

# **OpenTasks**

**Eine Team-basierte Web-Applikation zur  
Stoff-Vertiefung**

## **Bachelor-Thesis**

**zur Erlangung des akademischen Grades B.Sc.**

**Patrick Hilgenstock**

**2203656**



Hochschule für Angewandte Wissenschaften Hamburg  
Fakultät Design, Medien und Information  
Department Medientechnik

Erstprüfer: Prof. Dr. Edmund Weitz

Zweitprüfer: Prof. Dr Torsten Edeler

Hamburg, 20.02.2017

# Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>4</b>
1.1	Zielsetzung . . . . .	4
1.2	Bereits existierende Software . . . . .	4
1.3	Vorteile der neuen Lösung . . . . .	5
<b>2</b>	<b>Konzeption der Anwendung</b>	<b>6</b>
2.1	Analyse des Anwendungsfeldes . . . . .	6
2.2	Was ist eine Aufgabe . . . . .	6
2.3	Verwendete Technologien . . . . .	7
2.3.1	MariaDB . . . . .	7
2.3.2	H2 . . . . .	8
2.3.3	Dependency-Injection . . . . .	8
2.3.4	Spring(-Boot) . . . . .	10
2.3.5	Spring Data JPA . . . . .	12
2.3.6	Bootstrap . . . . .	15
2.3.7	Typescript / Angular 2 . . . . .	15
2.3.8	SocketIO . . . . .	17
2.3.9	JSON Web Tokens . . . . .	18
2.3.10	MathJax . . . . .	18
<b>3</b>	<b>Ein Rundgang durch das Frontend</b>	<b>19</b>
3.1	Die Ansicht des Nutzers . . . . .	19
3.1.1	Die Auswahl des Teams . . . . .	19
3.1.2	Erhalten und bearbeiten der Aufgaben . . . . .	20
3.1.3	Die Eingabeformulare . . . . .	21
3.1.3.1	Das Textformular . . . . .	22
3.1.3.2	Das Zahlen / Dezimalformular . . . . .	22
3.1.3.3	Das Matrixformular . . . . .	23
3.1.3.4	Das Bruchformular . . . . .	24
3.2	Die Sicht des Administrators . . . . .	25
3.2.1	Übersicht über die laufende und letzte Aufgabe . . . . .	25
3.2.2	Starten eines neuen Aufgaben-Generators . . . . .	26
3.2.3	Bearbeitung der Helper-Funktionen . . . . .	28
3.2.4	Bearbeitung der Aufgaben-Generatoren . . . . .	30
3.2.5	Der Fehler-Log . . . . .	32

<b>4</b>	<b>Ein Blick unter die Haube - das Backend</b>	<b>34</b>
4.1	Die Sicherung der REST-API . . . . .	34
4.1.1	Authentifizierung und Rollen-System . . . . .	34
4.1.2	Die Sicherung der Aufgaben-Generierung . . . . .	35
4.2	Dokumentation der REST-Schnittstellen . . . . .	35
4.2.1	Admin . . . . .	35
4.3	Die Generierung und Validierung der Aufgaben . . . . .	36
<b>5</b>	<b>Fazit</b>	<b>38</b>
5.1	Das Ergebnis . . . . .	38
5.2	Wie kann die Anwendung verbessert / erweitert werden . . . . .	38
	<b>Abbildungsverzeichnis</b>	<b>39</b>
	<b>Literaturverzeichnis</b>	<b>40</b>

# 1 Einleitung

## 1.1 Zielsetzung

Das Ziel dieser Arbeit ist es eine Webapplikation zu erstellen, welche bei der Gestaltung einer Mathematik-Vorlesung helfen soll. Dabei werden von der Anwendung Aufgaben generiert und an den Nutzer gesendet. Dieser kann diese nun lösen und seine Antwort abschicken. Nach der Validierung bekommt er ein Feedback und bei dem korrekten Lösen eine neue Aufgabe, sowie einige Punkte für sein Team. Der Administrator kann dabei den ganzen Vorgang überwachen und den aktuellen Punktestand einsehen.

Einer der Kernpunkte der Anwendung wird es sein, dass dynamisch Aufgaben generiert werden. Das bedeutet, dass wenn der Nutzer mehrere Aufgaben anfordert er jedes mal eine Aufgabe bekommt welche zwar aus dem selben Themenbereich kommen aber doch jedes mal unterschiedliche Variablen beinhalten und so ein erneutes Berechnen erfordern.

## 1.2 Bereits existierende Software

Wenn man sich die bereits existierenden Lösungen für dieses Problem anschaut stößt man auf sehr viele Programme die sich mit dem Vertiefen von Stoff befassen. Allerdings setzen diese lediglich darauf den Nutzer Aufgaben zu stellen und die Aufgaben zu validieren. Es gibt keinerlei Möglichkeit die vorhandenen Aufgaben zu erweitern oder die Ergebnisse mit anderen Leuten zu teilen, geschweige denn automatisch eine Übersicht über die Ergebnisse eines ganzen Kurses zu erschaffen. Ebenfalls als Nachteil aufzufassen ist hier die Tatsache, dass es sich bei fast allen Lösungen um Programme handelt welche käuflich zu erwerben sind

Als Variante zur Validierung der Ergebnisse eines ganzen Kurses besteht die Webanwendung “MARS” (Minimal Audience Response System). Dieses ist sehr gut dafür geeignet statische Fragen zu stellen und zu validieren. Allerdings ist es hier nicht möglich dynamisch Aufgaben zu erstellen. Zusätzlich kann nur ein einziges Mal eine Antwort abgegeben werden, es ist also nicht möglich mehrere Fragen des selben Bereiches zu beantworten und so sein Wissen intensiver zu testen.

Ansonsten gibt es bereits viele Webseiten die sich mit dem Stellen von Aufgaben aus verschiedenen Themenbereichen kümmern, zum Beispiel RegexGolf ( [https:](https://www.regexgolf.com/)

[//alf.nu/RegexGolf](http://alf.nu/RegexGolf) ). Hierbei handelt es sich um eine Anwendung bei der dem Nutzer eine Liste von Wörtern gegeben wird. Die Aufgabe besteht nun darin einen regulären Ausdruck zu finden welcher für einen Teil der Liste matcht und bei der anderen nicht.

Es gibt allerdings auch Projekte welche sich in der Kategorie aufhalten in die OpenTasks fallen wird. Darunter fallen zum Beispiel Maple TA <sup>1</sup>, WebWork <sup>2</sup> und Moodle <sup>3</sup>

Bei Maple TA handelt es sich um ein System welches alle Funktionalitäten abdeckt die gewünscht sind und dazu noch von einem vollwertigen Linear Algebra System unterstützt wird das für Berechnungen und Aufgabenstellungen zur Verfügung steht. Durch den großen Umfang an Funktionalität steigert allerdings auch die Komplexität des Programms enorm. Zusätzlich handelt es sich bei Maple TA um kostenpflichtige Software.

Auf WeBwork und Moodle wird hier nicht näher eingegangen, gesagt sei lediglich, dass es sich bei WebWork um FreeWare handelt (die allerdings selber gehostet werden muss), während Moodle kostenpflichtig sind.

Bei WeBWork, Maple TA, sowie Moodle handelt es sich um Anwendungen mit relativ hohem Lernaufwand für den Administrator, was eine schnelle Nutzung nur schwer möglich macht.

### 1.3 Vorteile der neuen Lösung

Bei der neuen Anwendung soll es sich um ein Projekt handelt welches alle Vorteile existierender Programme vereint, während es die Nachteile auslässt. Das bedeutet, dass die Anwendung viel Wert darauf legt erweiterbar zu sein, ein möglichst großes Spektrum an Aufgaben abdecken zu können und dynamisch Aufgaben zu generieren können.

Dabei ist es wichtig, dass die Anwendung nicht zu komplex wird. Sie soll für einen neuen Nutzer so intuitiv wie möglich verständlich sein. Zusätzlich soll die Installation sehr einfach sein. Im Zentrum steht die Idee des "Herunterladen und direkt starten". Gleichzeitig wird es sich bei der resultierenden Webapplikation um Open Source handeln, das heißt der Quellcode ist frei verfügbar und kostenlos, ebenso wie die Webapplikation selbst.

---

<sup>1</sup><http://www.maplesoft.com/products/mapleta/>

<sup>2</sup><http://webwork.maa.org>

<sup>3</sup><https://moodle.de/>

## 2 Konzeption der Anwendung

### 2.1 Analyse des Anwendungsfeldes

Bevor man sich nun an die Entwicklung der Anwendung macht bleibt eine Frage offen: Für wen ist sie eigentlich gemacht und wie soll sie eingesetzt werden?

Als grobe Eingrenzung lässt sich sagen, dass diese Anwendung für die Lehre geschaffen wird, um genau zu sein für den Teilbereich der Mathematik. Wichtig ist hierbei zu beachten, dass sich die Applikation nach dem momentanen Konzept nur während der Vorlesung / des Unterrichts benutzen lässt. Dies liegt daran, dass die Idee Teams vorsieht, welche zusammenarbeiten und so Punkte sammeln. Die Arbeit als einzelne Person ist nicht vorgesehen.

Da die Anwendung vorsieht, dass jeder Nutzer einen eigenen Zugang zum Internet hat verlagert sich das Hauptnutzungsfeld in höhere Bildungsstufen, zum Beispiel Universität und Fachhochschule, da dort jeder ein eigenes Smartphone oder sogar einen Laptop benutzt.

Zusammenfassend lässt sich also sagen, dass die Anwendung hauptsächlich im Bereich der Lehre der Mathematik in Universitäten und Fachhochschulen zum Einsatz kommen wird.

### 2.2 Was ist eine Aufgabe

Als letzten Schritt ist nun noch zu definieren was genau eine Aufgabe eigentlich ist, beziehungsweise wie sie im Sinne dieser Anwendung gesehen wird.

Für diese Arbeit werden wir Aufgaben immer in drei Teilen betrachten:

1. Die Variablen

Als erstes betrachten wir die Variablen. Sie werden für jede Aufgabe neu generiert. Dadurch, dass diese dynamisch generiert wird jedes mal wenn eine Aufgabe angefragt wird.

2. Die Darstellung

An zweiter Stelle befassen wir uns mit der Darstellung der Aufgabe. Um die vorher generierten Variablen für den menschlichen Nutzer angemessen darzustellen wird hier eine Zeichenkette generiert, welche die Aufgabe repräsentiert, in die dann die Variablen eingesetzt werden.

### 3. Die Lösung

Nachdem die Aufgaben generiert und dargestellt wurden bleibt noch eine Sache übrig. Die Validierung der Aufgabe. Es wird die Lösung des Nutzers übergeben und festgestellt ob diese korrekt gelöst wurde oder nicht.

## 2.3 Verwendete Technologien

Für die Anwendung ist nun bereits klar, dass es sich um eine Webanwendung handeln wird. Doch stehen noch einige Fragen zu dieser offen. Wie sieht das Frontend aus? Welcher Technologie wird zum Ausliefern der Backend Daten genutzt? Diese Fragen werden nachfolgend geklärt.

### 2.3.1 MariaDB

Bereits am Anfang der Entwicklung stand fest, dass viele Daten sowie die Accounts von Administratoren über mehrere Sitzungen hinweg persistiert werden müssen. Ebenfalls stand die Idee im Raum die durchschnittlichen Ergebnisse von Aufgaben zu speichern um so langfristige Beobachtungen durchführen zu können. Daher war recht früh klar, dass auf eine Datenbank kein Verzicht sein kann.

Wenn man sich nun aber überlegt was für eine Datenbank in einem Projekt eingesetzt werden sollte landet man vor einer großen Frage: Dokument-basiert oder doch lieber relational?

Bei relationalen Datenbanken besteht einer der Hauptvorteile liegt in ihrer Struktur. Ihr Aufbau in mehrere Tabellen erlaubt es, dass es sehr leicht ist Duplikate von Daten zu vermeiden (auch bekannt unter dem Begriff "Normalisierung"). Dadurch bleibt die Datenbank kleiner und leichter zu durchsuchen, was gut für die Performanz ist.

Sollte man allerdings Flexibilität benötigen so sind Dokument-basierte Datenbanken die richtige Wahl. Dadurch, dass die Daten einfach in ein Dokument abgelegt werden und nicht an ein Schema gebunden sind an das sich gehalten werden muss können die einzelnen Dokumente ohne Probleme sehr voneinander abweichen. Am ehesten vergleichbar ist dies mit einem normalen JSON-Objekt. Da die Daten hier nicht über mehrere Tabellen verteilt sind lässt sich ein Dokument jederzeit mit einem einzelnen Query aus der Datenbank extrahieren. Daher sind Dokument-basierte Datenbanken immer dann die richtige Wahl wenn viele Datensätze existieren, welche allerdings keine Beziehung zu einander haben (nicht-relationale Datensätze)

Da in dem Projekt die Notwendigkeit war eine Relation zwischen Nutzern und ihren Rollen (Administrator / Nutzer) herzustellen ist die Wahl auf eine relationale Datenbank gefallen.

Bei der Suche nach einer passenden Datenbank sind zwei in Frage gekommen : MariaDB oder MySQL. Hier die Wahl zu treffen ging recht schnell, da es sich bei MariaDB um nicht viel mehr als einen Fork von MySQL handelt, welcher auf einer performanteren Engine und einer intensiveren Betreuung durch das Entwickler-Team basiert.

### 2.3.2 H2

Ein wichtiger Punkt der Anwendung war allerdings auch, dass der Installationsprozess der Anwendung möglichst unkompliziert sein sollte. Dies ist leider nicht der Fall wenn es notwendig ist, dass eine Datenbank installiert und initialisiert wird. Daher wird bei OpenTasks die Alternative angeboten durch eine kleine Änderung der Konfiguration eine “in-memory-database”(IMDB) zu benutzen. Dabei handelt es sich um eine Datenbank die lediglich im Arbeitsspeicher der Maschine existiert und keinerlei Lese und Schreibzugriffe auf die Festplatte ausführt, während eine normale Datenbank den Großteil der Daten auf der Festplatte speichert. Dadurch wird der Installationsaufwand reduziert.

Einer der Hauptvorteile einer solchen Datenbank liegt in der schnellen Zugriffszeit, da Arbeitsspeicher erheblich performanter ist als Festplattenspeicher. Allerdings hat eine IMDB nicht nur Vorteile. Arbeitsspeicher ist erheblich teurer als regulärer Festplattenspeicher, daher sind die Kosten nicht zu vernachlässigen. Da es sich bei OpenTasks nur um kleine Datenmengen handelt ist dies in dieser Anwendung zu vernachlässigen.

Ein viel größeres Problem ist hingegen die Persistenz der Daten. Da bei einer IMDB alle Daten im Arbeitsspeicher gehalten werden sind sie im Falle eines Absturzes alle verloren. Viele moderne Datenbanken nutzen daher das Prinzip von “Snapshots”. Dabei wird in regelmäßigen Abständen (oder bei ordnungsgemäßem herunterfahren der Datenbank) der aktuelle Stand der Datenbank auf der Festplatte gespeichert um den Datenverlust zu reduzieren. Trotzdem sind diese Daten niemals so sicher wie bei einer Datenbank, welche ihre Daten auf der Festplatte speichert.

Daher ist es, für eine dauerhafte Installation, immer besser eine Festplattenbasierte Datenbank zu benutzen. Für eine kurze Testphase ist allerdings eine IMDB eine angenehme Alternative.

### 2.3.3 Dependency-Injection

*„Als Dependency Injection (englisch dependency ‚Abhängigkeit‘ und injection ‚Injektion‘; Abkürzung DI) wird in der objektorientierten Programmierung ein Entwurfsmuster bezeichnet, welches die Abhängigkeiten eines Objekts zur Laufzeit reglementiert: Benötigt ein Objekt beispielsweise bei seiner Initialisierung ein anderes Objekt, ist*



*diese Abhängigkeit an einem zentralen Ort hinterlegt – es wird also nicht vom initialisierten Objekt selbst erzeugt.* “<sup>1</sup>

Bei der Dependency-Injection handelt es sich um ein Pattern welches aus der heutigen Welt der Programmierung kaum wegzudenken ist. Anstatt das einzelne Komponenten ihre Abhängigkeiten selber erzeugen geben sie lediglich an, dass sie eine Abhängigkeit benötigen welche einen bestimmten Satz an Aufgaben erfüllen kann (oft durch ein Interface definiert). Daraufhin sucht der Dependency-Injector nach einer Komponente (in den meisten Frameworks als “Service” bezeichnet) die diese Anforderungen erfüllt und fügt sie in den Anfragenden ein (bei diesem Prozess handelt es sich dann um die “Injection”)

*Beispiel für eine Komponente zur Abfragung von Daten (Datenbankzugriff)*

---

```
public interface ScriptRepository extends CrudRepository<ScriptEntity,
    Integer> {
    public ScriptEntity findByName(String name);
    public List<ScriptEntity> findAll();
}
```

---

### Service

Der Begriff Dienst (auch Service oder Daemon) beschreibt in der Informatik allgemein eine technische, autarke Einheit, die zusammenhängende Funktionalitäten zu einem Themenkomplex bündelt und über eine klar definierte Schnittstelle zur Verfügung stellt.

*Beispiel für einen Service zum Sammeln von Fehlern*

---

```
public interface GlobalErrorService {
    public void appendError(String message);
    public List<String> getErrors();
}
```

---

Durch das Pattern der Dependency-Injection wird die Trennung zwischen der “Business-Logik” und der technischen Implementation stark getrennt. Dies fördert die Nachvollziehbarkeit von Programmabschnitten. Ebenfalls wird dank der loseren Kopplung die Wiederverwertbarkeit von Code verbessert. Daher wird Dependency-Injection immer dann verwendet wenn auf Modularität Wert gelegt wird

### Geschäftslogik

*„Geschäftslogik (englisch business logic, auch Anwendungslogik) ist ein abstrakter Begriff in der Softwaretechnik, der eine Abgrenzung der durch die Aufgabenstellung selbst motivierten Logik eines Softwaresystems von der technischen Implementierung zum*

---

<sup>1</sup>[https://de.wikipedia.org/wiki/Dependency\\_Injection](https://de.wikipedia.org/wiki/Dependency_Injection)

*Ziel hat. “<sup>2</sup>*

Ebenfalls zu erwähnen ist die Verbesserung des Verständnisses von Programmcode, insbesondere wenn man nicht den gesamten Kontext der Anwendung kennt. Dies liegt daran, dass jeder Service immer für einen Satz an Aufgaben steht welche logisch zusammenhängen (zum Beispiel das Verwalten der Punktzahl von Nutzern). Sollte ein Service ebenfalls Abhängigkeiten haben werden diese ebenfalls über den Dependency-Injector eingefügt.

Als letzter Punkt steht die Verbesserung der Testbarkeit. Dadurch, dass die einzelnen Komponenten ihre Abhängigkeiten an einer zentralen Stelle anfragen anstatt sie selber zu erzeugen können an dieser simplere Versionen der Abhängigkeiten hinterlegt werden (sogenannte Mock's), wodurch die Funktionen der zu testenden Komponente steigt da Fehler durch die Abhängigkeiten in dieser Umgebung nicht auftreten können.

### Mock

*„Mock-Objekte (auch Attrappe, von englisch to mock ‚etwas vortäuschen‘) sind in der Softwareentwicklung Objekte, die als Platzhalter für echte Objekte innerhalb von Modultests verwendet werden. Diese werden umgangssprachlich auch Mocks genannt.“<sup>3</sup>*

Das Pattern der Dependency-Injection wird von denn im Nachhinein folgenden Frameworks Spring(-Boot) sowie Angular 2 implementiert und intensiv genutzt.

### 2.3.4 Spring(-Boot)

Auf der Suche nach einem Framework für die Programmierung des REST-Backends stolpert man oft über drei die sich wiederholen. Express-JS, Sinatra und Spring-Boot.

*„Express ist ein einfaches und flexibles Node.js-Framework von Webanwendungen, das zahlreiche leistungsfähige Features und Funktionen für Webanwendungen und mobile Anwendungen bereitstellt.“<sup>4</sup>*

Schaut man sich als erstes Express an stößt man auf leichtes Package-Management (dank NPM), eine Sprache die exzellent für das Bearbeiten und Manipulieren von JSON gemacht ist. Gleichzeitig ist das Problem, dass viele Bibliotheken unterschiedliche Standards benutzen was der Übersicht in größeren Projekten schadet. Zusätzlich kommt das Problem, dass Javascript durch fehlende starke Typisierung eine Sprache ist die bei wachsenden Code-Mengen immer unübersichtlicher wird, während nicht für alle Bibliotheken eine Typescript Implementation vorhanden ist die die Arbeit

---

<sup>2</sup><https://de.wikipedia.org/wiki/Geschäftslogik>

<sup>3</sup><https://de.wikipedia.org/wiki/Mock-Objekt>

<sup>4</sup><http://expressjs.com/de/>

erleichtern würde. Dazu kommt, dass die Nutzung mehrerer Kerne (und so die Handhabung vieler Anfragen an den Server) nur mittels Prozess-Managern möglich ist, welche ebenfalls wieder Konfiguration und Testaufwand bedeuten. Als letztes ist zu sagen, dass Express-JS eine relativ kleine Entwickler-Basis besitzt, was zu weniger Möglichkeiten für Hilfestellungen bei Problemen führt

*„Das Spring Framework (kurz Spring) ist ein quelloffenes Framework für die Java-Plattform. Ziel des Spring Frameworks ist es, die Entwicklung mit Java/Java EE zu vereinfachen und gute Programmierpraktiken zu fördern. Spring bietet mit einem breiten Spektrum an Funktionalität eine ganzheitliche Lösung zur Entwicklung von Anwendungen und deren Geschäftslogiken; dabei steht die Entkopplung der Applikationskomponenten im Vordergrund.“<sup>5</sup>*

Schaut man sich als letztes das Spring-Framework an stößt man auf ein System welches über ein inzwischen gigantisches Ökosystem an Erweiterungen verfügt. Seit März 2004 hat es immer mehr an Popularität und an Features dazu gewonnen. Doch ein wirklich großer Sprung war der Release von Spring-Boot. Dieses übernahm ein Großteil der Konfiguration einer Anwendung und kam bereits mit integriertem Webcontainer mit. Einer der vielen Vorteile die Spring mitbrachte war die Dependency-Injection, welche von Haus aus implementiert war und direkt verwendet werden konnte. Die Absicherung einer Anwendung wird durch Spring-Security ebenfalls erheblich vereinfacht, welches direkt Schutz gegen einige Angriffe bietet, sowie einige Authentifizierungsverfahren erheblich vereinfacht. Eines der Probleme ist allerdings, dass es sich bei Spring (und all seinen Erweiterungen) inzwischen um ein immens großes Projekt handelt, was es sehr schwer machen kann mit Spring anzufangen und die erste Webanwendung zu schreiben.

Alternative : Express-JS, Sinatra

Für das Backend ist die Wahl auf Spring-Boot gefallen. Bei Spring-Boot handelt es sich um eine Erweiterung des bekannten Java-Frameworks Spring, welches den Großteil der Konfiguration automatisiert, beziehungsweise vereinfacht.

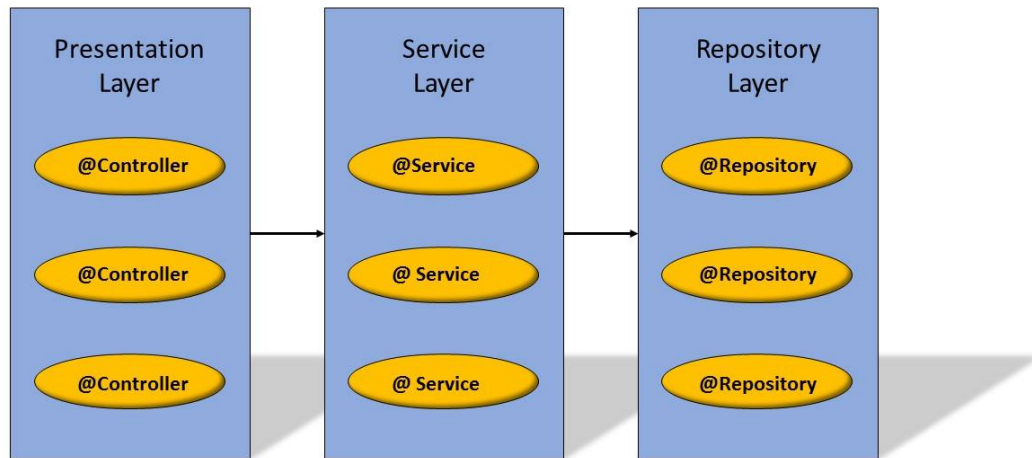
Kurz gesagt hilft das Framework dabei eine Anwendung modular aufzubauen, was insbesondere bei großen Anwendungen sehr hilfreich ist, da diese übersichtlich bleiben. Drei Kernkonzepte die dabei besonders wichtig sind sind die Controller, sowie die bereits erklärten Services und die Dependency-Injection

### Controller

Bei einem Controller handelt es sich um eine Klasse welche für die Verwaltung von Anfragen zuständig ist. Jede Anfrage kommt immer bei einem Controller an, von wo die Anfrage an den zuständigen Service weitergeleitet wird. Das Ergebnis von diesem wird dann über den Controller zurück an den Sender der Anfrage gesendet

---

<sup>5</sup>[https://de.wikipedia.org/wiki/Spring\\_\(Framework\)](https://de.wikipedia.org/wiki/Spring_(Framework))



**Abbildung 2.1:** Konzept einer Spring-Anwendung

Gleichzeitig stellt Spring-Boot auch direkt einen Container für die Webapplikation zur Verfügung. Das bedeutet, dass die Programmierung sich komplett mit der REST-API und ihren Funktionalitäten beschäftigen kann und nur sehr wenig Zeit für die Konfiguration des Web-Servers verbraucht wird.

Ein großer Teil der Konfiguration findet außerhalb des Codes statt. In dem Projekt kann eine Datei mit dem Namen "application.properties" hinterlegt werden. Hier können dann mehrere Sachen eingestellt werden, wie zum Beispiel der Port, unter dem der Server laufen soll.

*Konfiguration des Server-Ports*

---

```
spring.server.port=8080
```

---

### 2.3.5 Spring Data JPA

*„Implementing a data access layer of an application has been cumbersome for quite a while. Too much boilerplate code has to be written to execute simple queries as well as perform pagination, and auditing. Spring Data JPA aims to significantly improve the implementation of data access layers by reducing the effort to the amount that’s actually needed. As a developer you write your repository interfaces, including custom finder methods, and Spring will provide the implementation automatically.“*<sup>6</sup>

---

<sup>6</sup><http://projects.spring.io/spring-data-jpa/>

Bei Spring Data JPA (im nachfolgenden nur noch als JPA bezeichnet) handelt es sich um eines der Projekte aus dem Spring-Ökosystem welches in wahrscheinlich jedem Spring Projekt heutzutage genutzt wird. Das Hauptziel des Projektes ist es, dass überflüssiger Code der jedes mal geschrieben wird automatisch generiert werden kann und so dem Entwickler viel Arbeit abnimmt. Der Zugriff auf eine Datenbank wird enorm erleichtert, indem die Verbindung automatisch aufgebaut und die Querys automatisch aus Interfaces generiert werden.

Als erstes muss JPA also die Information erlangen wie die Datenbank erreichbar ist und wie die Zugangsdaten lauten. Dies wird in der `application.properties` angegeben

### *Konfiguration der Datenbankverbindung*

---

```
spring.datasource.url=jdbc:mysql://localhost:3306/bachelor
spring.datasource.username=root
spring.datasource.password=root
```

---

Nach dem JPA nun die Verbindung mit der Datenbank aufbauen kann muss definiert werden wie die Datenbank-Entitäten in Java-Objekte umgewandelt werden und in welchen Tabellen sie sich befinden. In JPA wird dies dadurch umgesetzt, dass für jede Entität eine Klasse angelegt wird in der durch Annotationen festgelegt wird wie die Daten in das Objekt gelegt werden.

### Beispiel für eine Datenbank-Entität

---

```
@Entity
@Table(name="scripts")
public class ScriptEntity implements Serializable{

    @Id
    @GeneratedValue(strategy= GenerationType.IDENTITY)
    private Integer id;

    @Column(nullable=false, name="name")
    private String name;

    @Column(nullable=false, name="variablesScript")
    private String variableScript;

    @Column(nullable=false, name="solutionscript")
    private String solutionScript;

    @Column(nullable=false, name="mathjaxscript")
    private String mathjaxScript;

    /*Getter/Setter/Konstruktor*/
}
```

---

Zu guter Letzt muss ein Interface erzeugt werden, welches definiert was für Querys zur Verfügung stehen. Um das zu realisieren wird ein Interface erzeugt, welches die Querys erzeugt indem es bei der Benennung der Methoden bestimmten Konventionen folgt. Wichtig ist hierbei, dass es das bereits gegebene Interface CrudRepository erweitern muss. Dadurch wird es von JPA erkannt und einige Funktionalität direkt gegeben (wie zum Beispiel findAll, save und delete). Um nun eigene Querys hinzufügen gibt es zwei Möglichkeiten. Entweder über eine Annotation oder über die Benennung der Funktion

#### *Beispiel für eigene Querys*

---

```
public interface ScriptRepository extends CrudRepository<ScriptEntity,
    Integer> {

    public ScriptEntity findByName(String name);

    @Query("SELECT p FROM ScriptEntity s WHERE s.name = :name")
    public ScriptEntity customQuery(@Param("name")String name);
}
```

---

### 2.3.6 Bootstrap

Gegenstück : Material

*„Bootstrap ist ein anpassungsfähiges und zuerst für mobile Geräte entwickeltes Framework. Egal, wie gut du dich auskennst, kannst du damit einfacher und schneller kleine wie große Projekte entwickeln, die auf Geräten in allen erdenklichen Formen funktionieren.“*<sup>7</sup>

Aus der heutigen Welt der Webentwicklung ist Bootstrap kaum wegzudenken. Durch die leichte Entwicklung für alle vorhandenen Formate und Endgeräte hat es sich als äußerst robust herausgestellt wenn es um die schnelle Entwicklung von Webseiten geht.

Gleichzeitig bietet es ein großes Repertoire an CSS-Klassen für alle Elemente der HTML-Entwicklung was es sogar Personen mit wenig Erfahrung im Design-Bereich ermöglicht spektakuläre Ergebnisse mit wenig Arbeit zu erzielen.

### 2.3.7 Typescript / Angular 2

Bevor man sich mit der Technologie beschäftigt die benutzt wurde um das Frontend zu bauen muss man sich vorher mit einer anderen Errungenschaft der Webentwicklung auseinander setzen. Typescript.

*„TypeScript is a free and open-source programming language developed and maintained by Microsoft. It is a strict superset of JavaScript, and adds optional static typing and class-based object-oriented programming to the language. TypeScript is designed for development of large applications and transcompiles to JavaScript. As TypeScript is a superset of JavaScript, any existing JavaScript programs are also valid TypeScript programs.“*<sup>8</sup>

Besonders für Entwickler welche aus der Welt der stark-typisierten Programmierwelt kamen war TypeScript ein großer Fortschritt. Durch die Möglichkeit Variablen stark zu typisieren entstand die Möglichkeit Schnittstellen klarer zu definieren. Gleichzeitig wurden die bestehenden Möglichkeiten Entwicklungsumgebungen mit Auto-Vervollständigung auszustatten verbessert.

Die Vorteile die sich daraus für die Entwicklung sind recht eindeutig: Eine Beschleunigung der Entwicklung, während die Testbarkeit vereinfacht wird und Fehler besser vermieden werden können. Kurz gesagt: Bessere Software in weniger Zeit.

Was für die Anwendung relativ früh klar war war, dass es sich um eine Single Page Application (kurz SPA) handeln sollte. Doch was genau ist das und wieso sollte es verwendet werden?

---

<sup>7</sup><http://holdirbootstrap.de//>

<sup>8</sup><https://en.wikipedia.org/wiki/TypeScript>

*„Als Single-Page-Webanwendung (englisch single-page application, kurz SPA) oder Einzelseiten-Webanwendung wird eine Webanwendung bezeichnet, die aus einem einzigen HTML-Dokument besteht und deren Inhalte dynamisch nachgeladen werden. Diese Art von Web-Architektur steht im Gegensatz zu klassischen Webanwendungen, die aus mehreren, untereinander verlinkten HTML-Dokumenten bestehen. Hierdurch wird allerdings die Grundlage geschaffen, eine Webanwendung in Form einer Rich-Client- bzw. Fat-Client-Verteilung zu entwickeln. Eine verstärkte clientseitige Ausführung der Webanwendung ermöglicht eine Reduzierung der Serverlast sowie die Umsetzung von selbstständigen Webclients, die beispielsweise eine Offline-Unterstützung anbieten.“*<sup>9</sup>

Warum also eine Single-Page-Application? Der Hauptvorteil liegt darin, dass bei dem Initialisieren der Seite fast alle benötigten Informationen geladen werden, somit fallen die Ladezeiten nach der ersten Initialisierung sehr gering aus. Das sorgt insbesondere bei Nutzern von mobilen Endgeräten für eine bessere User-Experience, da bei diesen (bedingt durch schwächere Prozessoren) die Ladezeiten besonders ins Gewicht fallen. Gleichzeitig reduzieren sich die Anforderungen an den Webserver, da der Content nur ein einziges mal ausgeliefert werden muss. Lediglich durch die Abfragung einiger Daten zur Laufzeit (wie zum Beispiel das Erhalten einer neuen Aufgabe) wird der Server nach dem Ausliefern der Webseite kontaktiert.

Nachdem nun evaluiert wurde, dass eine SPA ein Gewinn ist also die Frage nach dem Framework / der Library um das umzusetzen. Im Raum stehen zwei Varianten. Zum einen ReactJS (welches seit längerem existiert und weit bekannt ist) oder das gerade in der Beta angekommene Angular 2.

Direkt zu sagen ist, dass Angular 2 von Haus aus mit Typescript gebaut ist. Das bedeutet bester Typescript bei der Entwicklung von Angular 2, was bei ReactJS nur teilweise ist und viele Varianten wie man am besten mit ReactJS und Typescript arbeitet sich widersprechen da keine klare Konvention herrscht.

Zusätzlich implementiert Angular 2 eine bessere Trennung der verschiedenen Komponenten. Während in ReactJS HTML und Javascript Code in der selben Klasse stehen werden in Angular 2 klare Mechanismen aufgezeigt wie diese sauber zu trennen und in verschiedenen Dateien abzulegen sind. Was gleichzeitig auch einer der Nachteile von Angular 2 ist. Dadurch, dass HTML und Script sauber getrennt werden ist es schwerer die Script-Variablen im HTML zu referenzieren und alle Namen im Kopf zu behalten ohne die zweite Datei irgendwo geöffnet zu haben.

Bei Angular 2, sowie ReactJS handelt es sich standardmäßig um Technologien bei denen "Client-Side Rendering" betrieben wird. Das bedeutet, dass vom Server alle Dateien ausgeliefert werden die benötigt werden. Der Client nimmt diese entgegen und rendert die einzelnen HTML-Komponenten und fügt diese korrekt zusammen. Das bedeutet, dass die Last des Rendering auf den Client übertragen wird. Das be-

---

<sup>9</sup><https://de.wikipedia.org/wiki/Single-Page-Webanwendung>



deutet weniger Aufwand für den Server, gleichzeitig aber eine bessere Performanz für den Server (da dieser nicht selber rendern muss). Was bei Angular2 aber auch unterstützt wird ist die Möglichkeit auf der Seite des Server zu rendern. Das bedeutet wenn man die Ladezeiten auf dem Client reduzieren will und einen performanten Server zur Hand hat ist dies die korrekte Lösung. Möglich ist dies durch eine kürzlich herausgekommene Erweiterung für Angular 2 <https://universal.angular.io/>.

Ein Feature welches ebenfalls beide Bibliotheken implementieren ist das sogenannte “Two-Way-Binding”. Bei dem Two-Way-Binding wird eine Variable, welche in der View genutzt wird mit einer Variable aus dem Model verknüpft. Ein klassisches Beispiel dafür sind Textfelder. Wenn der Nutzer eine Eingabe betätigt wird dies ebenfalls an das Model weitergeleitet, welches daraufhin die zugehörige Variable erneuert. Gleichzeitig wird bei Veränderung der Variable im Model die View geupdatet. So kann man zum Beispiel ein Knopf der für einen Reset des Textfeldes verantwortlich ist einfach die Variable im Model ändern, die View wird automatisch erneuert.

Am Ende fiel die Wahl auf Angular 2, auf Grund der oben genannten Vorteile zusätzlich dazu, dass hinter Angular 2 ein Team von Entwicklern steht welche Erfahrungen aus einem vorherigen Projekt (AngularJS) gezogen haben um daraus von neu an ein komplett neues Framework zu erstellen welches aus den Problemen der alten Version gelernt hat und diese ausgemerzt hat.

### 2.3.8 SocketIO

Es stand bereits sehr früh klar, dass Daten vom Server zum Client übertragen werden müssen um den aktuellen Stand der Aufgaben für die Übersicht des Administrators aktuell zu halten. Früher hätte man dafür Long Polling verwendet, doch heute gibt es eine Technologie, welche erheblich besser für diesen Zweck geeignet ist: Websockets.

*„WebSockets ist eine fortschrittliche Technologie welche es möglich macht eine interaktive Kommunikations-Session zwischen dem Browser des Benutzers und dem Server herzustellen. Mit dieser API können Sie Nachrichten zum Server senden und ereignisorientierte Antworten erhalten ohne beim Server die Antwort abzufragen.“*<sup>10</sup>

Für dieses Protokoll gibt es inzwischen etliche Implementierungen (PubNub, WS, uWebSockets um nur einige zu nennen), da für dieses Projekt am wichtigsten war eine zuverlässige Lösung zu haben fiel die Entscheidung am Ende auf socket.io, da es ein Projekt ist welches schon sehr lange genutzt, gewartet und weiterentwickelt ist. Daher ist bei einem so erfahrenen Entwicklerteam zu rechnen, dass es recht reibungslos funktioniert und auch zukünftige Versionen stabil und performant sind.

---

<sup>10</sup><https://developer.mozilla.org/de/docs/WebSockets>

### 2.3.9 JSON Web Tokens

Wenn man im Web-Bereich Anwendungen entwickelt gibt es einen Abschnitt welcher fast immer relevant ist: Die Authentifizierung. Daher ist dieser Bereich inzwischen auch so vielschichtig und es gibt enorm viele Möglichkeiten und Verfahren Authentifizierung erfolgreich umzusetzen. Als Verfahren welches wohl am längsten dabei ist gibt es zum Beispiel “Basic Auth”. Hierbei wird bei jeder Anfrage im Header der Nutzernamen und das Passwort übertragen. Dieses wird auf Server Seite überprüft und bei Korrektheit die Daten ausgeliefert. Der Vorteil ist eine sehr einfache Logik, allerdings muss der Server bei jeder Anfrage die Korrektheit von Nutzernamen und Passwort überprüfen, sowie eventuelle Daten die mit dem Nutzer in Verbindung stehen aus der Datenbank extrahieren.

Zwei Ansätze die sich relativ ähneln sind Token und Cookie Authentication. Bei beiden wird ein Objekt generiert welches von dem Client als Authentifizierungs-Objekt bei jeder Anfrage mitgeschickt wird. Der Unterschied ist, dass Cookies direkt im Browser gespeichert werden, während die Speicherung von Tokens vom Programmierer selber entwickelt werden muss (und dann meistens im RAM gehalten werden).

Am Ende ist die Wahl auf JSON Web Tokens(JWT) gefallen. Zusätzlich zu den Authentifizierungsmöglichkeiten beinhaltet JWT die Möglichkeit Nutzdaten im Token mitzuschicken. Das bedeutet, dass der Nutzer nach der Authentifizierung Daten erhalten kann welche für die gesamte Nutzung der Anwendung wichtig sind, wie zum Beispiel welche Rolle er erhalten hat und welche Rechte damit verbunden sind.

### 2.3.10 MathJax

## 3 Ein Rundgang durch das Frontend

Das Frontend ist im wesentlichen in zwei große Abschnitte geteilt - Die Ansicht des Nutzers (Also die Anwendung wie sie von den Studenten gesehen wird die dort ihre Aufgaben erhalten und bearbeiten können) und die Ansicht des Administrators, welche hauptsächlich die Aufgabe hat die Anwendung zu verwalten und Einstellungen zu ändern. Daher werden diese beiden Sichten im Nachfolgenden getrennt voneinander behandelt, beziehungsweise ein Einblick in die Funktionsweise gewährt.

### 3.1 Die Ansicht des Nutzers

#### 3.1.1 Die Auswahl des Teams

Bevor mit der Bearbeitung der Aufgaben beginnen kann ist vorher der Login-Prozess von Nöten. Der Nutzer muss dafür lediglich ein Team auswählen und dies über Knopf-Druck bestätigen. Um die Auswahl besonders für Mobiltelefone gut auszulegen ist die Auswahl über Knöpfe gewählt worden. Um die aktuelle Auswahl hervorzuheben wird das ausgewählte Team als Text über den Knöpfen dargestellt, zusätzlich wird der entsprechende Knopf mit rotem Text versehen.

Willkommen bei Open Tasks. Bitte wählen dein Team aus um mit der Bearbeitung der Aufgaben zu beginnen.



The image shows a user interface for team selection. At the top, the text 'Team Eins' is displayed. Below it, there are two buttons: 'Team Eins' and 'Team Zwei'. The 'Team Eins' button is highlighted with a red border and red text, indicating it is the selected team. The 'Team Zwei' button has a grey border and grey text. Below these buttons is a blue button with the text 'Login' in white.

**Abbildung 3.1:** Team Auswahl für den Nutzer

Sobald der Login Knopf betätigt wurde werden mehrere Pakete an den Server geschickt. Bei dem ersten handelt es sich um das Erhalten des Tokens welches in Zukunft für die eindeutige Identifizierung und Authentifizierung zur Nutzung der REST-API genutzt wird. Sobald dieser Prozess beendet wurde wird die Auswahl des Teams übermittelt. Sind alle diese Prozesse erfolgreich wird auf die Bearbeitung der Aufgaben weitergeleitet.

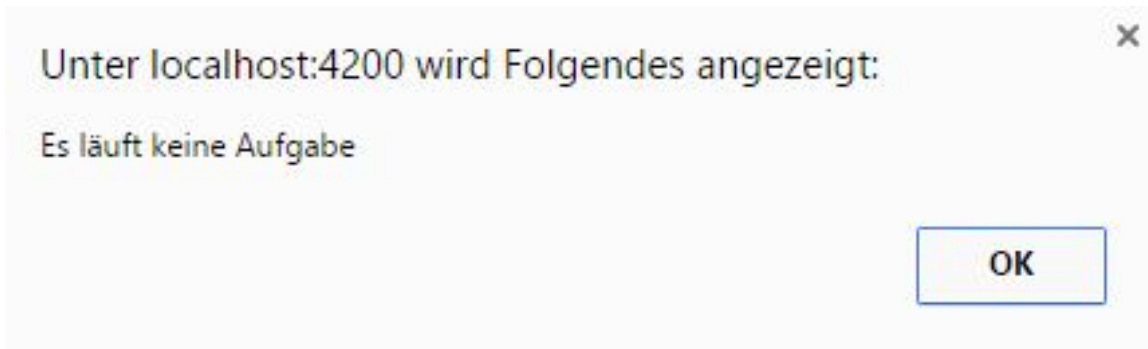
### 3.1.2 Erhalten und bearbeiten der Aufgaben

Nachdem nun die initiale Prozedur der Team-Auswahl beendet ist kommt der Student zu dem Hauptteil der Anwendung. Die Erhaltung und Bearbeitung der Aufgaben. Nach der Einwahl erscheint ein Hinweis, dass bisher keine Aufgabe gefunden wurde, mit einem Knopf, welcher das Anfragen einer Aufgabe ermöglicht.



**Abbildung 3.2:** Ansicht bei nicht geladener Aufgabe

Sollte dieser Knopf betätigt werden wird eine Anfrage an den Server gesendet, welche entweder mit dem Erhalt einer Aufgabe oder der Information, dass momentan kein Generator vorhanden ist beendet wird.



**Abbildung 3.3:** Information über keinen laufende Aufgabe

Sollte eine Aufgabe erhalten worden beginnt nun die Bearbeitung der Aufgabe. Der erhaltene Inhalt wird als mathematische Formel dargestellt, hierbei hilft die Bibliothek MathJax dabei den erhaltenen Text optisch ansprechend und mathematisch korrekt darzustellen. Der Student hat nun die Zeit die er braucht um die Aufgabe zu bearbeiten. Sollte er sich der Lösung sicher sein gibt er diese in das dafür vorgesehene Feld ein und bestätigt das Ganze mit einem Druck auf den “Senden” Knopf. Die Eingabe wird an den Server weitergeleitet welcher diese validiert. Für den Studenten gibt es nun drei Fälle.

1. Die Bearbeitungszeit für die Aufgabe ist abgelaufen. Der Nutzer erhält eine Informationsanzeige über den Ablauf der Zeit. Die momentan angezeigte Aufgabe wird entfernt

### 3 Ein Rundgang durch das Frontend

2. Die Aufgabe wurde korrekt gelöst. Der Nutzer erhält eine neue Aufgabe, sowie eine Notifikation über die korrekte Lösung
3. Die Lösung war falsch. Der Nutzer wird informiert, behält aber seine momentane Aufgabe bei um eine neue Lösung anzugeben

Um die Effizienz für den Studenten während der Nutzung des Studenten zu steigern wurden zwei weitere Features eingebaut. Bei dem Ersten handelt es sich um eine kleine Informationsanzeige wie lange die durchschnittliche Zeit beträgt bis eine Aufgabe korrekt gelöst wurde. Dies soll dem Studenten helfen seine Leistungen zu überprüfen und sich vielleicht auch mit anderen zu vergleichen.

Bei dem zweiten handelt es sich um die Eingabeformulare. Eine Sache die alle gemeinsam haben ist die Validierung ob die momentane Antwort des Nutzers korrekt sein könnte und das graphische Darstellen davon (grün / rot- Färbung des Antwort Textes). Wie genau diese Validierung aussieht und welche weiteren Funktionalitäten das Formular bietet ist allerdings von Formular zu Formular unterschiedlich.

The screenshot shows a math problem interface. At the top, there are three congruence equations:
$$\begin{aligned}x &\equiv 0 \pmod{5} \\ x &\equiv 7 \pmod{11} \\ x &\equiv 1 \pmod{3}\end{aligned}$$
Below the equations is the number 40. Underneath, the text "Antwort: 50" is displayed in green. Below this is an input field containing the number 50. To the right of the input field is a numeric keypad with buttons for digits 1 through 9, 0, and a "CE" button. Below the keypad is a green progress bar with the text "2 / 2". At the bottom, there is a text label "Die durchschnittliche Bearbeitungszeit beträgt 9.00 Sekunden" and two buttons: "Send" (blue) and "Refresh" (red).

**Abbildung 3.4:** Beispiel für eine komplette Seite zur Bearbeitungszeit

#### 3.1.3 Die Eingabeformulare

Insgesamt existieren fünf Eingabeformulare.

1. Das Textformular
2. Das Zahlenformular
3. Das Dezimalformular
4. Das Matrixformular
5. Das Bruchformular

In diesem Kapitel werden diese genauer unter die Lupe genommen und darüber gesprochen wie genau ihre graphische Darstellung ist, sowie die Validierung ob eine Antwort korrekt sein kann und abgeschickt werden darf.

Die Antwort die sich am Ende von dem Formular ergibt wird zusätzlich immer über dem Formular noch einmal angezeigt, so kann der Nutzer überprüfen, dass seine Antwort dem gewünschten entspricht.

Welches Formular angezeigt wird wird immer von dem Administrator der Anwendung angezeigt, für weitere Informationen bitte im Kapitel 3.2.3 Bearbeitung der Generatoren und ihrer Hilfsmittel nachlesen.

#### 3.1.3.1 Das Textformular

Bei dem Textformular handelt es sich um das wohl einfachste aller Formulare. Die graphische Darstellung besteht lediglich aus einer Textbox.

Ähnlich simpel ist ebenfalls die Validierung, es wird lediglich überprüft, dass die Antwort eine Länge von mindestens einem Zeichen hat, sodass keine leeren Antworten abgegeben werden können.

#### 3.1.3.2 Das Zahlen / Dezimalformular

Da sich das Zahlen und das Dezimal-Formular sehr ähnelt werden sie hier gemeinsam behandelt. Da es sich bei den Nutzern von OpenTasks zu großen Teilen um mobile Endgeräte nutzen wird wurde dieses Formular mit einem speziellen Fokus auf diese entworfen.



The image shows a digital calculator interface. At the top, the number '158.3' is displayed in green. Below it, a text input field contains '158.3|'. Underneath the input field is a numeric keypad with buttons for digits 1 through 9, 0, a decimal point (.), and a 'C' button. At the bottom of the keypad is a 'CE' button.

1	2	3
4	5	6
7	8	9
C	0	.
CE		

**Abbildung 3.5:** Das Dezimalformular

1	2	3
4	5	6
7	8	9
C	0	CE

**Abbildung 3.6:** Das Zahlenformular

Die Eingabe kann über zwei Wege erfolgen. Bei dem ersten handelt es sich um die Eingabe der Lösung über das Textfeld. Dies ist besonders für die Nutzer von Laptops gut, da hier eine Tastatur zur Verfügung steht.

Für die mobilen Endgeräte wurden Knöpfe implementiert, welche bei Druck die abgebildete Zahl der Antwort hinzufügen. Außerdem gibt es noch zwei weitere Knöpfe, einen um die letzte Eingabe zu entfernen (C) und einen um das gesamte Eingabefeld zu leeren (CE). Ein Unterschied zwischen den beiden Formularen ist, dass das Dezimalformular zusätzlich einen Knopf besitzt welcher einen Punkt einfügt und es so ermöglicht Kommazahlen einzugeben.

Der zweite Unterschied liegt in der Validierung der Eingaben. Während bei dem Zahlenformular nur Ganze Zahlen erlaubt sind werden bei dem Dezimalformular alle reellen Zahlen angenommen.

#### 3.1.3.3 Das Matrixformular

Für einige Aufgabentypen war es notwendig, dass der Nutzer in der Lage ist eine Matrix als Antwort einzugeben. Da dies in einem Textfeld sehr schwer ist wurde das Matrixformular entwickelt.

Mobilen Modus umschalten		
Reihe hinzufügen	Reihe entfernen	
Spalte hinzufügen	Spalte entfernen	
3	0	0
0	5	0
0	0	0

$$\begin{bmatrix} 3 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

**Abbildung 3.7:** Das Matrixformular

Bei dem Matrixformular handelt es sich um Textfelder im  $m \times n$  Format, genau wie die Matrix die erzeugt werden soll. Dabei wird die produzierte Matrix immer über dem Formular angezeigt, dabei hilft wieder die MathJax Bibliothek das Ganze optisch ansprechend aufzubereiten.

### 3 Ein Rundgang durch das Frontend

Ein weiterer wichtiger Einfluss bei der Erstellung des Formulars war die Tatsache, dass die Größe der Matrix nicht am Anfang bekannt ist. Zum einen liegt dies daran, dass dies ein Tipp für den Studenten sein kann (da nicht unbedingt direkt bekannt ist welche Größe die Matrix hat die am Ende herauskommen soll). Andererseits liegt dies auch daran, dass es ein größerer Aufwand wäre direkt bei der Generierung festzulegen wie groß die Matrix sein soll. Dieses Problem wurde über vier Knöpfe gelöst, welche die Manipulation der Größe der Matrix erlauben.

Ein weiteres Feature welche in diesem Formular implementiert wurde ist eine automatische Korrektur der Eingabe des Nutzers. Sollte der Nutzer sich vertippen und zum Beispiel aus Versehen einen Buchstaben in eines der Felder eintippen so wird dies automatisch korrigiert indem der Buchstabe entfernt und nicht in der Lösung angezeigt wird.

Für die mobilen Nutzer wurde ein Knopf implementiert welcher auf eine alternative, für mobile Nutzer angepasste Sicht umschaltet. Diese ähnelt sehr der Ansicht von dem Nummernformular. Der Nutzer wählt über Knopfdruck aus welches Feld er bearbeiten will. Daraufhin wird dieses durch blaue dickere Schrift markiert. Sollte der Nutzer nun einen der nummerierten Knöpfe drücken wird der Text des Knopfes der Eingabe hinzugefügt.

The image shows a user interface for a matrix calculator. At the top, there is a button labeled 'Mobilen Modus umschalten'. Below it are four buttons arranged in a 2x2 grid: 'Reihe hinzufügen', 'Reihe entfernen', 'Spalte hinzufügen', and 'Spalte entfernen'. The main area contains a 3x3 grid of buttons with numbers 1 through 9 and a 'CE' button. Below this grid is another 3x3 grid of buttons with numbers 0 through 9. The buttons are styled with different colors and fonts to indicate different states or functions.

**Abbildung 3.8:** Das Matrixformular

#### 3.1.3.4 Das Bruchformular

Als letztes Formular ist das Bruchformular zu erwähnen. Dies sollte jederzeit genutzt werden wenn von dem Nutzer erwartet wird einen Bruch einzugeben.

The image shows a user interface for a fraction calculator. It features a text input field containing the decimal value '0.2500'. Below this field are two separate input fields for the numerator and denominator, with the values '1' and '4' entered respectively.

**Abbildung 3.9:** Das Bruchformular

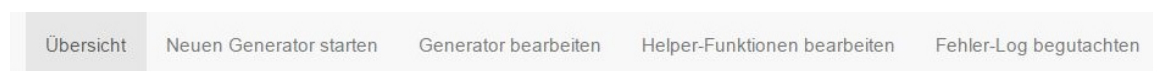


Es unterstützt den Nutzer dabei indem die Eingabe über zwei Textfelder erfolgt. Eines für den Zähler und eines für den Nenner. Sobald der Nutzer in beide Felder einen Wert eingegeben hat wird die daraus resultierende Zahl berechnet. Wenn es sich hierbei um eine Zahl aus dem Raum der rationalen Zahlen handelt wird die Lösung als valide eingestuft und kann abgeschickt werden.

## 3.2 Die Sicht des Administrators

Ein weiterer wichtiger Punkt der Anwendung war die Verwaltung und Übersicht über die Aufgaben und die Lösungen die von den Studenten abgegeben wurden. Nicht zu vergessen ist allerdings auch, dass die Aufgaben in diesem Abschnitt verwaltet und erweitert werden müssen.

Daher wurde diese Sektion in die Verwaltung von laufenden Aufgaben (Übersicht / Starten eines neuen Aufgaben-Generators), die Verwaltung und Erweiterung von Aufgaben (Bearbeitung der Aufgaben-Generatoren und ihren Hilfsmitteln) und das Evaluieren von Fehlerzuständen aufgeteilt (Der Fehler-Log). Die Navigation innerhalb diesem Teil der Anwendung funktioniert über eine Navigationsleiste



**Abbildung 3.10:** Navigationsleiste Admin-Bereich

### 3.2.1 Übersicht über die laufende und letzte Aufgabe

Die während der Laufzeit wahrscheinlich am meisten genutzte Seite : Die Übersicht. Hier wird eine Oberfläche dargestellt, welche optisch aufbereitet die Daten anzeigt die von der Anwendung erzeugt werden: Die Zeit wie lange eine Aufgabe noch läuft, sowie den aktuellen Stand an richtig / falsch gelösten Aufgaben der verschiedenen Teams

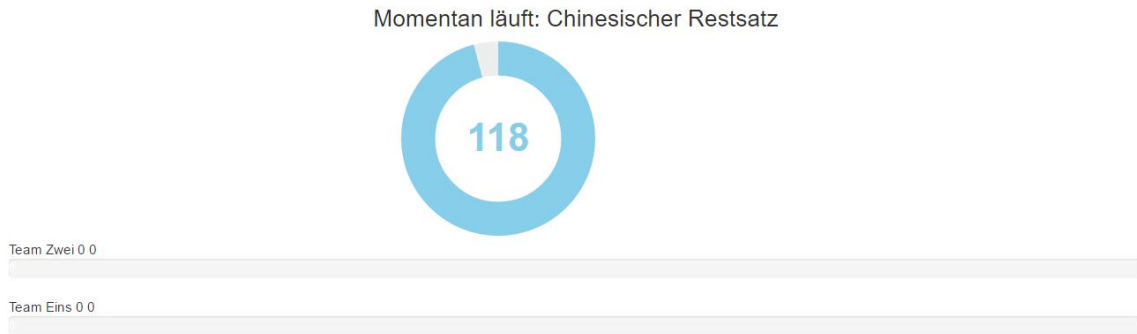
In der obersten Zeile ist immer zu sehen welcher Generator momentan läuft.

Um die Zeit die noch verbleibt darzustellen wurde die Bibliothek jQuery-Knob <sup>1</sup> verwendet, welche in Eigenarbeit in eine Angular 2 Komponente umgewandelt wurde um sie dann in die Übersicht Seite einzubetten.

Darunter ist abschließend der Stand der einzelnen Teams zu sehen. Die Namen der Teams werden über den Fortschrittsbalken angezeigt, welche das Verhältnis von richtig zu falsch gelösten Aufgaben darstellt. Zusätzlich ist, sobald die erste Aufgabe korrekt gelöst wurde, der absolute Stand der korrekt gelösten Aufgaben zu sehen.

<sup>1</sup><https://github.com/aterrien/jquery-Knob>

### 3 Ein Rundgang durch das Frontend



**Abbildung 3.11:** Übersicht Seite während einer laufenden Aufgabe



**Abbildung 3.12:** Fortschrittsbalken

Sollte die Zeit der Bearbeitung auslaufen ändert sich die Ansicht auf der Seite. Die Zeitanzeige verschwindet, und der Name des laufenden Generators wird mit der Info-Meldung ersetzt, dass zur Zeit keine aktive Aufgabe läuft. Die Anzahl der richtig und falsch gelösten Aufgaben der vergangenen Aufgabe wird allerdings weiterhin korrekt angezeigt und kann ausgewertet oder mit den Studenten besprochen werden.



**Abbildung 3.13:** Übersicht nach Ende der Aufgabe

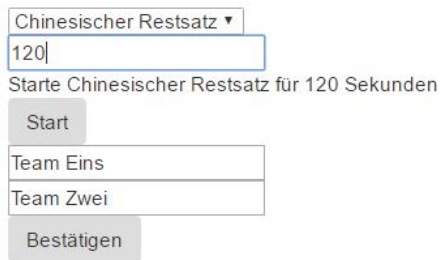
#### 3.2.2 Starten eines neuen Aufgaben-Generators

Damit das Programm in der Vorlesung genutzt werden kann reicht es allerdings nicht nur den Stand der laufenden Aufgabe anzuzeigen. Es ist ebenfalls notwendig einen neuen Generator starten zu können.

Dafür wurde ein Tab eingerichtet welcher unter dem Namen “Neuen Generator starten” erreichbar ist. Auf dieser Seite gibt es zwei verschiedene Ansichten, die je nach Status des Programmes gezeigt werden.

Bei der ersten Ansicht handelt es sich um die Ansicht die gezeigt wird wenn aktuell

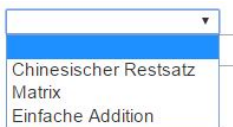
kein Generator am Laufen ist.



The screenshot shows a web form for starting a 'Chinesischer Restsatz' (Chinese Remainder Theorem) generator. At the top, there is a dropdown menu with 'Chinesischer Restsatz' selected. Below it is a text input field containing '120'. Underneath the input field, the text 'Starte Chinesischer Restsatz für 120 Sekunden' is displayed. To the right of this text is a 'Start' button. Below the 'Start' button are two text input fields for team names, labeled 'Team Eins' and 'Team Zwei'. At the bottom of the form is a 'Bestätigen' (Confirm) button.

**Abbildung 3.14:** Starten eines neuen Aufgaben-Generators

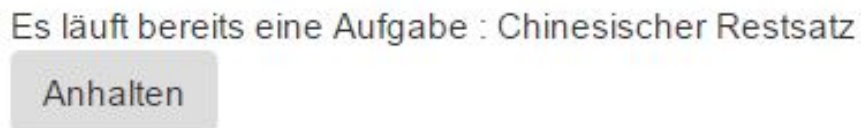
Die Ansicht besteht aus zwei Teilen. Der untere Teil besteht aus der Benennung der existierenden Teams. Der Administrator ist hier in der Lage in zwei Textfeldern die gewünschten Namen einzugeben. Danach bestätigt er die Eingabe durch Knopfdruck. Daraufhin werden die neuen Namen an den Server übertragen, welche diese nun für alle folgenden Generatoren verwenden wird.



The screenshot shows a dropdown menu for selecting a generator. The menu is open, showing three options: 'Chinesischer Restsatz' (highlighted in blue), 'Matrix', and 'Einfache Addition'.

**Abbildung 3.15:** Auswahl-Menü für den zu startenden Generator

Bei der Auswahl welcher Generator gestartet werden soll wurde wieder ein Drop-Down zur Auswahl des Generators gewählt. Die Daten die vom Server geholt werden, werden als Optionen zum Auswählen dargestellt, sollte der Start Knopf gedrückt werden ohne, dass ein Option ausgewählt wurde so erscheint eine Benachrichtigung für den Nutzer dies bitte zu korrigieren, selbiges geschieht bei einer ungültigen Angabe für die Laufzeit des Generators.



The screenshot shows a message box with the text 'Es läuft bereits eine Aufgabe : Chinesischer Restsatz'. Below the message is an 'Anhalten' (Stop) button.

**Abbildung 3.16:** Anzeige bei bereits laufendem Generator

Sollte aber bereits ein Generator gestartet sein wird eine andere Sicht gezeigt. Der Nutzer wird darüber informiert, dass bereits Aufgaben generiert werden. Er erhält

zudem den Namen des Generators und die Möglichkeit ihn vorzeitig abubrechen (zum Beispiel bei Start des falschen oder vorzeitigem Abbruch der Aufgabe)

#### 3.2.3 Bearbeitung der Helper-Funktionen

Es war von Anfang an klar, dass es Funktionalität gab die für viele Aufgaben-Generatoren benötigt werden würde. Daher wurden die Helper Funktionen eingeführt. Bei diesen handelt es sich um Funktionen welche erstellt werden und dann global verfügbar sind. Dadurch wird die Menge an überflüssigem Code der in jedem Generator erzeugt wird reduziert.

Bei den Helper-Funktionen handelt es sich um Funktionen, welche über den Tab “Helper-Funktionen bearbeiten” erstellt und bearbeitet werden können.

Wählen sie eine Entität aus dem Dropdown um sie zu bearbeiten

Bearbeiten einer bereits existierenden Helper-Funktion:

Name:

Wählen sie eine Entität aus dem Dropdown um sie zu bearbeiten

Script

```
if (! b) {
    return a;
}

return this.gcd(b, a % b);
```

Beschreibung

returns the greatest common divisor from a and b

Konstanten

Argumente

a,b

Testargumente

Testcount:

**Abbildung 3.17:** Ansicht zur Bearbeitung einer Helper-Funktion

Bevor man mit der Bearbeitung beginnt muss festgelegt werden ob es sich bei dem Skript um ein “Variablen” oder ein “MathJax” Skript handelt. Der Unterschied besteht darin, dass Variablen-Skripte dafür genutzt werden Variablen zu erzeugen, während MathJax-Skripte dabei helfen Variablen in MathJax umzuwandeln.

### 3 Ein Rundgang durch das Frontend

Um mit der Bearbeitung zu beginnen muss entweder eine zu bearbeitende Entität über das Dropdown-Menü ausgewählt oder ein Name eingegeben werden. Sobald dies erledigt ist müssen die anderen Felder ausgefüllt werden. Pflicht ist hierbei nur das Skript, sowie die Argumente die das Skript benötigt. Es wird allerdings empfohlen eine Beschreibung hinzuzufügen. Dies hilft dabei sich an die Funktionalität des Skriptes zu erinnern sollte man dies in Zukunft vergessen.

Sollte das Skript Konstanten benötigen können diese ebenfalls in der dafür vorgesehenen Spalte eingetragen.

Konstanten

primes:[2,3,5,7,11,13,17,19,23,29,31,37,41,43,47,53,59,61,67,71]
------------------------------------------------------------------

**Abbildung 3.18:** Beispiel zur Erstellung einer Konstanten

Sollte die Erstellung des Skriptes fertig sein kann getestet werden ob die erwarteten Ergebnisse produziert werden. Dafür muss der Administrator eingeben mit welchen Argumenten die Funktion aufgerufen werden soll und daraufhin den Test-Knopf betätigen. Dies stößt einen Prozess an der die Funktion in validen Javascript-kompiliert.

#### *Beispiel für eine transpilierte Funktion*

---

```
gcd:function(a,b){
if ( ! b) {
    return a;
}

return this.gcd(b, a % b);},
}
```

---

Diese Funktion wird an die global verfügbaren Objekte gehängt, Funktionen die bei der Erstellung von Variablen helfen an das “vf”-Objekt, Funktionen die die Erstellung von MathJax erleichtern an das “mj”-Objekt. Auf diese Objekte wird bei der Bearbeitung der Aufgaben-Generatoren noch einmal näher eingegangen.

Sobald der Transpilierungs-Prozess abgeschlossen ist wird die gerade erstellte Funktion mit den eingegeben Argumenten aufgerufen. Das Ergebnis wird daraufhin auf der Seite angezeigt um sich zu vergewissern, dass es den Erwartungen entspricht.

Wenn nun alle Tests den Erwartungen entsprechen und das Ergebnis zufrieden stellend ist besteht die letzte Aufgabe darin es zu persistieren. Dafür wurde ein Knopf neben dem Test-Knopf eingefügt welcher die Anfrage an den REST-Server stellt.

### 3.2.4 Bearbeitung der Aufgaben-Generatoren

Nachdem die Helper-Funktionen bearbeitet und erstellt werden können kann man sich nun dem Hauptpunkt zuwenden: Den Aufgaben-Generatoren. Die Bearbeitung wird über den Tab “Generator bearbeiten” gestartet.

Bearbeiten eines bereits existierenden Generators: Einfache Addition Reset Löschen

FormType: Zahlen

Name: Einfache Addition

Variable Skript

```
var a = vf.int(10,20)
var b = vf.int(10,20)
var solution = a+b
return {a:a, b:b, solution:solution}
```

Mathjax Skript

$a + b$

Test Solution

```
return variables.a + variables.b == userInput
```

Auto-Testen benutzen ☐

Testcount: 1

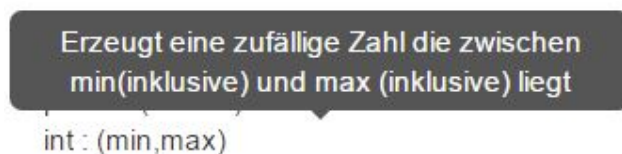
Test Create

vf:  
 prime : (bound)  
 int : (min,max)  
 mat : (x,y,fun)  
 dif : (fun)  
 gcd : (a,b)  
 tolerantSolution : (solution,userinput,tolerance)

**Abbildung 3.19:** Ansicht zur Bearbeitung einer Helper-Funktion

Ähnlich wie bei den Helper-Funktionen kann entweder ein bestehendes Skript über das Dropdown-Menü gewählt und editiert werden, ansonsten kann direkt mit der Editierung angefangen werden.

Bevor allerdings näher auf die Erstellung der drei verschiedenen Skripte eingegangen wird sollte zuerst die Nutzung der Helper-Funktionen näher erläutert werden. Jede dieser Funktionen wurde einem bestimmten Objekt zugeordnet, entweder dem “vf” (Variable-Functions) oder dem “mj” (MathJax) Objekt, je nachdem für welche Kategorie von Aufgabe sie erstellt wurden. Je nachdem welches der Felder nun ausgewählt wurde wird neben den Textfeldern angezeigt welche Funktionen vorhanden sind, so wie welche Parameter benötigt werden. Wenn nähere Informationen benötigt werden so lässt sich durch Bewegung der Maus über den Namen der Funktion die Beschreibung lesen.



**Abbildung 3.20:** Popup bei Bewegen der Maus über einen Funktionsnamen

### 3 Ein Rundgang durch das Frontend

Bei der Erstellung eines Aufgaben-Generators existieren drei Felder von denen alle Pflicht sind.

Bei dem Variablen-Skript handelt es sich um das Skript welches die Variablen erzeugt. Dadurch, dass in diesem Skript Funktionen genutzt werden können welche in der Lage sind zufällig Werte zu generieren wird gewährleistet, dass verschiedene Aufgaben erzeugt werden (auch wenn die Chance besteht, dass zwei generierte Aufgaben sich gleichen, diese Chance sinkt jedoch je größer der Zahlenraum der generierten Werte ist). Dieses Skript muss ein JSON-Objekt zurück geben welches alle Werte enthält die für die graphische Darstellung, sowie die Validierung der Lösung von Nöten sind.

Um die Erstellung von diesem Skript zu erleichtern wurde ein Transpiler-Feature eingebaut. Dieses sorgt dafür, dass jeder Code der von zwei Prozent-Zeichen umgeben wird zu einem Funktionsblock umgesetzt wird. Dies hilft dabei Code zu schreiben bei dem Funktionen aufgerufen werden die wiederum Funktionen als Parameter benötigen.

#### *Code vor der Transpilierung*

---

```
var mod1 = vf.prime(5)
var mod2= vf.dif(%vf.prime(5)%, mod1)
```

---

#### *Code nach der Transpilierung*

---

```
var mod1 = vf.prime(5)
var mod2= vf.dif(function(){return vf.prime(5)}, mod1)
```

---

Bei dem zweiten Kategorie handelt es sich um die graphische Darstellung. Der hier erzeugte Wert wird nachher von der MathJax-Bibliothek interpretiert und muss daher dessen Syntax folgen. Um die von dem ersten Skript erzeugten Werte zu nutzen werden diese über Dollar-Zeichen referenziert. So wird \$a immer durch den Wert ersetzt der für a errechnet wurde. Wenn hier Funktionen aufgerufen werden müssen, so sind diese immer durch Prozent-Zeichen von dem Rest des MathJax-Strings zu trennen. Der Transpiler erkennt diese und kann so den Code in gültigen JavaScript-Code übersetzen

Mathjax Skript

```
%mj.bmatrix(variables['matrix'])%
```

**Abbildung 3.21:** MathJax-Skript vor der Transpilierung

#### *Skript nach der Transpilierung*

```
return "$$ "+mj.bmatrix(variables['matrix'])+" $$"
```

---

Das von dem ersten Skript erzeugte JSON-Objekt ist im Code unter dem Namen “variables” referenzierbar.

Als letztes Skript wird die Validierung der Antworten des Nutzers benötigt. Diese Skript muss als Rückgabe einen Boolean Wert produzieren, welcher für die Korrektheit der übergebenen Antwort steht (also true bei korrekt und false bei falsch). Während der Validierung der Antwort stehen dem Skript zwei Variablen zur Verfügung.

1. Das Variablen-Skript erzeugte JSON-Objekt, unter dem Namen “variables”
2. Die vom Nutzer eingegebene Antwort, ein String unter dem Namen userInput”

Diese können nun dafür genutzt werden um die Nutzereingabe zu validieren und den korrekten Wert zurück zugeben. Für diese Skript gelten die selben Transpilierungs-Regeln wie für das Variablen-Skript.

#### 3.2.5 Der Fehler-Log

Eine wichtige Frage bei der Entwicklung einer Anwendung ist immer die Frage was alles schief gehen kann. Welche Fehler können auftreten und wie können sie behandelt oder sogar umgangen werden?

Einer der Hauptfehler der bei OpenTasks auftreten kann sind Fehler während der Generierung der Aufgaben. Da diese Skripte vom Nutzer geschrieben werden sind die Möglichkeiten diese zu überprüfen stark beschränkt. Eine der Möglichkeiten ist natürlich sie mehrmals laufen zu lassen und zu überprüfen ob ein Fehler zur Laufzeit erzeugt wird. Allerdings besteht hier immer die Gefahr, dass der Fehler der erzeugt wird genau bei diesen Testläufen nicht auftritt. Somit kann nie garantiert werden, dass alle Generatoren die sich in der Datenbank befinden korrekt sind und zur Laufzeit keine Fehler erzeugen. Die einzige Sache die durch solch empirische Tests gezeigt werden kann ist, dass die Wahrscheinlichkeit, dass die Generierung funktioniert recht hoch ist.

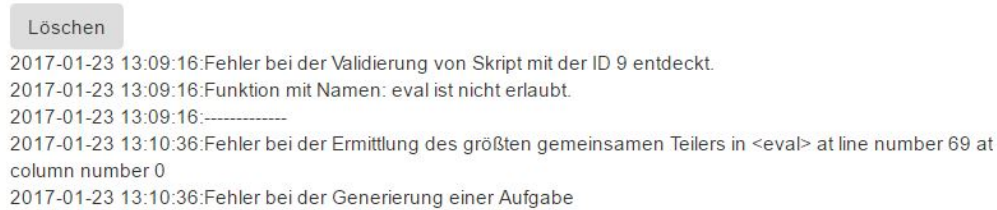
Was ist nun aber die Lösung für dieses Problem? In der Implementierung wird so vorgegangen, dass immer eine Aufgabe erzeugt wird. Sollte dieser Prozess fehlschlagen wird ein Fehler produziert. Dieser Fehler wird zum einen an den Nutzer (den Studenten) weitergeleitet und auf seiner Seite angezeigt. Zusätzlich wird jeder Fehler der registriert wird auf Server-Seite abgespeichert. Der Administrator hat nun die Möglichkeit in der Webanwendung auf den Tab “Fehler-Log” zu gehen. Sobald diese Seite besucht wird wird eine Anfrage an den Server gesendet. Dieser antwortet mit einem Datensatz der alle Fehler enthält die während der Laufzeit produziert wurden.



### 3 Ein Rundgang durch das Frontend

Daraufhin können diese evaluiert und der entsprechende Generator korrigiert werden, sodass dieser Fehler in Zukunft vermieden werden kann.

Die zweite Fehlerquelle ist die Validierung der Skripte. Da auf Server-Seite ebenfalls eine Validierung der Skripte statt findet (siehe Kapitel 4.1.2 Die Sicherung der Aufgaben-Generierung) müssen die Fehler die hier entdeckt werden für den Administrator sichtbar sein. Diese werden ebenfalls in dem Fehler-Log angezeigt und können dort nachgelesen werden.



Löschen

2017-01-23 13:09:16:Fehler bei der Validierung von Skript mit der ID 9 entdeckt.  
2017-01-23 13:09:16:Funktion mit Namen: eval ist nicht erlaubt.  
2017-01-23 13:09:16:-----  
2017-01-23 13:10:36:Fehler bei der Ermittlung des größten gemeinsamen Teilers in <eval> at line number 69 at column number 0  
2017-01-23 13:10:36:Fehler bei der Generierung einer Aufgabe

**Abbildung 3.22:** Der Fehler-Log

## 4 Ein Blick unter die Haube - das Backend

### 4.1 Die Sicherung der REST-API

Ein wichtiger Punkt bei der Entwicklung eines Backends ist immer die Sicherheit. Daher wird in diesem Kapitel darauf eingegangen wie in dieser Applikation die Sicherheit der REST-API, sowie die zusätzliche Sicherheit die auf dieser Seite für den Nutzer eingebaut wurde, erreicht wurde.

#### 4.1.1 Authentifizierung und Rollen-System

Eine der grundlegenden Fragen wenn es um Sicherheit geht ist die Authentifizierung. Für diese Anwendung wurden JSON Web Tokens gewählt um eine Authentifizierung zu gewährleisten.

In der Implementation bedeutet dies, dass bevor der erste Austausch von Nutzdaten erfolgen kann als erstes der Austausch des Tokens erfolgen muss. Von der Seite des Nutzers wird eine Anfrage gesendet welche Nutzernamen und Passwort beinhaltet. Dieses wird mit der Datenbank abgeglichen. Wird der Nutzer gefunden und ist das Passwort korrekt wird ein Token generiert und an den Anfrage-Steller zurück gesendet. Wichtig ist hierbei zu erwähnen, dass die Passwörter nur in kodierter Form in der Datenbank vorliegen (für diese Anwendung wurde als Kodierung der Algorithmus MD5 gewählt) und so gewährleistet wird, dass wenn ein fremder Lesezugriff auf die Datenbank auftreten sollte die Passwörter schwerer zu entschlüsseln sind.

Jedes mal wenn nun Daten abgefragt werden sollen schickt der Nutzer sein Token mit. Der Server überprüft die Validität des Tokens und wenn diese gewährleistet wurde erhält der Nutzer die angefragten Daten.

Allerdings ist dies nicht der einzige Mechanismus der in der REST-API eingebaut wurde. Jeder Nutzer hat eine Rolle welche in der Datenbank gespeichert ist. Bei der Authentifizierung wird ebenfalls die Rolle dieses Nutzers ausgelesen. Jede Rolle hat einen fest zugewiesenen Satz an Befugnissen, welche diesem Nutzer angehängt werden.

Der Großteil der Controller hat als Voraussetzung um die Daten abfragen zu dürfen vorher eine Abfrage ob die Person die aktuell die Daten anfragt die benötigte Befugnis

besitzt diese zu lesen. Ist diese Abfrage erfolgreich werden die Daten ausgeliefert, ansonsten kommt als Antwort lediglich ein 403 - Forbidden.

### 4.1.2 Die Sicherung der Aufgaben-Generierung

Neben der Sicherung der Rest-API an sich ist natürlich ein weiterer wichtiger Aspekt die Sicherung der Aufgaben-Generierung. Einerseits muss darauf geachtet werden, dass sich kein unerlaubter Code auf der Seite des Servers befindet und hier Schaden anrichten kann, andererseits ist es ebenfalls wichtig, dass keine Entitäten an den Administrator ausgeliefert werden die schadhaften Code enthalten (wenn er die Generatoren editiert und dabei auch testet).

Um dies zu bewerkstelligen wurde der selbe Ansatz wie im Frontend gewählt. Jeder Generator, sowie jede Helper-Funktion, wird bei Ankunft im Backend getestet. Dabei wird darauf geachtet welche Funktionsaufrufe getätigt wurden. Jeder einzelne wird mit der Liste von erlaubten Funktionen abgeglichen, sollte eine gefunden werden die nicht erlaubt ist wird das Speichern der Entität abgebrochen und ein Fehler an den Sender der Anfrage zurück geschickt.

Gleichzeitig wirft dieser Prozess eine Frage auf: Welchen Mehrwert hat es Code zweimal zu testen, beide Male mit der selben Methode? Um diese Frage zu beantworten ist es wichtig sich folgendes bewusst zu machen: Der Code auf der Client-Seite kann korrumpiert werden. Ein Nutzer mit dem Ziel Schaden zuzufügen könnte ihn editieren und so Anfragen schicken, welche normalerweise direkt auf Client Seite abgefangen und verboten werden würden. Sollte der Code allerdings nicht korrumpiert sein, so ist gute Praxis einen Generator bevor er abgeschickt wird zu testen, so wird der unnötige Austausch von Daten vermieden und dem Nutzer insgesamt eine flüssigere Erfahrung geboten.

## 4.2 Dokumentation der REST-Schnittstellen

### 4.2.1 Admin

Benötigte Befugnis : ADMIN

Pfad: admin/generators

Methode: GET

Beschreibung: Sucht alle Generatoren und gibt ihre Namen zurück

Rückgabe:

---

`["Euklidischer Algorithmus", "Matritzenmultiplikation"]`

---

Pfad: admin/generator

Methode: GET

Beschreibung: Sucht den Generator mit dem angegebenen Namen und gibt ihn zurück

URL-Parameter: name : Der Name des Generators

Rückgabe:

---

```
{
  "id": 19,
  "name": "Einfache Addition",
  "variableScript": "var a = vf.int(10,20)\nvar b = vf.int(10,20)\nvar
    solution = a+b\nreturn {a:a, b:b, solution:solution}",
  "solutionScript": "return variables.a + variables.b = solution",
  "mathjaxScript": "return \"$$ $a + $b $$\"",
  "formType": "Numbers"
}
```

---

## 4.3 Die Generierung und Validierung der Aufgaben

Kommen wir zu dem Hauptpunkt der Anwendung. Die Generierung und Validierung der Aufgaben. Jedes Mal wenn ein Generator angefragt wird wird dieser aus der Datenbank geladen. Die hier geladene Entität enthält allerdings nur Bruchstücke des Codes zur Generierung, wobei diese eindeutig reichen um den Generator eindeutig zu rekonstruieren. Sobald die Rekonstruktion abgeschlossen ist wird eine virtuelle Maschine gestartet innerhalb der Code ausgeführt und die Aufgaben generiert werden. Um die virtuelle Maschine(VM) zu starten und die Kommunikation zwischen Java und Javascript Code zu ermöglichen wird die `-NAME EINFÜGEN-` API genutzt, auf die hier nicht weiter eingegangen wird.

Sobald der Generator geladen wurde wird eine Rückmeldung an die Routine gegeben die ihn gestartet hat. Kommt nun eine Anfrage an den Server eine Aufgabe zu generieren wird dies an die Javascript-VM weitergeleitet. Der dafür benötigte Kontext wird hierbei der eindeutigen ID des Nutzers zugeordnet und abgespeichert. Bei erfolgreicher Generierung wird diese zurück gegeben, damit sie vom Nutzer bearbeitet werden kann, bei einem Fehler wird dieser Fehler in den Fehler-Log geschrieben und dem Nutzer die Meldung zurück gegeben, dass ein Fehler aufgetreten ist.

Wenn die Aufgabe bearbeitet wurde und die Lösung ankommt wird als erstes der korrekte Kontext vom Java-Code gesucht. Dafür wird geguckt ob eine Aufgabe mit der eindeutigen ID des Users verknüpft wurde. Wird keiner gefunden so wird ein Fehler ausgegeben. Sollte allerdings ein Kontext gefunden werden (was immer der Fall ist außer im Client-Seitigen Code liegt ein Fehler vor, außer die Bearbeitungszeit ist abgelaufen) so wird der Javascript-VM die Lösung, sowie der Kontext (Variablen

welche bei der Generierung erzeugt und für die Lösung benötigt werden) übergeben. Diese validiert die Lösung, sollte sie korrekt sein wird ein `true` zurück geben, ansonsten `false`. Der Java-Code nimmt diesen Wert nun an, verarbeitet ihn und gibt dem Nutzer die entsprechende Rückmeldung.

# **5 Fazit**

## **5.1 Das Ergebnis**

## **5.2 Wie kann die Anwendung verbessert / erweitert werden**

# Abbildungsverzeichnis

2.1	Der Grafik <a href="http://image.slidesharecdn.com/springsourceusi2009v3-0-090702135517-phpapp01/95/developing-modular-java-applications-13-728.jpg?cb=1246543977">http://image.slidesharecdn.com/springsourceusi2009v3-0-090702135517-phpapp01/95/developing-modular-java-applications-13-728.jpg?cb=1246543977</a> . . . . .	12
3.1	Login für den Nutzer / Teamauswahl . . . . .	19
3.2	Ansicht bei nicht geladener Aufgabe . . . . .	20
3.3	Information über keine laufende Aufgabe . . . . .	20
3.4	Beispiel für eine komplette Seite zur Bearbeitungszeit . . . . .	21
3.5	Das Dezimalformular . . . . .	22
3.6	Das Zahlenformular . . . . .	23
3.7	Das Matrixformular . . . . .	23
3.8	Das Matrixformular . . . . .	24
3.9	Das Bruchformular . . . . .	24
3.10	Navigationsleiste Admin-Bereich . . . . .	25
3.11	Übersicht Seite während einer laufenden Aufgabe . . . . .	26
3.12	Fortschrittsbalken . . . . .	26
3.13	Übersicht nach Ende der Aufgabe . . . . .	26
3.14	Starten eines neuen Aufgaben-Generators . . . . .	27
3.15	Auswahl-Menü für den zu startenden Generator . . . . .	27
3.16	Anzeige bei bereits laufendem Generator . . . . .	27
3.17	Ansicht zur Bearbeitung einer Helper-Funktion . . . . .	28
3.18	Beispiel zur Erstellung einer Konstanten . . . . .	29
3.19	Ansicht zur Bearbeitung einer Helper-Funktion . . . . .	30
3.20	Popup bei Bewegen der Maus über einen Funktionsnamen . . . . .	30
3.21	MathJax-Skript vor der Transpilierung . . . . .	31
3.22	Der Fehler-Log . . . . .	33

# Literaturverzeichnis

Spring-Dokumentation <https://spring.io/docs>, letzter Zugriff: 17.12.2016

Spring-Boot-Dokumentation <http://docs.spring.io/spring-boot/docs/current-SNAPSHOT/reference/htmlsingle/>, letzter Zugriff: 17.12.2016  
<https://blog.php-dev.info/2014/04/mariadb-vs-mysql/>  
<https://www.quora.com/What-are-the-advantages-and-disadvantages-of-using-Sinatra-vs-Express-for-a-web-service-API>  
<https://medium.freecodecamp.com/angular-2-versus-react-there-will-be-blood-66595faafd51.ufvfertqo>

Blu-ray Disc Association: *White paper Blu-ray Disc Format 2.B Audio Visual Application, Format Specifications for BD-ROM*, [http://www.blu-raydisc.com/Assets/downloadablefile/2b\\_bdrom\\_audiovisualapplication\\_0305-12955-15269.pdf](http://www.blu-raydisc.com/Assets/downloadablefile/2b_bdrom_audiovisualapplication_0305-12955-15269.pdf), 2005, letzter Zugriff: 1. 10. 2012

Dooley, Wesley L. & Streicher, Ronald D.: „M-S Stereo: A Powerful Technique for Working in Stereo“, *Journ. Audio Engineering Society* vol. 30 (10), 1982

Kuttruff, Heinrich: *Room Acoustics*, 3. Aufl., Elsevier 1991

Spehr, Georg (Hrsg.): *Funktionale Klänge*, transcript 2009

Sowodniok, Ulrike: „Funktionaler Stimmklang – Ein Prozess mit Nachhaltigkeit“, in: Spehr, Georg (Hrsg.): *Funktionale Klänge*, transcript 2009

Stephenson, Uwe: „Comparison of the Mirror Image Source Method and the Sound Particle Simulation Method“, *Applied Acoustics* vol. 29, 1990



Ich versichere, die vorliegende Arbeit selbstständig ohne fremde Hilfe verfasst und keine anderen Quellen und Hilfsmittel als die angegebenen benutzt zu haben. Die aus anderen Werken wörtlich entnommenen Stellen oder dem Sinn nach entlehnten Passagen sind durch Quellenangaben eindeutig kenntlich gemacht.

Ort, Datum

Patrick Hilgenstock