

# 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	Idehner	mbeier	2020-09-24	Draft	Create
1	mbeier	Idehner	2020-10-24	Draft	Backend - Überblick
2	mbeier	Idehner	2020-11-08	Draft	Layout finalisiert
3	mbeier	Idehner	2020-11-08	Draft	Backend - Docker
4	Idehner	mbeier	2020-11-09	Draft	Webdesign - Überblick

## Inhaltsverzeichnis

<b>1</b>	<b>Projektleitung &amp; Frontend - responsives Webdesign</b>	<b>4</b>
1.1	Überblick . . . . .	4
1.2	HTML, CSS, JS . . . . .	4
1.2.1	HTML . . . . .	4
1.2.2	Vanilla CSS . . . . .	6
1.2.3	JS . . . . .	6
1.3	SASS . . . . .	6
1.4	CSS Frameworks . . . . .	6
1.4.1	Bootstrap . . . . .	6
1.4.2	Materialize . . . . .	6
1.4.3	ZURB Foundation . . . . .	6
1.4.4	Tailwind CSS . . . . .	6
1.5	Vergleich . . . . .	6
1.6	Zielgruppenorientiertes Design . . . . .	6
1.7	Fragestellungen . . . . .	6
<b>2</b>	<b>Frontend - Webapplikation als REST-Client</b>	<b>7</b>
2.1	Überblick . . . . .	7
2.2	Design-Patterns . . . . .	7
2.2.1	MVC . . . . .	7
2.2.2	MVVM . . . . .	8
2.2.3	Vergleich . . . . .	9
2.3	Datenformate . . . . .	9
2.4	Umsetzungsmöglichkeiten . . . . .	9
2.4.1	Vue . . . . .	9
2.4.2	React . . . . .	9
2.4.3	Angular . . . . .	9
2.4.4	Ohne Framework . . . . .	9
2.4.5	Vergleich . . . . .	9
2.5	Aufbereitung der Daten . . . . .	9
2.6	Fragestellung . . . . .	9
<b>3</b>	<b>Backend - REST-Schnittstelle und Infrastruktur</b>	<b>10</b>
3.1	Überblick . . . . .	10
3.2	Docker . . . . .	11
3.2.1	Datenbank . . . . .	11
3.2.2	Backend-Container . . . . .	12
3.2.3	Webserver . . . . .	12
3.3	Deployment . . . . .	12
3.4	REST-Schnittstelle . . . . .	12
3.4.1	Framework . . . . .	12
3.4.2	Endpoints . . . . .	12

---

3.5	Funktionalität . . . . .	12
3.5.1	TGM-LDAP Schnittstelle . . . . .	12
3.5.2	Datenbank Schnittstelle . . . . .	12
3.5.3	Google Maps . . . . .	12
3.5.4	WebUntis . . . . .	12
3.5.5	E-Mails . . . . .	12
3.5.6	PDF-Dateien . . . . .	12
3.6	Kommunikation und Datenformate . . . . .	12
<b>4</b>	<b>Fazit</b>	<b>13</b>
	<b>Glossar</b>	<b>14</b>
	<b>Akronyme</b>	<b>15</b>
	<b>Literaturverzeichnis</b>	<b>16</b>
	<b>Abbildungsverzeichnis</b>	<b>17</b>

# 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.[13] 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.“[3]*

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:

```
1      <tagname>Tag Inhalt</tagname>
2      <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.

```
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>
```

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

## Überschrift

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. [8] [13] [3]

### **1.2.2 Vanilla CSS**

[Vanilla CSS](#)

### **1.2.3 JS**

## **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 **Entwurfsmuster** 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 Funktionen, 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.[7]



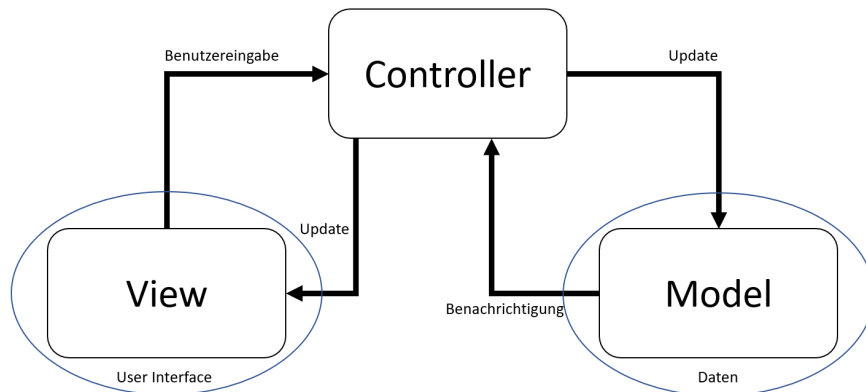


Abbildung 2: Übersicht über die Komponenten des MVC-Patterns und ihre Zusammenhänge

### 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.[11]

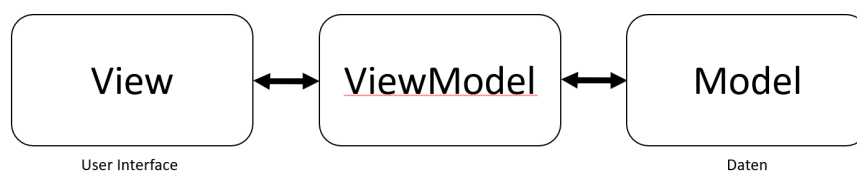


Abbildung 3: Übersicht über die Komponenten des MVVM-Patterns und ihre Zusammenhänge

### **2.2.3 Vergleich**

## **2.3 Datenformate**

## **2.4 Umsetzungsmöglichkeiten**

### **2.4.1 Vue**

### **2.4.2 React**

### **2.4.3 Angular**

### **2.4.4 Ohne Framework**

### **2.4.5 Vergleich**

## **2.5 Aufbereitung der Daten**

## **2.6 Fragestellung**

### 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.

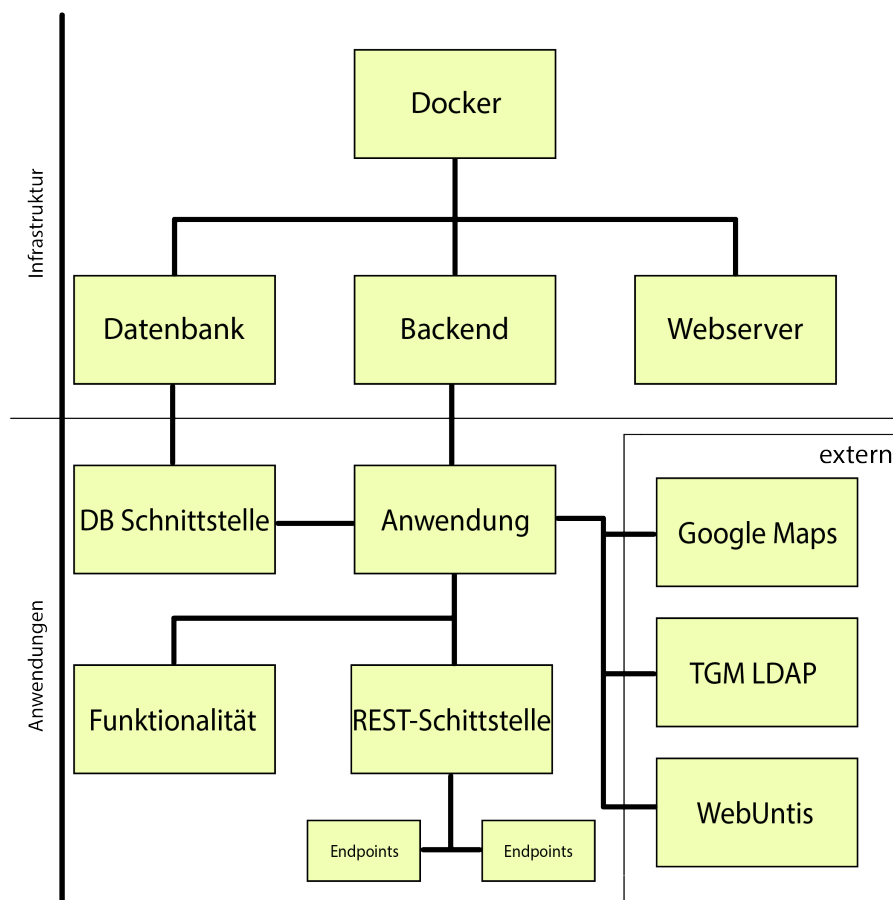


Abbildung 4: Übersicht über die verschiedenen Komponenten der Infrastruktur und der Anwendung

## 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.<sup>[12]</sup>

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.[9]

### 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.[1]

### 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.[6]

## 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**

## Glossar

**Auszeichnungssprache** Vereinfacht gesagt, eine Sprache die von verschiedenen Programmen lesbar ist und für die Gliederung und Formatierung von jeglichen Daten zuständig ist.[8]. 4

**Backend** Das Backend ist die Funktionalität im Hintergrund des Frontends.. 7

**CSS** Cascading Style Sheet. Zum umgestalten und verschönern der Weboberfläche.. 4, 6

**Docker** Docker ist eine Software, welche es ermöglicht Programme in einer abgeschnittenen Umgebung (genannt Container) laufen zu lassen. Das Erstellen dieser Umgebung und das Installieren und Laufen des Programms darin, gestaltet sich hierbei sehr einfach.[5]. 11

**Docker Compose** Docker Compose ist eine Erweiterung von Docker. 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.[10]. 11

**Entwurfsmuster** Eine Vorlage, wie man ein Programm oder Software intern strukturiert.. 7, 8

**Framework** Kann als Baukasten gesehen werden. Bietet Möglichkeiten um die normalen Vorhergehensweisen zu kürzen bzw. vereinfachen.. 4, 6

**Frontend** Das Frontend ist die Oberfläche einer Website - also das was zu sehen ist.. 4, 7

**HTML** Hypertext Markup Language. Die Grundbausteine bzw. Struktur der Weboberfläche. Zum Beispiel Text oder eine Eingabefläche.. 4

**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.. 12

**JavaScript** Java Script. Bietet einen Rahmen an Funktionalität und Animationen in Verbindung mit HTML und CSS.. 4, 7

**JavaScript Framework** Ein Framework für JavaScript.. 7

**JSON** Bei JSON (JavaScript Object Notation) handelt es sich um ein kompaktes textbasiertes Datenformat, welches für den Datenaustausch zwischen Schnittstellen entwickelt wurde.[4]. 11

**Konsistenz** Konsistenz bedeutet die Einhaltung von Regeln. Im Zusammenhang mit Daten in einer Datenbank ist gemeint, dass die Daten die im DBMS gespeichert sind, eingehaltet werden müssen.. [12](#)

**MVC** MVC (Model-View-Controller) ist ein Entwurfsmuster, welches die Logik eines Programms von dem Interface(View) trennt.. [7, 8](#)

**MVVM** MVVM (Model-View-ViewModel) ist ein Entwurfsmuster, welches eine Variante des [MVC](#)-Patterns ist.. [7, 8](#)

**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.. [11](#)

**relationale Datenbank** Datenbank, wo ein Typ von Daten durch eine Tabelle repräsentiert wird. Die Anzahl der Spalten ist hierbei für jeden Datensatz konstant.. [11](#)

**SASS** tba. [4](#)

**Tag** Markiert in HTML ein HTML-Element. Auch als Marken bezeichnet. . [5](#)

**Vanilla** Ein anderer Begriff für Basisausführung.. [4, 6](#)

**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.. [12](#)

**Webinterface** Ein Web Interface ist ein System, durch welches Anwender mit dem Netz interagieren. Der Begriff Web Interface steht zumeist für grafische Oberflächen.. [10](#)

**XML** XML (Extensible Markup Language) ist eine Markup-Sprache, die zur Beschreibung von Daten genutzt wird. Wobei die Datenstruktur von XML über ein Schema verifiziert werden kann.[2]. [11](#)

**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.. [11](#)



## Akronyme

**DB** Datenbank. 11

**DBMS** Datenbankmanagementsystem. 11, 12

## Literaturverzeichnis

- [1] *alpine - Docker Hub*. URL: [https://hub.docker.com/\\_/alpine](https://hub.docker.com/_/alpine) (besucht am 11. 11. 2020).
- [2] 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).
- [3] 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](https://doi.org/10.1007/978-3-662-53916-3).
- [4] 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>.
- [5] *Docker Engine overview | Docker Documentation*. URL: <https://docs.docker.com/engine/> (besucht am 08. 11. 2020).
- [6] *httpd - Docker Hub*. URL: [https://hub.docker.com/\\_/httpd](https://hub.docker.com/_/httpd) (besucht am 11. 11. 2020).
- [7] 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](https://doi.org/10.1109/EDOC.2001.950428).
- [8] *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).
- [9] *mongo - Docker Hub*. URL: [https://hub.docker.com/\\_/mongo](https://hub.docker.com/_/mongo) (besucht am 11. 11. 2020).
- [10] *Overview of Docker Compose | Docker Documentation*. URL: <https://docs.docker.com/compose/> (besucht am 08. 11. 2020).
- [11] Ralph Steyer. *Webanwendungen erstellen mit Vue.js : MVVM-Muster für konventionelle und Single-Page-Webseiten*. ger. 1st ed. 20. Wiesbaden: Springer Fachmedien Wiesbaden Imprint: Springer Vieweg, 2019. ISBN: 3658271701. DOI: [10.1007/978-3-658-27170-1](https://doi.org/10.1007/978-3-658-27170-1).
- [12] *Types of NoSQL Databases | MongoDB*. URL: <https://www.mongodb.com/scale/types-of-nosql-databases> (besucht am 11. 11. 2020).

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

## Abbildungsverzeichnis

1	HTML Beispielseite . . . . .	5
2	Übersicht des MVC-Patterns . . . . .	8
3	Übersicht des MVVM-Patterns . . . . .	8
4	Übersicht über die Komponenten . . . . .	10