STATE OF THE ART

REFUNDABLE

effiziente Reise- und Exkursions-verwaltung für Schulen

Dehner Linus, Foster Ryan, Beier Michael

tgm Die Schule der Technik

JUST DO IT
HÖHERE ABTEILUNG FÜR INFORMATIONSTECHNOLOGIE

Version	Autor	QS	Datum	Status	Kommentare
0.1	ldehner	mbeier	2020-09-24	Draft	Create
1	mbeier	ldehner	2020-10-24	Draft	Backend - Überblick
2	mbeier	ldehner	2020-11-08	Draft	Layout finalisiert
3	mbeier	ldehner	2020-11-08	Draft	Backend - Docker
4	ldehner	mbeier	2020-11-09	Draft	Webdesign - Überblick

Inhaltsverzeichnis

1 Projektleitung & Frontend - responsives Webdesign

1.1 Überblick

In diesem Kapitel werden die Anforderungen des Designs des Frontends geschildert. Da es mehrere Möglichkeiten gibt das Frontend zu realisieren werden hier drei wesentliche Methoden verglichen. Die Variante HTML, CSS und JavaScript zu verwenden. Die zweite Methode die zum Vergleich hergezogen wird ist statt Vanilla CSS SASS zu verwenden. Die dritte und auch letzte Methode in diesem Vergleich ist, dass die Arbeit durch die Verwendung eines CSS Frameworks vereinfacht wird. Anschließend werden die drei verschiedenen Methoden verglichen und die, die am Besten für unser Projekt Refundable passt ausgewählt. Zu guter Letzt wird das Design für die Zielgruppe analysiert, da Refundable explizit für Lehrer codiert wird und diese sich möglichst gut und schnell auf der Website zurecht finden sollen.

1.2 HTML, CSS, JS

Der eigentliche Standard HTML5 wird in der Praxis meist als Überbegriff für HTML, CSS und JS verwendet. In diesem Unterkapiteln beschäftigen wir uns genau damit.[**html5-css3-handbuch**] In den folgenden Kapiteln wird erklärt wozu HTML, CSS und JS da sind und welche Funktionalitäten sie bieten.

1.2.1 HTML

HTML ist eine Auszeichnungssprache steht für *Hypertext Markup Language* und wurde 1989 von dem britischen Informatiker Tim Burners-Kee veröffentlicht.
„Hypertext bezeichnet die Möglichkeit, Texte mit Hilfe von Hyperlinks oder kurz Links miteinander zu verbinden.“[**html5-css3-def**]
Dies heißt, dass man mittels Hypertexts auf der Seite beliebig Herumspringen kann. Auszeichnungssprachen werden im Fachjargon auch als Markup Language (ML) bezeichnet. 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 wären:

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

HTML ist hauptsächlich dafür da, um Texte, Grafiken und Hyperlinks (Links) darzustellen. Die Bearbeitung von HTML-Dokumenten ist relativ einfach und unkompliziert, da es eine reine 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:

```
<tagname>Tag Inhalt</tagname>
<einzeltag attribut="123">
```

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 Website angegeben und ein *meta-tag*, in diesem werden Meta-Informationen angegeben. In den *Body-Tag* kommen wiederum Tags, die den Inhalt der Seite beinhalten.

```
<!DOCTYPE html>
<html lang="de">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width,
        ↪  initial-scale=1.0">
        <title>Kurzes BSP</title>
    </head>
    <body>
        <h1>Überschrift</h1>
        <button>Drück mich!</button>
    </body>
</html>
```

Wenn man die Datei nun im Browser öffnet schaut dies wie folgt aus:

Abbildung 1: Beispiel einer HTML Seite mit einer Überschrift und einem Button

Da HTML nur für die Grundstruktur einer Website gedacht ist, also zum beschreiben der Struktur und des Inhalts schaut die Seite noch nicht sonderlich ansprechend aus. Um das zu verändern ist CSS benötigt. [**auszeichnungssprachen**] [**html5-css3-handbuch**] [**html5-css3-def**]

1.2.2 CSS

Cascading Stylesheets oder kurz CSS ist wie der Name schon sagt für den *Style*, also für das Aussehen der Website verantwortlich. Da HTML Anfangs nur im Printbereich verbreitet war, war es nicht notwendig die Seiten zu gestalten. 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 ist unter anderem folgende Dinge möglich:

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

Man kann CSS auf verschiedene Arten in HTML benützen. Als Beispiel werden wir eine Überschrift in die Mitte der Website setzen, die Schriftgröße auf 40pt stellen und die Schriftart auf sans-serif ändern.
Entweder man fügt 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 werden:

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

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 und diese im CSS auswählen und verändern:

```
<html lang="de">
    <head>
        <style>
        //Für das ganze Element
        h1 {
            text-align: center;
            font-size: 40pt;
            font-family: sans-serif;
        }

        //Für IDs
        #ueberschrift1 {
            text-align: center;
            font-size: 40pt;
            font-family: sans-serif;
        }

        //Für Klassen
        .ueberschriftenGruppe {
            text-align: center;
            font-size: 40pt;
            font-family: sans-serif;
        }
        </style>
    </head>
</html>
```

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

```
<head>
    <link rel="stylesheet" href="file.css" type="text/css">
</head>
```

Wie man erkennen kann ist dem ganzen keine Ende gesetzt und mit viel Aufwand kann man so ziemlich alles verändern. Um den Aufwand jedoch gering zu halten sind SASS und CSS Frameworks da, zu welchen wir später kommen. **[html5-css3-def]** **[html5-css3-handbuch]**

1.2.3 JS

JavaScript wird nur sehr kurz besprochen, da es für diesen Teil keine große Rolle spielt. Es wird verwendet um den Elementen Funktionen zu geben, also 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 brauch man JS fast nur, wenn man nur mit CSS oder SASS arbeitet, wenn man ein Framework verwendet haben diese meist ein JavaScript Framework inkludiert und man muss fast kein JavaScript mehr verwenden.

1.3 SASS

1.4 CSS Frameworks

CSS Frameworks

1.4.1 Bootstrap

1.4.2 Materialize

1.4.3 ZURB Foundation

1.4.4 Tailwind CSS

Tailwind CSS

1.5 Vergleich

1.6 Zielgruppenorientiertes Design

1.7 Fragestellungen

2 Frontend - Webapplikation als REST-Client

2.1 Überblick

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 2 verschiedene Entwurfsmuster angeschaut und verglichen. Umgesetzt wird dann ein Entwurfsmuster mithilfe von JavaScript. Hier kann ein JavaScript Framework zum Einsatz kommen. Dazu werden hier verschiedene JavaScript Frameworks angeschaut und verglichen. Für die Verarbeitung der Daten ist es wichtig Datenformate festzulegen. Die Aufbereitung der Elemente fürs Frontend mit den Daten des Backends wird ebenfalls angeschaut.

2.2 Design-Patterns

Dieser Teil des Projektes wird in Verwendung eines Entwurfsmusters umgesetzt. Zwei Entwurfsmusters werden hierbei in Betrachtung gezogen. Zum einen MVVM und zum anderen MVC. Es kommen diese zwei Entwurfsmuster in Frage, da MVC in sehr bekanntes Entwurfsmuster ist und MVVM eine neuere Variante von MVC ist.

2.2.1 MVC

MVC ist ein Entwurfsmuster mit dem eine Software in die drei Teile (Model, View und Controller) geteilt wird.

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, die Benutzer*innen sehen und mit denen sie interagieren. Dieser Teil der Software beinhaltet keine wichtigen Daten oder Funcktionen, welche Daten bearbeiten. Sie hört nur auf Benutzereingaben und zeigt die bereitgestellten Daten an.

Der Controller ist der Teil der Software, der Model und View verbindet. In dem Controller wird auf die Benutzereingaben reagiert und die richtigen Funktionen aus dem Model aufgerufen.[**mvc**]

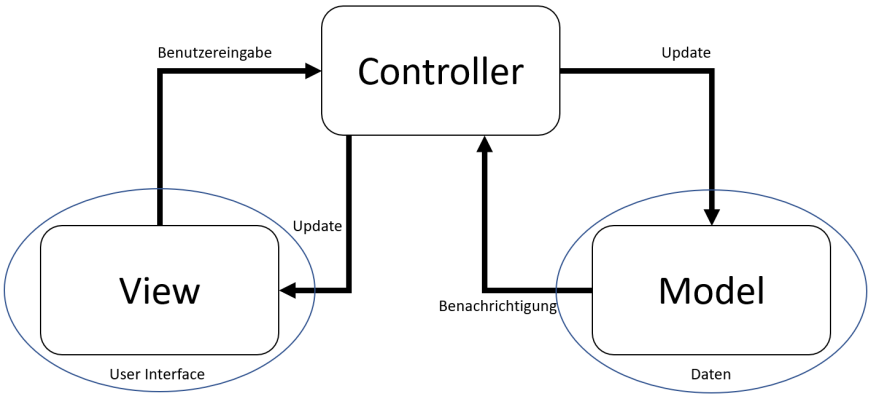

Abbildung 2: Übersicht über die Komponenten des MVC-Patterns und ihre Zusammenhänge

2.2.2 MVVM

MVVM ist ein Entwurfsmuster mit dem eine Software in drei Teile geteilt wird. 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 der Benutzer*innen interagieren.

Das ViewModel ist ein Bindeglied zwischen Model und View. Dabei stellt es der View offen 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_vue**]

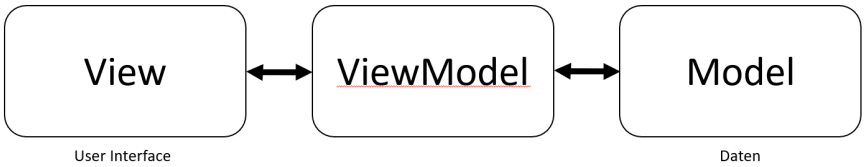

Abbildung 3: Übersicht über die Komponenten des MVVM-Patterns und ihre Zusammenhänge

3 Backend - REST-Schnittstelle und Infrastruktur

3.1 Überblick

Das Backend besteht aus mehreren Komponenten. Einerseits muss eine gewisse Software-Infrastruktur aufgebaut werden, um Webinterface und die REST-Schnittstelle bereitzustellen. Andererseits muss die Anwendung selbst auch entwickelt werden. Diese besteht wiederum auch aus mehreren Teilen. Darunter fällt die REST-Schnittstelle, inklusive der implementierten Endpoints, selbst, Schnittstellen zu diversen Diensten, wie dem TGM-LDAP Server, zur Datenbank, zu Google Maps und zu WebUntis, aber auch die weitere Funktionalität der Anwendung, unter anderem das Versenden von E-Mails oder Erstellen von PDF-Dateien.

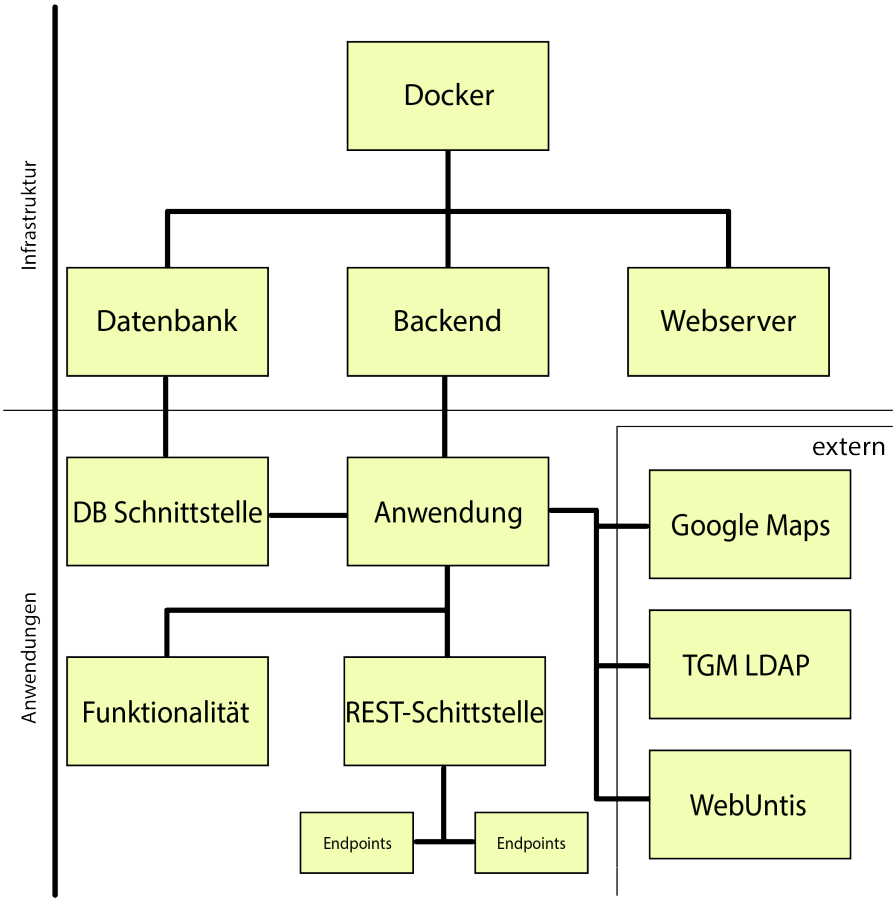

Abbildung 4: Übersicht über die verschiedenen Komponenten der Infrastruktur und der Anwendung

3.2 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 die der 1. Abbildung (Übersicht) entspricht. Für diese sind folgende Container vorgesehen, die in den nächsten Kapiteln noch ins Detail beschrieben werden.

3.2.1 Datenbank

Um die Daten, die durch Refundable erhoben und generiert werden, zu speichern, benötigen wir eine Datenbank (DB). 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:

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

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 vorrangig auf die Beziehung zwischen einzelnen Elementen geschaut. Daten werden hierbei 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 DB-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.[**nosqltypes**]
Als Datenbankmanagementsystem (DBMS) kommt bei dieser Auswahl einige Software

in Frage. Die am meisten verbreitete Software hier ist MongoDB und CouchDB. 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 DBMS verwendet.[**mongo**]

3.2.2 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 alpine-Image genutzt. Dieses stellt eine sehr sparsame Linux-Instanz dar, die in einem eigenem Container laufen kann. Um diesen Container noch entsprechend anzupassen, wird ein entsprechendes Docker-Image über ein Dockerfile gebaut.[**alpine**]

3.2.3 Webserver

Als Webserver um das Webinterface aufrufbar und verfügbar zu machen wird ein Apache2 Server genutzt. 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.[**apache**]

3.3 Deployment

3.4 REST-Schnittstelle

3.4.1 Framework

3.4.2 Endpoints

3.5 Funktionalität

3.5.1 TGM-LDAP Schnittstelle

3.5.2 Datenbank Schnittstelle

3.5.3 Google Maps

3.5.4 WebUntis

3.5.5 E-Mails

3.5.6 PDF-Dateien

3.6 Kommunikation und Datenformate

4 Fazit

Abbildungsverzeichnis