
2         headers: {
3             Authorization: "Basic " + this.refresh_token
4         }
5     }).then(resp => {
6         switch (resp.status) {
7             case 201:    // Die Anfrage hat funktioniert
8                 this.$emit("updateToken", resp.data.access_token,
9                         ↳ resp.data.refresh_token);
10                // Die Anfrage wird erneut durchgeführt
11                break;
12            default:
13                this.$emit("logout"); // Der Benutzer wird vom System abgemeldet
14                break;
15            }
16        });
17    );
```

Auflistung 7.69: Anfrage um Sitzung des Nutzers zu aktualisieren

Bei erfolgreicher Erneuerung der Sitzung, wird die anfangs durchgeführte Anfrage erneut gesendet. Diese reagiert, bis auf die fehlende Behandlung des *Fehlercodes 401*, ident wie die originale Anfrage.

7.3.3 Anmelden

Der Benutzer kann sich auf einer Anmeldeseite anmelden, um das System verwenden zu können. Dazu werden die Anmelddaten des Benutzers an das *Backend* gesendet. Um sich anmelden zu können, müssen die *Cookies* akzeptiert werden. Sollten sie nicht akzeptiert werden, ist es nicht möglich den Dienst zu verwenden. Sobald der Benutzer erfolgreich angemeldet ist, wird dieser auf die Hauptseite weitergeleitet.

```

1  axios.post(this.url + "/login", {    // Sendet eine POST-Anfrage an das Backend
2      username: this.email,        // Die eingegebene E-Mail Adresse
3      password: this.password // Das eingegebene Passwort
4  }).then(response => {
5      var data = response.data
6      // Es werden die Daten aus der Anfrage an den Manager gesendet, damit eine
7      // valide Sitzung erstellt wird.
8  })

```

Auflistung 7.70: POST-Anfrage für das Anmelden

7.3.3.1 Spezielle Situationen

In besonderen Fällen, wie in Unterunterabschnitt 7.3.1.4 beschrieben, wird der Benutzer nicht auf die Hauptseite weitergeleitet, sondern auf den *Antrag-Viewer* oder auf die *Antrag-Ansicht*.

```

1  switch (this.forward.name) {    // Es wird entschieden, auf welche Unterseite
2      // navigiert werden soll
3      case "ApplicationSearch":    // Es wird auf den Antrag-Viewer navigiert
4          this.$emit("change-component", this.forward.name);
5          break;
6      case "ApplicationView":      // Es wird auf die Antrag Ansicht navigiert
7          this.$emit(
8              "change-component",
9              this.forward.name,
10             true,
11             this.forward.id
12         );
13         break;
14     default:      // Es wird auf die Hauptseite navigiert
15         this.$emit("change-component", "Index");
16         break;
17     }

```

Auflistung 7.71: Weiterleitung nach erfolgreichem Anmelden

7.3.3.2 Passwort vergessen

Da Refundable die Anmelddaten von *LDAP* verwendet, wird der Benutzer auf *OWA* weitergeleitet, wenn auf *Passwort vergessen* gedrückt wird.

7.3.4 Abmelden

Wird auf den *Abmelden*-Knopf gedrückt, wird ein Anfrage mit dem derzeit aktiven *Token* an das *Backend* gesendet, um den *Authentifizierungstoken* zu terminieren. Dies bedeutet, dass der Benutzer keinen validen *Token* mehr besitzt und auf die *Login*-Seite weitergeleitet wird.

```
1 axios.post(this.url + "/logout", {      // Sendet eine POST-Anfrage an das Backend
2     headers: { Authorization: this.token }    // Der Token des Benutzers
3 }
4 .then(response => {
5     if (response.status === 200) {          // Falls die Anfrage erfolgreich war
6         this.token = "";
7         this.refresh_token = "";
8         this.deleteCookie();
9         this.changeComponent("Login");      // Es werden die Tokens und Cookies
10        ↪ gelöscht und der Benutzer wird auf die Login-Seite weitergeleitet.
11    }
12});
```

Auflistung 7.72: Gesamte Anfrage für das Abmelden

7.3.5 Antrag erstellen

Ein Antrag kann von jedem Lehrer, der im System angemeldet ist, erstellt werden. Dazu wird auf die *Antrag erstellen*-Unterseite navigiert, bei der der Benutzer aussuchen kann, ob er eine Schulveranstaltung, Fortbildung oder einen anderen Antrag erstellen möchte. Auf diesen Unterseiten werden die notwendigen Informationen vom Benutzer in die Eingabefelder eingegeben. Diese Informationen werden in ein JSON-Objekt gegeben und an das *Backend* gesendet.

In den folgenden *Codebeispielen* werden oft die Befehle *returnString*, *returnBoolean* und *returnValue* verwendet. Diese dienen dazu die Datentypen auf die des *Backends* anzupassen.

7.3.5.1 Schulveranstaltung

Allgemeine Informationen

Im ersten Schritt, werden allgemeine Informationen der Schulveranstaltung eingegeben.

The screenshot shows a form for entering general information about a school event. The fields and their descriptions are:

- Bezeichnung: Text input field for the name of the school event.
- Startdatum: Date input field for the start date of the school event.
- Startzeit: Time input field for the start time of the school event.
- Enddatum: Date input field for the end date of the school event.
- Endzeit: Time input field for the end time of the school event.
- Startadresse: Text input field for the starting address of the school event, with a placeholder "Wexstraße 19-23, 1200 Wien, Österreich".
- Zieladresse: Text input field for the target address of the school event, with a placeholder "Straße & Nr., Postleitzahl & Ort, Land".
- Begleitpersonen: Text input field for listing accompanying persons, with a placeholder "Einträge durch Leerzeichen trennen".
- Jahrgänge: Text input field for listing grades, with a placeholder "Einträge durch Leerzeichen trennen".
- Anzahl Schüler: Number input field for the number of students, with a placeholder "0".
- Anzahl Schülerinnen: Number input field for the number of female students, with a placeholder "0".
- Anmerkungen: Text input field for additional remarks.

A blue button labeled "weiter" (next) is located at the bottom right of the form area.

Abbildung 7.26: Eingabefelder für allgemeine Informationen einer Schulveranstaltung

Informationen zu Begleitlehrern

Nach der Eingabe aller allgemeinen Informationen wird der Benutzer nach dem Drücken des *weiter-*-Knopfes auf eine weitere Unterseite weitergeleitet. Auf dieser können Informationen zu den im vorherigen Schritt angegebenen Begleitlehrern angegeben werden.

Dominik Dolezal

Startdatum	<input type="text" value="Dienstag, 20. April 2021"/> ✓
Geben Sie das Startdatum der Schulveranstaltung ein.	
Startzeit	<input type="text" value="08:00"/> ✓
Geben Sie die Startzeit der Schulveranstaltung ein.	
Enddatum	<input type="text" value="Dienstag, 27. April 2021"/> ✓
Geben Sie das Enddatum der Schulveranstaltung ein.	
Endzeit	<input type="text" value="18:00"/> ✓
Geben Sie die Endzeit der Schulveranstaltung ein.	
Gruppe	<input type="radio"/> L1 <input type="radio"/> L2 <input type="radio"/> L3
Wählen Sie die Gruppe der Begleitperson	
Startadresse	<input type="text" value="Wexstraße 19-23, 1200 Wien, Österreich"/>
Geben Sie Ihre Startadresse für diese Schulveranstaltung ein.	
Treffpunkt	<input type="text" value="Wexstraße 19-23, 1200 Wien, Österreich"/>
Geben Sie den Treffpunkt für diese Schulveranstaltung ein.	
Akademischer Titel	<input type="text"/>
Geben Sie den akademischen Titel an.	
Akademischer Grad	<input type="text"/>
Geben Sie den Akademischen Grad an.	
Personalnummer	<input type="text"/>
Geben Sie die Personalnummer der Begleitperson ein.	

Abbildung 7.27: Eingabefelder für Informationen von Begleitlehrern (Teil 1)

Personalnummer	<input type="text"/>
<p>Geben Sie die Personalnummer der Begleitperson ein.</p>	
Fortbewegungsart	<input type="radio"/> Amtl. Businesskarte 2. Kl. <input type="radio"/> Beförderungszuschuss <input type="radio"/> Bahn 2. Kl. - gegen Beleg <input type="radio"/> Schlafwagen <input type="radio"/> MitfahrerInnen <input type="radio"/> Flug <input type="radio"/> Billigflug <input type="radio"/> Bus - gegen Beleg <input type="radio"/> Amtl. Businesskarte / Bahnverrechnung 1. Kl. <input type="radio"/> Eigener PKW
<p>Geben Sie die Fortbewegungsart an.</p>	
Begründung	<input type="text"/>
<p>Begründung der obigen Faktoren</p>	
Ausgangspunkt	<input checked="" type="radio"/> Dienststelle <input type="radio"/> Wohnung* <small>*nur wenn dadurch niedrigere Kosten anfallen</small>
Endpunkt	<input checked="" type="radio"/> Dienststelle <input type="radio"/> Wohnung* <small>*nur wenn dadurch niedrigere Kosten anfallen</small>
Begründung	<input type="text"/>
<p>Begründung der obigen Faktoren</p>	
Bonus-Meilen	<input type="checkbox"/> Ich bestätige, dass ich anlässlich von Dienstreisen im Rahmen personenbezogener Bonusprogramme erworbene Prämien nicht privat in Anspruch nehme <input type="checkbox"/> Für die Dienstreise verwende ich auf meinem Meilenkonto gutgeschriebene, dienstlich erworbene Meilen
<p>Akzeptieren Sie ggf. die Bedingungen für die Bonus-Meilen</p>	
Reisekosten - andere Stellen	<input checked="" type="radio"/> Nein <input type="radio"/> Ja <small>Werden Reisekosten von anderen Stellen getragen?</small>
Aufenthaltskosten - andere Stellen	<input checked="" type="radio"/> Nein <input type="radio"/> Ja <small>Werden Aufenthaltskosten von anderen Stellen getragen?</small>
Sonstige Kosten	<input type="text"/> €
<p>Geben Sie sonstige Kosten an.</p>	
Geschätzte Kosten	<input type="text"/> €
<p>Geben Sie die geschätzten Kosten an.</p>	
Businesskarte	<input checked="" type="radio"/> Nein <input type="radio"/> Hinfahrt <input type="radio"/> Rückfahrt <small>Businesskarte ausgefolgt?</small>
<input type="button" value="Einreichen"/>	

Abbildung 7.28: Eingabefelder für Informationen von Begleitlehrern (Teil 2)

Nachdem die Seite bearbeitet worden ist, kann auf den Knopf *Einreichen* gedrückt werden. Sobald der Knopf gedrückt worden ist, wird das JSON-Objekt mit allen Informationen, die eingegeben worden sind, erstellt und an das *Backend* gesendet.

```

1 var data = {           // Das JSON-Objekt mit allen Informationen
2   Name: this.returnString(this.escorts.description),      // Der Name des Antrags
3   Kind: 0,        // Der Typ des Antrags (0 = Schulveranstaltung)
4   MiscellaneousReason: this.returnString(""),           // Nicht benötigt, da es
   ↵  eine Schulveranstaltung ist
5   Progress: 1,    // Der Fortschritt ist standardmäßig auf 1 gesetzt
6   StartTime: this.setTimezone(   // Die Startzeit des Antrags
7     new Date(this.escorts.startDate + "T" + this.escorts.startTime)
8   ),
9   EndTime: this.setTimezone(   // Die Endzeit des Antrags
10    new Date(this.escorts.endDate + "T" + this.escorts.endTime)
11  ),
12  Notes: this.returnString(this.escorts.notes),       // Zusätzliche Notizen zum
   ↵  Antrags
13  StartAddress: this.returnString(this.escorts.start),   // Startadresse des
   ↵  Antrags
14  DestinationAddress: this.returnString(this.escorts.ziel), // Zieladresse des
   ↵  Antrags
15  SchoolEventDetails: {      // Allgemeine Informationen zu einer Schulveranstaltung
16    Classes: this.escorts.class,      // Klassen, die auf die Schulveranstaltung
   ↵  mitfahren
17    AmountMaleStudent: this.returnValue(      // Anzahl der männlichen Schüler
   ↵  der Schulveranstaltung
      this.escorts.count_student_male
    ),
18    AmountFemaleStudent: this.returnValue(      // Anzahl der weiblichen Schüler
   ↵  der Schulveranstaltung
      this.escorts.count_student_female
    ),
19    DurationInDays: this.returnValue(this.escorts.exkursLength),    // Die
   ↵  Länge der Schulveranstaltung in Tagen
20    Teachers: teachers    // Eine Liste aller Lehrer mit Ihren spezifischen
   ↵  Informationen
21  },
22  BusinessTripApplications: business,      // Die Reiseformulare für alle Lehrer
   ↵  beteiligt an dem Antrag
23  TravelInvoices: invoices    // Die Reiserechnungen für alle Lehrer beteiligt an
   ↵  dem Antrag
24
25  };
26
27
28 };

```

Auflistung 7.73: JSON-Objekt einer Schulveranstaltung

7.3.5.2 Fortbildung

Eine Fortbildung wird auf der Fortbildung-Unterseite erstellt. Diese sieht wie folgt aus und beinhaltet Eingabefelder für alle notwendigen Informationen der Fortbildung.

The form consists of the following fields:

- Titel:** Input field with placeholder "Geben Sie den Titel der Fortbildung ein."
- Startdatum:** Input field with placeholder "Datum auswählen" and a calendar icon.
- Startzeit:** Input field with placeholder "Zeit auswählen" and a clock icon.
- Enddatum:** Input field with placeholder "Datum auswählen" and a calendar icon.
- Endzeit:** Input field with placeholder "Zeit auswählen" and a clock icon.
- PH-Zahl:** Input field with placeholder "Geben Sie Ihre PH-Zahl ein."
- Veranstalter:** Input field with placeholder "Geben Sie den Namen des Veranstalters ein."
- Startadresse:** Input field with placeholder "Geben Sie die Startadresse ein."
- Fortbildungsadresse:** Input field with placeholder "Geben Sie die Adresse der Fortbildung ein."
- Art:** Radio button group with options: Seminar, Tagung, Lehrgang, Sonstiges.

Wählen Sie die Art der Fortbildung.
- Anmerkungen:** Input field with placeholder "Anmerkungen".

Geben Sie zusätzliche Anmerkungen an.
- Akademischer Titel:** Input field with placeholder "Geben Sie den akademischen Titel an."
- Akademischer Grad:** Input field with placeholder "Geben Sie den Akademischen Grad an."

Abbildung 7.29: Eingabefelder einer Fortbildung (Teil 1)

Personalnummer	<input type="text"/>
<small>Geben Sie die Personalnummer der Begleitperson ein.</small>	
Fortbewegungsart	<input type="radio"/> Amtl. Businesskarte 2. Kl. <input type="radio"/> Beförderungszuschuss <input type="radio"/> Bahn 2. Kl. - gegen Beleg <input type="radio"/> Schlafwagen <input type="radio"/> MitfahrerInnen <input type="radio"/> Flug <input type="radio"/> Billigflug <input type="radio"/> Bus - gegen Beleg <input type="radio"/> Amtl. Businesskarte / Bahnverrechnung 1. Kl. <input type="radio"/> Eigener PKW
<small>Geben Sie die Fortbewegungsart an.</small>	
Begründung	<input type="text"/>
<small>Begründung der obigen Faktoren</small>	
Ausgangspunkt	<input checked="" type="radio"/> Dienststelle <input type="radio"/> Wohnung* <small>*nur wenn dadurch niedrigere Kosten anfallen</small>
<small>Begründung der obigen Faktoren</small>	
Endpunkt	<input checked="" type="radio"/> Dienststelle <input type="radio"/> Wohnung* <small>*nur wenn dadurch niedrigere Kosten anfallen</small>
Begründung	<input type="text"/>
<small>Begründung der obigen Faktoren</small>	
Bonus-Meilen	<input type="checkbox"/> Ich bestätige, dass ich anlässlich von Dienstreisen im Rahmen personenbezogener Bonusprogramme erworbene Prämien nicht privat in Anspruch nehme <input type="checkbox"/> Für die Dienstreise verwende ich auf meinem Meilenkonto gutgeschriebene, dienstlich erworbene Meilen
<small>Akzeptieren Sie ggf. die Bedingungen für die Bonus-Meilen</small>	
Reisekosten - andere Stellen	<input checked="" type="radio"/> Nein <input type="radio"/> Ja <small>Werden Reisekosten von anderen Stellen getragen?</small>
<small>Werden Reisekosten von anderen Stellen getragen?</small>	
Aufenthaltskosten - andere Stellen	<input checked="" type="radio"/> Nein <input type="radio"/> Ja
<small>Werden Aufenthaltskosten von anderen Stellen getragen?</small>	
Sonstige Kosten	<input type="text"/> €
<small>Geben Sie sonstige Kosten an.</small>	
Geschätzte Kosten	<input type="text"/> €
<small>Geben Sie die geschätzten Kosten an.</small>	
Businesskarte	<input checked="" type="radio"/> Nein <input type="radio"/> Hinfahrt <input type="radio"/> Rückfahrt <small>Businesskarte ausgefolgt?</small>
Einreichen	

Abbildung 7.30: Eingabefelder einer Fortbildung (Teil 2)

Bei einer Fortbildung kann unterschieden werden, welche Art von Fortbildung erstellt werden soll. Es kann zwischen Seminar, Tagung, Lehrgang und Sonstiges unterschieden werden. Das JSON-Objekt wird nach dem Drücken des *Einreichen*-Knopfes erstellt und mit den angegebenen Daten befüllt. Das Objekt wird anschließend an das *Backend* gesendet.

Das JSON-Objekt wird, wie in Abschnitt 7.3.5.1 beschrieben, erstellt. Für eine Fortbildung gibt es nur wenige Änderungen. Statt dem *SchoolEventDetails*-Objekt wird ein *TrainingDetails*-Objekt in den Antrag hinzugefügt und die Art des Antrags wird auf 1 gesetzt:

```
1 var data = {           // Das JSON-Objekt mit allen Informationen
2   Name: this.returnString(this.escorts.description),      // Der Name des Antrags
3   Kind: 1,        // Der Typ des Antrags (1 = Fortbildung)
4   // ...
5   TrainingDetails: {
6     Kind: this.returnValue(this.selected),      // Die Art der Fortbildung
7     MiscellaneousReason: this.returnString(this.son),    // Falls "Sonstiges"
8     ↪ ausgewählt worden ist
9     PH: this.returnValue(this.phoneNumber),      // Die PH Zahl des Lehrers
10    Organizer: this.returnString(this.veran)      // Der Veranstalter der
11    ↪ Fortbildung
12  },
13  BusinessTripApplications: business,      // Das Reiseformular für den Lehrer
14  TravelInvoices: invoices      // Die Reiserechnung für den Lehrer
15};
```

Auflistung 7.74: JSON-Objekt einer Fortbildung

7.3.5.3 Sonstiges

Ein *Sonstiges*-Antrag kann auf der Unterseite *Anderer Grund* erstellt werden. Er sieht wie folgt aus und beinhaltet Eingabefelder für alle notwendigen Informationen für den Antrag.

Startdatum	<input type="button" value="Datum auswählen"/> Datum auswählen Geben Sie das Startdatum der Fortbildung ein.
Startzeit	<input type="button" value="Zeit auswählen"/> Zeit auswählen Geben Sie die Startzeit der Fortbildung ein.
Enddatum	<input type="button" value="Datum auswählen"/> Datum auswählen Geben Sie das Enddatum der Fortbildung ein.
Endzeit	<input type="button" value="Zeit auswählen"/> Zeit auswählen Geben Sie die Endzeit der Fortbildung ein.
Startadresse	<input type="text"/> Geben Sie die Startadresse ein.
Zieladresse	<input type="text"/> Geben Sie die Zieladresse des Antrags ein.
Gründe	<input type="radio"/> Pflegefreistellung <input checked="" type="radio"/> Dienstauftrag <input type="radio"/> Arzttermin <input type="radio"/> Sonstiges Wählen Sie Ihren Grund aus.
Sonstiger Grund	<input type="text"/> Geben Sie den Grund an.
Anmerkungen	<input type="text"/> Geben Sie zusätzliche Anmerkungen an.
Akademischer Titel	<input type="text"/> Geben Sie den akademischen Titel an.
Akademischer Grad	<input type="text"/> Geben Sie den Akademischen Grad an.

Abbildung 7.31: Eingabefelder für *Sonstiges*-Antrag (Teil 1)

Personalnummer

Geben Sie die Personalnummer der Begleitperson ein.

Fortbewegungsart

- Amtl. Businesskarte 2. Kl.
- Beförderungszuschuss
- Bahn 2. Kl. - gegen Beleg
- Schlafwagen
- MitfahrerInnen
- Flug
- Billigflug
- Bus - gegen Beleg
- Amtl. Businesskarte / Bahnverrechnung 1. Kl.
- Eigener PKW

Geben Sie die Fortbewegungsart an.

Begründung

Begründung der obigen Faktoren

Ausgangspunkt

Dienststelle Wohnung*

*nur wenn dadurch niedrigere Kosten anfallen

Endpunkt

Dienststelle Wohnung*

*nur wenn dadurch niedrigere Kosten anfallen

Begründung

Begründung der obigen Faktoren

Bonus-Meilen

- Ich bestätige, dass ich anlässlich von Dienstreisen im Rahmen personenbezogener Bonusprogramme erworbene Prämien nicht privat in Anspruch nehme
- Für die Dienstreise verwende ich auf meinem Meilenkonto gutgeschriebene, dienstlich erworbene Meilen

Akzeptieren Sie ggf. die Bedingungen für die Bonus-Meilen

Reisekosten - andere Stellen

Nein Ja

Werden Reisekosten von anderen Stellen getragen?

Aufenthaltskosten - andere Stellen

Nein Ja

Werden Aufenthaltskosten von anderen Stellen getragen?

Sonstige Kosten

Geben Sie sonstige Kosten an.

Geschätzte Kosten

Geben Sie die geschätzten Kosten an.

Businesskarte

Nein Hinfahrt Rückfahrt

Businesskarte ausgefolgt?

Einreichen

Abbildung 7.32: Eingabefelder für *Sonstiges*-Antrag (Teil 2)

Sonstiges-Anträge werden unterschieden in Pflegefreistellungen, Dienstaufträge, Arzttermine und Sonstiges. Das JSON-Objekt wird nach dem Drücken des *Einreichen*-Knopfes erstellt und mit den angegebenen Daten befüllt. Das Objekt wird anschließend an das *Backend* gesendet.

Das JSON-Objekt wird, wie in Abschnitt 7.3.5.1 beschrieben erstellt. Für einen *Sonstiges*-Antrag gibt es nur wenige Änderungen. Statt dem *SchoolEventDetails*-Objekt wird ein *OtherReasonDetails*-Objekt in den Antrag hinzugefügt und die Art des Antrags wird auf 6 gesetzt:

```

1  var data = {           // Das JSON-Objekt mit allen Informationen
2      Name: this.returnString(this.escorts.description),    // Der Name des Antrags
3      Kind: 6,      // Der Typ des Antrags (6 = Sonstiges Antrag)
4      // ...
5      OtherReasonDetails: {
6          Kind: this.returnValue(this.selected),      // Die Art des Sonstigen Antrags
7          MiscellaneousReason: this.returnString(this.son),   // Falls "Sonstiges"
8          ← ausgewählt worden ist
9          ServiceMandateGZ: this.returnValue(this.gz),    // Die GZ des Dienstauftrags
10         ServiceMandateTitle: this.returnString(this.title) // Titel des
11         ← Dienstauftrags
12     },
13     BusinessTripApplications: business,    // Das Reiseformular für den Lehrer
14     TravelInvoices: invoices      // Die Reiserechnung für den Lehrer
15 };

```

Auflistung 7.75: JSON-Objekt eines *Sonstiges*-Antrags

7.3.5.4 Antrag einreichen

Sobald der Benutzer auf den *Einreichen*-Knopf drückt, wird das JSON-Objekt erstellt und an das *Backend* gesendet. Dies erfolgt mit folgendem *Code*:

```

1  axios.post(this.url + "/createApplication", // Es wird eine
2      ← createApplication-Anfrage an das Backend gesendet
3      {
4          headers: {
5              Authorization: "Basic " + this.token
6          }
7      },
8      data // Der Antrag mit den eingegebenen Daten wird mit der Anfrage gesendet
9  )

```

Auflistung 7.76: Beispiel um einen Antrag im *Backend* zu erstellen

7.3.6 Antrag Ansicht

Wenn die Unterseite, die einen Antrag anzeigen soll, geladen wird, so wird der Unterseite die *ID* des Antrags mitgegeben. Sobald die Komponente geladen wird, wird mit dieser *ID* eine Anfrage an das *Backend* gesendet, um den gesamten Antrag zu laden. Die Informationen werden interpretiert und angezeigt.

7.3.6.1 Fortschrittsanzeige

Jeder Antrag besitzt einen Fortschritt, welcher graphisch durch eine *Progress-Komponente* angezeigt wird. Die Komponente erhält von der *Antrag-Ansicht* den Fortschritt des Antrags und die Art des Antrags als Nummer, um die Informationen für den Benutzer lesbar anzuzeigen.



Abbildung 7.33: Fortschrittsanzeige für einen Antrag

7.3.6.2 Dokumente

In einem Antrag werden mehrere Dokumente gespeichert. Die Dokumente können je nach Fortschritt des Antrags bearbeitet werden und werden in einer Liste angezeigt:

Dokument	Aktionen
Allgemeine Infos - Abwesenheitsformulare der Klassen	<input type="button" value="PDF öffnen"/> <input type="button" value="Bearbeitung öffnen"/>
Abgeltung für pädagogische Betreuung	<input type="button" value="PDF öffnen"/>
Begleitformular - Abwesenheitsformular	<input type="button" value="PDF öffnen"/> <input type="button" value="Bearbeitung öffnen"/>
Reiseformular	<input type="button" value="Excel herunterladen"/> <input type="button" value="PDF öffnen"/> <input type="button" value="Bearbeitung öffnen"/>
	<input type="button" value="Änderungen speichern"/> <input type="button" value="Antrag schließen"/> <input type="button" value="Antrag löschen"/>

Abbildung 7.34: Die Dokumente einer Schulveranstaltung

Folgende Dokumente werden je nach Antragsart erstellt:

Schulveranstaltung

Die Schulveranstaltung besitzt zwei Dokumente: zum Einen die allgemeinen Informationen zu der Veranstaltung und zum Anderen die Informationen zu den Begleitpersonen.

- **Allgemeine Informationen**

In diesem Dokument werden generelle Informationen über die Schulveranstaltung gespeichert. Diese können nur von dem Leiter des Antrags bearbeitet werden.

- **Informationen zu Begleitlehrern**

In diesem Dokument sieht jeder Begleitlehrer sein eigenes Formular, welches er je nach Fortschritt des Antrags bearbeiten kann. In diesem Dokument werden Informationen über die An- und Abreise sowie den Aufenthalt des Lehrers gespeichert.

Fortbildung

In einem Fortbildungsformular werden Informationen über die Fortbildung und den Lehrer gespeichert.

Sonstiges

In einem *Sonstiges*-Antrag werden Informationen über den Grund des Antrags sowie den Zeitraum gespeichert.

Reiseformular

Ein Reiseformular wird bei den meisten Arten von Anträgen erstellt. In einem Reiseformular werden Informationen über den Lehrer und über dessen An- und Abreise gespeichert.

Reiserechnung

Eine Reiserechnung wird bei den meisten Arten von Anträgen erstellt. In einer Reiserechnung werden Informationen über die Kosten des Antrags gespeichert und wie diese entstanden sind. Des Weiteren können Belege hochgeladen werden, um diese dem Antrag zu hinterlegen. Ein Beleg wird mit folgendem *Code* hochgeladen:

```
1 axios.post(this.url + "/saveBillingReceipt",
2 {
3     params: {
4         uuid: this.app.uuid, // Der Antrag wird spezifiziert, für den die
5             ← Belege hochgeladen werden
6         shortname: this.user.short // Es wird der Lehrer spezifiziert, der die
7             ← Belege hochlädt
8     }
9 },
10 {
11     headers: {
12         Authorization: "Basic " + this.token
13     }
14 },
15 belege // Die Belege, welche hochgeladen werden
)
```

Auflistung 7.77: Beispiel um Belege hochzuladen

7.3.6.3 Aktionen

Excel herunterladen

Der Benutzer kann Reiseformulare und Reiserechnungen als *Excel*-Datei herunterladen. Diese werden mit folgendem *Code* erstellt und heruntergeladen [13]:

```

1  var anchor_href =
2    "data:application/vnd.openxmlformats-officedocument.spreadsheetml.sheet;base64, "
3    ↪   "
4    ↪   +
5      excel; // Wandelt die Daten vom Backend in eine Excel-Datei um
6  var exportLinkElement = document.createElement("a");
7
8    exportLinkElement.hidden = true;
9    exportLinkElement.download = "Formular.xlsx";
10   exportLinkElement.href = anchor_href;
11   exportLinkElement.text = "downloading...";
12
13  document.body.appendChild(exportLinkElement);
14  exportLinkElement.click();
15  exportLinkElement.remove();

```

Auflistung 7.78: *Excel* erstellen und herunterladen

PDF öffnen

Der Benutzer kann die *PDFs* der Dokumente öffnen. Die *PDFs* werden in einem neuen Tab im *Browser* angezeigt.

Mit folgendem *Code* wird eine *PDF* erstellt und in einem neuen Tab geöffnet [37]:

```

1  let pdfWindow = window.open("");
2  var fileName = "PDF";
3  pdfWindow.document.write(
4    "<html<head><title>" +
5    fileName +
6    "</title><style>body{margin: 0px;}iframe{border-width: 0px;}</style></head>"
7  );
8  pdfWindow.document.write(
9    "<body><embed width='100%' height='100%' src='data:application/pdf;base64, " +
10   encodeURI(pdf) +
11   "#toolbar=0&navpanes=0&scrollbar=0'></embed></body></html>"
12 );

```

Auflistung 7.79: *PDF* erstellen und öffnen im *Browser*

Änderungen speichern

Der Benutzer kann die Änderungen eines Antrags speichern. Dies kann in bestimmten Phasen eines Antrags geschehen. Die Änderungen werden an das *Backend* gesendet, sobald auf den *Änderungen Speichern*-Knopf gedrückt wird. Das senden erfolgt mit folgendem *Code*:

```

1  if (this.belege.length >= 1) {
2      this.sendReceipts(this.belege); // Falls Belege angegeben worden sind,
3      // werden diese hochgeladen
4  }
5  if (this.checkProgression()) {
6      this.app.progress = 2; // Falls alle Begleitlehrer ihre Reiseformulare
7      // ausgefüllt haben, wird der Fortschritt auf den nächsten Schritt gesetzt
8  }
9  if (this.checkInvoices()) {
10     this.app.progress = 6; // Falls alle Begleitlehrer ihre Reiserechnungen
11     // ausgefüllt haben, wird der Fortschritt auf den nächsten Schritt gesetzt
12 }
13 this.app.last_changed = this.createNewDate(); // Das Datum der letzten Änderung
14 // wird im Antrag gespeichert.
15 axios.put(this.url + "/updateApplication?uuid=" + this.app.uuid,
16 {
17     headers: {
18         Authorization: "Basic " + this.token
19     }
20 },
21 this.app // Es wird der neue Antrag (Mit den neuen Änderungen) hochgeladen
22 )

```

Auflistung 7.80: Beispiel um die Änderungen eines Antrags zu speichern

Antrag schließen

Der Leiter eines Antrags kann den Antrag schließen. Sobald der Antrag geschlossen ist, wird dieser im *Backend* auf den Fortschritt abgeschlossen gesetzt. Dies wird mit folgendem *Code* erreicht:

```

1  if (this.app.kind === 0) { // Der Fortschritt wird je nach Antragsart auf
2      // Abgeschlossen gesetzt
3      this.app.progress = 7;
4  } else {
5      this.app.progress = 6;
6  }
7  this.save(); // Der Antrag wird gespeichert
this.hideClose(); // Das Modal wird geschlossen

```

Auflistung 7.81: Beispiel um einen Antrag zu schließen

Antrag löschen

Der Leiter eines Antrags kann den Antrag löschen. Sobald auf den Knopf *Antrag löschen* gedrückt wird, wird dieser im *Backend* gelöscht. Der Benutzer wird daraufhin auf die Startseite weitergeleitet. Ein Antrag wird mit folgender Anfrage gelöscht:

```
1 axios.delete(this.url + "/deleteApplication?uuid=" + this.app.uuid, { // Es
2   ← wird ein Delete-Request an das Backend gesendet
3   headers: {
4     Authorization: "Basic " + this.token
5   }
})
```

Auflistung 7.82: Beispiel um einen Antrag zu löschen

7.3.7 Antrag Übersicht

Um einen Überblick über die Anträge zu erlangen, gibt es zwei Übersichtsseiten, welche die Anträge auf-listen. In den Auflistungen werden die groben Daten des Antrags angezeigt und können in der *Antrag-Ansicht* geöffnet werden. Sobald eine Übersichtsseite aufgerufen wird, werden die Anträge abgefragt und in einer Liste angezeigt:

Titel	Leiter	Einreichdatum	Status	Aktionen
Pflegefreistellung	Stefan Zakall	14-04-2021	Abgeschlossen	Schnelle Information Antrag Betrachten
Sommersportwoche	Stefan Zakall	14-06-2021	In Bearbeitung	Schnelle Information Antrag Betrachten

Abbildung 7.35: Auflistung von Anträgen auf der Übersichtsseite

7.3.7.1 Alle Anträge

Auf dieser Übersichtsseite werden alle Anträge angezeigt, die mit dem angemeldeten Benutzer zusammenhängen.

7.3.7.2 Aktive Anträge

In der Auflistung der aktiven Anträge, werden alle Anträge angezeigt, welche nicht abgeschlossen sind. Ist nur ein Antrag in der Liste der aktiven Anträge, so wird der Antrag sofort in der *Antrag-Ansicht* geöffnet.

7.3.7.3 Aktionen

Schnelle Informationen

Wird auf den Knopf *Schnelle Informationen* gedrückt, öffnet sich ein Fenster, in dem die groben Informationen des Antrags angezeigt werden.

Antrag betrachten

Wird auf den Knopf *Antrag betrachten* gedrückt, wird der Antrag in der *Antrag-Ansicht* geöffnet.

7.3.8 Administrator

7.3.8.1 Antrag Ansicht

Der Aufbau der *Antrag-Ansicht* ähnelt dem aus Unterabschnitt 7.3.6.

Ein Administrator kann alle Informationen eines Antrags einsehen und diesen akzeptieren. Des Weiteren können alle *PDFs* des Antrags geöffnet werden. Der Antrag kann auch abgelehnt werden, wodurch der Fortschritt zurückgesetzt wird. Wenn der Antrag abgelehnt wird, kann ein Grund für das Ablehnen angegeben werden. Der Antragsteller kann daraufhin die Informationen des Antrags aktualisieren und damit den Antrag erneut einreichen oder den Antrag schließen.

7.3.8.2 Antrag Übersicht

Die Übersichtsseite für Administratoren ähnelt dem Aufbau aus Unterabschnitt 7.3.7.

Der Administrator sieht, wer den Antrag gestellt hat, und kann mehrere Anträge auswählen, um diese in einer *PDF* anzuzeigen. Des Weiteren kann auf den Knopf *Antrag betrachten* gedrückt werden, wodurch der Administrator auf die Antrag Ansicht für Administratoren weitergeleitet wird.

7.3.8.3 Rechte

Die Rechte für Benutzer können über ein Formular bearbeitet werden. Die E-Mail-Adresse des zu bearbeitenden Nutzers wird in dem Formular angegeben. Zusätzlich werden über die Auswahl die Rechte von dem Benutzer gesetzt.

E-Mail @tgm.ac.at

Rechte

- Admin
- AV
- Administration
- PEK
- Lehrer

Geben Sie die Rechte des Kontos an.

Rechte speichern

Abbildung 7.36: Formular, um Rechte eines Benutzers zu setzen

Sobald auf den *Rechte speichern*-Knopf gedrückt wird, wird das Formular ausgelesen und damit die Anfrage für das *Backend* erstellt. Die Anfrage an das *Backend* wird folgendermaßen gesendet:

```
1 axios.get(this.url + "/setTeacherPermissions?uuid=" + response.data.uuid,
2 {
3     headers: {
4         Authorization: "Basic " + this.token
5     }
6 }, rechte // Die im Formular eingestellten Rechte
7 )
8 .then(res => {
9     if (res.status === 200) {
10         this.saveComplete();
11         this.reset();
12     }
13 }) ;
```

Auflistung 7.83: Anfrage zum Setzen der Rechte eines Benutzers

Kapitel 8

Retrospektive

8.1 Probleme

Die Zusammenarbeit zwischen den Teammitgliedern war aufgrund der derzeitigen Pandemie stark beeinträchtigt, wodurch es viele neue Herausforderungen zu meistern gab.

Durch diese Situation wurde die Entwicklung der Datenschnittstelle deutlich erschwert und sie konnte aufgrund fehlender Informationen erst sehr spät entwickelt werden. Auch die Entwicklung der Webseitenlogik hat durch fehlende Kommunikation länger gedauert als geplant. Dadurch wurden Fehler in der Navigation, sowie bei den Eingaben des Nutzers einprogrammiert, welche später behoben werden mussten.

Durch herausragende Zusammenarbeit gegen Ende des Projekts konnten jedoch alle Herausforderungen bewältigt werden.

8.2 Erkenntnisse

Die Kommunikation ist ein essenzieller Teil eines Projektes, welcher auf gar keinen Fall vernachlässigt werden darf. Des Weiteren hat das Projektteam gelernt, sich nicht durch Rückschritte frustrieren zu lassen, sondern umso mehr daran zu arbeiten, dass das Endprodukt funktionsfähig ist.

8.3 Webdesign

Alle erforderlichen Seiten des *Frontends* wurden erfolgreich implementiert. Die Webseite wurde mittels *Bootstrap*-Komponenten gefüllt, welche im Nachhinein an das *Design* von *Refundable* angepasst wurden. Alle benötigten Formulare wurden in *HTML*-Form kreiert und sind für den Nutzer mit ausreichend Informationen und Hinweisen bestückt. Auch die Übersicht der normalen und administrativen Nutzer wurde vollständig erstellt.

Jedoch hat das Projektteam viel aus der Arbeit im Projekt gelernt, und hat deshalb folgende Verbesserungsvorschläge entworfen:

- Das Design der *Login*- und *Startseite* auf den restlichen Seiten stärker fortführen
- Die Darstellung auf mobilen Geräten verbessern
- Das Reisekostenformular verschönern

Auflistung 8.1: Auflistungen von Verbesserungsmöglichkeiten

8.4 Backend und Infrastruktur

Sowohl das *Backend*, als auch die Infrastruktur, wurden komplett implementiert. Die Hauptfunktionalität wurde fertig programmiert und in das System und in die Betriebsumgebung eingepasst. Zusätzlich wurde auch die Infrastruktur entworfen und dynamisch *deployable* designet, sodass dieses einfach aufbaubar und steuerbar ist. Die *REST*-Schnittstelle bereitet Daten entsprechend auf und stellt sie auf Anfrage eines *Clients* zur Verfügung. Auch die Schnittstellenherausforderungen im *Backend* mit Diensten und Protokollen wie *LDAP* oder *Untis* wurden gelöst und auf eine Art und Weise, dass die Implementierung klar und effizient ablaufen kann.

Jedoch gibt es immer meistens bei Softwareprojekten mit klaren *Deadlines* immer noch etwas, das noch implementiert, verbessert oder neu entworfen werden kann. So ist es auch hier im *Backend* und in der Infrastruktur. Aus dieser Erkenntnis heraus wurde eine Liste erstellt, welche weiteren Implementierungen und möglichen Verbesserungen wichtig für das System wären und die Software unterstützen könnten:

- Versenden von Mails (bei Statusänderungen oder als Erinnerungen)
- Ein umfangreichendes *Logging*-System, um jegliche Aktionen aufzzeichnen zu können
- Ein einfacheres Design des Datenmodells entwerfen
- PDF-Vorlagen besser und übersichtlicher designen
- Die *REST*-Schnittstelle mit weiteren Steuerungsmöglichkeiten und kleineren *Endpoints* zur Zustandsänderung ausstatten, wodurch große *Endpoints* (wie beispielsweise jener um Anträge zu bearbeiten) obsolet werden würden und die *Endpointstruktur* klarer und übersichtlicher werden würde.
- modulare Wahl zwischen Datenbanken inklusive Datenbankschnittstelle über *Strategy-Pattern* implementieren, und über Nutzerkonfiguration bei der Installation Entscheidung ermöglichen.
- Grenzwertüberprüfungen im *Backend* einbauen, sodass nur letztlich valide Werte genutzt werden können
- Reduzierung der *McCabe-Metrik* (zyklomatische Komplexität)
- Generelle *Performance* Verbesserungen
- Weitere Sicherheitsmaßnahmen entwerfen und implementieren (beispielsweise HTTPS Implementation für *Webserver* und *REST*-Schnittstelle)

Auflistung 8.2: Auflistungen von Ideen zur weiteren Implementierung und Verbesserung

8.5 Datenschnittstelle und Webseitenlogik

Die Webseitenlogik wurde vollends implementiert und lässt den Benutzer auf jede gewünschte Seite zugreifen. Die Datenschnittstelle ist ebenfalls implementiert. Die Datenschnittstelle ist so konfiguriert, dass eine *REST*-Schnittstelle angesprochen wird, um die Daten eines Benutzers zu erhalten, damit dieser das System verwenden kann. Die Rechte, die bei jedem Benutzer hinterlegt sind, lassen den Nutzer auf unterschiedlichste Unterseiten gelangen. Sobald ein Antrag fertig erstellt worden ist, wird dieser mit allen eingegebenen Informationen an die *REST*-Schnittstelle gesendet, damit dort der Antrag gespeichert wird.

Trotzdem, gibt es einige Verbesserungen, die in Zukunft umgesetzt werden könnten. Diese sind in der folgenden Liste aufgezählt:

- Google-Maps Schnittstelle zur Berechnung der Fahrtdistanz
- Überprüfung, ob ein ähnlicher Antrag bereits existiert
- Vorlagen von Veranstaltungen speichern
- Verbesserung der Auswahl der Teilnehmenden Lehrer einer Schulveranstaltung
- Verbesserung der Fehlerbehandlung der Anfragen an die *REST*-Schnittstelle
- Verbesserung der allgemeinen Performanz der Datenschnittstelle sowie der Webseitenlogik

Auflistung 8.3: Auflistungen von Verbesserungen zur zukünftigen Implementierung, der Datenschnittstelle und Webseitenlogik

Kapitel 9

Conclusio

Das Diplomprojekt *Refundable* hat das gesamte Team vor viele technische Herausforderungen gestellt. Trotzdem wurden während des Projekts viele Erfahrungen in Verbindung mit Webdesign, Datenschnittstellen, Schnittstellenentwicklung und Systemintegration gesammelt. Durch die intensive Auseinandersetzung mit den Themenbereichen konnten einige neuen Fähigkeiten erlernt und angewandt werden. Die Anforderung an das Diplomprojekt, eine völlig neue, maßgeschneiderte Software zu entwickeln, zog unzählige Umsetzungsmöglichkeiten auf. Die durchgeführte Studie im Diplomprojekt war essenziell, um eine konkrete Umsetzungsmöglichkeit zu finden. Auch in sozialer Hinsicht haben die Teammitglieder einiges dazugelernt. Nicht immer ist alles rund gelaufen, jedoch hat das Team gelernt, produktiver zu arbeiten und sich zusammenzureißen, falls mal etwas nicht so geklappt hat, wie es ursprünglich geplant war. Das gesamte Team freut sich, die einzigartige Möglichkeit gehabt zu haben, das Diplomprojekt *Refundable* umzusetzen.

Glossar

Angular Angular ist ein JavaScript-Framework entwickelt von Google. 45, 174

Array Ein Array ist eine geordnete Ansammlung an Daten. 40, 174

Auszeichnungssprache Vereinfacht gesagt, eine Sprache die von verschiedenen Programmen lesbar ist und für die Gliederung und Formatierung von jeglichen Daten zuständig ist. 26, 174

axios axios ist eine Bibliothek, welche HTTP anfragen senden kann. 51, 146, 174

Backend Das Backend ist die Funktionalität im Hintergrund des Frontends. 43, 48, 174, 176

Bash Bash ist eine Unix-Shell, welche die Eingabe von Befehlen ermöglicht. Ein Bash-Skript ist im weiteren Sinne eine Ansammlung von solchen Befehlen, wobei der Kontrollfluss mit Hilfe von Verzweigungen gesteuert werden kann. 37, 174

Boolean Booleans sind ein Datenformat, welches einen Wahrheitswert, also entweder wahr (true; 1) oder falsch (false; 0), repräsentiert. 40, 174

CDN CDN oder auch Content Delivery Network. Darunter kann man sich eine Online Datenbank vorstellen, welche optionale Komponenten (wie vorgefertigte CSS Skripte) beinhaltet, damit man sie nicht lokal auf dem Server speichern muss. 31, 174

Code-Pen Eine Website, auf der man Live Code ausführen und testen kann. 34, 174

CSS Cascading Style Sheet. Zum umgestalten und verschönern der Weboberfläche. 26, 45, 174, 176

Docker Docker ist eine Software, welche es ermöglicht Programme in einer abgeschnittenen Umgebung (genannt Container) laufen zu lassen [10]. Das Erstellen dieser Umgebung und das Installieren und Laufen des Programms darin, gestaltet sich hierbei sehr einfach. 36, 174

Docker Compose Docker Compose ist eine Erweiterung von Docker [39]. Mit ihr kann man multiple Container gleichzeitig aufbauen, womit es ermöglicht wird komplexe Infrastruktur - wie in Refundable benötigt - einfach aufzubauen, zu reproduzieren und letztlich auf die Computer, auf denen während der Produktion die Infrastruktur laufen wird, zu liefern. 36, 174

Entwurfsmuster Eine Vorlage, wie man ein Programm oder Software intern strukturiert.. 43, 44, 174

Framework Kann als Baukasten gesehen werden. Bietet Möglichkeiten um die normalen Vorhergehensweisen zu kürzen bzw. vereinfachen. 25, 26, 43, 45, 48–50, 174, 176

Frontend Das Frontend ist die Oberfläche einer Website - also das was zu sehen ist. 26, 43, 48, 51, 174–176

Full Stack Framework Ein Framework für Backend als auch Frontend. 174

Git Git ist eine Software zur verteilten Versionsverwaltung. Dies bedeutet, dass Code, der sich in einem Repository, einem von Git verwaltetem Ordner, und speziell die Veränderungen im Code erfasst werden und als verschiedene Versionen des Codes gespeichert werden. 37, 174

GitHub GitHub ist ein online Service, welcher Git-Repositories hosted. Hierdurch wird unter anderem Kollaboration im Team, die Verteilung von Programmen, aber auch generell die Versionsverwaltung auf mehreren Geräten ermöglicht. 37, 174

Google Maps Eine Online-Karten Service entwickelt von Google.. 174

HTML Hypertext Markup Language. Die Grundbausteine bzw. Struktur der Weboberfläche. Zum Beispiel Text oder eine Eingabefläche. 26, 45, 48, 51, 174, 176

HTTP Unter HTTP (Hypertext Transfer Protocol) versteht man ein zustandsloses Protokoll zur Übertragung von Daten[23]. 51, 174

HTTPS Unter HTTPS (Hypertext Transfer Protocol Secure) versteht sich ein Kommunikationsprotokoll, welches über Verschlüsselung durch Zertifikate eine Verbindung zwischen Server und Client im Web sicherer macht. 37, 174

Java Eine Programmiersprache der Firma Oracle, wird meist für Desktopapplikationen bzw. für das Backend. 174

JavaScript Java Script. Bietet einen Rahmen an Funktionalität und Animationen in Verbindung mit HTML und CSS. 26, 43, 45, 46, 48–50, 174

JSON Bei JSON (JavaScript Object Notation) handelt es sich um ein kompaktes textbasiertes Datenformat, welches für den Datenaustausch zwischen Schnittstellen entwickelt wurde [8]. 36, 40, 51, 174

JSON Web Token Ein JSON Web Token ist ein kompaktes Übertragungsformat für die sichere Übertragung zwischen zwei Punkten. Die Informationen sind dabei signiert und können dadurch verifiziert werden. [27] . 36, 174

Konsistenz Konsistenz bedeutet die Einhaltung von Regeln. Im Zusammenhang mit Daten in einer Datenbank ist gemeint, dass die Daten die im DBMS gespeichert sind die vordefinierten Regeln einhalten müssen. 37, 174

Mockup Ein Design Prototyp einer Website, welcher rein optisch ist und keine Funktionalität besitzt. 174

MV* MV* (Model-View-*) ist eine Zusammenfassung aller Model-View Patterns. MVVM und MVC fallen beide in dieses Schema hinein. 43, 174

MVC MVC (Model-View-Controller) ist ein Entwurfsmuster, welches die Logik eines Programms von dem Interface(View) trennt. 43, 44, 174

MVVC MVVC (Model-View-ViewController) ist ein anderer Name für MVVM. 44, 174

MVVM MVVM (Model-View-ViewModel) ist ein Entwurfsmuster, welches eine Variante des MVC-Patterns ist. 43, 44, 51, 174

nicht-relationale Datenbank (auch *NoSQL Datenbank*) Datenbank die von dem Konzept, dass Daten durch eine Tabelle repräsentiert wird, abweicht. Hierbei gibt es sehr viele verschiedene Ansätze, die von einer Datenbank-Software zur anderen verschieden sind. 36, 174

Null Null ist ein Datentyp, welcher die Abwesenheit von Daten beschreibt. Er ist nicht zu verwechseln mit einer 0. Null wird meistens dann genutzt, wenn Daten noch nicht verfügbar sind. 40, 174

Objekt Ein Objekt ist eine ungeordnete Ansammlung an verschiedenen benannten Feldern, welche alle eigene Daten speichern können. 40, 174

Open-Source Open-Source bedeutet, dass der Sourcecode öffentlich ist und von Dritten einsehbar ist. 46, 174

React React ist ein JavaScript-Framework entwickelt von Facebook. 46, 50, 51, 174

relationale Datenbank Datenbank, wo ein Typ von Daten durch eine Tabelle repräsentiert wird. Die Anzahl der Spalten ist hierbei für jeden Datensatz konstant. 36, 174

Repository Ein Repository ist ein Verzeichnis, in welchem Git aktiv als Versionsverwaltung arbeitet. 37, 174

SASS SASS oder auch Syntactically Awesome Style Sheets ist eine Skriptsprache, welche auf CSS basiert. Jedoch bietet SASS deutlich mehr Agilität und einen hohen Grad an Automatisierung. 26, 174

String Ein String ist ein Datenformat, welches in der Informatik Text und Zeichenketten repräsentiert. 40, 174

Tag Markiert in HTML ein HTML-Element. Auch als Marken bezeichnet. 27, 174

Third Party Packages Third Party Packages sind Programme bzw. Teile von Programmen, die von Entwicklern bereits erstellt und programmiert wurden. Sie wurden daraufhin für die Wiederverwendung von anderen Entwicklern veröffentlicht. Der Gedanke dahinter orientiert sich an dem Sprichwort „Man muss das Rad nicht neu erfinden.“. 35, 41, 174

Vanilla Ein anderer Begriff für Basisausführung. 26, 174

Verfügbarkeit Verfügbarkeit im Zusammenhang mit einem Dienst bedeutet, dass jener Dienst zu einer bestimmten Zeit erreichbar ist. Eine hohe Verfügbarkeit bedeutet, dass der Dienst (fast) immer erreichbar ist. 37, 174

VueJS VueJs ist ein JavaScript-Framework entwickelt von Evan You. 45, 46, 50, 51, 174

Webapplikation Wie eine Programm auf dem PC, nur dass das Programm nicht auf dem PC installiert wird, sondern im Internet aufgerufen und geladen wird. 174

Webinterface Ein Web Interface ist ein System, durch welches Anwender mit dem Netz interagieren.
Der Begriff Web Interface steht zumeist für grafische Oberflächen. 35, 174

XML XML (Extensible Markup Language) ist eine Markup-Sprache, die zur Beschreibung von Daten genutzt wird [3]. Wobei die Datenstruktur von XML über ein Schema verifiziert werden kann. 36, 41, 174

YAML YAML (YAML Ain't Markup Language) ist eine Markup-Sprache, die zur Beschreibung von Daten genutzt wird. Hierbei zeichnet sich YAML genau dabei aus, dass es nicht nur für die Maschine, sondern auch für den Menschen gut lesbar ist. 36, 174

Akronyme

CDN Content Delivery Framework. 174

CSS Cascading Style Sheets. 174

DB Datenbank. 36, 174

DBMS Datenbankmanagementsystem. 37, 174, 176

DIN Deutsches Institut für Normung. 174

HTML Hypertext Markup Language. 174

ITP Informationstechnische Projekte. 21, 174

JS Java Script. 174

LoC Lines of Code. 49, 174

PM Projektmanagement. 21, 174

PMBOK Project Management Body of Knowledge. 21, 174

PMI Project Management Institut. 21, 174

SASS Syntactically Awesome Style Sheets. 174

Literaturverzeichnis

- [1] *Agiles Projektmanagement im Berufsalltag - Für mittlere und kleine Projekte* / Ursula Kusay-Merkle / Springer. URL: <https://www.springer.com/de/book/9783662567999>.
- [2] *Angular-Tutorial für Einsteiger*. URL: <https://angular.de/artikel/angular-tutorial-deutsch/> (besucht am 12.11.2020).
- [3] Tim Bray u. a. *Extensible Markup Language (XML) 1.0 (Fifth Edition)*. 2008. URL: <http://www.w3.org/TR/2008/REC-xml-20081126/> (besucht am 07.11.2020).
- [4] Peter Bühler, Patrick Schlaich und Dominik Sinner. *HTML5 und CSS3: Semantik - Design - Responsive Layouts*. ger. Bibliothek der Mediengestaltung. Berlin, Heidelberg: Springer Berlin Heidelberg Imprint: Springer Vieweg, 2017. ISBN: 3662539160. DOI: 10.1007/978-3-662-53916-3.
- [5] „Create Golang API Doc with Swag“. In: 0. URL: <https://adityarama1210.medium.com/create-golang-api-doc-with-swag-d73be1767d39> (besucht am 19.04.2021).
- [6] *Creating a Project / Vue CLI*. URL: <https://cli.vuejs.org/guide/creating-a-project.html> (besucht am 31.03.2021).
- [7] „Creating PDFs using Golang“. In: 0. URL: <https://johnathanfercher.medium.com/creating-pdfs-using-golang-98b722e99d6d> (besucht am 19.04.2021).
- [8] D Crockford. *The application/json Media Type for JavaScript Object Notation (JSON)*. RFC 4627. RFC Editor, 2006. URL: <https://www.rfc-editor.org/rfc/rfc4627.txt>.
- [9] *Das Wasserfallmodell - Online-Projektmanagement*. URL: <https://www.online-projektmanagement.info/agiles-projektmanagement-scrum-methode/scrum-versus-wasserfallmodell/das-wasserfallmodell/> (besucht am 19.04.2021).
- [10] *Docker Engine overview / Docker Documentation*. URL: <https://docs.docker.com/engine/> (besucht am 08.11.2020).
- [11] *Documentation - Materialize*. URL: <https://materializecss.com/> (besucht am 15.10.2020).
- [12] *Documentation - The Go Programming Language*. URL: <https://golang.org/doc/> (besucht am 26.11.2020).
- [13] *Download file approaches*. URL: <https://codepen.io/DanIgnatov/pen/RvbeeB> (besucht am 18.04.2021).
- [14] *excelize - pkg.go.dev*. URL: <https://pkg.go.dev/github.com/360EntSecGroup-Skylar/excelize/v2> (besucht am 14.04.2021).
- [15] *Flask-RESTful – Flask-RESTful 0.3.8 documentation*. URL: <https://flask-restful.readthedocs.io/en/latest/> (besucht am 26.11.2020).
- [16] *Foundation for Sites 6 Docs*. URL: <https://get.foundation/sites/docs/> (besucht am 26.11.2020).

- [17] *Getting Started / BootstrapVue*. URL: <https://bootstrap-vue.org/docs> (besucht am 31.03.2021).
- [18] *gin-gonic - pkg.go.dev*. URL: <https://pkg.go.dev/github.com/gin-gonic/gin> (besucht am 19.04.2021).
- [19] Hugo Giraudel und Miriam Suzanne. *Jump Start Sass*. eng. 1. Aufl. SitePoint. ISBN: 0994182678.
- [20] *go-ldap - pkg.go.dev*. URL: <https://pkg.go.dev/github.com/go-ldap/ldap/v3> (besucht am 19.04.2021).
- [21] *golang - Docker Hub*. URL: https://hub.docker.com/_/golang (besucht am 16.04.2021).
- [22] *httpd - Docker Hub*. URL: https://hub.docker.com/_/httpd (besucht am 11.11.2020).
- [23] *Hypertext Transfer Protocol – Wikipedia*. URL: <https://de.wikipedia.org/wiki/HypertextTransferProtocol> (besucht am 26.11.2020).
- [24] *Installation / Vue CLI*. URL: <https://cli.vuejs.org/guide/installation.html> (besucht am 31.03.2021).
- [25] *Introduction · Bootstrap v4.5*. URL: <https://getbootstrap.com/docs/4.5/getting-started/introduction/> (besucht am 20.10.2020).
- [26] *JDK 11 Documentation - Home*. URL: <https://docs.oracle.com/en/java/javase/11/> (besucht am 26.11.2020).
- [27] Michael Jones, John Bradley und Nat Sakimura. *JSON Web Token (JWT)*. RFC 7519. RFC Editor, 2015. URL: <https://www.rfc-editor.org/rfc/rfc7519.txt>.
- [28] *jwt-go - pkg.go.dev*. URL: <https://pkg.go.dev/github.com/dgrijalva/jwt-go> (besucht am 14.04.2021).
- [29] Jörg Krause. *Introducing Bootstrap 4: Create Powerful Web Applications Using Bootstrap 4.5*. eng. 2. Aufl. Berkeley, CA: Apress. ISBN: 9781484262030. DOI: 10.1007/978-1-4842-6203-0.
- [30] Matt Lambert. *Learning Bootstrap 4*. eng. Packt Publishing. ISBN: 1785881000.
- [31] A Leff und J.T Rayfield. „Web-application development using the Model/View/Controller design pattern“. eng. In: *Proceedings Fifth IEEE International Enterprise Distributed Object Computing Conference*. IEEE, 2001, S. 118–127. ISBN: 9780769513454. DOI: 10.1109/EDOC.2001.950428.
- [32] *Lehre - Begriff: Markup Language [Zentrum für Informationsmodellierung in den Geisteswissenschaften]*. URL: <http://www-gewi.uni-graz.at/zim/lehre/markuplanguages.html> (besucht am 11.11.2020).
- [33] *maroto - pkg.go.dev*. URL: <https://pkg.go.dev/github.com/johnfercher/maroto> (besucht am 14.04.2021).
- [34] *mongo - Docker Hub*. URL: https://hub.docker.com/_/mongo (besucht am 11.11.2020).
- [35] *mongo-driver - pkg.go.dev*. URL: <https://pkg.go.dev/go.mongodb.org/mongo-driver/mongo> (besucht am 19.04.2021).
- [36] Nurzhan Nurseitov u. a. *Comparison of JSON and XML Data Interchange Formats: A Case Study*. Techn. Ber.
- [37] *Opening PDF String in new window with javascript - Stack Overflow*. URL: <https://stackoverflow.com/questions/2805330/opening-pdf-string-in-new-window-with-javascript> (besucht am 05.04.2021).

- [38] *Our Documentation / Python.org*. URL: <https://www.python.org/doc/> (besucht am 26.11.2020).
- [39] *Overview of Docker Compose / Docker Documentation*. URL: <https://docs.docker.com/compose/> (besucht am 08.11.2020).
- [40] Sanjay Patni. *Pro RESTful APIs*. eng. 1. Aufl. Berkeley, CA: Apress L. P, 2017. ISBN: 978-1-4842-2665-0. DOI: 10.1007/978-1-4842-2665-0.
- [41] *pdcpu - pkg.go.dev*. URL: <https://pkg.go.dev/github.com/hhrutter/pdcpu> (besucht am 19.04.2021).
- [42] Anirudh Prabhu und Aravind Shenoy. *Introducing Materialize*. eng. 1. Aufl. Berkeley, CA: Apress L. P, 2016. ISBN: 9781484223482. DOI: 10.1007/978-1-4842-2349-9.
- [43] *React – A JavaScript library for building user interfaces*. URL: <https://reactjs.org/> (besucht am 07.11.2020).
- [44] *Revision von Projektmanagement (PM) vom Di., 26.02.2013 - 18:46 • Definition / Gabler Wirtschaftslexikon*. URL: <https://wirtschaftslexikon.gabler.de/definition/projektmanagement-pm-46130/version-176073> (besucht am 19.04.2021).
- [45] „Secured MongoDB container“. In: (). URL: <https://medium.com/@leondeng/set-up-a-secured-mongodb-container-e895807054bd> (besucht am 19.04.2021).
- [46] Y Shafrazevich. *Common Format and MIME Type for Comma-Separated Values (CSV) Files*. RFC 4180. RFC Editor, 2005. URL: <https://www.rfc-editor.org/rfc/rfc4180.txt>.
- [47] Aravind Shenoy. *Introducing Zurb Foundation* 6. eng. 1. Aufl. Berkeley, CA: Apress L. P, 2015. ISBN: 9781484217955.
- [48] *Simple bash subcommands*. URL: <https://gist.github.com/waylan/4080362> (besucht am 19.04.2021).
- [49] *Spring / Why Spring?* URL: <https://spring.io/why-spring> (besucht am 26.11.2020).
- [50] Ralph Steyer. *Webanwendungen erstellen mit Vue.js : MVVM-Muster für konventionelle und Single-Page-Websiten*. ger. 1st ed. 20. Wiesbaden: Springer Fachmedien Wiesbaden Imprint: Springer Vieweg, 2019. ISBN: 3658271701. DOI: 10.1007/978-3-658-27170-1.
- [51] *swaggo - pkg.go.dev*. URL: <https://pkg.go.dev/github.com/swaggo/swag> (besucht am 19.04.2021).
- [52] *Types of NoSQL Databases / MongoDB*. URL: <https://www.mongodb.com/scale/types-of-nosql-databases> (besucht am 11.11.2020).
- [53] „Using JWT for Authentication in a Golang Application“. In: (). URL: <https://codeburst.io/using-jwt-for-authentication-in-a-golang-application-e0357d579ce2> (besucht am 19.04.2021).
- [54] *Vue.js*. URL: <https://vuejs.org/> (besucht am 07.11.2020).
- [55] *Was ist Agiles Projektmanagement? / Definition und Methoden*. URL: <https://www.projektmagazin.de/glossarterm/agiles-projektmanagement> (besucht am 19.04.2021).
- [56] *Wasserfallmodell / Definition & Anwendungsbereiche - IONOS*. URL: <https://www.ionos.at/digitalguide/websites/web-entwicklung/wasserfallmodell/> (besucht am 19.04.2021).
- [57] WebUntis. *WebUntis JSON-RPC API*. (Besucht am 19.04.2021).

- [58] Jürgen Wolf. *HTML5 und CSS3 : das umfassende Handbuch.* ger. 3., aktual. Rheinwerk Computing. Bonn: Rheinwerk Verlag, 2019. ISBN: 3836262266.

Abbildungsverzeichnis

4.1	Wasserfallmodell	22
4.2	Scrum	23
4.3	Kanban	24
5.1	<i>HTML</i> Beispieleseite	27
5.2	Übersicht über die Komponenten	35
5.3	Übersicht des <i>MVC-Patterns</i>	43
5.4	Übersicht des <i>MVVM-Patterns</i>	44
5.5	Die Beispielwebseite	49
6.1	Mockup Login	55
6.2	<i>Use-Case</i> Diagramm Login	56
6.3	Mockup Startseite	57
6.4	<i>Use-Case</i> Diagramm Startseite	58
6.5	Mockup neuer Antrag	59
6.6	Mockup Antrag erstellen	59
6.7	<i>Use-Case</i> Diagramm Neuer Antrag	60
6.8	Mockup Alle Anträge	61
6.9	<i>Use-Case</i> Diagramm Alle Anträge	61
6.10	Mockup aktive Anträge	62
6.11	<i>Use-Case</i> Diagramm Aktive Anträge	62
6.12	Mockup Adminansicht	63
6.13	<i>Use-Case</i> Diagramm Adminansicht	63
6.14	Mockup Antragsansicht	64
6.15	<i>Use-Case</i> Diagramm Antragsansicht	64
6.16	<i>Docker-Compose Stack</i>	65
6.17	Flussdiagramm über den Installationsvorgang	67
6.18	Flussdiagramm über den Startvorgang	68
6.19	Flussdiagramm über den Stoppvorgang	69
6.20	Flussdiagramm über den Neustartvorgang	70
6.21	Flussdiagramm über den Aktualisierungsvorgang	71
6.22	Flussdiagramm über den Bereinigungsvorgang	72
6.23	Flussdiagramm über die Deinstallation	73
6.24	Zentrales Datenmodell	74
6.25	<i>Untis API-Client UML-Klassendiagramm</i>	77
6.26	Authentifizierung über <i>LDAP</i>	78

6.27	<i>LDAP UML-Klassendiagramm</i>	79
6.28	Kopfleiste PDF-Datei	80
6.29	Hierarchie der Webseite	81
6.30	Seite nicht gefunden	81
6.31	Prozess der Daten zur Anzeige	83
6.32	Neue Nachrichten	83
6.33	Prozess der Befehlssendung	84
7.1	Login Seite	88
7.2	Dashboard Seite	93
7.3	Dashboard Mobil Seite	94
7.4	Neuen Antrag erstellen Seite	98
7.5	Neuer Schulantrag Seite (Teil 1)	99
7.6	Neuer Schulantrag Seite (Teil 2)	99
7.7	Tags	101
7.8	Begleitformular	102
7.9	Fortbildungsantrag	103
7.10	Dienstreiseauftrag	103
7.11	Reiseantrag	104
7.12	Reisekostenabrechnung	105
7.13	Aktive Anträge Seite	108
7.14	Aktive Anträge <i>Pop-Up Fenster</i>	108
7.15	Alle Anträge Seite	110
7.16	Antragsansicht	111
7.17	Antragsansicht geöffnet	112
7.18	Antrag schließen Warnung	112
7.19	Antrag löschen Warnung	112
7.20	Administrator Ansicht	116
7.21	Antrag ablehnen Warnung	117
7.22	Nicht gefunden Seite	118
7.23	Antrag suchen Seite	119
7.24	Rechte vergeben Seite	119
7.25	Webseite <i>Antrag-Viewer</i>	145
7.26	Schulveranstaltungsinformationen eingeben	151
7.27	Begleitlehrerinformationen eingeben (Teil 1)	152
7.28	Begleitlehrerinformationen eingeben (Teil 2)	153
7.29	Fortbildungsinformationen eingeben (Teil 1)	155
7.30	Fortbildungsinformationen eingeben (Teil 2)	156
7.31	<i>Sonstiges-Antrag</i> Informationen eingeben (Teil 1)	158
7.32	<i>Sonstiges-Antrag</i> Informationen eingeben (Teil 2)	159
7.33	Fortschrittsanzeige	161
7.34	Dokumente eines Antrags	161
7.35	Liste der Anträge	166
7.36	Formular der Rechte	168

Tabellenverzeichnis

5.1	Vergleich <i>HTML Frameworks</i>	32
5.2	Vergleich zwischen Java, Python und Golang	39
5.3	Vergleich <i>JavaScript-Frameworks</i>	49
6.1	<i>REST-Endpoints</i> 1	75
6.2	<i>REST-Endpoints</i> 2	76
6.3	Dateierstellung - Dateien	80
7.1	Methoden der <i>MongoDB</i> Schnittstelle	129
7.2	Weitere Methoden der <i>Untis API</i>	135
7.3	Methoden des Token Management	137

Auflistungsverzeichnis

5.1	PML/DML Beispiele	26
5.2	<i>HTML Tags</i>	27
5.3	Kurzes <i>HTML</i> Beispiel	27
5.4	Beispiele Verwendung von CSS	28
5.5	Beispiele Verwendung von Inline-CSS	28
5.6	Beispiel CSS	29
5.7	CSS Verlinkung	29
5.8	Fragestellungen - Frameworks	30
5.9	Bootstrap Pro	30
5.10	Bootstrap Contra	30
5.11	Materialize Pro	31
5.12	Materialize Contra	31
5.13	ZURB Foundation Pro	32
5.14	ZURB Foundation Contra	32
5.15	<i>NoSQL</i> Datenbank-Typen	36
5.16	Prinzipien des <i>Representation State Transfer</i>	38
5.17	<i>JSON</i> - Datentypen	40
5.18	<i>JSON</i> Beispiel	40
5.19	<i>XML</i> Beispiel	41
5.20	<i>CSV</i> Beispiel	42
5.21	<i>VueJS</i> Beispiel	45
5.22	<i>React</i> -Webseite Beispiel	46
5.23	<i>React</i> -Komponente Beispiel	47
5.24	Ohne <i>Framework</i> Beispiel	48
5.25	<i>HTTP</i> -Anfrage mittels <i>Axios</i>	51
5.26	<i>Axios</i> Anfrage mit genauem Pfad	51
7.1	Benötigte Befehle um <i>Bootstrap</i> im <i>VueJS</i> Projekt einzubinden	86
7.2	CSS Klassen für die Haupt- Contianer und Reihen	87
7.3	HTML Code der Login-Maske	90
7.4	CSS Code der Login-Maske	91
7.5	HTML Code Cookie akzeptieren	92
7.6	HTML-Code-Teile des Startseitenmenüs	95
7.7	Teile des CSS Codes, des Startseitenmenüs	95
7.8	Teile des HTML-Codes der mobilen Startseite	96
7.9	HTML Code, des Neuigkeiten Elementes	97

7.10	HTML Code, einer Auswahlmöglichkeit	98
7.11	Beispiel für eine <i>Eingabegruppe</i>	100
7.12	<i>Timepicker</i>	100
7.13	<i>Datepicker</i>	101
7.14	<i>Tags</i>	102
7.15	Datei auswählen Code	106
7.16	Beispielcode Tabelle	107
7.17	Anträge Suchfunktion Code	109
7.18	Anträge filtern Code	110
7.19	<i>Pop-Up Fenster</i> Code	110
7.20	Codeausschnitt - Fortschrittsanzeige	113
7.21	CSS - Fortschrittsanzeige Farbklassen	113
7.22	Dokumente anzeigen Code	114
7.23	Antrag schließen Sicherheitshinweis Code	115
7.24	Badge Beispielcode	116
7.25	Antrag ablehnen Code	118
7.26	Skeleton Beispielcode	118
7.27	<i>MongoDB Dockerfile</i>	120
7.28	<i>MongoDB Initskript</i>	121
7.29	<i>Docker-Compose File: MongoDB-Container</i>	121
7.30	<i>Backend Dockerfile</i>	122
7.31	<i>Docker-Compose File: Backend-Container</i>	122
7.32	<i>Frontend Dockerfile</i>	123
7.33	<i>Docker-Compose File: Web-Container</i>	123
7.34	<i>Docker-Compose File: Secrets</i>	124
7.35	<i>Docker Installation</i> - Steuerungsskript	124
7.36	Installationsvorgang - Steuerungsskript	125
7.37	Startvorgang - Steuerungsskript	126
7.38	Stoppvorgang - Steuerungsskript	126
7.39	Neustartvorgang - Steuerungsskript	126
7.40	Aktualisierungsvorgang - Steuerungsskript	127
7.41	Bereinigungsvorgang - Steuerungsskript	127
7.42	Deinstallation - Steuerungsskript	128
7.43	<i>Subcommand parsing</i> - Steuerungsskript	128
7.44	<i>MongoDB Treiber</i> Beispiel	130
7.45	<i>LDAP-Schnittstelle: Authentifizierungsvorgang</i>	131
7.46	<i>LDAP-Schnittstelle: Namensabfrage</i>	132
7.47	Grundstruktur des <i>Untis Teil 1 Clients</i>	133
7.48	Grundstruktur des <i>Untis Clients</i> Teil 2	133
7.49	Hilffunktion für Anfragen	134
7.50	<i>Maroto</i> Beispiel	136
7.51	Token System Beispiel	137
7.52	<i>REST router</i>	138
7.53	Implementierung Login- <i>Endpoint</i>	138
7.54	<i>REST-Schnittstelle: Swagger Imports</i>	139
7.55	<i>REST-Schnittstelle: Swagger Routen</i>	139
7.56	Hauptmethode des <i>Backends</i>	139

7.57	Funktion Anzeige ändern	140
7.58	Funktion Management der <i>Cookies</i> und <i>History</i>	141
7.59	<i>Event Handling</i>	141
7.60	Signal senden	141
7.61	Importieren und Laden einer Komponente	142
7.62	Anzeige der Komponenten	143
7.63	Auffangen übergebener Variablen	143
7.64	Pfadverwaltung	144
7.65	Route für den <i>Antrag-Viewer</i>	145
7.66	Navigation zuletzt besuchte Seite	146
7.67	Unterstützung <i>Browser</i> -Pfeile	146
7.68	Laden von Neuigkeiten	147
7.69	Sitzung aktualisieren	148
7.70	Umsetzung Anmelden	149
7.71	Anmelden Weiterleitung	149
7.72	Anfrage Abmelden	150
7.73	<i>JSON Schulveranstaltung</i>	154
7.74	<i>JSON Fortbildung</i>	157
7.75	<i>JSON Sonstiges-Antrag</i>	160
7.76	Antrag einreichen	160
7.77	Belege hochladen	162
7.78	<i>Excel</i> hinzufügen	163
7.79	<i>PDF</i> erstellen und öffnen	163
7.80	Antrag speichern	164
7.81	Antrag schließen	164
7.82	Antrag löschen	165
7.83	Anfrage Rechte setzen	168
8.1	Auflistung von Verbesserungsvorschlägen	170
8.2	Auflistung von weiteren Ideen	171
8.3	Datenschnittstelle und Webseitenlogik Verbesserungsideen	172

Appendix

Abwesenheitsmeldung eines Jahrgangs



Abwesenheitsmeldung eines Jahrgangs



Allgemeine Informationen:

Jahrgang:	3BHIT	Lehrkraft:	<i>Stefan Zakall</i>
Anzahl m/w Schüler/innen:	95 / 5	Begleitpersonen:	<i>Wolfgang Fejan, Gottfried Koppensteiner, Markus Schabel</i>
Von:	<i>Montag, 12. 05. 2025 08:00</i>	Bis:	<i>Samstag, 17. 05. 2025 16:00</i>

Anmerkungen:

Die Sommersportwoche

Schulveranstaltung:

Veranstaltung:	<i>Sommersportwoche 2025</i>		
Treffpunkt:	<i>TGM, Wexstraße 19-23, 1200 Wien</i>	Uhrzeit:	<i>08:00</i>
Dauer der Veranstaltung:	<i>mehr als 3-tägig (004)</i>		

Supplierungen:

H/R/E	Jahrgang	Datum	Stunde	Saal	LK Supp.	LK Entf.	Paraphe
	3BHIT	12.05.2025	1. - 4.	H298	Stefan Zakall	Hans Brabenetz	
	3BHIT	12.05.2025	4. - 8.	H932	Stefan Zakall	Kerstin Kollitsch	

Kenntnisnahme:

Stelle	Datum	Paraphe
AV		
AV		
WL		
Begleitperson		
Ersteller/in		
UNTIS Eintragung		

Abwesenheitsmeldung eines Lehrers

Abwesenheitsmeldung
eines Lehrers**Allgemeine Informationen:**

Name:

szakall

Von:

Montag, 12.05.2025 08:00

Bis: Samstag, 17.05.2025 16:00

Anmerkungen:

*Die Sommersportwoche***Abwesenheitsgrund:**

Schulveranstaltung:

*Sommersportwoche 2025***Supplierungen:**

H/R/E	Jahrgang	Datum	Stunde	Saal	LK Supp.	LK Entf.	Paraphe
5B HIT		12.05.2025	1. - 4.	H298	Hans Brabenetz	Stefan Zakall	
5A HIT		12.05.2025	4. - 8.	H932	Kerstin Kollitsch	Stefan Zakall	

Kenntnisnahme:

Stelle	Datum	Paraphe
AV		
WL		
Ersteller/in		
UNTIS Eintragung		

Abgeltung für pädagogische Betreuung



Abgeltung für
pädagogische
Betreuung
gemäß §63a



Allgemeine Informationen:

Formular ist vom Leiter bzw. der Leiterin der Schulveranstaltung mit der Reiserechnung in der PEK abzugeben

Veranstaltung:	Sommersportwoche 2025		
Datum:	Montag, 12. 05. 2025 08:00 - Samstag, 17. 05. 2025 16:00		
Leitung:	Stefan Zakall	Verwendungsgruppe:	L1
Beginn:	Montag, 12.05.2025 08:00	Ende:	Samstag, 17.05.2025 16:00

Pädagogisch-inhaltliche Betreuung:

Name	Verwendungsgruppe	Beginn	Ende
Stefan Zakall	L1	Montag, 12.05.2025 08:00	Samstag, 17.05.2025 16:00
Wolfgang Fejan	L1	Montag, 12.05.2025 08:00	Samstag, 17.05.2025 16:00
Gottfried Koppensteiner	L1	Montag, 12.05.2025 08:00	Samstag, 17.05.2025 16:00
Markus Schabel	L1	Montag, 12.05.2025 08:00	Samstag, 17.05.2025 16:00

Datum und Unterschrift des Leiters der Schulveranstaltung

1. Dem Lehrer gebührt für die Teilnahme an mindestens zweitägigen Schulveranstaltungen mit Nächtigung, sofern er die pädagogisch-inhaltliche Betreuung einer Schülergruppe innehat, eine Abgeltung.

2. Weiters gebührt dem Leiter einer mindestens viertägigen Schulveranstaltung als Abgeltung die Einrechnung in die Lehrverpflichtung von 4.55 WE in jener Woche in der die Schulveranstaltung endet.

Reiserechnung



Reiserechnung Inland



Familienname:	Zakall	Vorname:	Stefan	Akademischer Grad:	DI	Amtsstiel:
Beginn:	12. 05. 2025 08:00	Ende:	17. 05. 2025 16:00	Reisekostenvorschuss:	0.00 €	Anzahl der Beilagen: 0
Personalnr:	94567812	Bearbeiter:		Prüfer:		Eingelangt: 19. 04. 2021 11:34
Ausgangsort:	TGM, Wexstraße 19-23, 1200 Wien		Zielort:	Karl Höneck Heim, Obersammelsdorf 17, 9122 Obersammelsdorf		

Zusätzliche Daten:

Anzahl und namentliche Angabe der Mitfahrer: Angeführte andere Reisekosten (nur gegen Beleg)

Tagessgebühr: Tarif I
5 Frühstück, 6 Mittagessen, 5 Abendessen

Nächtigungsgeld: ..

kein Anspruch
gem. § 37 RGV 55 für die Richtigkeit der Angaben:

(Datum: Unterschrift der/s Anweisungsberechtigten)

(Datum: Unterschrift der/s Rechnungslegers/in)

Berechnungsblatt										
Nr.	Tag	Beginn	Ende	Art Gebühren	Kilometer	Reisek.	Tagk.	Nachtk.	Nebenk.	Summe
1	12.05	08:00	18:00	Tagesgebühr, Nebenkosten			23.12		4.42	27.54
2	14.05	08:00	16:00	Tagesgebühr			55.66			55.66
3	15.05	08:00	17:30	Tagesgebühr, Nebenkosten			13.45		2.42	15.87
4	16.05	12:00	19:00	Tagesgebühr			13.45			13.45
				Summe:			0.00	105.68	0.00	6.84 112.52

Die sachliche Richtigkeit wird bestätigt:

Dienstreiseantrag



Dienstreiseantrag
Inland



Name:	Zakall Stefan	Akad. Grad:	DI	Amtstitel:
Personalnr.:	94567812	Reiseziel:	Karl Höneck Heim, Obersammelsdorf 17, 9122 Obersammelsdorf	
Dienstreise	Beginn:	12. 05. 2025 08:00 Uhr	Ende:	17. 05. 2025 16:00 Uhr
Dienstverrichtung	Beginn:	12. 05. 2025 08:00 Uhr	Ende:	17. 05. 2025 16:00 Uhr
Reisezweck:	Sportwoche der 3. Klassen			
Reiseart:	BUS - (Beleg erford.)			
Ausgangspunkt:	Dienststelle	Endpunkt:	Dienststelle	
Begründung:	Sportliche Weiterbildung und Projektwoche der 3. Klassen			
Sonstige Teilnehmer/innen:	Wolfgang Fejan, Gottfried Koppensteiner, Markus Schabel			
Ich bestätige, dass ich anlässlich von Dienstreisen im Rahmen personenbezogener Bonusprogramme erworbene Prämien nicht privat in Anspruch nehme.		Für die Dienstreise verwende ich auf meine Meilenkonto gutgeschriebene, dienstlich erworbene Meilen.		
Es werden keine Kosten von anderer Stelle getragen				
Sonstige Kosten:	0.00 €	Geschätzte Kosten:	150.66 €	

Antragsteller/in

Instituts-/Abteilungsleiter/in

Die vorstehend beantragte Dienstreise wird mit 01. 06. 2025 genehmigt.

Ort, Datum

Unterschrift

Eingabedatum: 19. 04. 2021

Referent/in:

Reiserechnung - Excel

Reiserechnung INLAND																																																																											
Personalaufteilung (der PH)																																																																											
<table border="1"> <tr> <td>Zl.</td> <td colspan="9"></td> </tr> <tr> <td>Eingelangt:</td> <td colspan="9">19.04.2021</td> </tr> <tr> <td colspan="10">Freilassen zum Protokollieren</td> </tr> </table>										Zl.										Eingelangt:	19.04.2021									Freilassen zum Protokollieren																																													
Zl.																																																																											
Eingelangt:	19.04.2021																																																																										
Freilassen zum Protokollieren																																																																											
<p>Geschäftsfall (Z/A) elektronisch erfasst</p> <p>Die technische Richtigkeit wird bestätigt:</p>																																																																											
<p>Zakall Stefan Vorname Akad.Grad Amtstitel</p> <p>Di</p> <p>Ausgangspunkt: TGM Wexstraße 19-23, 1200 Wien</p> <p>Zielort: Karl Hönek Heim, Obersammelsdorf 17, 9122 Obersammelsdorf</p> <p>jeweil. genauer Adress</p>																																																																											
<p>SACHBEARBEITERIN</p>																																																																											
<table border="1"> <tr> <td><input type="checkbox"/> Amtl. Businesskarte erhalten</td> <td><input type="checkbox"/> Beförderungsauszuss</td> <td><input type="checkbox"/> Ersatz für Vorreiseland (Beleg anschließen)</td> <td><input type="checkbox"/> Ersatz für Bahnfahrt 2. Kl. (nur gegen Beleg)</td> <td><input type="checkbox"/> Amtl. Kilometergeld für eigenen PKW</td> <td><input checked="" type="checkbox"/> Anzahl der km:</td> <td><input checked="" type="checkbox"/> Angelturte andre Reisekosten (nur gegen Beleg)</td> <td><input checked="" type="checkbox"/> Anzahl und namehliche Angabe dem Mitfahrer</td> <td><input checked="" type="checkbox"/> Angelturte andre Reisekosten</td> </tr> <tr> <td colspan="4"><input checked="" type="checkbox"/> Tagessgebuhr gem. § 17</td> <td colspan="4"><input type="checkbox"/> Nächtigungsgebühr</td> <td><input type="checkbox"/> Nachtrügungen ohne Nachweis</td> </tr> <tr> <td colspan="4"><input type="checkbox"/> Tagessgebuhr nach Tarif I</td> <td colspan="4"><input type="checkbox"/> Tagessgebuhr nach Tarif II zu kurzen um (Anzahl)</td> <td><input type="checkbox"/> Nachtrügungen gegen Nachweis</td> </tr> </table>										<input type="checkbox"/> Amtl. Businesskarte erhalten	<input type="checkbox"/> Beförderungsauszuss	<input type="checkbox"/> Ersatz für Vorreiseland (Beleg anschließen)	<input type="checkbox"/> Ersatz für Bahnfahrt 2. Kl. (nur gegen Beleg)	<input type="checkbox"/> Amtl. Kilometergeld für eigenen PKW	<input checked="" type="checkbox"/> Anzahl der km:	<input checked="" type="checkbox"/> Angelturte andre Reisekosten (nur gegen Beleg)	<input checked="" type="checkbox"/> Anzahl und namehliche Angabe dem Mitfahrer	<input checked="" type="checkbox"/> Angelturte andre Reisekosten	<input checked="" type="checkbox"/> Tagessgebuhr gem. § 17				<input type="checkbox"/> Nächtigungsgebühr				<input type="checkbox"/> Nachtrügungen ohne Nachweis	<input type="checkbox"/> Tagessgebuhr nach Tarif I				<input type="checkbox"/> Tagessgebuhr nach Tarif II zu kurzen um (Anzahl)				<input type="checkbox"/> Nachtrügungen gegen Nachweis																																							
<input type="checkbox"/> Amtl. Businesskarte erhalten	<input type="checkbox"/> Beförderungsauszuss	<input type="checkbox"/> Ersatz für Vorreiseland (Beleg anschließen)	<input type="checkbox"/> Ersatz für Bahnfahrt 2. Kl. (nur gegen Beleg)	<input type="checkbox"/> Amtl. Kilometergeld für eigenen PKW	<input checked="" type="checkbox"/> Anzahl der km:	<input checked="" type="checkbox"/> Angelturte andre Reisekosten (nur gegen Beleg)	<input checked="" type="checkbox"/> Anzahl und namehliche Angabe dem Mitfahrer	<input checked="" type="checkbox"/> Angelturte andre Reisekosten																																																																			
<input checked="" type="checkbox"/> Tagessgebuhr gem. § 17				<input type="checkbox"/> Nächtigungsgebühr				<input type="checkbox"/> Nachtrügungen ohne Nachweis																																																																			
<input type="checkbox"/> Tagessgebuhr nach Tarif I				<input type="checkbox"/> Tagessgebuhr nach Tarif II zu kurzen um (Anzahl)				<input type="checkbox"/> Nachtrügungen gegen Nachweis																																																																			
<p>Zusätzliche Daten</p>																																																																											
<table border="1"> <tr> <td><input type="checkbox"/> Amtl. Businesskarte erhalten</td> <td><input type="checkbox"/> Ersatz für Vorreiseland (Beleg anschließen)</td> <td><input type="checkbox"/> Ersatz für Bahnfahrt 2. Kl. (nur gegen Beleg)</td> <td><input type="checkbox"/> Amtl. Kilometergeld für eigenen PKW</td> <td><input checked="" type="checkbox"/> Anzahl der km:</td> <td><input checked="" type="checkbox"/> Angelturte andre Reisekosten (nur gegen Beleg)</td> <td><input checked="" type="checkbox"/> Anzahl und namehliche Angabe dem Mitfahrer</td> <td><input checked="" type="checkbox"/> Angelturte andre Reisekosten</td> </tr> <tr> <td colspan="4"><input checked="" type="checkbox"/> Tagessgebuhr gem. § 17</td> <td colspan="4"><input type="checkbox"/> Nächtigungsgebühr</td> <td><input type="checkbox"/> Nachtrügungen ohne Nachweis</td> </tr> <tr> <td colspan="4"><input type="checkbox"/> Tagessgebuhr nach Tarif I</td> <td colspan="4"><input type="checkbox"/> Tagessgebuhr nach Tarif II zu kurzen um (Anzahl)</td> <td><input type="checkbox"/> Nachtrügungen gegen Nachweis</td> </tr> </table>										<input type="checkbox"/> Amtl. Businesskarte erhalten	<input type="checkbox"/> Ersatz für Vorreiseland (Beleg anschließen)	<input type="checkbox"/> Ersatz für Bahnfahrt 2. Kl. (nur gegen Beleg)	<input type="checkbox"/> Amtl. Kilometergeld für eigenen PKW	<input checked="" type="checkbox"/> Anzahl der km:	<input checked="" type="checkbox"/> Angelturte andre Reisekosten (nur gegen Beleg)	<input checked="" type="checkbox"/> Anzahl und namehliche Angabe dem Mitfahrer	<input checked="" type="checkbox"/> Angelturte andre Reisekosten	<input checked="" type="checkbox"/> Tagessgebuhr gem. § 17				<input type="checkbox"/> Nächtigungsgebühr				<input type="checkbox"/> Nachtrügungen ohne Nachweis	<input type="checkbox"/> Tagessgebuhr nach Tarif I				<input type="checkbox"/> Tagessgebuhr nach Tarif II zu kurzen um (Anzahl)				<input type="checkbox"/> Nachtrügungen gegen Nachweis																																								
<input type="checkbox"/> Amtl. Businesskarte erhalten	<input type="checkbox"/> Ersatz für Vorreiseland (Beleg anschließen)	<input type="checkbox"/> Ersatz für Bahnfahrt 2. Kl. (nur gegen Beleg)	<input type="checkbox"/> Amtl. Kilometergeld für eigenen PKW	<input checked="" type="checkbox"/> Anzahl der km:	<input checked="" type="checkbox"/> Angelturte andre Reisekosten (nur gegen Beleg)	<input checked="" type="checkbox"/> Anzahl und namehliche Angabe dem Mitfahrer	<input checked="" type="checkbox"/> Angelturte andre Reisekosten																																																																				
<input checked="" type="checkbox"/> Tagessgebuhr gem. § 17				<input type="checkbox"/> Nächtigungsgebühr				<input type="checkbox"/> Nachtrügungen ohne Nachweis																																																																			
<input type="checkbox"/> Tagessgebuhr nach Tarif I				<input type="checkbox"/> Tagessgebuhr nach Tarif II zu kurzen um (Anzahl)				<input type="checkbox"/> Nachtrügungen gegen Nachweis																																																																			
<p>Berechnungsblatt:</p>																																																																											
<table border="1"> <thead> <tr> <th>Post Nr.</th> <th>Tag</th> <th>Beginn</th> <th>Ende</th> <th>Art des Gebührenanspruches</th> <th>Gesamt Kilometer</th> <th>Reisekosten in Euro und Cent</th> <th>Tagessgebühr in Euro und Cent</th> <th>Nächtigungsgebühr in Euro und Cent</th> <th>Sonstige Nebenkosten in Euro und Cent</th> <th>Summe in Euro und Cent</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>12.05</td> <td>08:00</td> <td>18:00</td> <td>Tagesgebühr, Sonstige Nebenkosten</td> <td></td> <td>23,12</td> <td></td> <td></td> <td>4,42</td> <td>27,54</td> </tr> <tr> <td>2</td> <td>14.05</td> <td>08:00</td> <td>16:00</td> <td>Tagesgebühr</td> <td></td> <td>55,66</td> <td></td> <td></td> <td></td> <td>55,66</td> </tr> <tr> <td>3</td> <td>15.05</td> <td>08:00</td> <td>17:30</td> <td>Tagesgebühr, Sonstige Nebenkosten</td> <td></td> <td>13,45</td> <td></td> <td></td> <td>2,42</td> <td>15,87</td> </tr> <tr> <td>4</td> <td>16.05</td> <td>12:00</td> <td>19:00</td> <td>Tagesgebühr</td> <td></td> <td>13,45</td> <td></td> <td></td> <td></td> <td>13,45</td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> <td>SUMME:</td> <td></td> <td>105,68</td> <td></td> <td></td> <td>6,84</td> <td>112,52</td> </tr> </tbody> </table>										Post Nr.	Tag	Beginn	Ende	Art des Gebührenanspruches	Gesamt Kilometer	Reisekosten in Euro und Cent	Tagessgebühr in Euro und Cent	Nächtigungsgebühr in Euro und Cent	Sonstige Nebenkosten in Euro und Cent	Summe in Euro und Cent	1	12.05	08:00	18:00	Tagesgebühr, Sonstige Nebenkosten		23,12			4,42	27,54	2	14.05	08:00	16:00	Tagesgebühr		55,66				55,66	3	15.05	08:00	17:30	Tagesgebühr, Sonstige Nebenkosten		13,45			2,42	15,87	4	16.05	12:00	19:00	Tagesgebühr		13,45				13,45					SUMME:		105,68			6,84	112,52
Post Nr.	Tag	Beginn	Ende	Art des Gebührenanspruches	Gesamt Kilometer	Reisekosten in Euro und Cent	Tagessgebühr in Euro und Cent	Nächtigungsgebühr in Euro und Cent	Sonstige Nebenkosten in Euro und Cent	Summe in Euro und Cent																																																																	
1	12.05	08:00	18:00	Tagesgebühr, Sonstige Nebenkosten		23,12			4,42	27,54																																																																	
2	14.05	08:00	16:00	Tagesgebühr		55,66				55,66																																																																	
3	15.05	08:00	17:30	Tagesgebühr, Sonstige Nebenkosten		13,45			2,42	15,87																																																																	
4	16.05	12:00	19:00	Tagesgebühr		13,45				13,45																																																																	
				SUMME:		105,68			6,84	112,52																																																																	
<p>Die sachliche Richtigkeit wird bestätigt:</p>																																																																											
<p>(Datum, Unterschrift der/s Anweisungsberechtigten)</p>																																																																											
<p>(Datum, Unterschrift der Rechnungslegern bzw. des Rechnungsgebers)</p>																																																																											

Dienstreiseantrag - Excel

tgm - Schule der Technik		D I E N S T R E I S E A N T R A G - I N L A N D						
Dienststelle								
Zakall	Stefan	DI		AMTSTITEL	TELEFON-NR. / Klappe			
FAMILIEN- OD. NACHNAME	VORNAME	AKAD.GRAD						
PERSONALNUMMER	9 4 5 6 7 8 1 2	VGr.	*EGr.	DKI.	GSt.	*EST.	/ Gebührenstufe	
DIENSTREISE	BEGINN:	12.05.2025	/ 08:00	(Datum / Uhrzeit)	ENDE:	17.05.2025	/ 16:00	
DIENSTVERRICHTUNG	BEGINN:	12.05.2025	/ 08:00	(Datum / Uhrzeit)	ENDE:	17.05.2025	/ 16:00	
REISEZIEL: Karl Hönck Heim, Obersammelsdorf 17, 9122 Obersammelsdorf								
REISEZWECK UND BEGRÜNDUNG DER NOTWENDIGKEIT:								
Sportwoche der 3. Klassen								
<input type="checkbox"/> Amtl. BUSINESSKARTE 2. KL. <input type="checkbox"/> BEFÖRDERUNGZUSCHUSS <input type="checkbox"/> BAHN 2. KL. - nur gegen Beleg <input type="checkbox"/> SCHLAFWAGEN <input type="checkbox"/> MITFAHRER <input type="checkbox"/> FLUG <input type="checkbox"/> BILLIGFLUG <input checked="" type="checkbox"/> BUS - nur gegen Beleg <input type="checkbox"/> BUSINESSKARTE / BAHNVERRECHNUNG 1. KL. - nur wenn in Dienstinteresse gelegen - Begründung erforderlich <input type="checkbox"/> EIGENER PKW - Begründung erforderlich								
Ausgangspunkt	<input checked="" type="checkbox"/> Dienststelle	Endpunkt	<input checked="" type="checkbox"/> Dienststelle					
	<input type="checkbox"/> Wohnung*		<input type="checkbox"/> Wohnung*					
*nur wenn dadurch niedrigere Reisegebühren anfallen								
BEGRÜNDUNG:								
Sportliche Weiterbildung und Projektwoche der 3. Klassen								
SONSTIGE TEILNEHMER / INNEN:								
Wolfgang Fejan, Gottfried Koppensteiner, Markus Schabel								
<input checked="" type="checkbox"/> Ich bestätige, dass ich anlässlich von Dienstreisen im Rahmen personenbezogener Bonusprogramme erworbene Prämien nicht privat in Anspruch nehme. "BONUS-MEILEN": <input checked="" type="checkbox"/> Für die Dienstreise verwende ich auf meinem Meilenkonto gutgeschriebene, dienstlich erworbene Meilen.								
Werden	Reisekosten	-	Aufenthaltskosten	von anderen Stellen getragen?				
<input checked="" type="checkbox"/> NEIN	<input type="checkbox"/> JA		<input checked="" type="checkbox"/> NEIN	<input type="checkbox"/> JA	von _____			
Sonstige Kosten:	0			Geschätzte Kosten: 150,66				
(z.B. Tagungsgebühr, Eintrittskarten, usw.)								
Antragsteller/in (Datum/Unterschrift)				Instituts-/Abteilungsleiter/in (Bestätigung der Notwendigkeit)				
Die vorstehend beantragte Dienstreise wird mit 01.06.2025 genehmigt.								
Ort, Datum				Rektor/in				
Eingabedatum: 19.04.2021								
Referent / in: _____								
Businesskarte ausgeføgt: Hinfahrt <input type="checkbox"/> Rückfahrt <input type="checkbox"/>								