HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

Puzzles

Bachelorarbeit

Abteilung Informatik
Hochschule für Technik Rapperswil

Frühjahrssemester 2020

Autoren:	Anojan Shanmuganathan
	Manuel Sonder
Betreuer:	Prof. Dr. Markus Stolze
Gegenleser:	Prof. Stefan Richter
Experte:	Thomas Kälin

Abstract

Coding Plattformen erfreuen sich bei Studenten grosser Beliebtheit. Sie erleichtern den Einstieg in neue Programmiersprachen und Themengebiete. Ausserdem bieten sie bei Schwierigkeiten die Möglichkeit das entsprechende Themengebiet tiefer zu erarbeiten.

Bestehende Coding Plattformen erlauben nur teilweise die Erstellung von eigenen Aufgaben. Somit ist die Adaption auf den gewünschten Unterrichtsstoff, oder eine Vertiefung in einem bestimmten Gebiet schwierig. Für Cascading Style Sheets (CSS) existieren zwar verschiedene Plattformen, diese decken allerdings nur einzelne abgegrenzte Themenbereiche ab.

Diese Arbeit befasst sich mit der Implementation eines webbasierten Prototyps einer Coding Plattform, die die Strukturierung von Aufgaben in Kursen zulässt und zusätzlich auch die Definition von Meilensteinen für die Kurse erlaubt. Dabei soll die Erweiterung mit anderen Sprachen mit minimalem Aufwand verbunden sein. Ausserdem wurde ein Ansatz gesucht, um auch die Validierung von CSS-Aufgaben anzubieten.

Der entwickelte Prototyp bietet die Möglichkeit, CSS- und Javascript-Aufgaben zu lösen, kann jedoch mit geringem Aufwand um weitere Sprachen erweitert werden. Die Validierung von CSS-Aufgaben erfolgt visuell und erlaubt somit einen offenen Lösungsansatz. Die Ausführung der Tests findet hierbei in einem Docker Container statt.
Darüber hinaus ermöglicht der Prototyp dem Dozenten eine detaillierte Auswertung der Leistungen der Studenten. Somit kann bei Problemen in bestimmten Bereichen der Unterrichtsstoff oder -schwerpunkt dynamisch angepasst oder vertieft werden.

Inhalt

i. Abbildungsverzeichnis

ii. Änderungsgeschichte

Datum	Version	Änderung	Autor
22.03.2020	0.1 – 0.8	Erarbeitung Dokument	Anojan Shanmuganathan Manuel Sonder
10.06.2020	1.0	Fertigstellung	Anojan Shanmuganathan Manuel Sonder

iii. Konventionen

► Hinweise

```
Befehle/Commandos und Code Snippets (Lucid Consola)
```

[Sprache] | [Datei]

Querverweise innerhalb des Dokuments sind kursiv dargestellt

Links auf externe Ressourcen sind blau unterstrichen dargestellt

1 Management Summary

Ausgangslage
Coding Plattformen erfreuen sich grosser Beliebtheit. Sie bieten einen strukturierten und geführten Einstieg in neue Sprachen oder Themengebiete. Ausserdem können der Lernfortschritt und der Vergleich zu anderen Benutzern anhand der gelösten Aufgaben oder gesammelten Punkte einfach festgestellt werden.
Die Plattformen weisen jedoch die Limitierung auf, dass sie sich nicht oder nur bedingt in die HSR Infrastruktur integrieren lassen und dass sie nicht auf den Unterricht und deren Schwerpunkte angepasst werden können. Des Weiteren erlauben sie dem Dozenten keine Übersicht über allfällige Stolpersteine und dementsprechend eine allfällige Vertiefung im Themengebiet.

Vorgehen
Nachdem die Kernpunkte identifiziert wurden, wurden sie anhand einfacher Prototypen auf ihre Lösbarkeit untersucht. Hierbei wurden verschiedene Ansätze verfolgt und ausgewertet. Schlussendlich fiel die Wahl auf den Einsatz von Docker für die Validierung der Aufgaben, das Backend wurde in ExpressJS und das Frontend in Angular implementiert. Aufgrund der gewählten Vorgehensweise und Architektur konnten die Komponenten mit einer minimalen Kopplung zueinander implementiert werden. Ausserdem wurde der Kernpunkt der Erweiterbarkeit der Anwendung mittels Beweis überprüft.

Ergebnisse
Es konnte ein Prototyp entwickelt werden, der in der Lage ist, CSS- sowie Javascript-Aufgaben entgegenzunehmen und zu validieren. Die Aufgaben können in Kursen organisiert werden. Um den Fortschritt der Studenten zu steuern, können Meilensteine für die Kurse definiert werden. Die Anmeldung erfolgt entweder über GitHub oder über die Credentials der Hochschule für Technik Rapperswil. Bei CSS-Aufgaben wird die abgegebene Lösung visuell mit der hinterlegten Musterlösung verglichen. Die Bedienung erfolgt ausschliesslich im Browser und bedarf keinerlei Installation. Im Gegensatz zu bisherigen Plattformen bietet Puzzles den Vorteil CSS-Aufgaben in allen Bereichen abdecken zu können.

2 Aufgabenstellung

Aufgabenstellung Bachelorarbeit Abteilung I, FS 2020 Manuel Sonder, Anojan Shanmuganathan

Puzzle – Gamifizierte Platform für Aufgaben im Bereich Web-Engineering & Design

1. Betreuer

Betreuer dieser Arbeit ist
Prof. Dr. Markus Stolze mstolze@hsr.ch

Co-Referent dieser Arbeit ist
Prof. Stefan Richter

Externer Experte dieser Arbeit ist
Thomas Kälin, Noser

2. Ausgangslage

Online Platformen für Programmieraufgaben wie CodeWars www.codewars.com) und Online-Lernspiele zu Themen wie Layout mit CSS (z.B. GridGarden www.gridgarden.com) sind beliebt bei Studierenden. Bei CodeWars erhalten Studierende eine klar umrissene Programmieraufgaben welche mit JavaScript gelöst werden können. Die Aufgaben müssen so gelöst werden, dass eine Reihe von Unit-Tests erfüllt werden. Dabei gibt es vorgegebene (einsehbare) und weitere (versteckt und häufig zufällig generierte) Unit-Tests. Nach der erfolgreichen Lösung einer Aufgabe können Studierende ihre Lösung mit anderen Lösungen vergleichen, Bewertungen von Lösungen durch die Community einsehen und selber Bewertungen abgeben. Bei GridGarden müssen Studierende CSS-Grid Properties so setzen, dass ein gewünschter visueller Effekt erreicht wird. Studierende erhalten direktes Feedback dazu welches Layout durch die aktuell genutzten CSS-Grid Properties erreicht wird. Das direkte Feedback und der Vergleich und Analyse ihrer Lösung erlaubt Studierenden den eigenen Lernstand anhand motivierender Aufgaben zu überprüfen.

Wichtige Limitierung der aktuellen Platformen und Lernspiele ist, dass sie sich nur sehr schwierig in die HSR Lern-Infrastruktur integrieren lassen und dass sie vielfach nicht anpassbar sind auf neue Entwicklungen im Bereich Web-Engineering & Design. So ist es zum Beispiel nicht möglich das Lern-Spiel «Grid-Garden» um die Platzierung von Zellen mittels «Named Areas» zu erweitern.

3. Ziele der Arbeit

Ziel dieser Arbeit ist die Erstellung einer Online-Platform für Aufgaben aus dem Bereich Web-Engineering. Die Platform soll folgende Features umfassen - Details sind durch die Studierenden zu erheben und zu priorisieren.

- Dozent: Erstellen von Aufgaben inklusive automatischer Tests zum Thema JavaScript
  - Einfache Erstellung der Aufgaben mit Text und (Prio 2) Texte mit zusätzlichen Grafiken
  - Einfache Erstellung statischer Tests
  - Einfache Erstellung parametrisierter bzw. «gescripteter» Test.
    Prio 1: Konzept. Prio 2: Implementation
- Dozent: Erstellung von Aufgaben inklusive automatischer Tests zum Thema CSS (Selectors und Rules)
  - Einfache Erstellung der Aufgaben (Text mit Grafiken)
  - Einfache Erstellung statischer Tests
  - Analyse ob dynamische (parametrierte) Tests zum Thema CSS sinnvoll sein können z.B. CSS welches für unterschiedliche, dynamisch generiertes HTML funktionieren muss.
- Student: Lösen von JavaScript Aufgaben
  - Hilfe beim «Debuggen» von Code (warum nicht richtiges Resultat, warum abgestürzt)
  - Code Completion im Editor (Prio 3)
  - Vergleich der eigenen Lösung mit anderen Lösungen nachdem die eigene Lösung alle Tests bestanden hat.
  - Sichtung von Ratings für Lösungen entsprechend zu bestimmender Kriterien
  - Eigene Ratings vergeben
- Student: Lösen von CSS Aufgaben
  - Vorschlag von verschiedenen Hilfen beim «Debuggen» von CSS (Prio 1) und Implementation einer der vorgeschlagenen Hilfen (Prio 2)
  - Code Completion im Editor (Prio 2)
  - Vergleich der eigenen Lösung mit anderen Lösungen nachdem die eigene Lösung alle Tests bestanden hat.
  - Sichtung von Ratings für Lösungen entsprechend zu bestimmender Kriterien
  - Eigene Ratings vergeben
- Authentisierung von Studierenden und Dozenten
  - Entwicklung Konzept von Authentisierungs-Plug-In
  - Authentisierungs-Plug-In für HSR/Switch Single-Sign-On (Prio 2)
- Unterschiedliche Sichtbarkeit von Infos und unterschiedliche Zugriffes-/Modifikationsrechte für Dozenten und Studierende (Autorisation / Rollen) (Prio 2)
- Admin-Funktionen (Rollen-Zuteilung, bzw. Self-Service für Dozenten) (Prio 3)
- Ranking/Scoring von Studierenden:
  - Berechnung von Testresultaten auf dem Server um «Mogeln» schwieriger zu machen.
  - Dozent: Definition von Testat-Anforderungen durch Dozierende (z.B. 80% der Punkte einer gegebenen Menge von Aufgaben)
  - Dozent: Übersicht über Studierende in einem Modul. Möglichkeit Studierende mit Top-Score zu identifizieren (z.B. nach Score sortierte Liste von Studierenden in einem Modul)

4. Vereinbarte Rahmenbedingungen

Die Studierenden können sich Hilfe von Michael Gfeller (Mitarbeiter IFS) holen.

5. Dokumentation

Über diese Arbeit ist eine Dokumentation gemäss den Richtlinien der Abteilung Informatik zu verfassen. Die Dokumentation (inkl. Source-Code) ist vollständig entsprechend den Instruktionen des Studiengangs abzugeben. Zudem sind ein Download-Link für Prof. Dr. Markus Stolze sowie weitere Exemplare - nach Absprache mit dem Co-Referenten und dem Experten – bereitzustellen.

6. Weitere Regeln und Termine

Darüber hinaus gelten die allgemeinen Regeln zu Bachelor und Studienarbeiten „Abläufe und Regelungen Studien- und Bachelorarbeiten im Studiengang Informatik" (HSR Intranet): https://www.hsr.ch/Ablaeufe-und-Regelungen-Studie.7479.0.html

Der Terminplan ist hier ersichtlich (HSR Intranet): https://www.hsr.ch/Termine-Bachelor-und-Studiena.5142.0.html

6. Rechte

Die resultierende Software und Dokumentation soll möglichst als Open-Source Software publiziert werden, so dass sie ohne Einschränkungen sowohl von den Studenten, wie auch von der HSR weiter genutzt und erweitert werden kann. Bei der Programmierung soll darauf geachtet werden, dass möglichst keine Libraries mit viraler Open-Source Lizenz (z.B. GNU) genutzt werden.

7. Beurteilung

Eine erfolgreiche Bachelorarbeit zählt 12 ECTS-Punkte pro Studierenden. Für 1 ECTS Punkt ist eine Arbeitsleistung von ca. 25 bis 30 Stunden budgetiert. Entsprechend sollten ca. 350h Arbeit für die Bachelorarbeit aufgewendet werden. Dies entspricht ungefähr 25h pro Woche (auf 14 Wochen) und damit ca. 3 Tage Arbeit pro Woche, pro Student.

Für die Beurteilung ist der HSR-Betreuer verantwortlich. Die Bewertung der Arbeit erfolgt entsprechend der verteilten Kriterienliste.

Diese definitive Aufgabenstellung wurde am 15.4.2020 beschlossen.

Rapperswil, 15.4.2020
Prof. Dr. Markus Stolze, Institut für Software, Hochschule für Technik Rapperswil

Teil I

Bericht

3 Ausgangslage

Aktuell werden Übungen anhand von abgegebenem Code durchgeführt. Dies bedeutet für den Ersteller der Aufgaben einen beachtlichen Aufwand, da neben der eigentlichen Aufgabe auch die umgebende Struktur (Projekt, Code, etc.) aufgebaut und abgegeben werden muss. Ausserdem müssen Testataufgaben von Hand ausgewertet werden und es fällt dem Dozenten oder Übungsleiter je nachdem schwer, den jeweiligen Kenntnisstand der Kursteilnehmer festzustellen.
Zusätzlich finden die Studenten zwar Tutorials im Internet. Diese variieren jedoch stark in der Qualität und haben unter Umständen einen anderen Fokus als der Unterricht oder decken gewisse Teilgebiete gar nicht ab.
Es existieren diverse online Plattformen (wie CodeWars[1] oder GridGarden[2]), die das Erlernen von Programmiersprachen unterstützen.
Die Plattformen weisen jedoch die Limitierung auf, dass sie sich nicht oder nur bedingt in die HSR Infrastruktur integrieren lassen und dass sie nicht auf den Unterricht und dessen Schwerpunkte angepasst werden können. Des Weiteren erlauben sie dem Dozenten keine Übersicht über allfällige Stolpersteine. Darüber hinaus bieten die existierenden Plattformen für CSS nur abgegrenzte Themenbereiche an (Bsp.: Grid).

3.1 **Vision**

Es soll eine Plattform entstehen, die auf die jeweiligen Bedürfnisse und Schwerpunkte angepasst werden kann. Die Plattform soll neben Javascript auch CSS-Aufgaben validieren können. Die Grundfunktionalität soll ähnlich wie jene von CodeWars sein; es sollen öffentliche Aufgaben erstellt werden können, die durch vordefinierte Unit Tests (oder visuell bei CSS-Aufgaben) validiert werden. Neben den öffentlichen Fragen und Aufgaben sollen auch kursbezogene Aufgaben sowie Testate eröffnet werden können. Dafür sollen Kurse eröffnet werden, für die sich Studenten einschreiben können. Der Dozent erhält einen Einblick in den Lernfortschritt und allfällige Probleme der Studierende und kann den Schwerpunkt des Unterrichts entsprechend anpassen. Zusätzlich soll die Plattform einfach um andere Sprachen (C#, Java, etc.) erweitert werden können.

[1] www.codewars.com
[2] www.gridgarden.com

3.2 Umgebende Arbeiten und Projekte

Dieses Unterkapitel betrachtet umgebende Arbeiten, die gewisse Teilaspekte oder Ziele dieser Arbeit abdecken.

3.2.1 AutoExercise

Die Bachelorarbeit «AutoExercise» (Durmaz & Alfred, 2018) befasst sich mit der automatisierten Auswertung von Programmieraufgaben mittels Docker und einer Webanwendung, welche die Ergebnisse zusammenfasst und grafisch aufarbeitet. Für die Lösung der Übungen muss sich der Student ein Repository auf den Rechner klonen, die Auswertung erfolgt mittels Commit/Push in das Repository.

Für den visuellen Vergleich von Aufgaben (CSS) bietet AutoExercise allerdings keine Out-of-the-box-Lösung an. Man hätte ein entsprechendes Docker Image erstellen können, das den abgegebenen Code mit einer Musterlösung vergleicht. Jedoch existiert im Fehlerfall keine Möglichkeit dem Benutzer mitzuteilen, welcher Teil seiner Lösung nicht stimmt. Die Rückmeldung beinhaltet lediglich die erreichte Punktzahl gemäss dem Bewertungsraster des Dozenten und keinerlei Hinweise oder Feedback zu den Tests.
Ein weiterer Unterschied liegt darin, dass der Autor einer Übung das dazugehörige Dockerfile jeweils selbst erstellen und hochladen muss.
Bei der Bearbeitung von Aufgaben muss sich der Student das gesamte Repository via Batch Ausführung auf seine Maschine klonen. Dies hat den Vorteil auch komplexere Aufgaben (inkl. Datei- und Ordnerstruktur) anbieten zu können.

Puzzles bietet den Vorteil, dass keinerlei «Installation» auf dem Client notwendig ist. Ausserdem ist die Konfiguration eines Docker Images für einen Aufgabentyp einmalig und muss nicht von jedem Dozenten durchgeführt werden. Puzzles soll dabei helfen die Grundlagen einer Sprache oder bestimmte Gebiete davon zu erarbeiten. Sobald eine Datei- oder Ordnerstruktur benötigt wird, gerät Puzzles an seine Grenzen. Der grösste Unterschied liegt allerdings darin, dass Puzzles einen visuellen Vergleich inklusive Rückmeldung dazu erlaubt. Somit können auch CSS-Aufgaben gelöst werden. Darüber hinaus erlaubt Puzzles im Gegensatz zu AutoExercise die Definition von Testaten / Meilensteinen.

3.2.2 Jupyter Notebook

Jupyter Notebook (früher "Ipython Notebook") ist eine Anwendung für die interaktive Darstellung von Code(-Fragmenten) und anderen Visualisierungen in einer Web-Ansicht. Mittlerweile wird eine beachtliche Anzahl an Programmiersprachen unterstützt. Unter anderem auch Javascript, PHP, C# und Haskell. (Jupyter, 2020)
Jupyter hatte allerdings nie zum Ziel als «Übungslöser» oder als Weg um eine Programmiersprache zu erlernen, verwendet zu werden. Es bietet eine Methode, um Daten nachvollziehbar auszuwerten, zu analysieren und zu visualisieren. Es ist in gewissen Zügen vergleichbar mit dem Produkt «Microsoft Excel». Beide ermöglichen Berechnungen in Tabellenform und die Darstellung in Grafiken oder Diagrammen.
Jupyter eignet sich auch hervorragend, um Tutorials anzufertigen oder gewisse Aspekte einer Sprache genauer zu beleuchten.
Folgendes Beispiel ist in Javascript verfasst:

In [6]:
```
path.pointRadius(1.5);
svg3 = svg2
    + '<path d="'
    + path(topojson.feature(airports, airports.objects.airports))
    + '"/>';
$$.svg(svg3);
```

Out[6]:

Abbildung 3-1: JS Notebook (Jupyter, 2020)

Weitere Beispiele können im Wiki[3] vom offiziellen GitHub Repository von Jupyter eingesehen werden.

Jupyter würde sich also theoretisch eignen, um Studenten das Verständnis von Sprachen zu vermitteln. Allerdings besteht (noch) keine Möglichkeit, reines CSS in einem Notebook einzubinden. Des Weiteren bietet Jupyter keine Möglichkeit, die Ergebnisse zu validieren.

[3] https://github.com/jupyter/jupyter/wiki

4 Lösungskonzept

In diesem Kapitel wird das Konzept beschrieben wie die Vision erreicht werden soll. Die Anforderungsspezifikation kann dem Kapitel *Anforderungsanalyse* entnommen werden. Designentscheidungen und die Architektur sind im Kapitel *Architektur und Design* aufgeführt.

4.1 Grundstruktur

Das System kann in drei Hauptkomponenten aufgeteilt werden:

Frontend ist für die Darstellung zuständig und enthält minimale Programmlogik. Das Frontend ist in Angular geschrieben und wird auf mit einem Apache Tomcat Server bereitgestellt. Es übernimmt die Authentifizierung (OAuth) und kommuniziert einerseits mit dem/den Client/s und andererseits mit dem Backend.

Backend ist für die Daten und deren Aufbereitung und Persistenz zuständig. Das Backend ist in Javascript implementiert und wird via NodeJS Webserver bereitgestellt. Es stellt dem Frontend eine API zur Verfügung.

Runner ist für das Ausführen der Validierung (Tests) verantwortlich, dafür werden Docker Container verwendet. Für die Erweiterung mit einer anderen Sprache ist ein neuer Docker Container notwendig.

> ► Details zur Systemarchitektur können dem Kapitel *Architektur und Design* entnommen werden.

4.2 Aufgabenvalidierung

Die Kernlogik liegt in der Validierung der Aufgaben. Dafür werden Docker Container verwendet. Dies fördert eine lose Kopplung und erleichtert somit die Erweiterung um andere Sprachen, darüber hinaus wird durch die Kapselung im Docker Image die Sicherheit erhöht. Das Ausführen von eingegebenem Code ist sicherheitstechnisch nicht einfach und bedarf sorgfältiger Planung.

Javascript Aufgaben sollen anhand vordefinierter Tests validiert werden. Die Validierung erfolgt dabei gänzlich im dafür vorgesehen Docker Container. Lediglich die Testresultate gelangen wieder vom Docker zum Backend und werden anschliessend wieder zurück ans Frontend geliefert.

Die Validierung von CSS-Aufgaben stellt zudem eine weitere Herausforderung dar, wie anhand vom folgenden einfachen Beispiel deutlich wird:
Als Ausgangslage sei der folgende Code gegeben:

```
<style>
    .warning {
                ???
    }
</style>
<p class="warning">
    Puzzles
</p>
```

HTML/CSS

Ziel ist es, den Text rot zu färben. Dies kann mit verschiedenen Ansätzen erreicht werden:

Lösungsvariante 1:

```
<style>
    .warning {
                color: red;
    }
</style>
<p class="warning">
    Puzzles
</p>
```

HTML/CSS

Lösungsvariante 2:

```
<style>
    .warning {
                color: rgb(255, 0, 0);
    }
</style>
<p class="warning">
    Puzzles
</p>
```

HTML/CSS

Beide Varianten erfüllen die Aufgabenstellung. Falls die Überprüfung mit Regex oder durch Abfrage auf bestimmte Attribute erfolgt, erhöht das die Komplexität enorm, da jede mögliche Lösung abgefangen werden muss. Diese Art der Validierung eignet sich somit nur für äusserst einfache Aufgabenstellungen.
Ausserdem stellt sich die Frage wie die folgende Lösungsvariante bewertet wird:

```
<style>
    .warning {
                color: rgb(255, 1, 1);
    }
</style>
<p class="warning">
    Puzzles
</p>
```

HTML/CSS

Visuell ist kaum einen Unterschied festzustellen. Bei einer Validierung via Code/Regex müsste man alle möglichen Werte definieren. Dies gestaltet sich allerdings als zeitraubend und ineffizient.
Aus diesem Grund sollen CSS-Aufgaben visuell überprüft und validiert werden. Der Runner soll den abgegebenen Code sowie die Musterlösung rendern und Bilder mit verschiedenen

Fenstergrössen erstellen. Anschliessen werden die Bilder verglichen. So ist der Lösungsweg frei wählbar, wodurch auch komplexere Aufgabenstellungen abgedeckt werden können.

4.3 Parametrisierte Tests

Durch eine clevere Wahl des Testframeworks können parametrisierte Tests vergleichsweise einfach angeboten werden.
Bei der Erstellung einer Aufgabe müssen die Tests lediglich in folgender Form angegeben werden (Beispiel mit Testframework «jest»[4] für Javascript-Aufgaben):

```
describe('test multiplicate method', () => {
    it.each`
      a     | b     | result
      ${7}  | ${3}  | ${21}
      ${20} | ${3}  | ${60}
      ${-2} | ${-3} | ${6}
    `('should return $result when $a and $b are used', ({a, b, result}) => {
      expect(multiplicate(a, b)).toEqual(result);
    });
  });
```

Javascript (Jest)

4.4 Authentisierung

Damit die Applikation möglichst benutzerfreundlich ist, soll die Authentisierung mit bereits vorhandenen Credentials erfolgen (SSO). Aufgrund der im Kapitel *10.1.1 Designentscheidung: OAuth* begründeten Entscheidung fiel die Wahl auf OAuth.

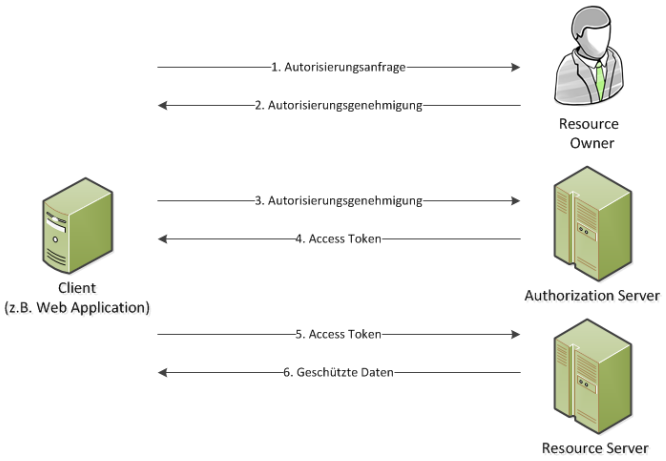


Abbildung 4-1: OAuth Flow (Wikipedia, 2020)

Um Erfahrung zu sammeln, soll in einem ersten Schritt OAuth mit GitHub als IDP realisiert werden. Anschliessend soll die Plattform so angepasst und erweitert werden, dass auch das Azure Active Directory (AAD) der Hochschule für Technik Rapperswil eingebunden werden kann.

[4] https://jestjs.io/

5 Implementation

Das folgende Diagramm zeigt den vereinfachten Ablauf bei der Validierung einer Aufgabe. Aus Gründen der Übersichtlichkeit wurden hier nicht alle alternativen Möglichkeiten abgebildet, sondern lediglich der Base Case:

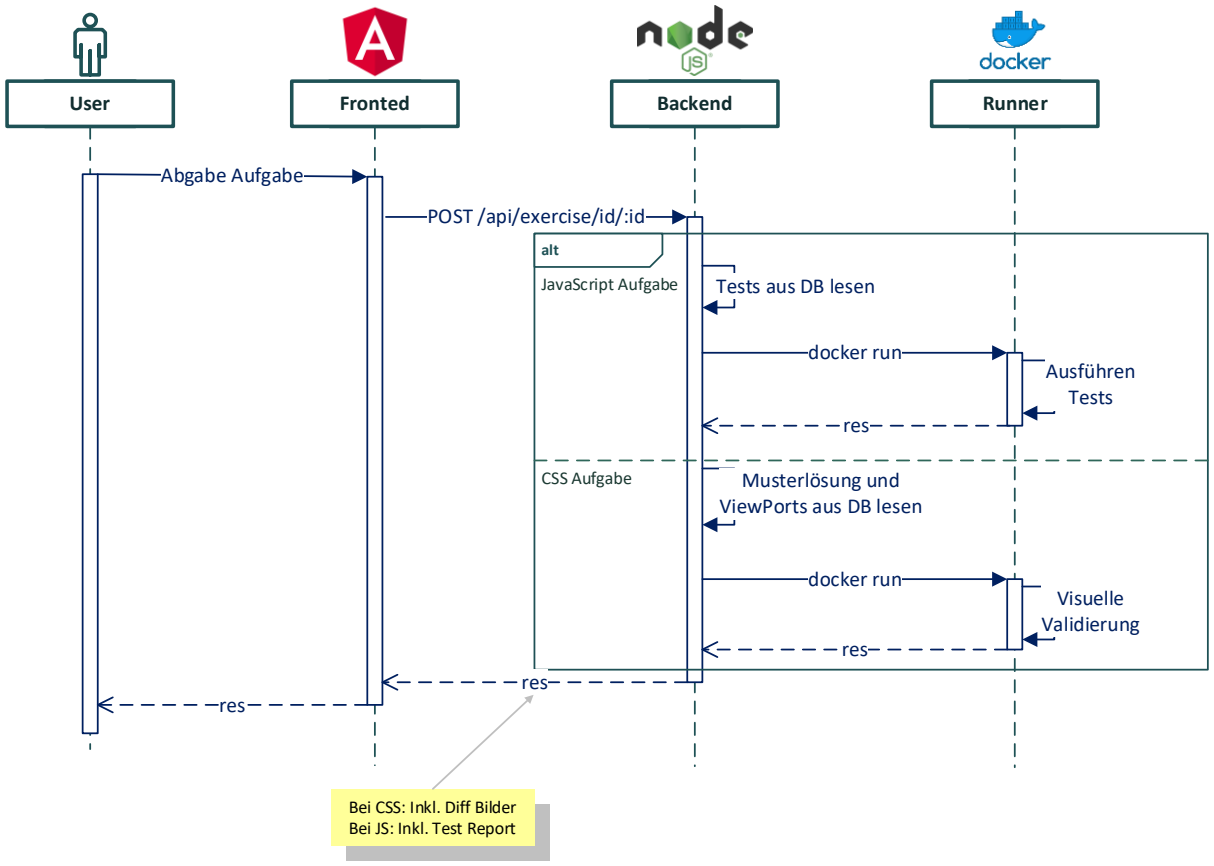


Abbildung 5-1: Seq-Chart Validierung (Base)

Das Frontend nimmt die Eingaben des Benutzers entgegen und schickt diese mit dem JSON Web Token (JWT) zusammen an das Backend, hierbei wird der Code «escaped». Das Backend lädt die relevanten Informationen aus der Datenbank und stösst den Runner mit dem entsprechenden Docker Image an. Nachdem die Tests durchgeführt wurden, meldet das Backend das Resultat zurück ans Frontend.

5.1 **Authentisierung**

Damit verschiedene Identity Provider unterstützt werden können, wurde folgendes Vorgehen ausgearbeitet:

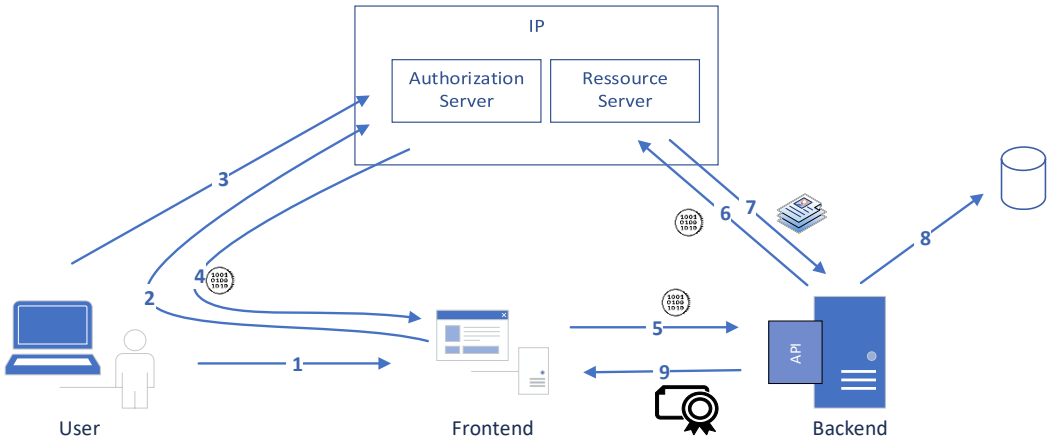


Abbildung 5-2: Authentication Flow Puzzles

Abbildung 5-3: Legende zu Authentication Flow Puzzles

1. Aufruf der Applikation (nicht angemeldet) und Klick auf den Login-Button
2. Applikation schickt einen Redirect zum IDP (Authorization Server)
3. Access Grant durch Benutzer
4. Callback mit Access Token an das Frontend
5. Weiterleiten des Access Token an das Backend
6. Abfragen der Ressourcen beim Ressource Server mit dem erhaltenen Access Token
7. Auslieferung der Ressourcen durch den Ressource Server
8. Persistieren der Benutzerdaten in der Datenbank
9. Auslieferung JWT (JSON Web Token) durch das Backend an das Frontend (und anschliessend durch das Frontend an den Benutzer)

Bei jedem Request erwartet das Backend einen gültigen JWT. Anhand der im JWT enthaltenen Daten wird überprüft, ob der Benutzer die angeforderte Operation ausführen darf oder nicht.
Der Vorteil dieses Vorgehens besteht darin, dass der IDP mit minimalem Aufwand ausgetauscht werden kann. Das Plugin im Frontend muss dem Backend neben dem Access Token (Schritt 5) allerdings zwei weitere Informationen mitteilen:

- URL des Ressource Servers (IDP spezifisch)
  Damit das Backend die Ressourcen beziehen kann
- Name der benötigten Felder (IDP spezifisch)
  Damit das Backend die entsprechende Ressource abfragen bzw. herausfiltern kann

Sämtliche Informationen werden in diesem Schritt im Request Body übermittelt.

5.2 **Frontend**

Das Frontend wurde in Angular geschrieben und wurde auf einem Windows Server implementiert. Hierzu wurde ein Apache Tomcat Webserver installiert. Sämtliche Kommunikation zwischen Benutzer und Frontend sowie Frontend und Backend ist mit http über TLS/SSL verschlüsselt. Für die Gestaltung der Benutzeroberfläche wurde «Bootstrap»[5] mit «Font Awesome»[6] verwendet.

5.2.1 Code Editor

Für das direkte Editieren von Code im Web Browser wurde der Web Code Editor «Ace»[7] verwendet. Ace unterstützt «syntax highlighting» und «auto completion» für über 110 Sprachen und kann einfach nach den eigenen Wünschen angepasst werden. Der Hauptgrund für die Auswahl von Ace ist die hohe Benutzerfreundlichkeit und die einfache Einbindung in Angular.

5.2.2 Factory Method Pattern

Um mehrere Sprachen im Frontend zu unterstützen und zusätzlich redundanten Code zu vermeiden, wurde eine Factory Methode entwickelt. Dadurch muss bei einer Erweiterung nur eine weitere Subklasse implementiert werden. Die Benutzeroberfläche wird anhand der vorhandenen Informationen dynamisch zusammengestellt und aufgebaut.

5.2.3 Meilensteine vs. Testate

Auf Wunsch des Betreuers wurde im Frontend der Begriff «Testat» in «Meilenstein» geändert. Aus Zeitgründen war es allerdings nicht möglich, dies auch im Backend anzupassen.

> ► Da der Begriff «Meilenstein» besser passt, wurde das *Product Backlog* mit einem entsprechenden Arbeitspaket ergänzt.

5.3 **Backend**

Aufgrund der Verwendung von Docker wurde das Backend sowie der Runner auf einem Linux-Server implementiert. Docker weist auf Linux eine deutlich bessere Performance auf, ausserdem wird die Konfiguration sowie das Testen signifikant erleichtert. Für das Backend selbst wurde ExpressJS[8] verwendet.
Für die asynchronen Aufrufe (Datenbank, Runner) wurde das Promise-Pattern angewendet. Wo möglich wurden die Abfragen parallelisiert.

[5] https://getbootstrap.com/
[6] https://fontawesome.com/
[7] https://ace.c9.io/
[8] https://expressjs.com

5.3.1 Logging

Das Backend bedient grundsätzlich drei Logsenken. Einerseits die Konsole, welche je nach Variante des Startens vom Backend gar nicht ausgegeben bzw. angezeigt wird. Andererseits auch noch zwei Dateien. Das erste Log «`puzzles.log`» enthält Systemrelevante Informationen und Fehlermeldungen. Das zweite Log «`puzzles_access_logYYYY-MM-DD.log`» enthält, wie es der Dateiname beschreibt, das Zugriffslog auf das Backend.
Via Anpassung des Parameters «`LOG_LEVEL_FILE`» bzw. «`LOG_LEVEL_CONSOLE`» kann der entsprechende Log-Level angepasst werden.

5.3.2 Views / ORM

Zu Beginn wurde kein ORM eingesetzt. Grund dafür war vor allem die Zeiteffizienz. Nach anfänglichen Untersuchungen und einfachen Prototypen wurde entschieden, die Abfragen in SQL auszuformulieren sowie die Beziehungen mittels Datenbanklogik abzubilden und via Data Access Layer (Services) auf die Daten zuzugreifen. Durch die Definition von Views konnten die Abfragen auf ein Minimum reduziert werden. Ausserdem wurde darauf geachtet die SQL-Abfragen «neutral» zu formulieren, in diesem Kontext bedeutet das, dass keine herstellerspezifischen Elemente (Dialekt) verwendet wurden.
Ein Beispiel hierzu wäre die Abfrage von den ersten zehn Werten (Zeilen) einer Tabelle. Je nach Hersteller geschieht dies mit einem «`SELECT TOP(10) …;`» am Anfang des Statements oder einem «`SELECT … LIMIT(10);`» an dessen Ende.
Ein weiterer Nachteil eines ORM ist, dass ein gewisser Teil der Logik, die durch die Datenbank übernommen wurde (durch Views, Joins, und FK-Beziehungen), durch zusätzliche Logik im Source Code abgefangen werden muss.

Nach dem Code Review wurde allerdings entschieden, trotz des erhöhten Mehraufwands nachträglich einen ORM einzusetzen.
Dies vor allem, da die Beziehungen zwischen den Models dadurch einfacher zugreifbar wurden. Darüber hinaus wurden die CRUD Operationen vereinfacht und die Kopplung zur Datenbank verringert. Ausserdem wird ein allfälliger Technologiewechsel beim Datenbankhersteller vereinfacht.
Nichtsdestotrotz war die Abbildung gewisser Beziehung sehr aufwändig und wurde teilweise nicht in den Modellen, sondern in den Abfragen implementiert. Gerade bei komplexeren Beziehungen oder Abfragen, wie beispielsweise Joins mit Bedingungen über mehrere Spalten, erwies sich die Abbildung der Beziehungen in den Abfragen als signifikant einfacher.

5.3.3 Validierung

In der Konfiguration vom Backend ist spezifiziert, welcher Aufgabentyp zu welchem Docker Container gehört. Das Backend startet via CLI-Befehl einen Docker und übergibt diesem als Parameter die Tests sowie die Aufgabe. Sämtlicher Code ist hierbei «escaped».
Bei der Aufgabenerstellung sowie bei dessen Überarbeitung wird die Aufgabe ebenfalls validiert und somit auf ihre Lösbarkeit überprüft.

5.4 **Runner**

Der Runner beinhaltet Skripte und Dateien, die bei der Erzeugung des Docker Images kopiert werden. Je nach Sprache muss ein entsprechender «Reporter» eingebunden werden (Pfad: `./Runner/src/language/[lang]`). Der Docker Container schreibt den Output in den «`stdout`». Dieser Output kann je nach eingestelltem Reporter angepasst werden. Anhand bestimmter Keywords kann schliesslich auf das Testresultat zurückgeschlossen werden.

5.4.1 Validierung CSS

Bei der CSS-Validierung wird innerhalb des Docker Containers ein Bild pro spezifizierten Viewport erzeugt. Hierzu werden der abgegebene Code sowie das Lösungsbeispiel innerhalb des Docker Images in eine HTML-Datei geschrieben. Anschliessend werden die Dateien mittels Headless Browser geöffnet und gerendert. Nachdem die Fenstergrösse eingestellt wurde, wird jeweils ein Screenshot erzeugt. Dies wird für alle Viewports wiederholt. Bei diesem Vorgang wurde ein Zeitlimit eingebaut. Dieses Limit kann in der Konfiguration des Runners angepasst werden. Falls bis zum Ablauf dieser Zeit kein Bild erstellt werden konnte, wird die Aufgabe als falsch angesehen. Nachdem die Bilder angelegt wurden, werden sie miteinander verglichen. Daraus resultiert ein Diff Bild. Somit entstehen pro Viewport jeweils drei Bilder, welche zu einem einzigen Bild mit Überschriften zusammengefügt werden.

5.4.2 Validierung Javascript

Für die Validierung von Javascript Aufgaben werden zuerst die Eingaben des Benutzers und die Tests innerhalb des Docker Images in eine Datei geschrieben. Anschliessend wird ein Child-Prozess erstellt, in welchem die Tests ausgeführt werden. Für die Ausgabe wurde der Reporter so adaptiert, dass er die Testresultate und das Endresultat in den Standardoutput schreibt. Zusätzlich dazu wird auch ein Testreport im HTML-Format erstellt. Dieser soll als Unterstützung für den Benutzer im Frontend angezeigt werden. Ausserdem wurde aus Sicherheitsgründen ein Timeout eingebaut. Somit wird sichergestellt, dass bei einem Fehler oder gar einer Attacke im Child-Prozess, der Parent-Prozess nicht vergeblich wartet.
Als Testing Framework wurde Jest ausgewählt. Der Grund für diese Entscheidung liegt daran, dass Jest als Testrunner, Assertion Library sowie auch als Mocking Library fungiert.

5.4.3 Dockerfile

Mittels Dockerfile wurde ein Image erstellt. Als Basis wird das Image von Node[9] verwendet, dieses installiert Node auf einem Linux OS. Nachdem das Basis Image importiert wurde, wird ein Benutzer angelegt, der die Ausführung der Tests sowie die Generierung der Bilder übernimmt. Zusätzlich werden alle notwendigen Libraries für Linux installiert. Darüber hinaus wird der Source Code vom Runner in das Docker Image hinüberkopiert und der Befehl «npm install» ausgeführt. Dadurch wird z.B. das Testing Framework Jest für Javascript und der Headless Browser für CSS installiert. Um die erstellten Bilder und Testreports vom Docker Image zu erhalten, wurde ein Ordner im «tmp»-Verzeichnis erstellt. Mittels Volumes[10] werden die Daten von dort aus zurück zum Server geschickt. Am Ende des Dockerfiles wird ein Shellscript angegeben, welches den Runner anstösst und als Einstiegspunkt dient.

[9] https://hub.docker.com/_/node/
[10] https://docs.docker.com/storage/volumes/

6 Resultate

Es konnte ein Prototyp entwickelt werden, der CSS- sowie Javascript-Aufgaben entgegennimmt und validieren kann. Für die Anmeldung werden GitHub und das Azure Active Directory der HSR als IDP angeboten.

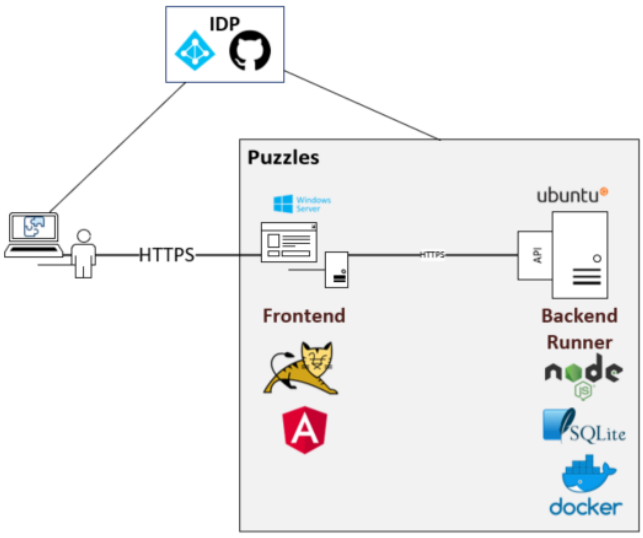


Abbildung 6-1: Grobübersicht Puzzles

Hierbei verteilen sich die Hauptkomponenten Frontend, Backend und Runner auf zwei voneinander getrennte Server.
Die API-Spezifikation kann unter der Base URL des Backendservers aufgerufen werden. Die Base URL besteht aus dem Servernamen und dem konfigurierten Port. In diesem konkreten Beispiel lautet sie https://puzzles-backend.hsr.ch:40000.

6.1 Validierung

Die Validierung von Javascript-Aufgaben erfolgt mittels dem Testframework jest[11].
Dementsprechend lassen sich auch parametrisierte Tests definieren:

```
describe('test filterArr method', () => {
  it.each`
    a               | result
    ${[]}           | ${[]}
    ${[2]}          | ${[]}
    ${[0]}          | ${[]}
    ${[0, 1, 2, 3]}| ${[1, 3, 5]}
    ${[-2, -1, 0]} | ${[-1, 1]}
    `('should return $result when $a is used', ({ a, result }) => {
    expect(filterArr(a)).toEqual(result);
  });
});
```

Javascript | Jest

[11] https://jestjs.io/

Das Bearbeiten der Aufgaben erfolgt in einem Code Editor, der unter anderem auch Auto Completion unterstützt. Nach der Validierung wird das Resultat (erfüllt/nicht erfüllt) im oberen Teil des Bildschirms angezeigt. Im unteren Teil des Bildschirms wird ein Testreport publiziert.

Bei CSS-Aufgaben können verschiedene Viewports definiert werden. Bei der Validierung wird der abgegebene Code in einem Headless Browser (im Docker) gerendert und visuell mit der Musterlösung verglichen. Der Benutzer erhält bei der Validierung ein dreiteiliges Bild, das aus der abgegebenen Lösung, dem Diff-Bild und der Musterlösung besteht.
Damit der Benutzer seine Lösung einsehen und gegebenenfalls bereits vorher korrigieren kann, besteht die Möglichkeit, den Code ohne Einbezug vom Runner im Browser zu rendern (rechts vom Code Editor).

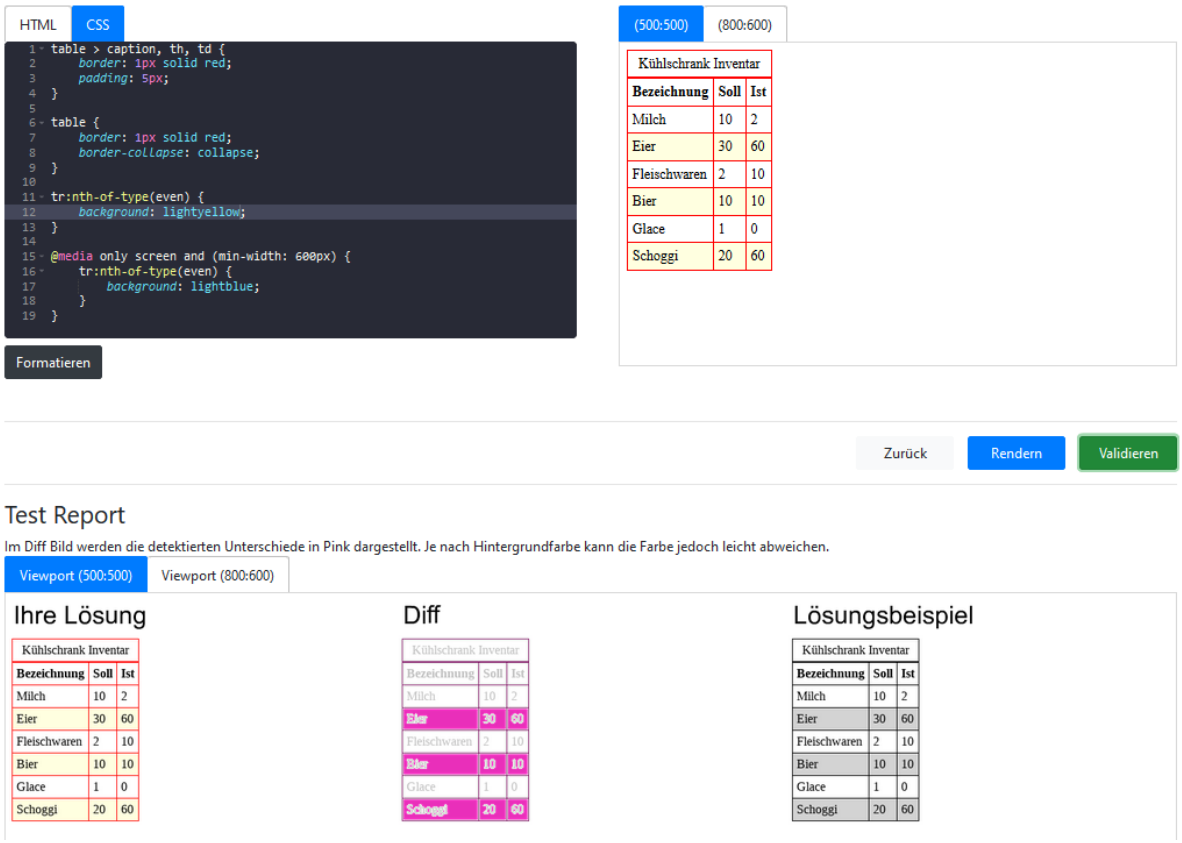


Abbildung 6-2: Beispiel lösen einer CSS-Aufgabe

► Die Transparenz des Diff Bildes lässt sich in der Konfiguration des Runners anpassen.

Ausserdem kann der Benutzer bereits bei der Aufgabenbearbeitung ein Bild des Lösungsbeispiels einsehen.
Bevor der Benutzer eine Aufgabe nicht erfolgreich gelöst hat, erhält er zu keinem Zeitpunkt Zugriff auf den Code des Lösungsbeispiels.

6.2 **Kursübersicht**

Der Dozent erhält einen Überblick über die Leistungen der Studenten und über die Testate resp. Meilensteine des Kurses. Ausserdem lassen sich die abgegebenen Lösungen anschauen und analysieren:

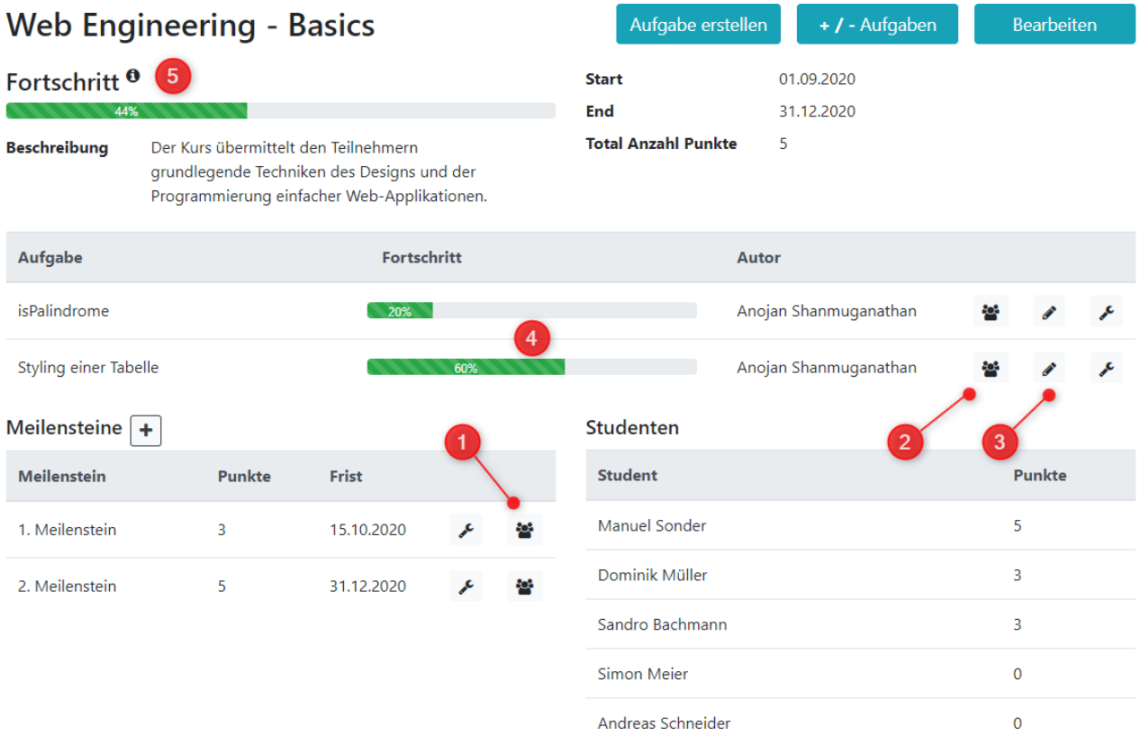


Abbildung 6-3: Dozentenansicht

1) Übersicht des Meilensteins, beinhaltet eine Auflistung der Studenten, die den Meilenstein erreicht haben, resp. welche Studenten dies (noch) nicht haben.
2) Abgegebene Lösung anzeigen
3) Aufgabe in der Studentenansicht
4) Fortschritt der Aufgaben, wird daraus berechnet, wie viele Studenten die Aufgabe erfolgreich gelöst haben.
5) Gesamtfortschritt des Kurses, wird aus den erreichten Punkten der Studenten berechnet.

6.3 **Performance**

Die im *NFR3: Effizienz* beschriebenen Performance-Ziele wurden nicht erreicht. Bei einer niedrigen Anzahl an Tests wurde die angestrebte Grenze überschritten. Sobald die Anzahl Tests bei Javascript-Aufgaben allerdings in einen etwas höheren Bereich kommt (>10 Tests), werden die angestrebten Zeiten eingehalten. Grund dafür ist, dass die Zeit bis zum Anlaufen der Tests unterschätzt wurde. Bei der Validierung von CSS-Aufgaben sieht es allerdings etwas anders aus. Je nach Anzahl Viewports kann die Zeit für die Validierung bis zu 10 Sekunden dauern. Die Zeit die für die Generierung der Bilder sowie für den Vergleich resp. für die Generierung des Diff-Bildes und für das Zusammenführen der drei Bilder benötigt wird, wurde unterschätzt. Die ursprünglich eingeplante Zeit, sah nicht vor, dass die generierten Bilder auch noch zusammengeführt werden müssen. Dieser Punkt hat sich allerdings im Laufe des Projekts als vorteilhaft erwiesen.

Damit der Benutzer beim Lösen der Aufgabe trotzdem ein zeitnahes Feedback zu seinem Code erhält, kann der Code während der Bearbeitung im Browser gerendert werden. Hierbei wird weder das Backend noch der Runner angestossen.

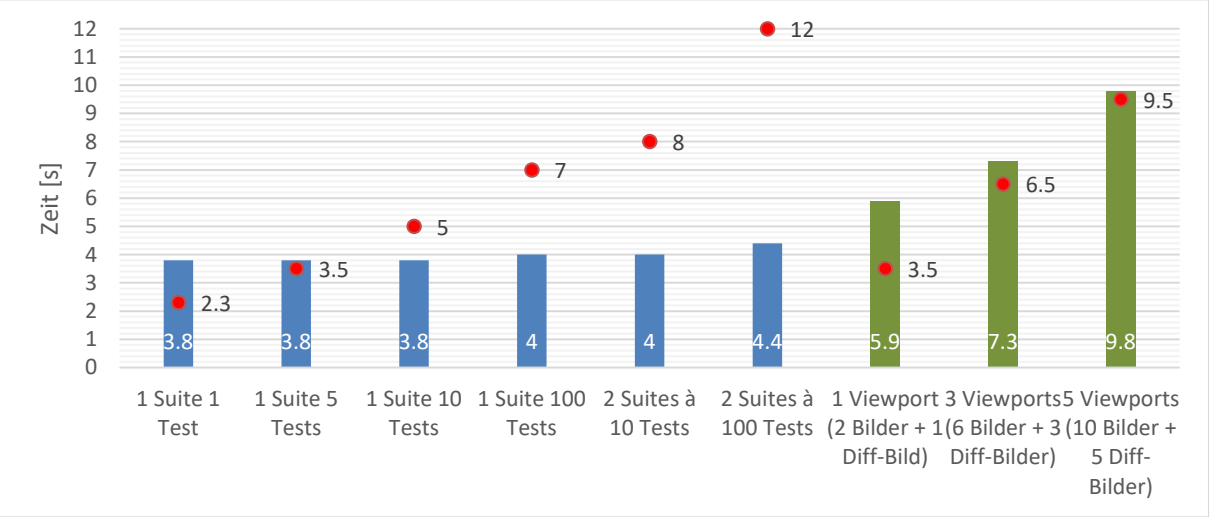


Abbildung 6-4: Zeitverhalten Validierung

► Die roten Punkte sind die Grenzen, welche im *NFR3: Effizienz* beschrieben sind.

6.4 Usability

Es wurde darauf geachtet das Auge des Benutzers möglichst gut zu führen und wo nötig mit Tipps zu unterstützen.
Durch die Usability-Tests wurden verschiedene Verbesserungsmöglichkeiten aufgezeigt, welche durch entsprechende Anpassungen übernommen wurden. Beispielsweise wurde der bei der Lösungsansicht ein Hinweis eingebaut, dass die Lösungen ausgeklappt werden können. Darüber hinaus wurden an verschiedenen Stellen Info-Icons mit Tooltips[12] eingebaut.

Ein weiteres Beispiel einer Anpassung ist der Slider bei der Definition der Schwierigkeit einer Aufgabe:

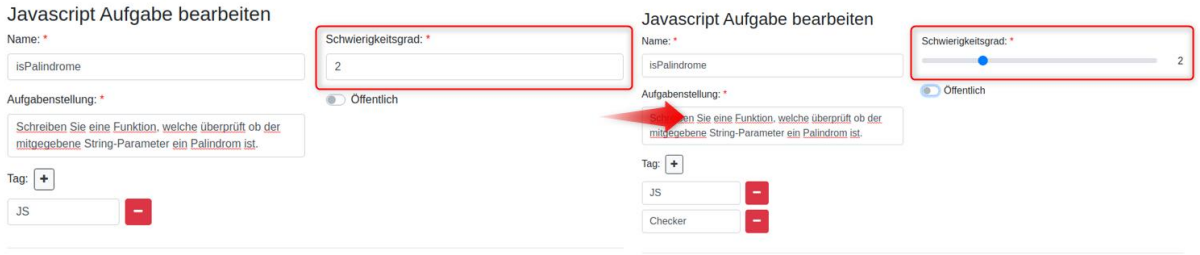


Abbildung 6-5: Anpassung Schwierigkeitsgrad (Usability)

Um die Benutzerfreundlichkeit zu steigern, sollte der Autor von CSS-Aufgaben darauf achten die Viewports nur so gross wie unbedingt nötig zu definieren. Die Gründe dafür sind:

- Einerseits wird so die Performance des Bildvergleichs optimal ausgenutzt. Wenn der Algorithmus viele weisse Pixel oder Bereiche miteinander vergleicht benötigt dies unnötig viel Zeit.
- Zweitens wird die Benutzerfreundlichkeit erhöht. Die Bilder der der Lösung sowie der gerenderte Code werden im Browser angezeigt, wenn diese Bilder resp. Bereiche gross sind entsteht zwangsläufig ein Platzproblem.

6.5 Sicherheit

Bei der Ausführung von Code, der vom Benutzer manipuliert werden kann, muss der Sicherheit besondere Aufmerksamkeit gewidmet werden. Das wurde bereits bei der Wahl der Architektur und Technologien berücksichtigt.

Durch die Ausführung des Codes innerhalb des Dockers besteht bereits eine erste Barriere, die nur schwer überwunden werden kann. Der Docker wird nach Ablauf eines konfigurierbaren Zeitlimits terminiert und abgeräumt. Bei der nächsten Validierung wird ein frischer Docker Container angelegt.

Attacken wie Spectre/Meltdown[13] sind schon aufgrund des Betriebssystems (Ubuntu 18.04 / Kernel 4.15.0-88-generic) sowie der darunter liegenden Architektur (x86-64) ausgeschlossen. (Ubuntu Wiki, 2020).

[12] https://de.wikipedia.org/wiki/Tooltip
[13] https://meltdownattack.com/

Zur weiteren Reduktion des Risikos sind ausserdem noch folgende Massnahmen getroffen worden:

- Backend/Runner laufen unter dediziertem Account mit den minimal notwendigen Rechten.
- Es sind nur jene Ports, die unbedingt benötigt werden gegen aussen geöffnet werden. Hierbei wird auch das jeweilige Protokoll berücksichtigt.

7 Schlussfolgerungen

Der implementierte Prototyp eignet sich hervorragend für die erweiterten Grundlagen einer Sprache oder Technologie.
Beide angebotenen Aufgabentypen werden zuverlässig validiert und mithilfe des Diff Bildes/Test Report kann der Benutzer die Fehler einfach lokalisieren und verbessern.

Sobald die Aufgaben sich etwas komplexer gestalten und/oder eine Datei- und Ordnerstruktur erfordern, gerät der Prototyp an seine Grenzen. Dazu ist Puzzles jedoch auch nicht konzipiert worden.

Durch Puzzles erhält der Dozent schnell einen Überblick über die Stärken und Schwächen der Studenten. Darüber hinaus kann er den Fortschritt durch die Meilensteine gut steuern und beeinflussen. Durch eine Analyse der abgegebenen Lösungen kann der Dozent ausserdem auf Tendenzen der Studenten reagieren. Wenn beispielsweise bei einer Javascript-Aufgabe, die sich mit Array Operationen befasst, nur ein minimaler Teil der Lösungen Array-Functions (`map()`, `forEach()`, etc.) enthalten, können im Unterricht noch einige Folien dazu eingebaut werden und die Studenten darauf aufmerksam gemacht werden.
Der Student kann anhand der anderen abgegebenen Lösungen alternative Ansätze und Lösungswege analysieren und mit der eigenen Variante vergleichen. Dies könnte sich positiv auf den Lerneffekt auswirken. Darüber hinaus kann anhand der Meilensteine leicht ermittelt werden, ob der geforderte Wissensstand erreicht wurde.

Ein grosser Vorteil von Puzzles ist, dass es gänzlich im Browser bedient werden kann. Dies wurde auch von den Testpersonen bei den Usability-Tests positiv bemerkt. Es bedarf keinerlei Installation und als Dozent kann man theoretisch bereits ab der ersten Unterrichtsstunde auf das Tool verweisen, ohne das die Studenten noch IDEs oder sonstige Werkzeuge installieren müssen.

Um die Sicherheit zu erhöhen, könnte die verwendete Datenbank auf PostgreSQL migrieren werden. Dies bietet den Vorteil von parallel (schreibenden) Zugriffen und von Array-Datentypen in den Tabellen. Darüber hinaus kann die Datenbank mit einem Passwort und weiteren Sicherheitsmechanismen geschützt werden. Durch die Array-Datentypen in den Tabellen könnte das Datenbankmodell vereinfacht werden. Die Nachteile bestünden in der geringeren Performance sowie im erhöhten Konfigurations- und Wartungsaufwand.
Eine weitere mögliche Verbesserung wäre die Integration von Regex bei den Aufgaben. So könnte der Dozent gewisse Lösungswege erzwingen. Weitere Vorschläge sind im *Anhang E: Product Backlog* ersichtlich.

Der Output dieses Projekts kann als erfolgreich angesehen werden und mit ein paar wenigen Erweiterungen ist es für den Unterricht einsetzbar.

Teil II

Software Engineering

8 Anforderungsanalyse

8.1 Gültigkeit

Die unten aufgeführten Anforderungen betreffen den umzusetzenden Prototyp und nicht ein fertiges, im Produktionsbetrieb einsetzbares Produkt. Die im *Anhang E: Product Backlog* aufgeführten Requirements ergänzen die in diesem Kapitel aufgeführten Anforderungen. Der Scope dieser Arbeit beschränkt sich auf die hier beschriebenen FR und NFR. Es sind keine optionalen Anforderungen definiert, falls die hier aufgeführten Requirements alle erfüllt sind und es die verbleibenden Ressourcen zulassen, werden Anforderungen aus dem Product Backlog in den Scope aufgenommen.

8.2 Funktionale Anforderungen

Die funktionalen Anforderungen werden anhand von User Stories beschrieben. Die Überprüfung erfolgt zusätzlich mit einer Definiton of Done. Die dort aufgeführten Punkte müssen vollständig erfüllt werden, damit die Anforderung als erfüllt angesehen werden kann. Eine Ausnahme bilden die mit «OPTIONAL:» gekennzeichneten Punkte.

8.2.1 FR1: Systemzugang

ALS	Benutzer
MÖCHTE ICH	via Browser auf das System zugreifen können,
DAMIT	ich die Anwendung unabhängig von meinem lokalen Betriebssystem nutzen kann.

Definition of Done:

- ☐ Die Applikation kann mit Mozilla Firefox, Google Chrome und Safari unter der URL https://puzzles.hsr.ch aufgerufen werden.
- ☐ Grafiken und sonstige Elemente werden korrekt angezeigt
- ☐ Test erfüllt

Test

ID	Testschritt
1.1	https://puzzles.hsr.ch aufrufen.
1.2	Das Zertifikat der Seite ist valide und es werden keine Sicherheitswarnungen angezeigt.
1.3	In der Konsole werden keine Fehler ausgegeben.
1.4	Der Benutzer wird begrüsst und wird aufgefordert sich anzumelden.
1.5	Bei Fehlern wird der Benutzer mit einer benutzerfreundlichen Meldung informiert.

8.2.2 FR2: Login

ALS	Benutzer
MÖCHTE ICH	mich mit bestehenden Credentials der Hochschule für Technik Rapperswil anmelden können (OPTIONAL: auch via GitHub),
DAMIT	ich meine Fortschritte nicht verliere und meine Leistungen und gelöste Aufgaben eindeutig zuweisbar sind.

Definition of Done:

- ☐ Die Anmeldung ist mit den HSR Credentials möglich.
- ☐ Die Anwendung kann nur mit gültigem Login erfolgreich durchgeführt werden
- ☐ OPTIONAL: Die Anmeldung ist mit einem anderen Identity Provider möglich
- ☐ Test erfüllt

Test

ID	Testschritt
2.1	Beim Aufruf von https://puzzles.hsr.ch wird eine Login-Option (Button) angezeigt (falls der Benutzer nicht bereits angemeldet ist).
2.2	Nach dem Klick auf den Button, leitet der Browser den Benutzer zur Login Seite der HSR (Azure Active Directory) weiter.
2.3	Nach erfolgreichem Login (Eingabe der Credentials sowie Erteilung der Berechtigungen) leitet der Browser den Benutzer zur Übersicht der eigenen Kurse weiter.
2.5	Der Name des Benutzers ist auf der Navigation ersichtlich.
2.6	Bei Fehlern wird der Benutzer mit einer benutzerfreundlichen Meldung informiert.

8.2.3 FR3: Kurs eröffnen

ALS	Dozent
MÖCHTE ICH	einen Kurs eröffnen können, welcher einen Namen sowie eine Beschreibung enthalten soll, (OPTIONAL: der Kurs kann mit einem optionalen Icon versehen werden)
DAMIT	die Studierenden sich für den Kurs einschreiben können.

Definition of Done:

- ☐ Jeder angemeldete Benutzer kann einen Kurs eröffnen
- ☐ Der Ersteller des Kurses wird bei diesem Kurs automatisch zum Dozenten.
- ☐ Andere Benutzer können den Kurs nicht bearbeiten.
- ☐ OPTIONAL: Der Kurs kann mit einem Icon versehen werden
- ☐ Test erfüllt

Test

ID	Testschritt
3.1	Der angemeldete Benutzer kann auf der Startseite den Button «Kurs eröffnen» klicken.
3.2	Nach dem Klicken wird er aufgefordert alle nötigen Daten (Name, Beschreibung, Start sowie Enddatum) einzugeben. Bei invalider Eingabe wird er aufgefordert, diese Eingaben zu korrigieren.

3.3	Nach dem Klick auf den «Speichern» Button, wird der Kurs angelegt und der Benutzer wird automatisch zum Dozenten dieses Kurses. Dies wird in der Auflistung der Kurse (Startseite) auch so angezeigt.
3.4	Wenn die Erstellung aus technischen Gründen nicht funktioniert hat, wird der Benutzer mit einer Meldung informiert und gebeten es später erneut zu versuchen.

8.2.4 FR4: Kurs überarbeiten

ALS	Dozent (Ersteller des Kurses)
MÖCHTE ICH	Kursdaten bearbeiten können,
DAMIT	ich die Daten stets aktuell halten kann und allfällige Änderungen vornehmen oder Fehler korrigieren kann.

Definition of Done:

- ☐ Der Dozent (Ersteller des Kurses) kann die Kursdaten überarbeiten. Die Bearbeitung kann unter der Kursübersicht gestartet werden.
- ☐ Test erfüllt

Test

ID	Testschritt
4.1	Der Ersteller des Kurses kann via Klick auf den Kurs > Bearbeiten die Kursdaten überarbeiten.
4.2	Andere Benutzer können den Kurs nicht überarbeiten. Weder im Web UI noch mit direktem Call auf das Backend.
4.3	Bei einer erfolgreichen Speicherung werden die Daten korrekt übernommen und in der Kursübersicht entsprechend angezeigt.
4.4	Wenn die Bearbeitung aus technischen Gründen nicht funktioniert hat, wird der Benutzer mit einer Meldung informiert und gebeten es später erneut zu versuchen.

8.2.5 FR5: Kurs löschen

ALS	Dozent (Ersteller des Kurses)
MÖCHTE ICH	den Kurs löschen können,
DAMIT	bei einer Absage des Kurses oder sonstigen Gründen für dessen Nicht-Durchführung den Kurs aus dem Angebot entfernen kann.

Definition of Done:

- ☐ Der Dozent (Ersteller des Kurses) kann den Kurs löschen.
- ☐ Der Löschvorgang kann in der Maske der Kursbearbeitung initiiert werden.
- ☐ Andere Benutzer können den Kurs nicht löschen.
- ☐ Test erfüllt

Test

ID	Testschritt
5.1	Der Ersteller des Kurses kann via Klick auf den Kurs > Bearbeiten > Löschen den Löschvorgang des Kurses initiieren.
5.3	Bei der Löschung des Kurses muss dies in einem Dialog zusätzlich bestätigt werden.
5.3	Bei einer erfolgreichen Löschung gelangt der Benutzer auf die Startseite und der Kurs ist nicht mehr auffindbar.

5.4	Wenn die Löschung aus technischen Gründen nicht funktioniert hat, wird der Benutzer mit einer Meldung informiert und gebeten es später erneut zu versuchen.
5.5	Abgesehen vom Ersteller des Kurses kann niemand den Kurs löschen. Weder im UI noch via Call zum Backend.

8.2.6 FR6: Für Kurs anmelden/einschreiben

ALS	Student
MÖCHTE ICH	mich für einen Kurs einschreiben,
DAMIT	ich die entsprechenden Aufgaben und Testate lösen kann.

Definition of Done:

- ☐ Ein angemeldeter Benutzer kann sich für einen Kurs einschreiben. Das Einschreiben erfolgt mittels Klicks auf einen Button.
- ☐ Es können mehrere Kurse angemeldet werden.
- ☐ Die angemeldeten Kurse werden auf der Startseite angezeigt
- ☐ Test erfüllt

Test

ID	Testschritt
6.1	Auf der Startseite erscheint ein Button «Kurs anmelden».
6.2	Nach Klick auf den Knopf «Kurs anmelden» erscheint eine Liste von Kursen.
6.3	Wenn das Laden aus technischen Gründen nicht funktioniert hat, wird der Benutzer mit einer Meldung informiert und gebeten es später wieder zu versuchen.
6.4	Neben den Kursen besteht die Möglichkeit sich mittels Buttons «Anmelden» dafür anzumelden.
6.5	Ein bereits angemeldeter Kurs kann nicht erneut angemeldet werden.
6.6	Angemeldete Kurse erscheinen auf der Startseite unter «Meine Kurse»
6.7	Von angemeldeten Kursen kann die Kursübersicht (Student) aufgerufen werden.
6.8	Wenn die Anmeldung aus technischen Gründen nicht funktioniert hat, wird der Benutzer mit einer Meldung informiert und gebeten es später erneut zu versuchen.

8.2.7 FR7: Aufgaben zum Kurs hinzufügen/entfernen

ALS	Dozent
MÖCHTE ICH	Aufgaben zu meinem Kurs hinzufügen und entfernen können
DAMIT	ich den Kurs entsprechend dem Lehrplan aufbauen kann.

Definition of Done:

- ☐ Der Dozent (Ersteller des Kurses) kann Aufgaben zu einem Kurs hinzufügen. Dabei kann er entweder aus bereits bestehenden Aufgaben auswählen oder neue Aufgaben erstellen.
- ☐ Die hinzugefügten Aufgaben werden in der Kursübersicht (Student/Dozent) korrekt angezeigt.
- ☐ Der Dozent kann Aufgaben aus dem Kurs entfernen
- ☐ Test erfüllt

Test

ID	Testschritt
7.1	Der Ersteller des Kurses kann via Klick auf den Kurs > «+/-Aufgabe» Aufgaben zum Kurs hinzufügen.
7.2	Nachdem Klicken auf «+/-Aufgabe» wird eine Suchmaske angezeigt, in welcher er nach seinen erfassten Aufgaben suchen kann. (Suche mittels Tags, Name oder Dozent)
7.3	Neben den Aufgaben erscheint ein «+»-Button, um Aufgaben hinzuzufügen.
7.4	Aufgaben die schon hinzugefügt wurden, können nicht noch einmal hinzugefügt werden.
7.5	Aufgaben die bereits im Kurs sind, können via Klick auf den «-»-Button aus dem Kurs entfernt werden.
7.6	Nach dem Bearbeiten sind die Aufgaben auf der Kursansicht ersichtlich und können von den Kursteilnehmern eingesehen werden.
7.7	Besitzt der Ersteller keine eigenen Aufgaben und es existieren keine öffentlichen Aufgaben wird der Benutzer darüber informiert und wird gebeten eine Aufgabe zu erstellen.
7.8	Wenn das Laden aus technischen Gründen nicht funktioniert hat, wird der Benutzer mit einer Meldung informiert und gebeten es später wieder zu versuchen.
7.9	Abgesehen vom Dozenten kann niemand Aufgaben zum Kurs hinzufügen/entfernen. Weder im UI noch via Call zum Backend.

8.2.8 FR8: Javascript-Aufgabe erstellen

ALS	Benutzer
MÖCHTE ICH	eine Javascript bezogene Aufgabe erstellen können. Hierbei muss ich folgende Angaben angeben: • Aufgabentitel • Aufgabenstellung • Ausgangscode (Optional) • Unit Test für die Überprüfung • Lösungsbeispiel • Schwierigkeitsgrad (1 bis 5) • Tags • Öffentlich: Ja/Nein[14]
DAMIT	andere Benutzer ihr Javascript Wissen überprüfen und verbessern können.

Definition of Done:

- ☐ Jeder angemeldete Benutzer kann Javascript Aufgaben erstellen
- ☐ Aufgaben werden nur den eingeloggten Benutzer angezeigt
- ☐ Die erstellten Aufgaben werden in der Aufgabenübersicht aufgelistet
- ☐ Test erfüllt

[14] Öffentliche Aufgaben sind für sämtliche Benutzer einsehbar und auch lösbar, nicht-öffentliche Aufgaben sind nur für Benutzer sichtbar, die im entsprechenden Kurs eingeschrieben sind.

Test

ID	Testschritt
8.1	Auf der Startseite ist der Button «Aufgaben erstellen» ersichtlich.
8.2	Nachdem Klicken wird gefragt, ob eine CSS oder Javascript Aufgaben erstellt werden soll. >Javascript auswählen.
8.3	Es können die entsprechenden Daten (Titel, Fragestellung, Ausgangscode, Unit Tests, Musterlösung, Schwierigkeit sowie Tags) angegeben werden.
8.4	Bei Eingabe invalider Daten wird der Benutzer aufgefordert, diese Eingaben zu korrigieren.
8.5	Bei Speichern werden die Daten in der Datenbank abgelegt und die Aufgabe erscheint in der Übersicht der eigenen Aufgaben.
8.6	Falls die Aufgabe zu einem Kurs zugewiesen wurde, ist sie anschliessend in der Kursansicht ersichtlich.
8.7	In der Kursansicht des Dozenten ist der Button «Aufgaben erstellen» ersichtlich. Mit diesem Button wird die neu erstellte Aufgabe direkt zum Kurs hinzugefügt.
8.8	Wenn die Erstellung aus technischen Gründen nicht funktioniert hat, wird der Benutzer mit einer Meldung informiert und gebeten es später wieder zu versuchen.

8.2.9 FR9: CSS-Aufgabe erstellen

ALS Benutzer

MÖCHTE ICH eine CSS bezogene Aufgabe erstellen können. Hierbei muss ich folgende Angaben machen:

- Aufgabentitel
- Aufgabenstellung
- Ausgangscode (Optional)
- Viewports (für die die Lösung validiert werden soll)
- Lösungsbeispiel
- Schwierigkeitsgrad (1 bis 5)
- Tags
- Öffentlich: Ja/Nein[15]

Die Aufgabe soll dabei unabhängig von Browserversion und Lösungsvariante (Bsp. bei Farben color-Attribut oder rgb()-Attribut) validiert werden.

DAMIT andere Benutzer ihr CSS Wissen überprüfen und verbessern können.

Definition of Done:

- ☐ Jeder angemeldete Benutzer kann CSS-Aufgaben erstellen
- ☐ Es werden nur Aufgaben akzeptiert, die über eine valide Musterlösung verfügen.
- ☐ Die erstellten Aufgaben werden in der Aufgabenübersicht aufgelistet
- ☐ Test erfüllt

[15] Öffentliche Aufgaben sind für sämtliche Benutzer einsehbar und auch lösbar, nicht-öffentliche Aufgaben sind nur für Benutzer sichtbar, die im entsprechenden Kurs eingeschrieben sind.

Test

ID	Testschritt
9.1	Auf der Startseite ist der Button «Aufgaben erstellen» ersichtlich.
9.2	Nachdem Klicken wird gefragt, ob eine CSS oder Javascript Aufgaben erstellt werden soll. >CSS auswählen.
9.3	Es können die entsprechenden Daten (Titel, Fragestellung, Ausgangscode, Unittests, Musterlösung, Schwierigkeit, Viewports sowie Tags) angegeben werden.
9.4	Bei Eingabe invalider Daten wird der Benutzer aufgefordert, diese Eingaben zu korrigieren.
9.5	Bei Speichern werden die Daten in der Datenbank abgelegt und die Aufgabe erscheint in der Übersicht der eigenen Aufgaben.
9.6	Falls die Aufgabe zu einem Kurs zugewiesen wurde, ist sie anschliessend in der Kursansicht ersichtlich.
9.7	In der Kursansicht des Dozenten ist der Button «Aufgaben erstellen» ersichtlich. Mit diesem Button wird die neu erstellte Aufgabe direkt zum Kurs hinzugefügt.
9.8	Wenn die Erstellung aus technischen Gründen nicht funktioniert hat, wird der Benutzer mit einer Meldung informiert und gebeten es später wieder zu versuchen.

8.2.10 FR10: Testat erstellen

ALS	Dozent
MÖCHTE ICH	für meine Kurse Testate definieren können. Testate bestehen aus «Levels» welche zu einem Zeitpunkt erreicht werden müssen. Die Levels berechnen sich aus den gelösten Aufgaben in Verbindung mit deren Schwierigkeit. Die Punkte werden aufsummiert und wenn zum bestimmten Zeitpunkt die bestimmte Punktemenge erreicht ist, gilt das Testat als bestanden.
DAMIT	ich das Wissen und den Fortschritt der Studenten überprüfen kann.

Definition of Done:

- ☐ Der Dozent (Ersteller des Kurses) kann für den Kurs Testate definieren.
- ☐ Der Dozent (Ersteller des Kurses) kann für den Kurs definierte Testate freigeben.
- ☐ Test erfüllt

Test

ID	Testschritt
10.1	Beim Ersteller des Kurses erscheint in der Kursübersicht der Button «+» im Meilenstein-Bereich.
10.2	Nachdem der Auswahl wird der Benutzer aufgefordert alle nötigen Daten (Anzahl Punkte, Name sowie Datum) einzugeben. Bei invalider Eingabe wird man aufgefordert, diese Eingaben zu korrigieren.
10.3	Es ist möglich mehrere Meilensteine für einen Kurs zu erfassen.
10.4	Nach der Speicherung sind die Meilensteine in der Kursübersicht des Dozenten sowie in der Kursübersicht des Studenten ersichtlich.
10.5	Wenn die Erstellung aus technischen Gründen nicht funktioniert hat, wird der Benutzer mit einer Meldung informiert und gebeten es später wieder zu versuchen.
10.6	Abgesehen vom Dozenten kann niemand Meilensteine zum Kurs hinzufügen. Weder im UI noch via Call zum Backend.

8.2.11 FR11: Javascript-Aufgabe lösen

ALS	Student
MÖCHTE ICH	eine Lösung für eine Javascript-Aufgabe abgeben,
DAMIT	ich mein Wissen überprüfen kann.

Definition of Done:

- ☐ Jeder angemeldete Benutzer kann öffentliche (freigegebene) Aufgaben lösen.
- ☐ Kursbezogene (private) Aufgaben können nur vom Autor und von den im Kurs eingeschriebenen Studenten gelöst werden.
- ☐ Nur freigegebene Aufgaben weisen die Möglichkeit auf sie zu lösen
- ☐ Die Validierung der Aufgabe erfolgt anhand der definierten Unit Tests
- ☐ Die Validierung erfolgt in der Runner-Komponente
- ☐ Nach der Abgabe einer korrekten Lösung wird die Aufgabe als «gelöst» angerechnet
- ☐ Nach der Abgabe einer korrekten Lösung wurde der Punktestand des Benutzers (im entsprechenden Kurs) entsprechend angepasst
- ☐ Nach der Abgabe einer korrekten Lösung wird die Lösungsbeispiel angezeigt.
- ☐ Test erfüllt

Test

ID	Testschritt
11.1	Beim Lösen einer Aufgabe werden die benötigten Informationen und Daten (Aufgabentitel, Aufgabenstellung und Ausgangscode) angezeigt.
11.2	Wenn das Laden aus technischen Gründen nicht funktioniert hat, wird der Benutzer mit einer Meldung informiert und gebeten es später wieder zu versuchen.
11.3	Nach der Eingabe im Editor kann der Student mittels «Validieren» Button seine Lösung abschicken.
11.4	Nach der Validierung wird ein Testreport angezeigt. Wenn alles Tests erfolgreich waren ist die Aufgabe erfüllt, ansonsten muss die Eingabe angepasst werden.
11.5	Wenn das Validieren aus technischen Gründen nicht funktioniert hat, wird der Benutzer mit einer Meldung informiert und gebeten es später wieder zu versuchen.

8.2.12 FR12: CSS-Aufgabe lösen

ALS	Student
MÖCHTE ICH	eine Lösung für eine CSS-Aufgabe abgeben
DAMIT	ich mein Wissen überprüfen kann.

Definition of Done:

- ☐ Jeder angemeldete Benutzer kann öffentliche (freigegebene) Aufgaben lösen.
- ☐ Kursbezogene (private) Aufgaben können nur vom Autor und von den im Kurs eingeschriebenen Studenten gelöst werden.
- ☐ Nur freigegebene Aufgaben weisen die Möglichkeit auf sie zu lösen
- ☐ Die Validierung der Aufgabe erfolgt anhand von einem visuellen Bildvergleich
- ☐ Die Validierung berücksichtigt die definierten Viewports
- ☐ Die Validierung erfolgt in der Runner-Komponente
- ☐ Nach der Abgabe einer korrekten Lösung wird die Aufgabe als «gelöst» angerechnet

- ☐ Nach der Abgabe einer korrekten Lösung wurde der Punktestand des Benutzers (im entsprechenden Kurs) entsprechend angepasst
- ☐ Nach der Abgabe einer korrekten Lösung wird die Musterlösung angezeigt.
- ☐ Test erfüllt

Test

ID	Testschritt
12.1	Beim Lösen einer Aufgabe werden die benötigten Informationen und Daten (Aufgabentitel, Aufgabenstellung, Ausgangscode und Lösungsbeispiel als Bild) angezeigt.
12.2	Wenn das Laden aus technischen Gründen nicht funktioniert hat, wird der Benutzer mit einer Meldung informiert und gebeten es später wieder zu versuchen.
12.3	Nach der Eingabe im Editor kann der Student mittels «Validieren» Button seine Lösung abschicken.
12.4	Nach der Validierung wird seine Eingabe gerendert und angezeigt. Enthält die Aufgabe mehrere Viewports, dann wird seine Lösung in verschiedenen Viewports angezeigt.
12.5	Wenn das Validieren aus technischen Gründen nicht funktioniert hat, wird der Benutzer mit einer Meldung informiert und gebeten es später wieder zu versuchen.

8.2.13 FR13: Aufgabe überarbeiten

ALS Ersteller einer Aufgabe

MÖCHTE ICH eine von mir gestellte Aufgaben überarbeiten und die folgenden Eigenschaften überarbeiten können:

- Aufgabentitel
- Aufgabenstellung
- Ausgangscode (Optional)
- Unit Test für die Überprüfung
- Lösungsbeispiel
- Schwierigkeitsgrad (1 bis 5)
- Tags
- Öffentlich

DAMIT ich Fehler oder sonstige Anpassungen vornehmen kann (anpassen der Schwierigkeit oder Ähnliches)

Definition of Done:

- ☐ Der Autor der Aufgabe kann die Aufgabe überarbeiten
- ☐ Freigegebene Aufgaben könne nicht mehr überarbeitet werden.
- ☐ Andere Benutzer können die Aufgabe nicht überarbeiten
- ☐ Test erfüllt

Test

ID	Testschritt
13.1	Der Autor einer Aufgabe kann bei der Übersicht der entsprechenden Aufgabe auf «Bearbeiten» klicken. Bei anderen Aufgaben erscheint diese Möglichkeit nicht.
13.2	Die Felder (Aufgabentitel, Aufgabenstellung, Ausgangscode, Unit Test, Musterlösung, Schwierigkeitsgrad, Öffentlich sowie Tags) sind bearbeitbar.
13.3	Bei der Bearbeitung werden die Felder auf ihre Gültigkeit überprüft.

13.4	Bei einer Speicherung werden die Änderungen korrekt übernommen und die Aufgabenübersicht wird angezeigt.
13.5	Die überarbeiteten Daten werden korrekt in der Aufgabenübersicht angezeigt.
13.6	Wenn die Bearbeitung/ das Laden aus technischen Gründen nicht funktioniert hat, wird der Benutzer mit einer Meldung informiert und gebeten es später wieder zu versuchen.
13.7	Abgesehen vom Autor kann niemand die Aufgab bearbeiten. Weder im UI noch via Call zum Backend.

8.2.14 FR14: Testat überarbeiten

ALS	Dozent
MÖCHTE ICH	ein von mir erstelltes Testat überarbeiten können.
DAMIT	ich die Daten des Testats anpassen kann.

Definition of Done:

- ☐ Der Dozent (Ersteller des Kurses) kann für den Kurs Testate überarbeiten.
- ☐ Andere Benutzer können das Testat nicht überarbeiten.
- ☐ Test erfüllt

Test

ID	Testschritt
14.1	Der Ersteller eines Kurses kann bei den Meilensteinen im Kurs in der Kursübersicht auf «Bearbeiten» klicken. Ansonsten besteht diese Möglichkeit nicht.
14.2	Bei der Bearbeitung können die Felder Punktzahl sowie Datum angepasst werden.
14.3	Bei der Bearbeitung werden die Felder auf ihre Gültigkeit überprüft.
14.4	Bei einer Speicherung werden die Daten korrekt übernommen und die Kursübersicht wird wieder angezeigt.
14.5	Die überarbeiteten Daten werden korrekt in der Kursübersicht des Dozenten sowie jener vom Studenten angezeigt-
14.6	Wenn die Bearbeitung/ das Laden aus technischen Gründen nicht funktioniert hat, wird der Benutzer mit einer Meldung informiert und gebeten es später wieder zu versuchen.
14.7	Abgesehen vom Dozenten kann niemand Meilensteine bearbeiten. Weder im UI noch via Call zum Backend.

8.2.15 FR15: Testat löschen

ALS	Dozent
MÖCHTE ICH	ein von mir erstelltes Testat löschen können.
DAMIT	ich falsch erstellte Testate löschen kann. Ausserdem kann ich so auf Anpassungen im Kurs reagieren.

Definition of Done:

- ☐ Der Löschvorgang kann in der Maske der Kursbearbeitung initiiert werden.
- ☐ Andere Benutzer können das Testat nicht löschen.
- ☐ Test erfüllt

Test

ID	Testschritt
15.1	Der Ersteller des Kurses (und somit auch der Meilensteine) kann bei der Kursübersicht bei den Meilensteinen auf «Bearbeiten» klicken.
15.2	Nach dem Löschvorgang wird die Kursübersicht angezeigt.
15.3	Die gelöschten Meilensteine sind weder in der Kursübersicht des Dozenten noch in jener von den Studenten ersichtlich.
15.4	Wenn die Löschung aus technischen Gründen nicht funktioniert hat, wird der Benutzer mit einer Meldung informiert und gebeten es später wieder zu versuchen.
15.5	Abgesehen vom Dozenten kann niemand einen Meilenstein löschen. Weder im UI noch via Call zum Backend.

8.2.16 FR16: Aufgabe löschen

ALS	Ersteller einer Aufgabe
MÖCHTE ICH	eine Aufgabe löschen können,
DAMIT	ich fehlerhafte, zu schwere oder nicht passende Aufgaben löschen kann.

Definition of Done:

- ☐ Der Löschvorgang erfordert eine Bestätigung in einem Dialog.
- ☐ Der Löschvorgang kann in der Maske der Aufgabenbearbeitung initiiert werden.
- ☐ Andere Benutzer können die Aufgabe nicht löschen.
- ☐ Test erfüllt

Test

ID	Testschritt
16.1	Der Ersteller der Aufgabe kann unter «Bearbeiten» (Aufgabeübersicht) den Löschvorgang der Aufgabe mittels Klicks auf den Knopf «Löschen» initiieren.
16.2	Der Löschvorgang muss mittels Bestätigung in einem Dialog bestätigt werden.
16.3	Bei einer erfolgreichen Löschung ist die Aufgabe aus der DB gelöscht und sie erscheint nicht mehr in der Aufgabenübersicht.
16.4	Die gelöschte Aufgabe wurde aus den entsprechenden Kursen entfernt und erscheint nicht mehr in der Kursübersicht.
16.5	Wenn die Löschung aus technischen Gründen nicht funktioniert hat, wird der Benutzer mit einer Meldung informiert und gebeten es später wieder zu versuchen.
16.6	Abgesehen vom Autor kann niemand die Aufgabe löschen. Weder im UI noch via Call zum Backend.

8.2.17 FR17: Kursübersicht einsehen (Student)

ALS	Student
MÖCHTE ICH	eine Kursübersicht einsehen, in der mein Fortschritt im entsprechenden Kurs visuell dargestellt wird und ich einen Überblick über meine Leistungen erhalte. Ausserdem will ich sehen welche Meilensteine ich erfüllt habe und welche noch offen sind.
DAMIT	ich entsprechend meinem Fortschritt meine Arbeiten besser planen kann.

Definition of Done:

- ☐ Der Student (eingeschriebener Benutzer) kann eine Kursübersicht aufrufen. Dort sind verschiedene Informationen zum Kurs und zum Kursverlauf aufgeführt.
- ☐ Die Daten werden pro Benutzer korrekt angezeigt
- ☐ Der Student (eingeschriebener Benutzer) sieht nur die eigene Kursübersicht.
- ☐ Die Kursübersicht kann unter Meine Kurse > [Kurs] eingesehen werden
- ☐ Test erfüllt

Test

ID	Testschritt
17.1	Der Student kann auf der Startseite aus einer seiner Kurse auswählen und gelangt somit auf die Kursübersicht dieses Kurses.
17.2	Alle Details über den Kurs in der perspektive vom Studenten sind ersichtlich.
17.3	Es werden nur Daten bezüglich des Studenten geladen. Der Zugriff auf weitere Daten vom Kurs oder anderen Studenten ist nicht möglich.
17.4	Wenn das Laden aus technischen Gründen nicht funktioniert hat, wird der Benutzer mit einer Meldung informiert und gebeten es später wieder zu versuchen.

8.2.18 FR18: Kursübersicht einsehen (Dozent)

ALS	Dozent
MÖCHTE ICH	eine Übersicht über den Kurs einsehen, damit ich sehe, wo die Studenten stehen und welche Studenten welche Meilensteine und Aufgaben bereits gelöst haben.
DAMIT	ich den Unterrichtsstoff und -tempo entsprechend anpassen kann.

Definition of Done:

- ☐ Der Dozent (Ersteller des Kurses) kann eine Kursübersicht aufrufen. Dort sind verschiedene Informationen zum Kurs und zum Kursverlauf aufgeführt. Ausserdem sind auch Information zu den Studenten und Meilensteinen aufgeführt.
- ☐ Die Kursübersicht kann unter Meine Kurse > [Kurs] eingesehen werden
- ☐ Test erfüllt

Test

ID	Testschritt
18.1	Der Benutzer kann auf der Startseite auf einen seiner Kurse klicken und gelangt somit auf die Kursübersicht für diesen Kurs.
18.2	Alle Details (Aufgaben, Studenten, Meilensteine, welcher Student welche Meilensteine (nicht) erfüllt) sind in dieser Ansicht ersichtlich.
18.3	Wenn das Laden aus technischen Gründen nicht funktioniert hat, wird der Benutzer mit einer Meldung informiert und gebeten es später wieder zu versuchen.
18.4	Abgesehen vom Dozenten kann niemand den Kurs als Dozent ansehen.

8.2.19 FR19: Antworten einsehen

ALS	Benutzer
MÖCHTE ICH	Antworten die zu einer Aufgabe eingegangen sind, ansehen und analysieren können. Dies soll allerdings nur möglich sein, nachdem ich eine korrekte Antwort zu einer Aufgabe abgegeben habe.
DAMIT	ich andere Lösungsansätze sehen kann.

Definition of Done:

- ☐ Nach der Abgabe einer korrekten Lösung erscheint in der Kursübersicht ein Button, mit dem sich andere Lösungen anzeigen lassen
- ☐ Test erfüllt

Test

ID	Testschritt
19.1	In der Kursübersicht eines Kurses besteht die Möglichkeit mittels Buttons die Antworten der (Mit-)Studenten einzusehen.
19.2	Die Möglichkeit Antworten einzusehen ist nur gegeben, wenn der Student die entsprechende Aufgabe erfolgreich gelöst hat.
19.3	Wenn das Laden aus technischen Gründen nicht funktioniert hat, wird der Benutzer mit einer Meldung informiert und gebeten es später wieder zu versuchen.

8.2.20 FR20: Kurs abmelden

ALS	Student
MÖCHTE ICH	mich von einem Kurs abmelden
DAMIT	ich im Falle einer Falschanmeldung meinen Fehler korrigieren kann. Ausserdem damit ich falls ich den Kurs nicht mehr besuche, mich entsprechend abmelden kann.

Definition of Done:

- ☐ Der Student kann einen angemeldeten Kurs wieder abmelden
- ☐ Test erfüllt

Test

ID	Testschritt
20.1	Auf der Kursübersicht des Studenten ist der Button «Abmelden» ersichtlich.
20.2	Nach der Abmeldung wird der Benutzer wieder auf die Startseite weitergeleitet
20.3	Ein bereits abgemeldeter Kurs kann nicht erneut abgemeldet werden.
20.4	Der abgemeldete Kurs erscheint nicht mehr in der Übersicht (Auflistung unter angemeldete Kurse).
20.5	Auf der Liste der anmeldbaren Kurse ist der abgemeldete Kurs wieder anmeldbar.

8.2.21 FR21: Öffentliche Aufgaben lösen

ALS	Benutzer
MÖCHTE ICH	öffentliche Aufgaben durchsuchen und lösen können
DAMIT	ich mein Wissen erweitern kann. Ausserdem damit ich (falls ich Themengebiete habe, wo ich noch nicht ganz auf dem Soll-Wissensstand bin) weitere Aufgaben suchen / lösen kann, um mich zu verbessern.

Definition of Done:

- ☐ Aufgabenübersicht einsehbar (öffentliche Aufgaben)
- ☐ Öffentliche Aufgaben lösbar
- ☐ Test erfüllt

Test

ID	Testschritt
21.1	Auf der Navigation ist ein Link mit «Öffentliche Aufgaben lösen».
21.2	Anschliessend werden auf der Seite eine Sucheingabe und eine Tabelle mit allen öffentlichen Aufgaben angezeigt.
21.3	Sind keine öffentlichen Aufgaben vorhanden, wird der Benutzer mittels Meldung darüber informiert.
21.4	Wenn das Laden aus technischen Gründen nicht funktioniert hat, wird der Benutzer mit einer Meldung informiert und gebeten es später wieder zu versuchen.
21.5	Nach dem Klick auf eine beliebige Aufgabe wird diese angezeigt.
21.6	Aufgabe kann gelöst werden (FR11 / FR12).

8.2.22 Abhängigkeiten

Um die Planung der Anforderungen zu vereinfachen wurden ihren Hauptabhängigkeiten zu anderen Anforderungen untersucht. Daraus entstand folgende Abbildung:

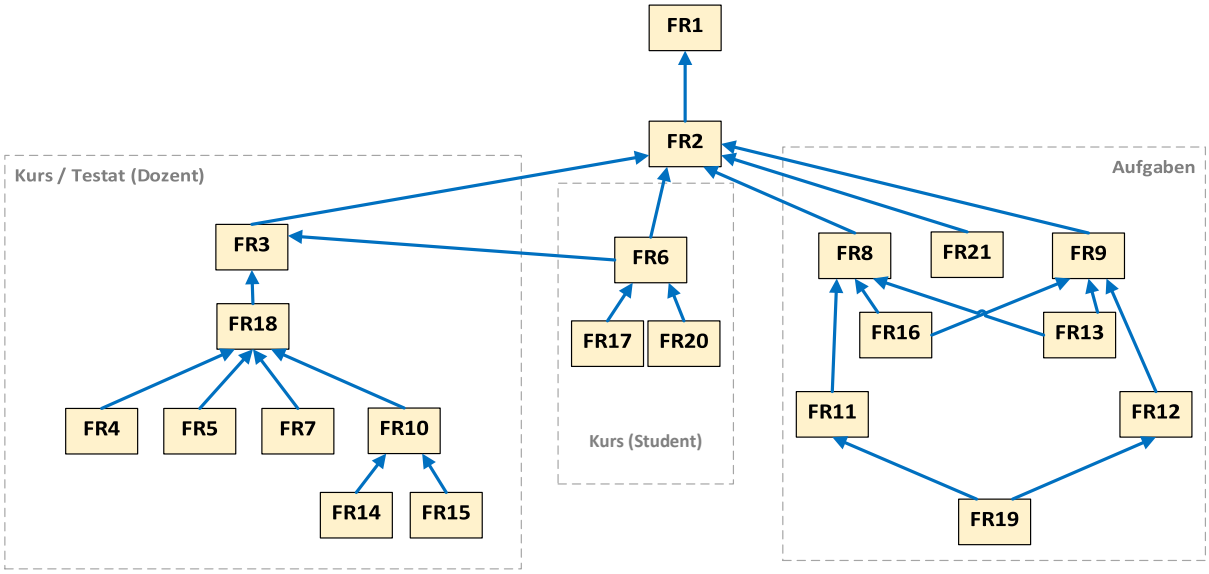


Abbildung 8-1: Abhängigkeiten Requirements

FR1 Ist der Systemzugang via Browser; FR2 beschreibt das Login.

Anhand dieser Analyse ist die zeitliche Abfolge der Implementierung einfacher zu definieren.

8.2.23 Use Case Diagramm

Dieses Use Case Diagramm dient als Übersicht über alle, als Use Story beschriebenen, funktionalen Anforderungen.

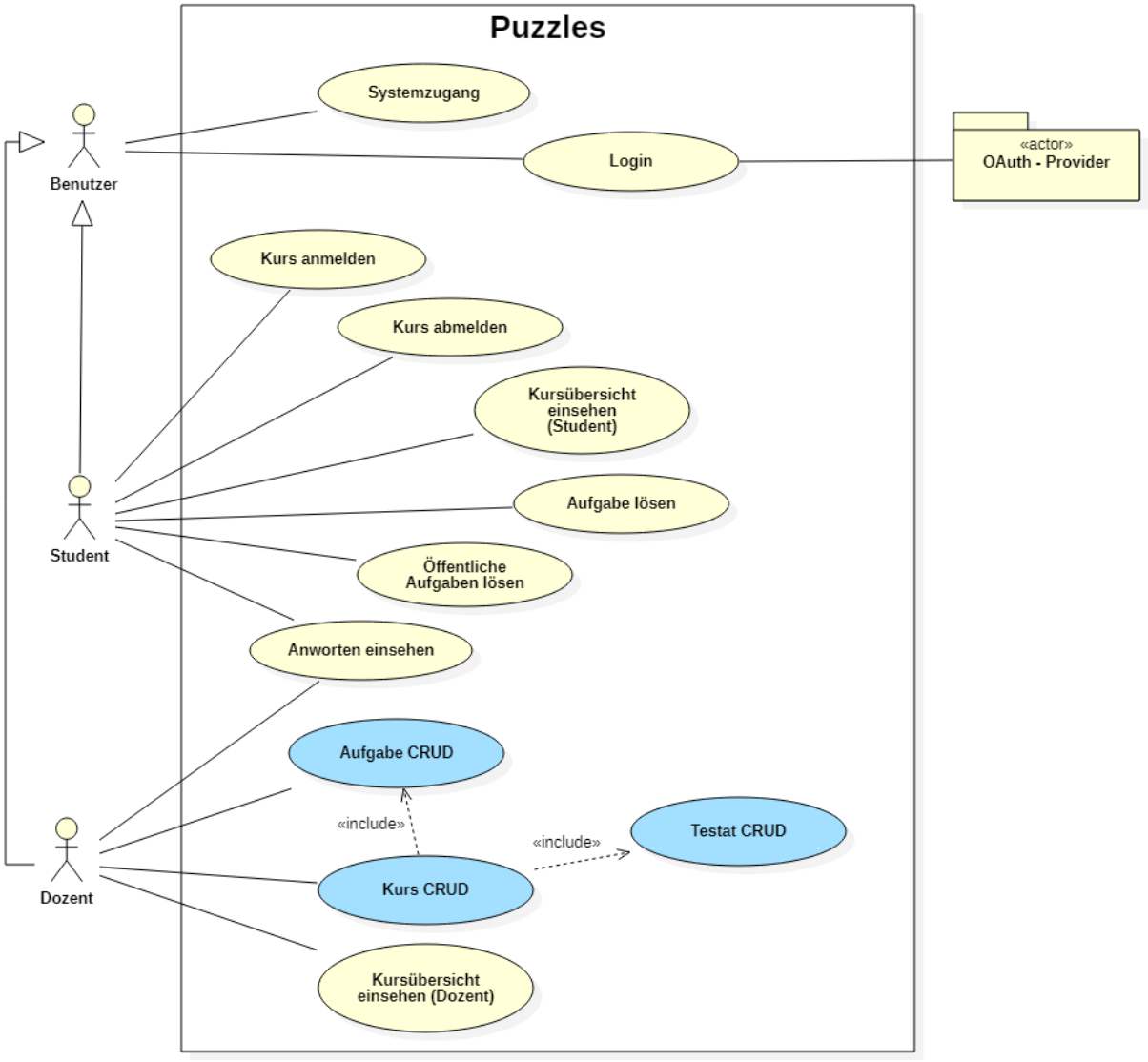


Abbildung 8-2: Use Case Diagramm

Farbe	Bedeutung
○	Normaler Use Case
○	CRUD Use Case (**C**reate **R**ead **U**pdate **D**elete)

8.3 Nicht funktionale Anforderungen

Die nicht funktionalen Anforderungen beruhen auf dem ISO Standard 9126-1. (ISO, 2019) Jedoch werden Kriterien, welche nicht angewendet werden können, ausgelassen.

8.3.1 NFR1: Änderbarkeit/Wartbarkeit

Modifizierbarkeit	Die Anwendung soll möglichst einfach durch andere Sprachen erweitert werden können. Ausserdem soll die Authentifizierung generisch gelöst werden, so dass durch Austauschen einer Komponente die Authentisierung beispielsweise durch das Azure Active Directory oder durch GitHub als Identity Provider erfolgen kann.
Testbarkeit	Änderungen an der Software sollen automatisch getestet werden indem bei jedem Push, unabhängig vom Branch, die definierten Tests durchgeführt werden.

8.3.2 NFR2: Benutzbarkeit

Bedienbarkeit und Erlernbarkeit	Die Anwendung soll ohne Schulung oder Einweisung bedienbar sein. Die Schritte sollen intuitiv implementiert werden und wo nötig sollen weitere Beschreibungen dem Bediener eine Hilfestellung geben. Die Überprüfung erfolgt durch Usability-Tests. Die Accessibilty soll mittels Verwendung von HTML Semantic Elements eingehalten werden.
Verständlichkeit	Die Anwendung soll auf Deutsch implementiert werden. Eine Ausnahme sind englische Fachbegriffe oder Namen. Sie soll von Personen, die der deutschen Sprache mächtig sind, bedient werden können.

8.3.3 NFR3: Effizienz

Zeitverhalten	Die Überprüfung einer Lösung soll folgende die mit folgender Formel berechneter Zeit in 95% der Fälle nicht übersteigen. Gemessen wird die Zeit vom Zeitpunkt des Drückens des «Submit»-Button bis zur Anzeige des Resultats (richtig/falsch). $$\boldsymbol{Javascript}: \boldsymbol{t = 2s + n * 0.05s}$$ $$\boldsymbol{CSS}: \boldsymbol{t = 2s + n * 1.5s}$$ $$\boldsymbol{wobei\ n}: \boldsymbol{Anzahl\ Tests}$$ Die Überprüfung erfolgt mittels Ausgabe der benötigten Zeit. Bei 10 Durchläufen darf das beschriebene Zeitlimit maximal 1 Mal überschritten werden.

8.3.4 NFR4: Funktionalität

Sicherheit	Die Kommunikation soll sobald sie über die Maschine hinaus geht verschlüsselt erfolgen. Der Browser soll das Zertifikat als Valid erkennen. Bei einem Absturz/Angriff während der Validierung soll der Schaden auf eine abgeschottete Umgebung begrenzt sein. Die Validierung darf nicht clientseitig durchgeführt werden.
Interoperabilität	Die Applikation soll für die gängigsten Browser funktionieren (Google Chrome ab V80, Safari ab V13, Mozilla Firefox ab V72). Die Applikation selbst soll allerdings nur auf einem Rechner mit dem OS Windows Server 2016 (Frontend) resp. Ubuntu 18.04 LTS (Backend + Runner) betrieben werden können.

8.3.5 NFR5: Übertragbarkeit

Installierbarkeit	Die Installation kann von fachkundigem Personal durchgeführt werden. Die Installation erfolgt mittels manueller Installation gemäss Anleitung. Die Anleitung soll von einer unabhängigen fachkundigen Person auf ihre Verständlichkeit überprüft werden.

8.3.6 NFR6: Zuverlässigkeit

Fehlertoleranz	Die Applikation soll auf Fehleingaben entsprechend reagieren und den Benutzer über den Fehler informieren, ohne der Person technische Details zu offenbaren. Die Fehlermeldungen dürfen keine Pfade, Exceptions oder Ähnliches enthalten. Mittels negativen Testcases sollen die Fehlermeldungen überprüft werden.

8.4 **Schnittstellen**

Schnittstelle	Beschreibung
Backend API	Schnittstelle zwischen Frontend und Backend Default-Port: 40000 (konfigurierbar)
Datenbank	Dateibasierender Zugriff auf die Datenbank
Runner	Aufruf via Docker-Befehl

8.5 **Rahmenbedingungen**

Für die Applikation müssen folgende Bedingungen erfüllt sein:

- Mindestens ein Server auf dem die Applikation läuft. Als Alternative kann das Backend auch auf einer separaten Maschine gestartet werden.
- Die Frontend-Maschine muss via Port 443 erreichbar sein.
- Die Backend-Maschine muss via beliebigen Port erreichbar sein (muss entsprechend konfiguriert werden).
- Auf der Backend-Maschine muss Docker installiert sein. Es empfiehlt sich die Backend-Maschine mit einem Linux OS zu wählen (Docker ist performanter und die Installation einfacher).
- Falls Frontend und Backend getrennt werden, müssen die beiden Maschinen miteinander kommunizieren können. (Port und Protokoll (HTTP/ HTTPS) sind konfigurierbar).

9 Domainanalyse

9.1 **Domain Modell**

Um die Aufgabenstellung zu erfüllen wurde folgendes Domain Modell ausgearbeitet:

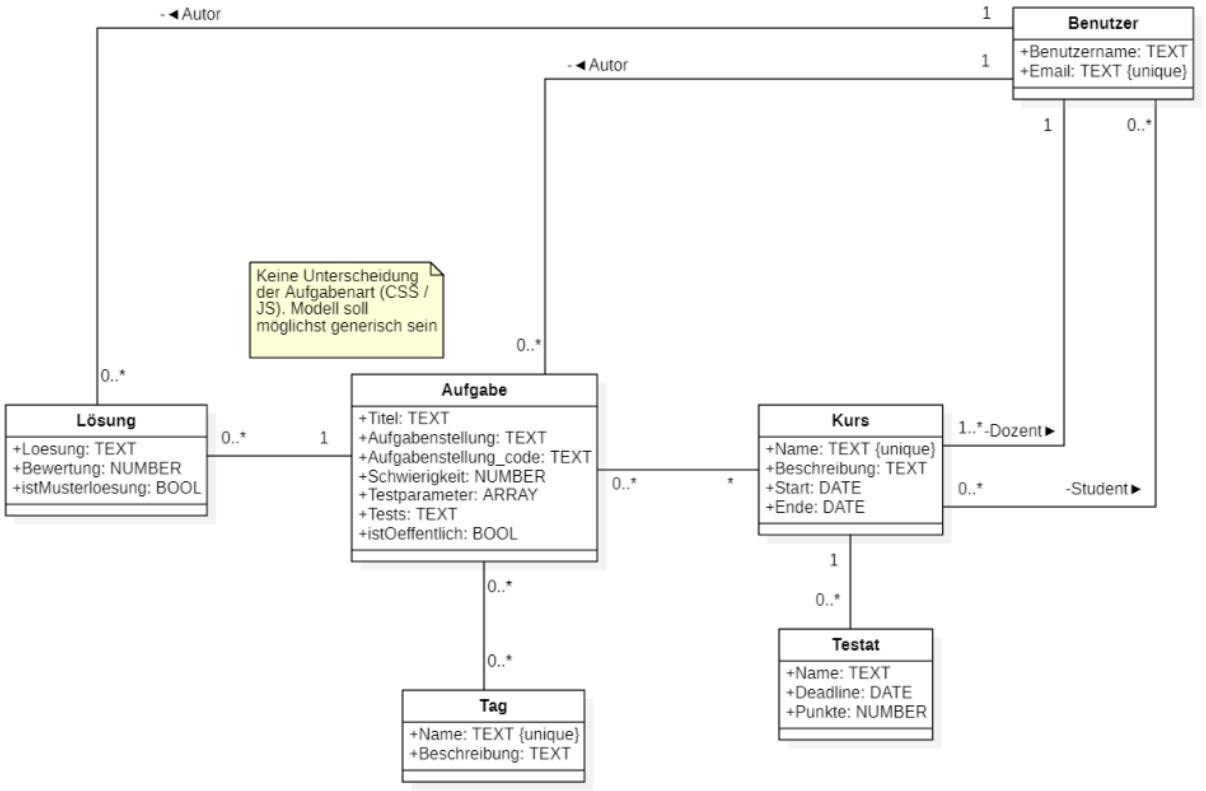


Abbildung 9-1: Domain Modell

Klasse	Attribut	Erläuterung
Benutzer	Benutzername	Benutzername des Benutzers
	Email	E-Mail-Adresse des Benutzers
Kurs	Name	Name des Kurses
	Beschreibung	Beschreibung des Kurses
	Start	Startdatum des Kurses
	Ende	Enddatum des Kurses
Testat	Name	Name des Testats («Testat Grundkenntnisse»)
	Deadline	Datum für Erfüllung des Testats
	Punkte	Anzahl Punkte die für die Erfüllung des Testats erreicht werden müssen.
Aufgabe	Titel	Titel der Aufgabe
	Aufgabenstellung	Aufgabenstellung als Text
	Aufgabenstellung_code	Aufgabenstellung als Code / Gerüstcode für Aufgabe
	Schwierigkeit	Schwierigkeitsgrad 1 bis 5
	Testparameter	Parameter für Tests (bei CSS sind das die Viewports, bei anderen Aufgaben das Testframework)
	Tests	Auszuführende Tests für Aufgabe
	istOeffentlich	FLAG: Falls true → Aufgabe ersichtlich für auch nicht eingeschriebene Benutzer
Tag	Name	Name vom Tag
Lösung	Loesung	Lösung (Code)
	Bewertung	Bewertung der Lösung 1 bis 5
	istMusterlösung	FLAG: Falls true → Lösung ist Beispiellösung

9.2 **Datenmodell**

Aus dem im vorangegangenen Kapitel aufgeführten Domain Modell entsteht folgendes Datenmodell.

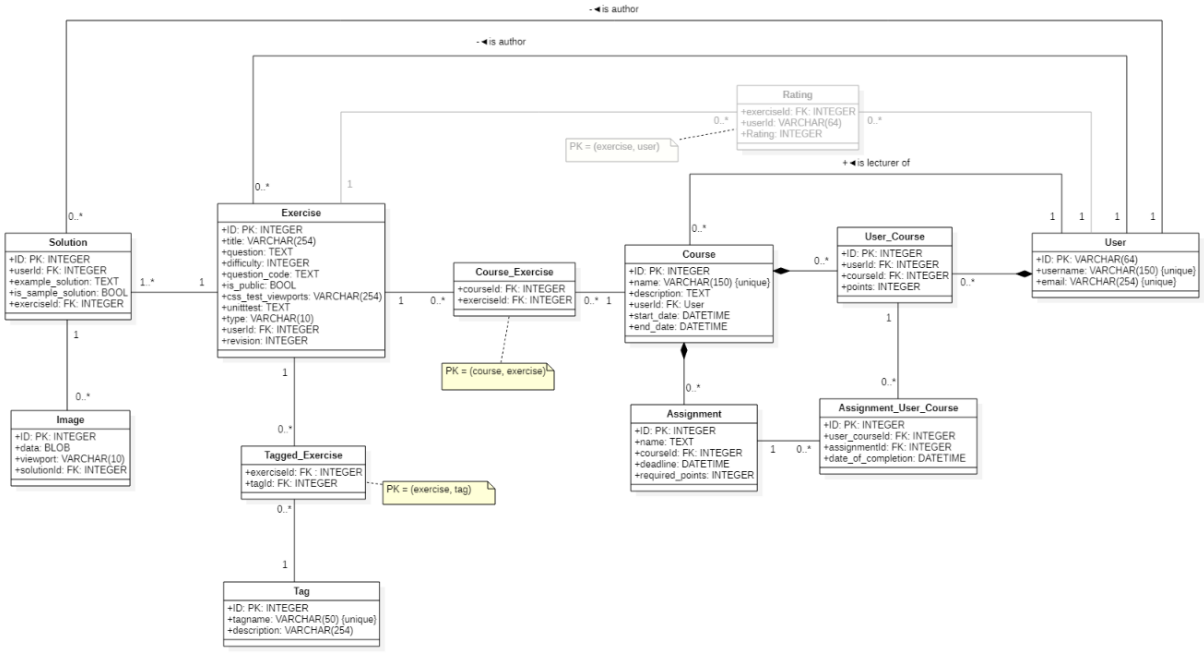


Abbildung 9-2: Datenmodell

► Eine vergrösserte Ansicht befindet sich im *Anhang D: Datenmodell*

Die Tabelle «Rating» ist ausgegraut, da die Bewertung der Aufgaben im Verlauf der Arbeit aus dem Project Scope gestrichen wurde und die Tabelle somit nicht mehr benötigt wird. Die Verkleinerung des Project Scope ging mit einer nachträglichen Änderung auf Wunsch des Gegenlesers, des Experten sowie des Betreuers einher.

10 Architektur und Design

Die Designentscheidungen sind in der Form von Y-Statements (Zdun, Capilla, Tran, & Zimmermann, 2020) aufgeführt. Für die Visualisierung der Architektur wurde das C4 Modell gewählt.

► Eine vergrösserte Ansicht der Diagramme befindet sich im *Anhang C: Diagramme*

Weder die Aufgabenstellung noch der Betreuer haben für diese Arbeit vorab Design Constraints definiert.

10.1 **Kontextdiagramm**

Das System befindet sich im folgenden Kontext:

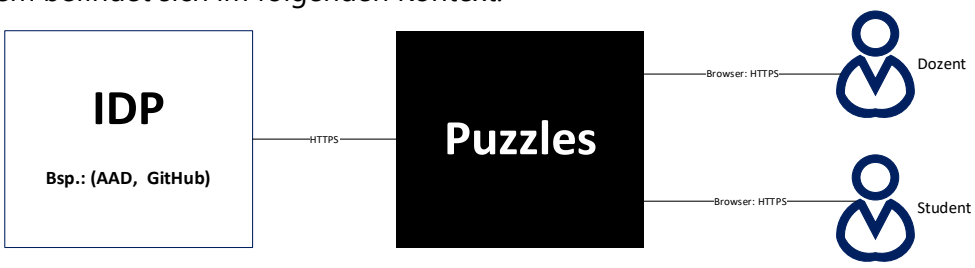


Abbildung 10-1: Kontextdiagramm

10.1.1 Designentscheidung: OAuth

IM KONTEXT	der Bachelorarbeit «Puzzles»
KONFRONTIERT MIT	den beiden Anforderungen *FR2*: Login und *NFR1: Änderbarkeit/Wartbarkeit*
FIEL DIE ENTSCHEIDUNG AUF	die Verwendung von OAuth
UND NICHT AUF	das Anlegen und Pflegen von eigenen Benutzern / -konten
UM	die Authentifizierung offen und generisch zu halten
UNTER ANNAHME/AKZEPTANZ	einer erhöhten Komplexität im Frontend und einer Steigerung der dort vorhandenen Programmlogik.

10.1.2 Designentscheidung: Browserzugriff

IM KONTEXT	der Bachelorarbeit «Puzzles»
KONFRONTIERT MIT	den beiden Anforderungen *FR1: Systemzugang* und *NFR4: Funktionalität*
FIEL DIE ENTSCHEIDUNG AUF	die Implementation einer Webanwendung
UND NICHT AUF	auf eine Client-Server Applikation
UM	unabhängig vom Client-OS zu sein und ein grösseres Zielpublikum erreichen zu können

UNTER ANNAHME/AKZEPTANZ	der verminderten Stabilität sowie der Eigenheiten der einzelnen Hersteller.

10.2 **Containerdiagramm**

Puzzles besteht aus den folgenden Containern:

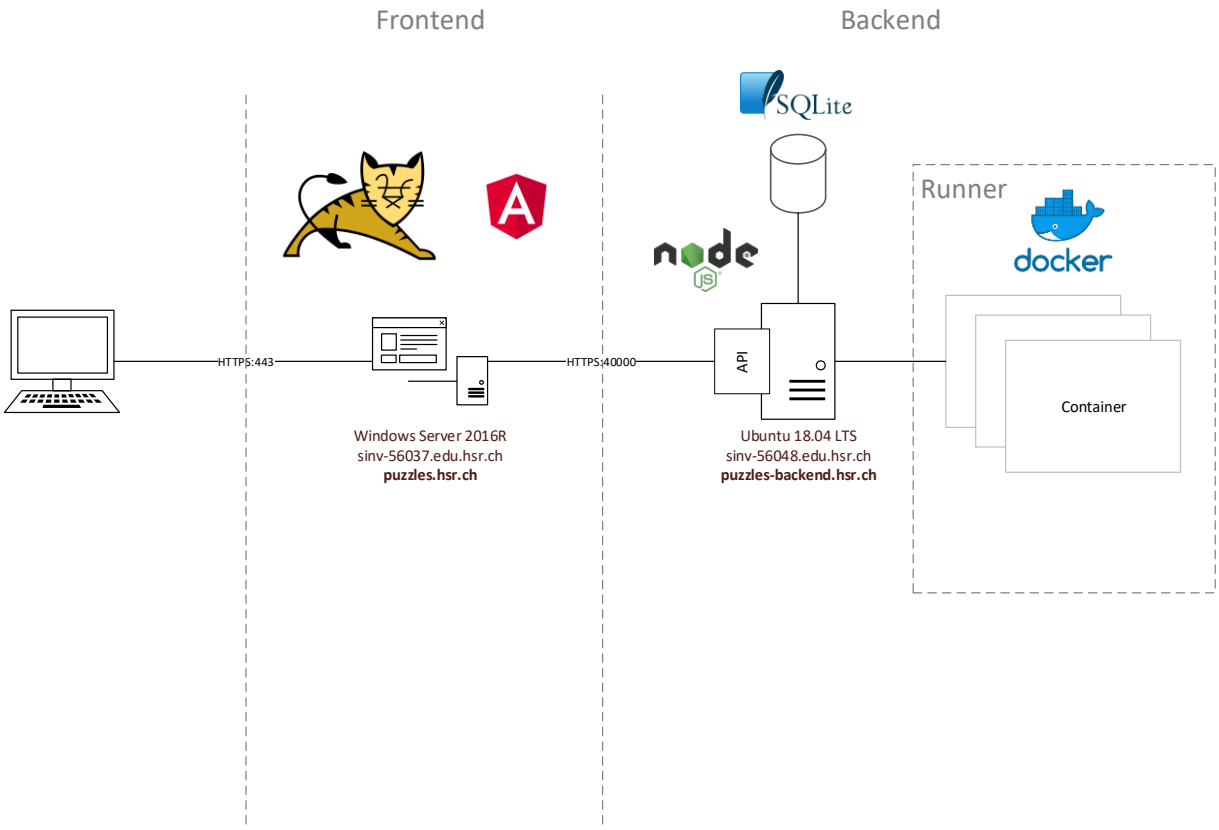


Abbildung 10-2: Containerdiagramm

10.2.1 Designentscheidung: Aufteilung Frontend / Backend / Runner

IM KONTEXT	der Bachelorarbeit «Puzzles»
KONFRONTIERT MIT	den Anforderungen *NFR1: Änderbarkeit/Wartbarkeit* und *NFR4: Funktionalität*
FIEL DIE ENTSCHEIDUNG AUF	die Separierung von Frontend, Backend und Runner in einzelne Komponenten
UND NICHT AUF	die Implementierung in einer «Big Box»
UM	die Entkoppelung zu begünstigen und somit die Austauschbarkeit und Erweiterbarkeit zu vereinfachen. Ausserdem wird dadurch das Schadensausmass im Worst-Case auf eine Komponente begrenzt
UNTER ANNAHME/AKZEPTANZ	der höheren Komplexität der Dateistruktur und des Deployments.

10.2.2 Designentscheidung: HTTPS

IM KONTEXT	der Bachelorarbeit «Puzzles»
KONFRONTIERT MIT	der Anforderung *NFR4: Funktionalität*
FIEL DIE ENTSCHEIDUNG AUF	die Verwendung von HTTPS für die Kommunikation zwischen den Komponenten
UND NICHT AUF	die Verwendung von HTTP
UM	die sichere (verschlüsselte) Kommunikation zwischen dem Frontend und dem Backend sowie zwischen Client und Frontend zu gewährleisten
UNTER ANNAHME/AKZEPTANZ	des Mehraufwands ein gültiges Zertifikat zu erhalten und einzubinden.

10.2.3 Designentscheidung: Angular

IM KONTEXT	der Bachelorarbeit «Puzzles»
KONFRONTIERT MIT	Der Anforderung *NFR4: Funktionalität*
FIEL DIE ENTSCHEIDUNG AUF	die Verwendung von Angular für das Frontend
UND NICHT AUF	die Verwendung von React oder Vue
UM	den komponenten-basierten Aufbau der Applikation zu unterstützen und zu erleichtern. Ausserdem bietet Angular eine leicht zu implementierende Hierachical Dependency Injection an und unterstützt durch eingebaute Features (Auto Reload, etc.) beim Entwickeln
UNTER ANNAHME/AKZEPTANZ	der erhöhten Komplexität und des Zusatzaufwands um sich mit Angular vertraut zu machen.

10.2.4 Designentscheidung: Apache Tomcat

IM KONTEXT	der Bachelorarbeit «Puzzles»
KONFRONTIERT MIT	den Anforderungen *FR1: Systemzugang* und *NFR4: Funktionalität*
FIEL DIE ENTSCHEIDUNG AUF	die Verwendung von Apache Tomcat als Webserver für das Frontend
UND NICHT AUF	die Implementation eines eigenen Webservers oder die Verwendung von anderen Produkten (IIS, NGINX, etc.)
UM	das Frontend zu bedienen
UNTER ANNAHME/AKZEPTANZ	der notwendigen Installation und Konfiguration von Apache Tomcat auf der Frontend-Maschine.

10.2.5 Designentscheidung: NodeJS / ExpressJS

IM KONTEXT	der Bachelorarbeit «Puzzles»
KONFRONTIERT MIT	Der Anforderung *NFR4: Funktionalität*
FIEL DIE ENTSCHEIDUNG AUF	die Verwendung von NodeJS für das Backend
UND NICHT AUF	die Verwendung von Django, Spring Boot, etc.
UM	eine einheitliche Sprache (Javascript / Typescript) durch das ganze System zu erhalten
UNTER ANNAHME/AKZEPTANZ	-

10.2.6 Designentscheidung: SQLite

IM KONTEXT	der Bachelorarbeit «Puzzles»
KONFRONTIERT MIT	der Anforderungen *NFR5: Übertragbarkeit*
FIEL DIE ENTSCHEIDUNG AUF	die Verwendung einer SQLite Datenbank
UND NICHT AUF	andere DB Provider (Postgres, MySQL, MongoDB)
UM	das Deployment und die Installation der Applikation zu vereinfachen (es ist keine Installation von sonstigen Tools notwendig und für das Deployment muss nur eine Datei erstellt resp. verteilt werden) und um eine möglichst hohe Performance zu erreichen (schnelle Zugriffe).
UNTER ANNAHME/AKZEPTANZ	der mit SQLite einhergehenden Restriktionen (paralleler Zugriff) sowie der weiteren Annahme der SQLite spezifischen Annotationen und Dateitypen.

10.2.7 Designentscheidung: Docker

IM KONTEXT	der Bachelorarbeit «Puzzles»
KONFRONTIERT MIT	den Anforderungen *NFR1: Änderbarkeit/Wartbarkeit* und *NFR4: Funktionalität*
FIEL DIE ENTSCHEIDUNG AUF	die Verwendung von Docker und Docker Container für die Ausführung von abgegebenem Code
UND NICHT AUF	die Validierung von Code/Aufgaben auf dem Backend-Server
UM	die benötigte Sicherheit und Kapselung zu erreichen und die Gefahr für den Server durch schadhaften Code auf ein Minimum zu reduzieren.
UNTER ANNAHME/AKZEPTANZ	einer Minderung der Performance durch das Aufstarten und Abräumen der Docker Container. (Gefahr für *NFR3: Effizienz*)

10.3 **Komponentendiagramm**

10.3.1 Frontend

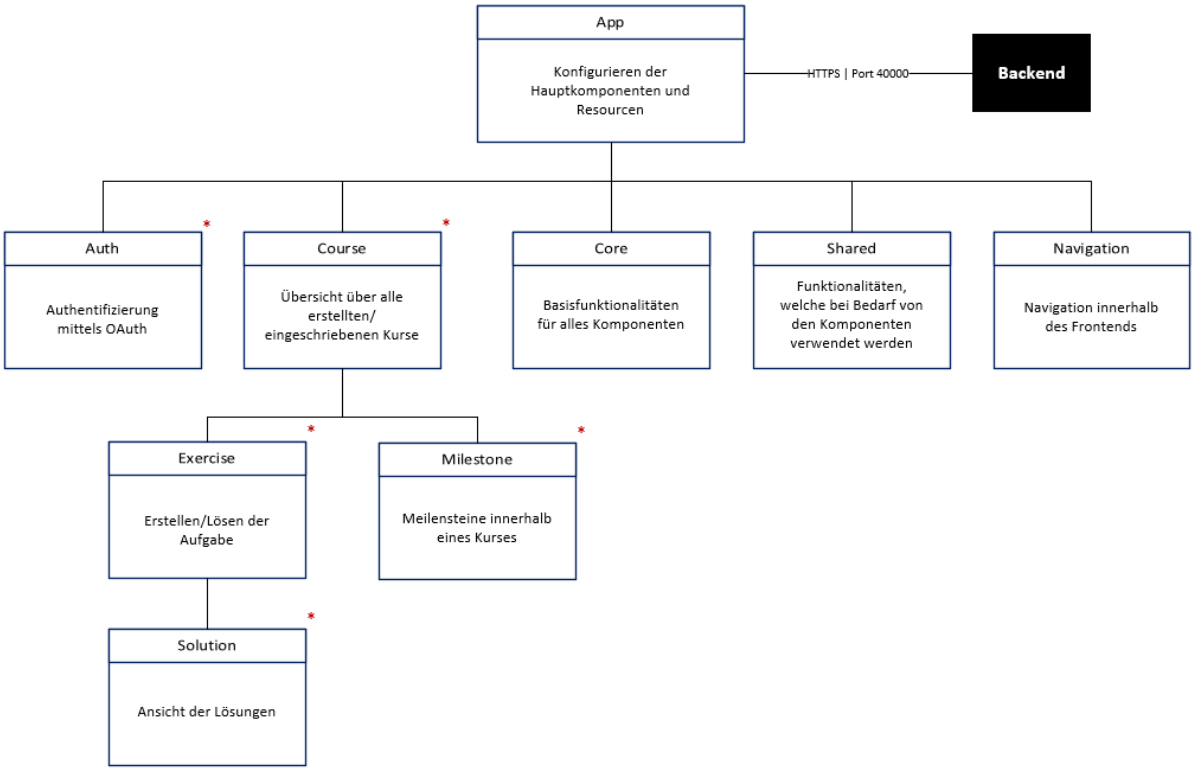


Abbildung 10-3: Frontend Komponentendiagramm

Die Strukturierung des Frontend hält sich an die Guidelines von Angular. (Angular, Angular, 2020)

* = Komponenten, welche mit dem Backend kommunizieren. Diese Komponenten fungieren im Angular als Module und besitzen neben der View-Logik weitere Eigenschaften.

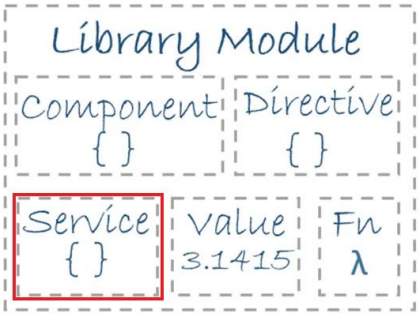


Abbildung 10-4: Eigenschaft Angular Modul (Angular, Angular, 2020)

Für den Datenaustausch wurde ein Service implementiert, welcher HTTPS Request zum Backend durchführt.

10.3.2 Backend

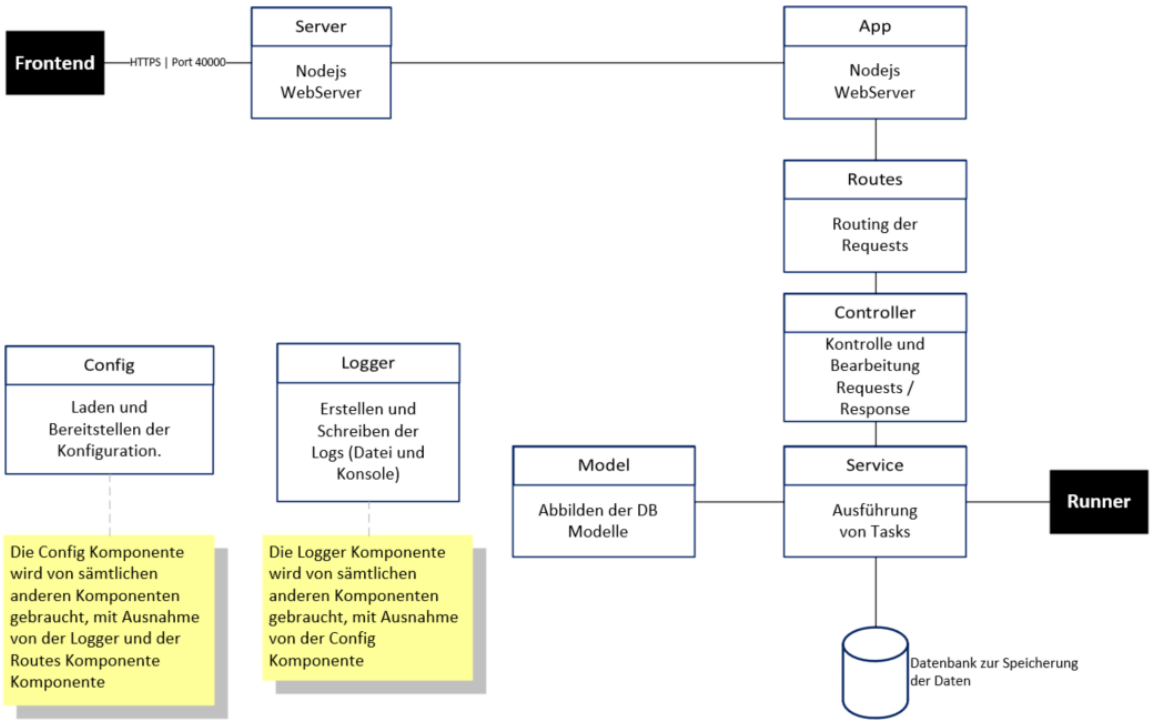


Abbildung 10-5: Backend Komponentendiagramm

Die Strukturierung des Backens wurde anhand von Best Practices definiert.

10.3.3 Runner

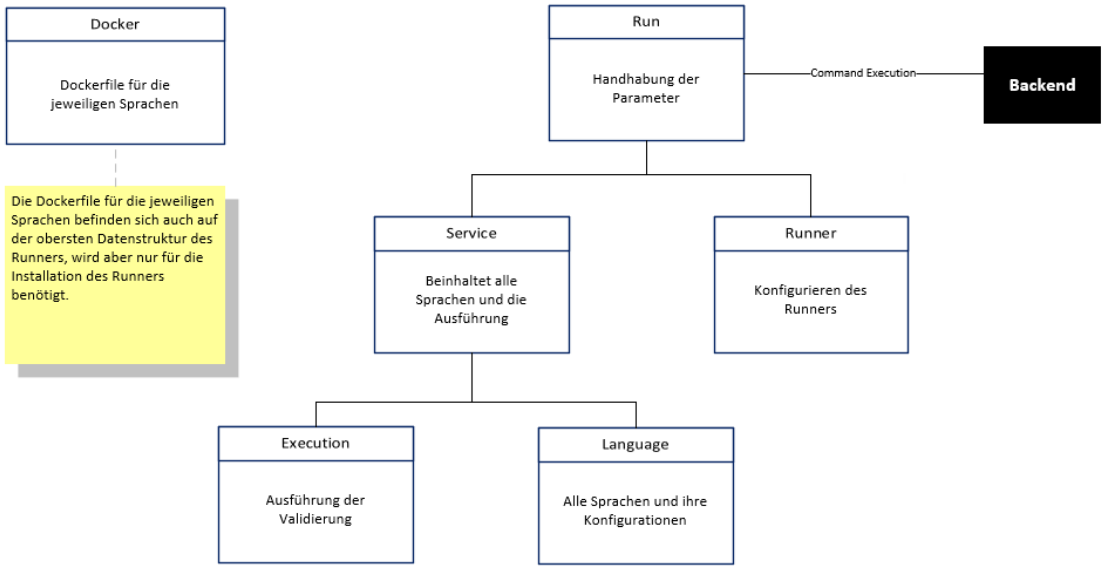


Abbildung 10-6: Runner Komponentendiagramm

Beim Aufbau des Runners wurde darauf geachtet, dass neue Sprachen einfach eingebaut werden können.

10.4 Analyse visueller Vergleich (CSS-Aufgaben)

Der visuelle Vergleich von CSS-Aufgaben ist ein Kernpunkt dieser Arbeit, darum wurde grosser Wert auf die Wahl des Vergleichsalgorithmus gelegt. In diesem Kapitel ist die Analyse von vier verschiedenen Bibliotheken beschrieben.

Für den visuellen Vergleich wurden folgenden Libraries genauer untersucht:

- Pixematch https://www.npmjs.com/package/pixelmatch
- Rembrandt https://www.npmjs.com/package/rembrandt
- ResembleJS https://www.npmjs.com/package/resemblejs
- Pixel-diff https://www.npmjs.com/package/pixel-diff

► Die folgenden Angaben wurden Anfangs Mai 2020 ermittelt.

10.4.1 Entscheidungsmatrix

Zusammengefasst sind folgende Ergebnisse aus der Analyse entstanden:

Kriterium	Gewichtung	Pixelmatch	Pixel-diff	Rembrandt	ResembleJS
Aktuell/Aktiv	5	🟢	🟡	🔴	🟢
Verbreitet	3	🟢	🟡	🔴	🟡
Gute Dokumentation	4	🟢	🟡	🟡	🟢
Diff Bild	5	🟡	🟡	🟢	🟢
Parametrisierbarer Vergleich	5	🟡	🟡	🟢	🟡
Einfache Handhabung	3	🟡	🟢	🟢	🟢
	Summe:	**62**	**55**	**47**	**67**

🟢	Übertroffen	3
🟡	Erfüllt	2
🟡	Teilweise erfüllt	1
🔴	Nicht erfüllt	0

► Die Punkte setzten sich aus der Summe der Produkte der Gewichtung des Kriteriums und der Anzahl erreichter Punkte im entsprechenden Kriterium zusammen.

10.4.2 Pixelmatch

Lizenz	ISC
Version	5.2.0
Letzte Änderung	April 2020

«Pixelmatch» ist bei Weitem die beliebteste Library (>1 Mio. wöchentliche Downloads). Der Einstieg ist leicht komplizierter als jener bei den anderen Libraries, es wurde keine Möglichkeit gefunden beim Vergleich einen farblichen Output zu generieren. Der Vergleich berücksichtigt zwar die Farben aber das Diff-Image ist immer schwarz-weiss. Um dem Benutzer besser auf die Fehler aufmerksam zu machen, sollten das Diff die Farben vom Ursprungsbild enthalten.

10.4.3 Pixel-diff

Lizenz	MIT
Version	1.0.1
Letzte Änderung	2017

Wie bei den beiden Bibliotheken «Rembrandt» und «Pixelmatch» ist auch bei «pixel-diff» der Einstieg einfach. Auch bei dieser Library lassen sich nur wenige Parameter konfigurieren (Threshold, AntiAliasing, etc.). Ausserdem wurde sie seit geraumer Zeit nicht mehr weiterentwickelt. Aus diesem Grund wurde diese Library ebenfalls verworfen,

10.4.4 Rembrandt

Lizenz	MIT
Version	0.1.3
Letzte Änderung	2018

Die Bibliothek «Rembrandt» bietet trotz ihrer geringen Verbreitung (~400 wöchentliche Downloads auf npm) eine gute Dokumentation und einen einfachen Einstieg. Ausserdem erlaubt diese Library eine einfache Konfiguration des Bildvergleichs. Ein erster Prototyp war in wenigen Minuten implementiert. Die Ergebnisse dieser Library sind sehr zufriedenstellend, sie ist die einzige die einen Parameter für den Offset. Allerdings hat sie einige schwerwiegende Nachteile:

- Geringe Verbreitung
- Seit 2 Jahren nicht mehr weiterentwickelt
- Der momentane Build Status lautet «Error»

Aus diesen Gründen wurde diese Library nicht weiterverwendet. Wenn die erwähnten Nachteile nicht so gravierend wären, wäre diese Library die erste Wahl gewesen. Durch den konfigurierbaren Offset bietet sie einen Vorteil, den keine andere untersuchte Library ermöglicht.

10.4.5 Resemblejs

Lizenz	MIT
Version	3.2.4
Letzte Änderung	März 2020

Die Library «Resemblejs» erlaubt die Konfiguration verschiedener Parameter, ausserdem kann auch ein farbliches Diff erzeugt werden (beinhaltet auch die Farben vom Ursprungsbild). Die Library erfreut sich akzeptabler Beliebtheit (30'000 wöchentliche Downloads) und die Dokumentation ist gut. Ausserdem ist sie einfach einsetzbar. Aus diesen Gründen fiel die Wahl auf «resemblejs».

10.4.5.1 Analyse

Um diese Library genauer zu erläutern werden im Folgenden einige Beispiele analysiert. Hierzu wurde ein Codefragment[16] von W3Schools[17] verwendet. Das Ausgangsbild entspricht der Musterlösung und bleibt immer gleich. Es sieht folgendermassen aus (500x300):

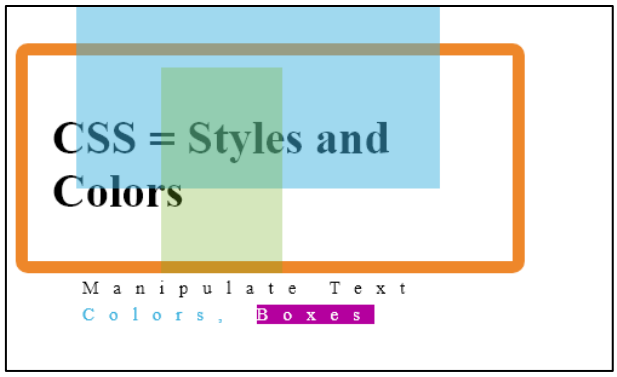


Abbildung 10-7: Ausgangsbild f. Vergleichsanalyse

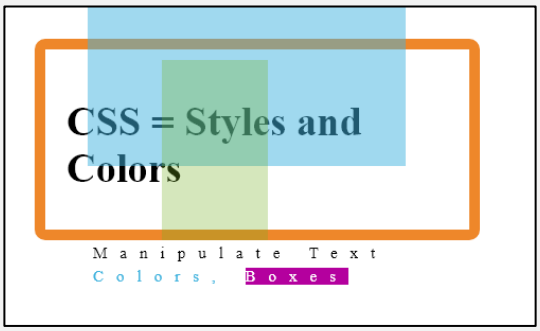
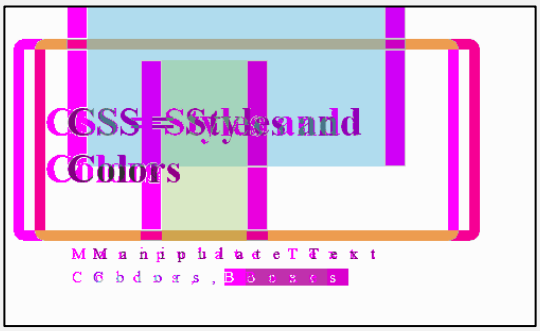
Dieses Bild wurde gewählt, weil es verschiedene gute Kriterien, wie beispielsweise transparente und/oder überlappende farbliche Bereiche, enthält.

- Der Rahmen gehört nicht zum Ursprungsbild und wurde nachträglich hinzugefügt, um die Änderungen in Bezug auf die Position besser sichtbar zu machen.
- Für die Analyse wurde der Threshold auf 1% gesetzt.

[16] https://www.w3schools.com/html/tryit.asp?filename=tryhtml_css_full
[17] https://www.w3schools.com/

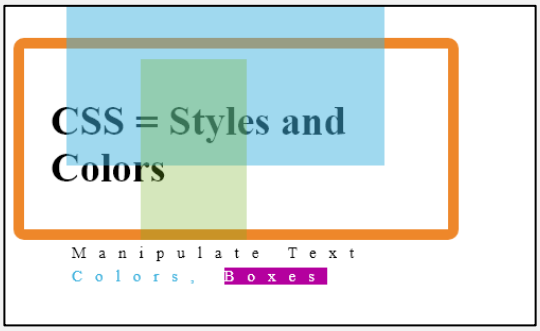
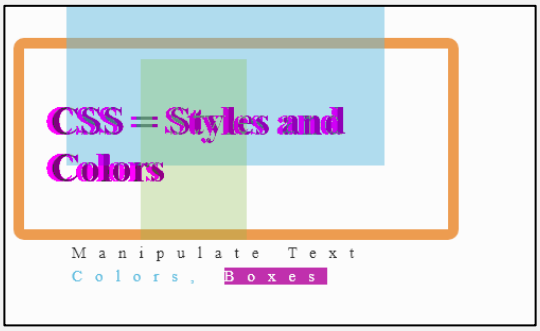
10.4.5.2 Position

Als erstes wurde untersucht welche Unterschiede festgestellt werden, wenn das ganze Bild (resp. der gesamte Inhalt) verschoben ist. Hierzu wurde ein Margin-Left von 20px eingefügt.

Erhaltenes Bild	Diff Bild
	

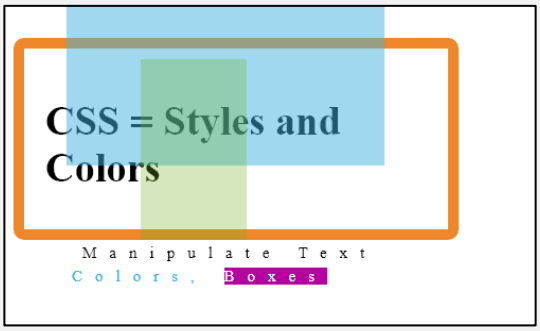
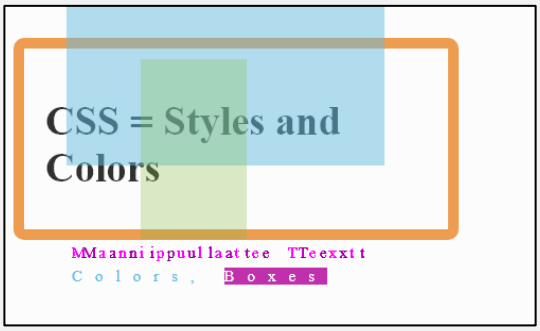
Die Abweichungen werden korrekt erkannt und das Resultat ist dementsprechend negativ.

Anschliessend wurde untersucht, wie sich der Algorithmus verhält, wenn nur ein einzelnes Element nicht an der vorgesehenen Position befindet. Hierzu wurde die Überschrift «CSS = Styles and Colors» mit einem Margin-Left von 5px nach rechts verschoben.

Erhaltenes Bild	Diff Bild
	

Da auch hier die Abweichungen prozentual gross sind, fällt das Resultat ebenfalls negativ aus.

Kleinere Änderungen oder Abweichungen werden allerdings je nach Einstellung des Thresholds, zugelassen. Der folgende Versuch beinhaltet hat ein Margin-Left von 10px beim kleinen Text.

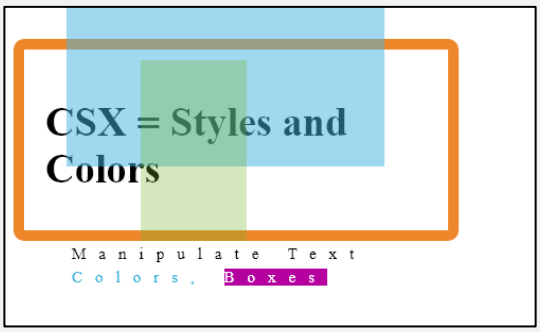
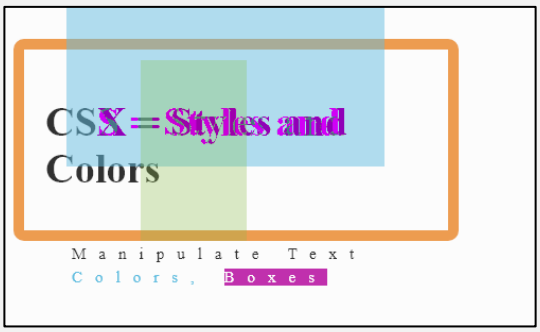
Erhaltenes Bild	Diff Bild
	

Bei einem Threshold von 1% ist das Resultat positiv, verkleinert man den Threshold allerdings auf 0.5% ist das Resultat negativ.

10.4.5.3 Inhalt

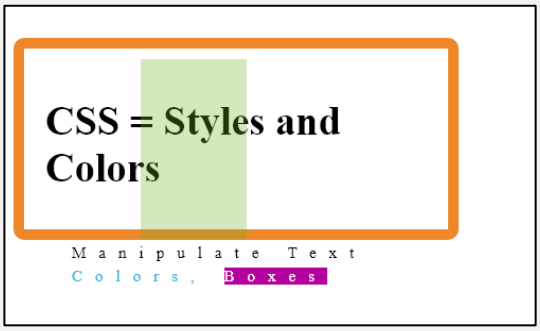
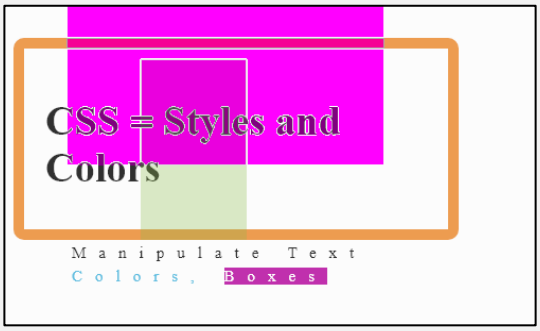
Anschliessend wurde untersucht, wie der Algorithmus auf einen anderen Inhalt reagiert. Hierbei ist nicht der Inhalt des Codes gemeint, sondern der Inhalt der Seite (Text, Rahmen, etc.).

Tippfehler:

Erhaltenes Bild	Diff Bild
	

Durch die Grösse des Texts wird je nachdem welcher Buchstabe aus dem Haupttitel abgeändert wird, das Resultat beeinflusst. Das Resultat des obigen Tests ist negativ. «Vertippt» man sich allerdings beim kleinen Text «Manipulate Text, Colors, Boxes» ist das Resultat bei 1% Threshold trotzdem positiv. Der gleiche Effekt entsteht, wenn man ähnliche Buchstaben verwendet, beispielsweise statt kleines «L» ein grosses «i». Die Pixel-Abweichung ist in diesem Fall minimal.

Ohne Box (hellblau):

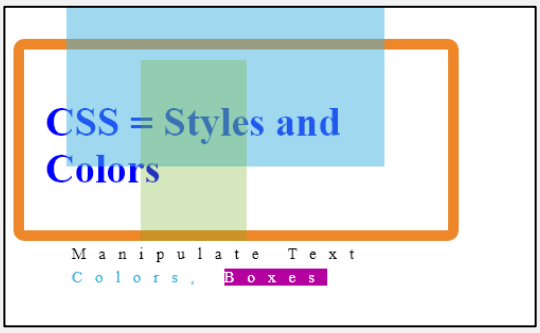
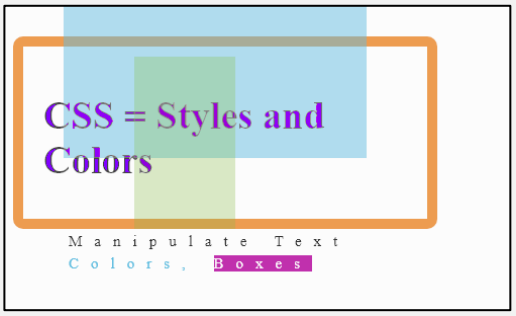
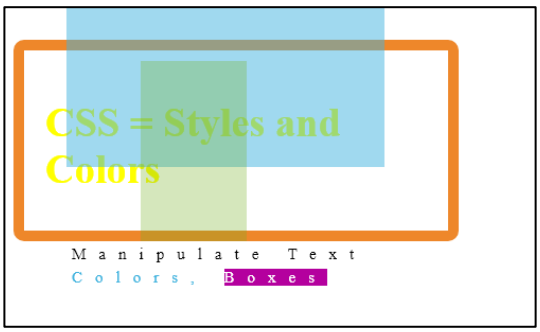
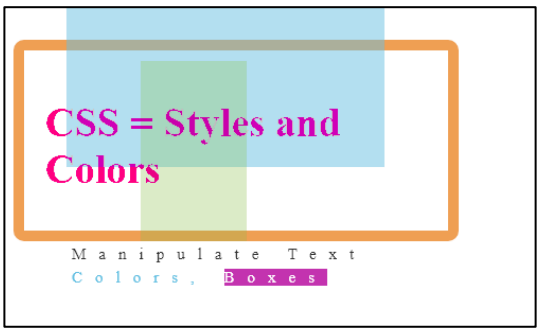
Erhaltenes Bild	Diff Bild
	

Grössere Elemente werden zuverlässig erkannt.

Es wurde festgestellt, dass fehlende Elemente zuverlässiger detektiert werden als abgeänderte. Gerade wenn die Elemente durch ähnliche ausgetauscht werden, hat der Algorithmus Mühe.

10.4.5.4 Farbe

Für den folgenden Versuch wurde allein die Textfarbe der Hauptüberschrift angepasst.

Erhaltenes Bild	Diff Bild
	
	

Je nach «Ähnlichkeit» wird das Resultat akzeptiert. Wenn die Überschrift statt schwarz beispielsweise rot (#FF0000) oder blau (#0000FF) ist, wird das Resultat akzeptiert, wenn sie allerdings gelb (#FFFF00) ist nicht. Dies lässt sich auch am Diff Bild erkennen. Bei der blauen Variante wurde weniger eingefärbt als bei der gelben. Dem kann entgegengewirkt werden,

indem man den Threshold auf einen kleineren Wert setzt. Bei einem Threshold von 0.5% wird auch blau und rot nicht mehr akzeptiert.

10.4.5.5 Sicherheit

Da es sich grundsätzlich um eine HTML Datei handelt, kann auch Javascript eingeschleust werden. Auch wenn der Code ausgeführt wird, passiert dies schlussendlich nur im Docker Container. Somit ist der Zugriff auf das Filesystem vom Backend Server nicht möglich.

Wenn beispielsweise folgendes Codefragment eingefügt wird, wird es auch ausgeführt:

```
<script>
  alert('Puzzles is awesome!')
</script>
```
Javascript

Um Angriffe auf die Infrastruktur abzufangen, wurde die Designentscheidung *10.2.7 Designentscheidung: Docker* getroffen. Zusätzlich wurde ein Timeout eingebaut.

10.4.5.6 Fazit

Die Bibliothek ResembleJS bietet eine zuverlässige Methode, um einen einfachen Bildvergleich durchzuführen. Trotz ihrer geringeren Beliebtheit im Vergleich zu Pixelmatch liefert sie die gleichen Ergebnisse und zusätzlich ein farblichen Diff Bildes. Der optimale Threshold hängt allerdings stark von der Aufgabe und der Grösse sowie Farbe der einzelnen Elemente ab.

10.5 **Erweiterbarkeit**

Ein Kernpunkt dieser Arbeit ist die Erweiterbarkeit der Anwendung durch andere Sprachen. Im Umfang dieser Arbeit sind die Sprachen Javascript und CSS enthalten. Es wurde überprüft, ob die gewählte Architektur mit annehmbarem Aufwand die Erweiterung um eine weitere Sprache (in diesem Beispiel Java) zulässt. Für den in diesem Kapitel beschriebenen Test wurde die Runner Komponenten sowie das Backend angepasst.
Die Änderungen im Frontend sind minimal. Es bedarf keiner neuen Maske, es muss lediglich ein neuer Aufgabentyp bei der Erstellung der Aufgaben hinzugefügt werden. Dieser Wert muss schlussendlich mit dem im Backend konfigurierten Wert für den entsprechenden Docker Container übereinstimmen.

10.5.1 Dockerfile

Zuerst muss ein neues Dockerfile erstellt werden. Dadurch werden alle notwendigen Bibliotheken und Einstellung in einem Docker Image bereitgestellt. Für das Java-Image wird neben dem Java Development Kit (JDK) auch Apache Maven benötigt. Letzteres wird benötigt, um die Tests auszuführen und einen Testreport zu generieren.

Zusätzlich zur sprachspezifischen Einstellung muss der Docker aufgrund des Runners, welcher in Javascript geschrieben ist, Node unterstützen. Bei mehreren Dockerfiles ist es sinnvoll, ein Basis-Dockerfile zu schreiben, welches dann von allen anderen importiert werden kann.

Schlussendlich sieht das Dockerfile für die Sprache Java, ohne Basis Konfigurationen, folgendermassen aus:

```
# JDK
RUN apt-get install -y openjdk-8-jdk-headless

# MAVEN
ARG MAVEN_VERSION=3.6.3
ARG USER_HOME_DIR=/home/puzzles
ARG
SHA=c35a1803a6e70a126e80b2b3ae33eed961f83ed74d18fcd16909b2d44d7dada3203f1ffe726c17e
f8dcca2dcaa9fca676987befeadc9b9f759967a8cb77181c0
ARG BASE_URL=https://apache.osuosl.org/maven/maven-3/${MAVEN_VERSION}/binaries

RUN mkdir -p /usr/share/maven /usr/share/maven/ref \
  && curl -fsSL -o /tmp/apache-maven.tar.gz ${BASE_URL}/apache-maven-
${MAVEN_VERSION}-bin.tar.gz \
  && echo "${SHA}  /tmp/apache-maven.tar.gz" | sha512sum -c - \
  && tar -xzf /tmp/apache-maven.tar.gz -C /usr/share/maven --strip-components=1 \
  && rm -f /tmp/apache-maven.tar.gz \
  && ln -s /usr/share/maven/bin/mvn /usr/bin/mvn

ENV MAVEN_HOME /usr/share/maven
ENV MAVEN_CONFIG "$USER_HOME_DIR/.m2"
```

Dockerfile

10.5.2 Runner

Neben dem Dockerfile benötigt der Runner ebenfalls eine kleine Erweiterung. Für jede Sprache wird ein Service benötigt. Dieser muss gleich wie die Sprache benannt werden und muss mindestens die Datei «`index.js`» beinhaltet. Der Grund dafür ist, dass der Code per Factory Methode ausgeführt wird. Jeder Service der jeweiligen Sprache muss den Code in ein Datei schreiben und einen Konsolenbefehl für die spätere Ausführung retournieren. Aufgrund der Factory Methode ist es möglich weitere Sprachen zu unterstützen, ohne weitere Codestellen anzupassen.

Bei Java reicht eine simple Datei, um den Code zu validieren leider nicht aus. Damit Maven die Tests durchführen kann, wird eine ganze Ordnerstruktur benötigt. Aufgrund dessen wird der Produktionscode in «`src/main`» abgelegt und die Unit Tests werden im «`src/test`» hinterlegt. Zusätzlich muss ein «`pom.xml`» bereitgestellt werden. Dort sind Bibliotheken und Maven spezifische Builds definiert. Nach Bedarf können weitere Bibliothek dort hinzugefügt werden.

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>ch.hsr.puzzles</groupId>
    <artifactId>attempt</artifactId>
    <version>1.0-SNAPSHOT</version>

    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <maven.compiler.source>1.8</maven.compiler.source>
        <maven.compiler.target>${maven.compiler.source}</maven.compiler.target>
        <junit.jupiter.version>5.6.2</junit.jupiter.version>
        <junit.platform.version>1.6.2</junit.platform.version>
    </properties>
```

```
    <dependencies>
        <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter-engine</artifactId>
            <version>${junit.jupiter.version}</version>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.junit.platform</groupId>
            <artifactId>junit-platform-runner</artifactId>
            <version>${junit.platform.version}</version>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.8.1</version>
            </plugin>
            <plugin>
                <artifactId>maven-surefire-plugin</artifactId>
                <version>2.22.2</version>
            </plugin>
        </plugins>
    </build>
</project>
```

pom.xml

10.5.3 Backend

Nun muss das Backend nur noch das richtige Docker Image starten und nach der Validierung erhält dieser den Output von Maven. Wenn «Build Success» ausgegeben wurde, war die Validierung erfolgreich.

```
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  10.400 s
[INFO] Finished at: 2020-05-17T21:40:51+02:00
[INFO] ------------------------------------------------------------------------
```

Maven

10.5.4 Fazit

Für das Integrierern weiterer Sprachen sind zwar minimale Anpassungen im Source Code notwendig, jedoch muss dieser nur leicht ergänzt werden. Der grösste Aufwand liegt in der Definition eines Docker Images bzw. in der Erstellung des Dockerfiles.

10.6 **Lizenzen**

Sämtliche verwendeten Bibliotheken weisen eine der folgenden Lizenzen auf:

- MIT
- Apache 2.0
- BSD 2 Clause
- BSD 3 Clause

Alle erlauben die Bibliotheken für den kommerziellen sowie für den privaten Gebrauch zu verwenden. In Rücksprache mit der Hochschule für Technik Rapperswil wurde entschieden, den Code dieser Arbeit unter der MIT Lizenz zu lizensieren.

10.6.1 MIT

- Jeder kann diese Software kopieren, modifizieren und verteilen.
- Die Lizenz- und Copyrightinformationen jeder einzelnen Distribution muss beifügt werden.
- Privater Gebrauch ist erlaubt.
- Kommerzieller Gebrauch ist erlaubt.
- Alle Änderungen müssen mit der gleichen Lizenz weitergegeben werden.
- Es gibt keine Garantie.
- Der Autor oder die Lizenz kann nicht für Schäden, die durch die Software verursacht werden, haftbar gemacht werden.

Copyright (c) 2020 Puzzles, https://puzzles.hsr.ch

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

► Eine detaillierte Auflistung aller 3rd Party Libraries sind im *Anhang F: Lizenzen* aufzufinden.

Teil III

Anhang

Anhang A: Glossar

Abkürzung / Begriff	Erläuterung
CSS	Cascading Style Sheets – Stylesheet Sprache
KeyStore Explorer	Tool zur Erstellung und Bearbeitung von Keystores.
Win-acme	ACMEv2 Client für Windows. https://www.win-acme.com/
Apache Tomcat	Open Source Webserver. https://tomcat.apache.org/
LetsEncrypt	Freie und offene Zertifizierungsstelle. https://letsencrypt.org/
FR	Abkürzung «Functional Requirement(s)». Funktionale Anforderungen
NFR	Abkürzung «Non-Functional Requirement(s)». Nicht-Funktionale Anforderungen
PPA	Abkürzung für "Personal Packet Archive". Paketquelle für LINUX
ACME	Abkürzung für «Automatic Certificate Management Environment». Protokoll für die automatische Prüfung der Inhaberschaft von Domains, dient der Vereinfachung der Ausstellung von Zertifikaten.
JWT	Abkürzung für «JSON Web Token».
IDP	Abkürzung für «Identity Provider».
Jest	Test Framework für Javascript. https://jestjs.io/
ORM	Abkürzung für Object-Relational Model (dt.: Objektrelationale Abbildung).
Diff / Diff-Bild	Bild, das die Unterschiede zwischen zwei anderen Bildern visuell darstellt

Anhang B: Quellenverzeichnis

Angular. (27. 03 2020). *Angular*. Von Overall Structure Guidelines: https://angular.io/guide/styleguide#overall-structural-guidelines abgerufen

Angular. (09. 04 2020). *Angular*. Von Architecture Module: https://angular.io/guide/architecture-modules abgerufen

Apache. (01. 04 2020). *GitHub*. Von moby: https://github.com/moby/moby/blob/master/LICENSE abgerufen

Durmaz, P., & Alfred, S. (14. 06 2018). AutoExercise. *Bachelorarbeit FS 18 - AutoExercise*. HSR.

ISO. (10. 10 2019). *ISO/IEC* . Von FDIS 9126-1: http://www.cse.unsw.edu.au/~cs3710/PMmaterials/Resources/9126-1%20Standard.pdf abgerufen

Jupyter. (02. 05 2020). Von Jupyter: https://jupyter.org/ abgerufen

Jupyter. (18. 05 2020). Von https://jupyter.org: https://nbviewer.jupyter.org/gist/Fil/efb1c9f3f0a9092c420dfe4cef8def96 abgerufen

MIT. (01. 04 2020). *GitHub*. Von Angular License: https://github.com/angular/angular.js/blob/master/LICENSE abgerufen

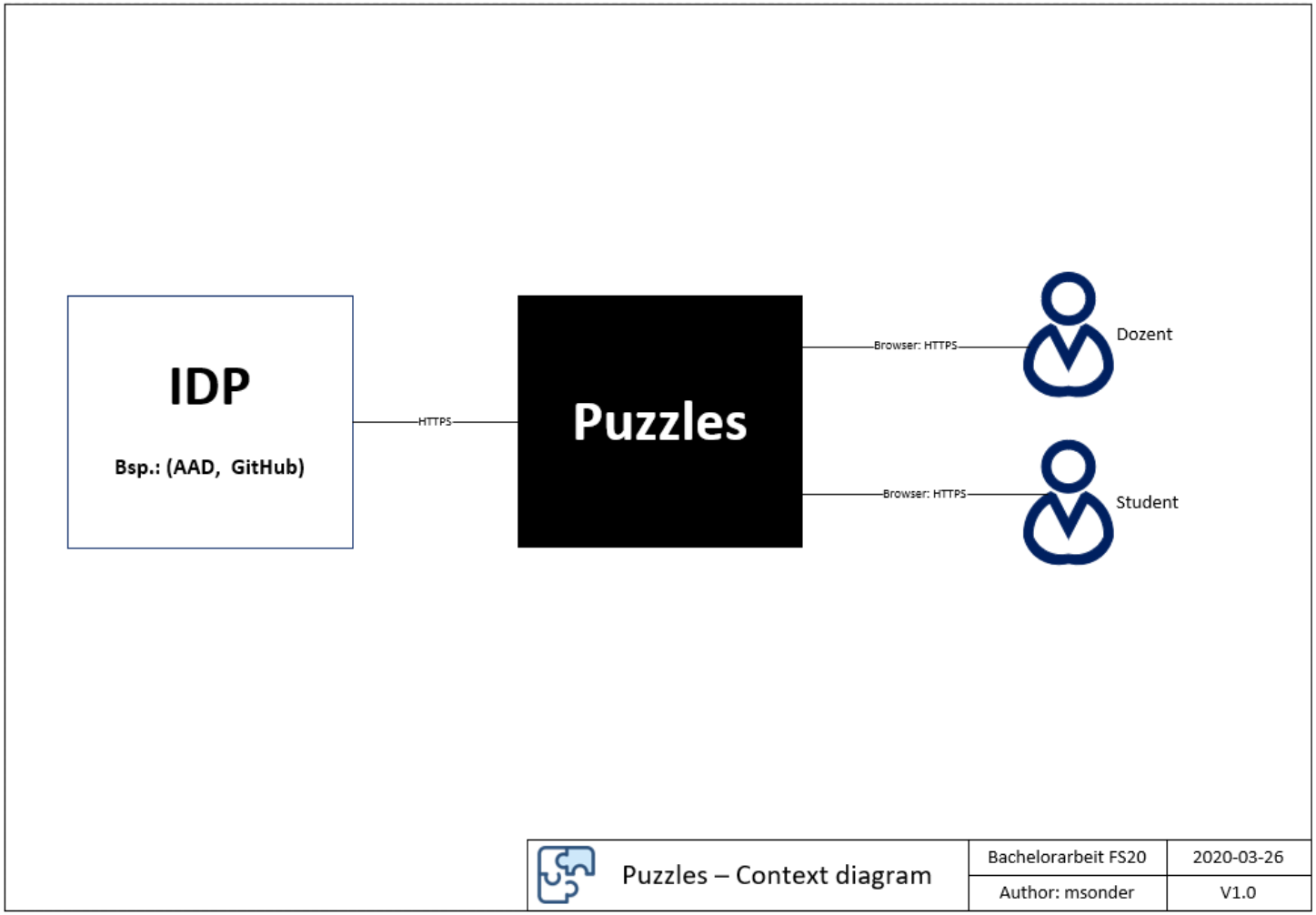
Ubuntu Wiki. (04. 06 2020). Von https://wiki.ubuntu.com/: https://wiki.ubuntu.com/SecurityTeam/KnowledgeBase/SpectreAndMeltdown abgerufen

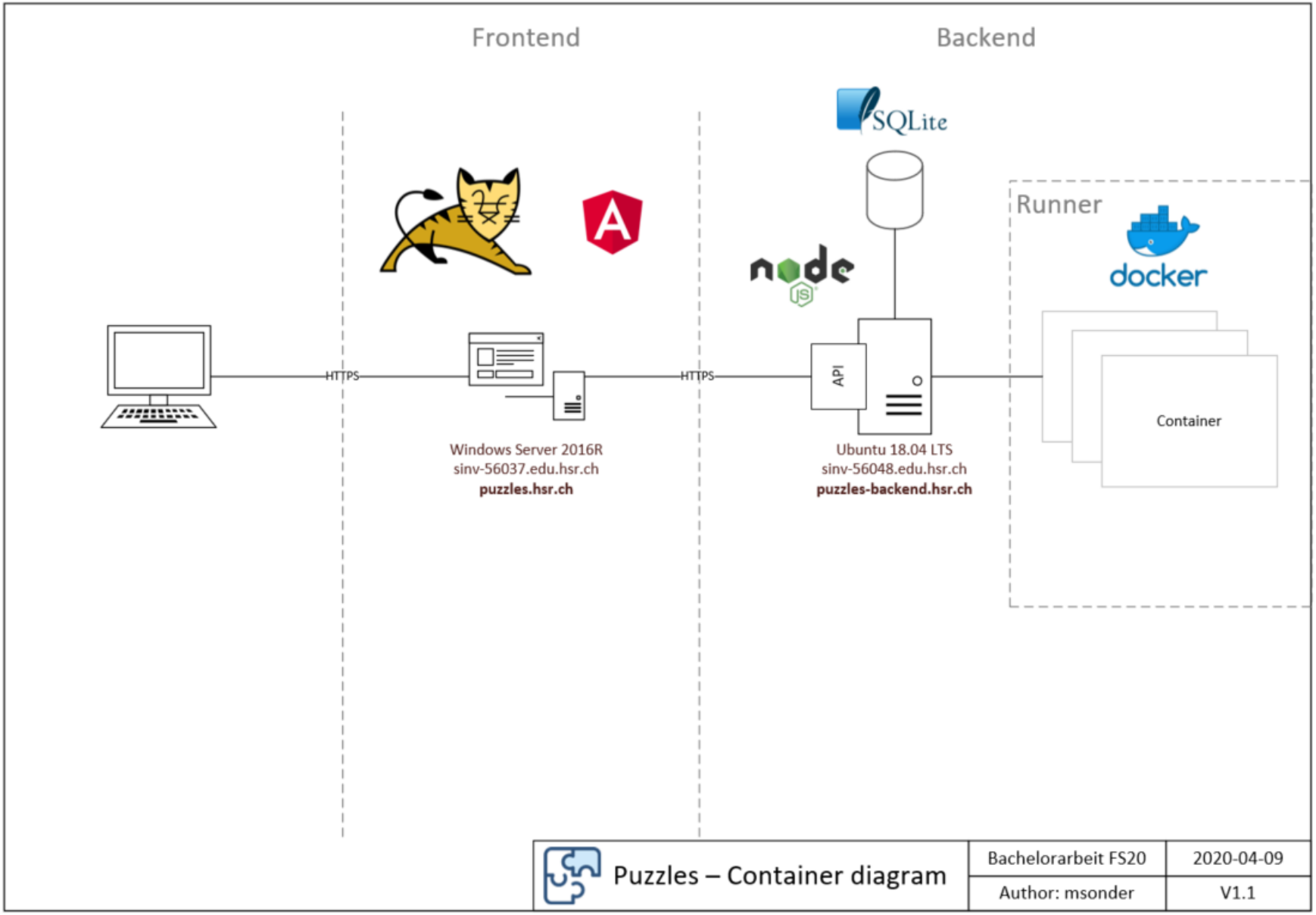
Wikipedia. (29. 03 2020). Von https://de.wikipedia.org/wiki/OAuth: https://de.wikipedia.org/wiki/OAuth#/media/Datei:Protocol-Flow.png abgerufen

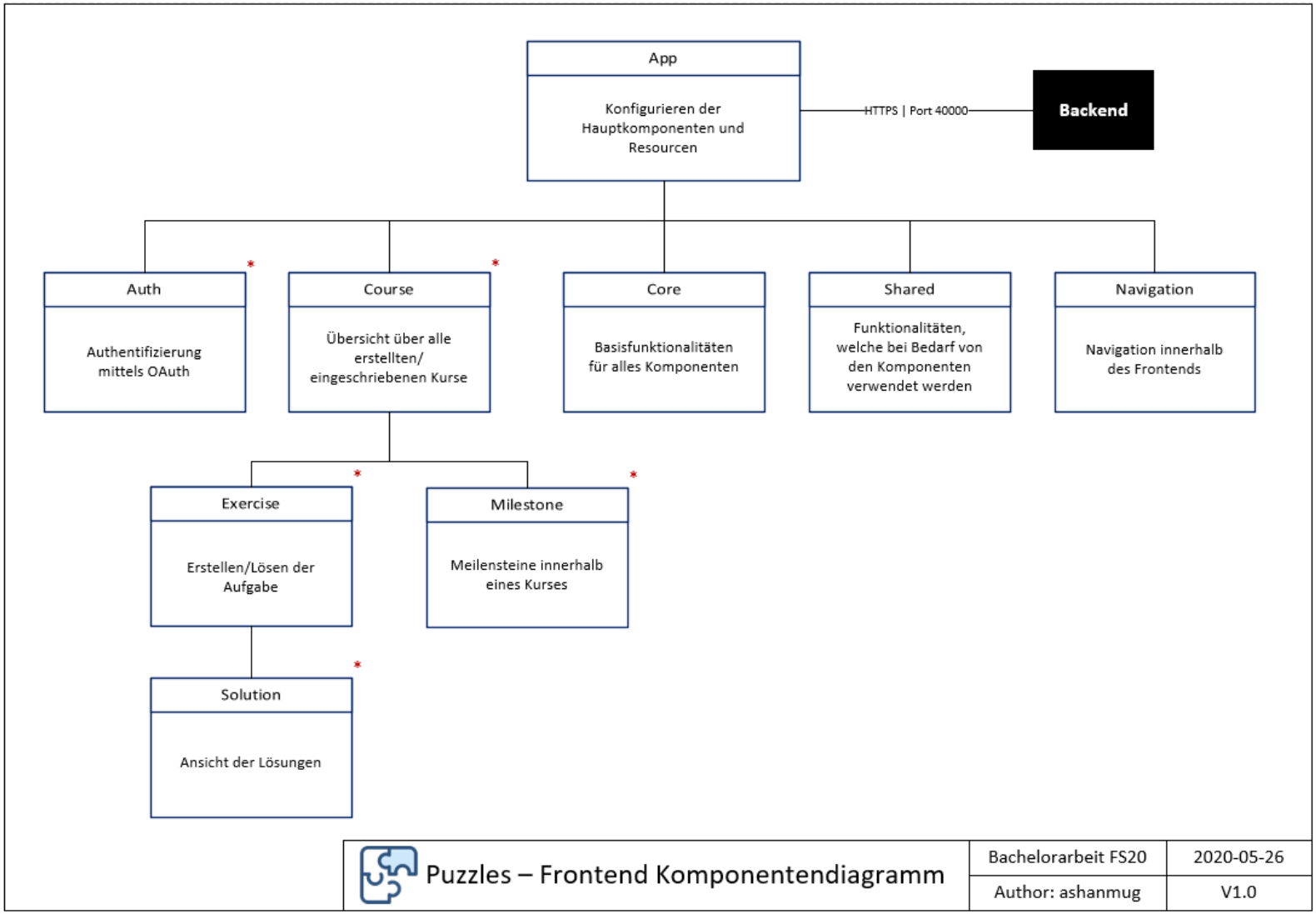
Wikipedia. (01. 04 2020). *Wikipedia*. Von BDS-Lizenz: https://de.wikipedia.org/wiki/BSD-Lizenz abgerufen

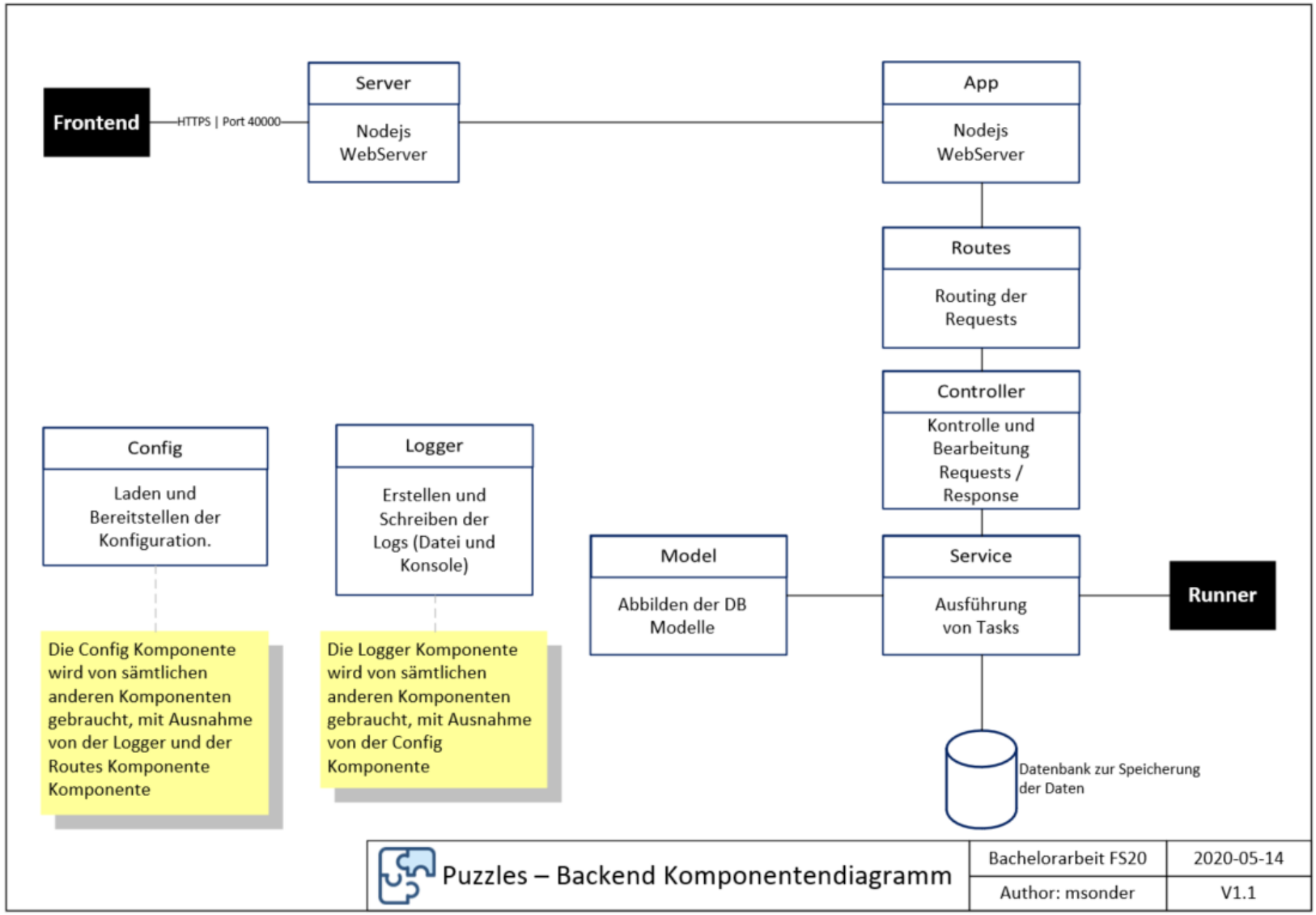
Zdun, U., Capilla, R., Tran, H., & Zimmermann, O. (26. 03 2020). *infoq*. Von https://www.infoq.com/: https://www.infoq.com/articles/sustainable-architectural-design-decisions/ abgerufen

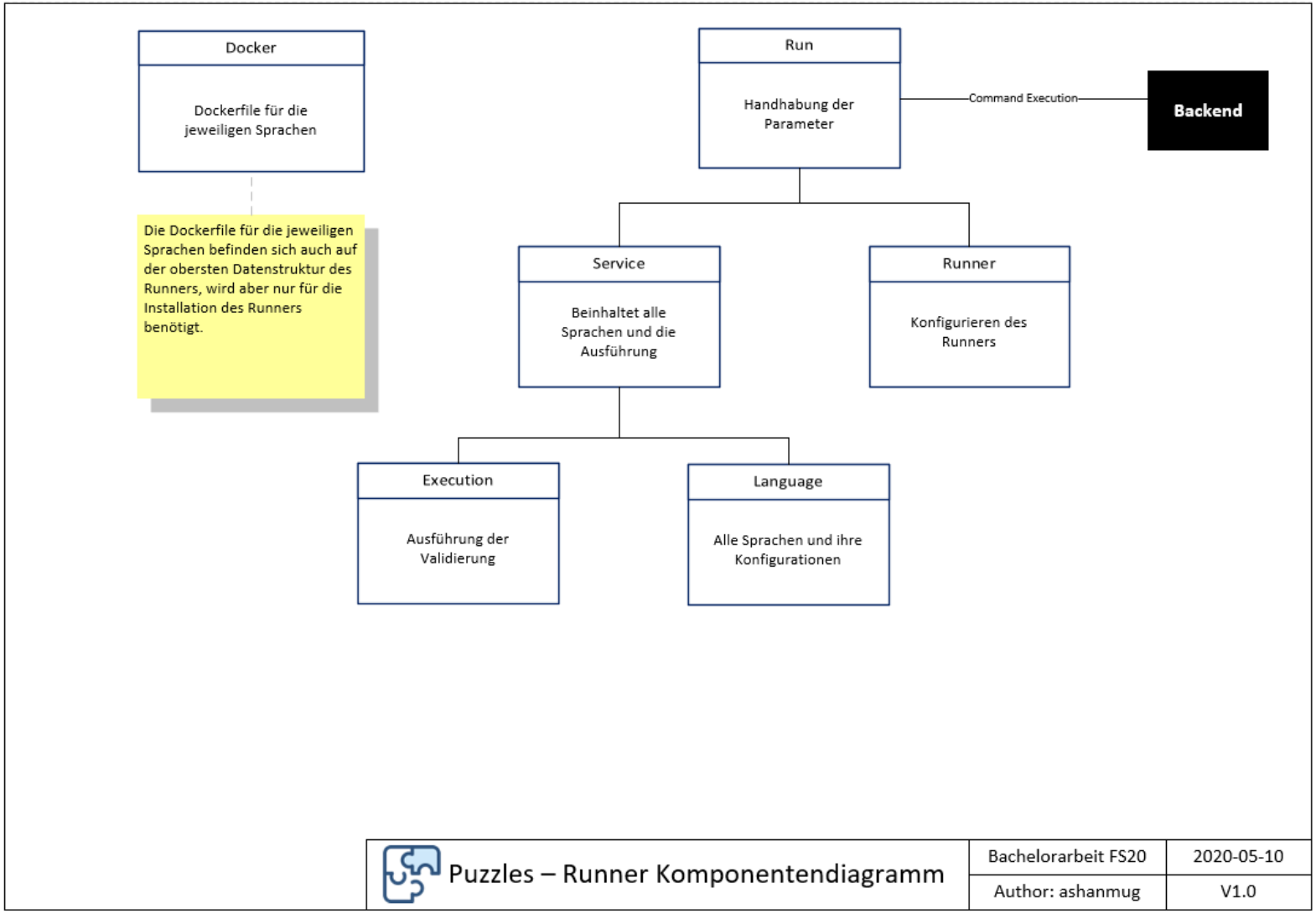
Anhang C: Diagramme



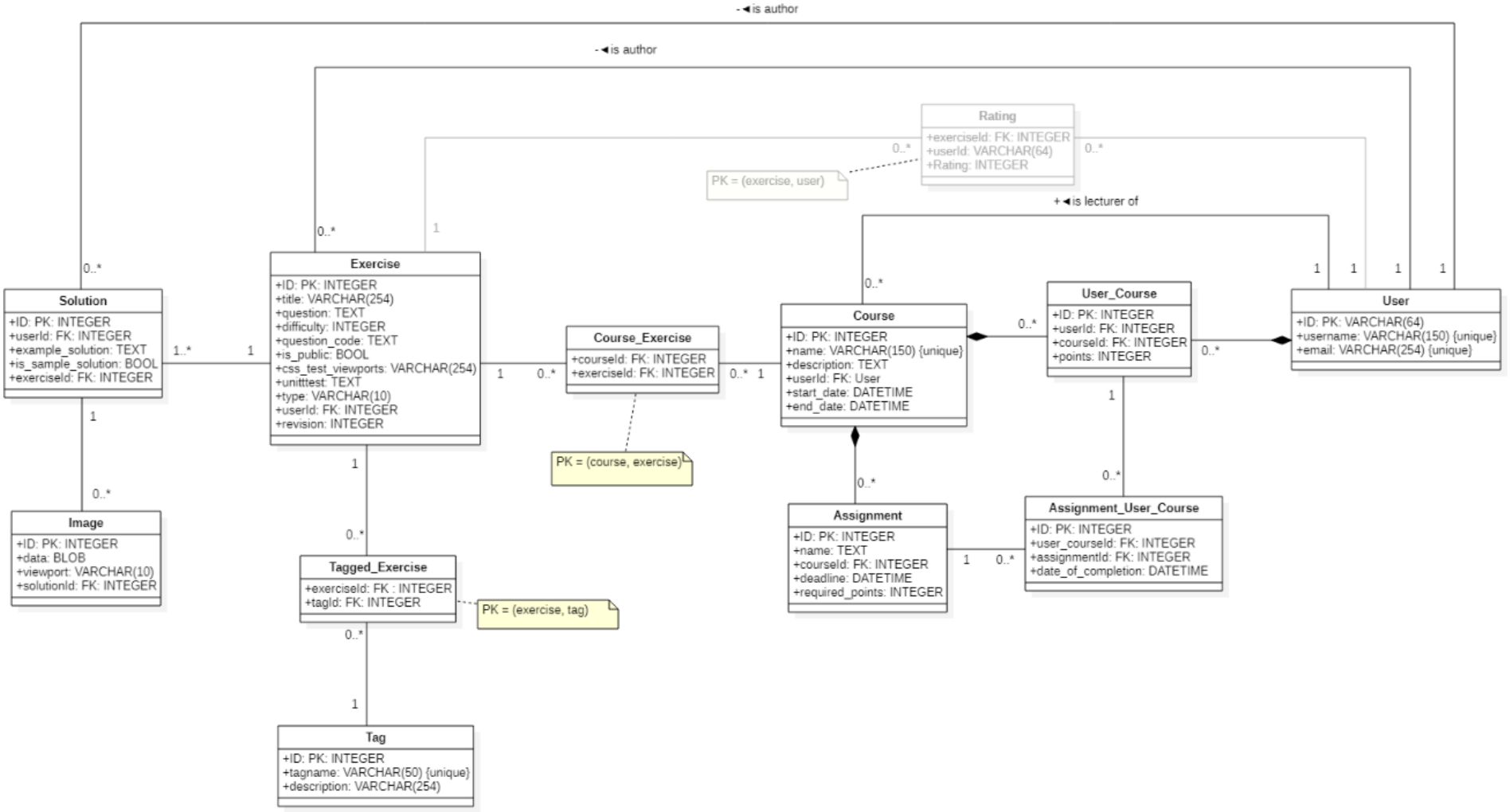

Frontend
Backend
SQLite
Runner
docker
node
HTTPS
HTTPS
API
Container
Windows Server 2016R
sinv-56037.edu.hsr.ch
puzzles.hsr.ch
Ubuntu 18.04 LTS
sinv-56048.edu.hsr.ch
puzzles-backend.hsr.ch
Puzzles – Container diagram
Bachelorarbeit FS20
2020-04-09
Author: msonder
V1.1


App
Konfigurieren der Hauptkomponenten und Resourcen
HTTPS | Port 40000
Backend
Auth
Authentifizierung mittels OAuth
Course
Übersicht über alle erstellten/ eingeschriebenen Kurse
Core
Basisfunktionalitäten für alles Komponenten
Shared
Funktionalitäten, welche bei Bedarf von den Komponenten verwendet werden
Navigation
Navigation innerhalb des Frontends
Exercise
Erstellen/Lösen der Aufgabe
Milestone
Meilensteine innerhalb eines Kurses
Solution
Ansicht der Lösungen
Puzzles – Frontend Komponentendiagramm
Bachelorarbeit FS20
2020-05-26
Author: ashanmug
V1.0


Frontend
HTTPS | Port 40000
Server
Nodejs
WebServer
App
Nodejs
WebServer
Routes
Routing der
Requests
Controller
Kontrolle und
Bearbeitung
Requests /
Response
Config
Laden und
Bereitstellen der
Konfiguration.
Logger
Erstellen und
Schreiben der
Logs (Datei und
Konsole)
Model
Abbilden der DB
Modelle
Service
Ausführung
von Tasks
Runner
Die Config Komponente wird von sämtlichen anderen Komponenten gebraucht, mit Ausnahme von der Logger und der Routes Komponente Komponente
Die Logger Komponente wird von sämtlichen anderen Komponenten gebraucht, mit Ausnahme von der Config Komponente
Datenbank zur Speicherung der Daten
Puzzles – Backend Komponentendiagramm
Bachelorarbeit FS20
2020-05-14
Author: msonder
V1.1


Docker
Dockerfile für die jeweiligen Sprachen
Die Dockerfile für die jeweiligen Sprachen befinden sich auch auf der obersten Datenstruktur des Runners, wird aber nur für die Installation des Runners benötigt.
Run
Handhabung der Parameter
Command Execution
Backend
Service
Beinhaltet alle Sprachen und die Ausführung
Runner
Konfigurieren des Runners
Execution
Ausführung der Validierung
Language
Alle Sprachen und ihre Konfigurationen
Puzzles – Runner Komponentendiagramm
Bachelorarbeit FS20
2020-05-10
Author: ashanmug
V1.0

Anhang D: Datenmodell



Anhang E: Product Backlog

Aufgabe kommentieren

ALS	Benutzer
MÖCHTE ICH	Aufgaben kommentieren können
DAMIT	ich auf dem Aufgabensteller Feedback zur Frage geben kann.

Testat-Benachrichtigungen per Mail

ALS	Dozent
MÖCHTE ICH	beim Erreichen der Deadline eines Testats darüber informiert werden, welche Studenten das Testat erfüllt haben und welche nicht.
DAMIT	ich einen Überblick über den Fortschritt der Studenten habe.

Beispiellösung hinzufügen

ALS	Autor einer Aufgabe
MÖCHTE ICH	gute Lösungsbeispiele von Studenten als weiteres Lösungsbeispiel markieren.
DAMIT	gute Lösungsvarianten von den Studenten eingesehen werden können

Maintainer zum Kurs hinzufügen

ALS	Dozent
MÖCHTE ICH	andere Personen als Maintainer zu meinem Kurs hinzufügen.
DAMIT	auch andere Personen Aufgaben und Testate erstellen und verwalten können

Globale Rangliste

ALS	Benutzer
MÖCHTE ICH	einen Vergleich mit anderen Benutzern haben. Dabei soll der Vergleich nicht auf einen Kurs beschränkt sein, sondern sämtliche Benutzer umfassen.
DAMIT	ich weiss, wo ich stehe und wo ich mich noch verbessern kann

Aufgabentestate

ALS	Dozent
MÖCHTE ICH	Testate so definieren können, dass Studenten eigenen Aufgaben erstellen müssen
DAMIT	ich eine weitere Art von Testat anbieten kann und meinen Kurs abwechslungsreicher gestalten kann. Ausserdem zwinge ich so die Studenten, sich mit der Materie tiefer auseinanderzusetzen.

Kurse taggen

ALS	Dozent (Ersteller des Kurses)
MÖCHTE ICH	meinen Kurs mit Tags versehen
DAMIT	der Kurs beim Suchen effizienter gefunden werden kann und den Benutzern schnell klar ist, was der Kursinhalt / -themen ist/sind

Filterabfragen Kursübersicht (Dozent)

ALS	Dozent
MÖCHTE ICH	Studenten in meinem Kurs anhand von bestimmten Kriterien suchen/auflisten (Beispiel: «Alle Studenten, die weniger als 120 Punkte haben» oder «alle Studenten, die das Testat noch nicht erfüllt haben»)
DAMIT	ich Studenten die Probleme haben oder abdriften kontaktieren kann.

CSS-Aufgabe: HTML editierbar

ALS	Autor einer CSS-Aufgabe
MÖCHTE ICH	definieren ob das vorgegebene HTML editierbar ist
DAMIT	ich den Lösungsweg / die Lösungsvarianten beschränken kann

Markup-Unterstützung bei Aufgabetexten

ALS	Autor einer Aufgabe
MÖCHTE ICH	den Aufgabentext mit Layout-Elementen versehen
DAMIT	ich eine strukturierte und besser verständliche Aufgabenstellung erstellen kann

Mehrere Sprachen

ALS	Benutzer
MÖCHTE ICH	die Applikation auch in anderen Sprachen als nur Deutsch bedienen
DAMIT	ich in meiner bevorzugten Sprache Puzzles bedienen kann.

Meine Testate bzw. Meine angemeldeten Testate

ALS	Benutzer
MÖCHTE ICH	eine Übersicht von allen meinen angemeldeten Testaten haben
DAMIT	einen besseren Überblick über meine Testate habe.

Breadcrumbs

ALS	Benutzer
MÖCHTE ICH	auf jeder Seite ein Breadcrumbs sehen
DAMIT	ich weiss in welcher Verzweigung ich mich innerhalb der Applikation befinde.

Anpassung Testat/Assignment

ALS	Projektteam
MÖCHTE ICH	wie im Frontend schon erledigt auch im Backend und in der Datenbank den Namen von «Testate/Assignment» zu «Meilenstein/Milestone» wechseln
DAMIT	überall der gleiche Begriff steht und Meilenstein ist der bessere Begriff.

REGEX bei Aufgaben

ALS	Dozent
MÖCHTE ICH	den Lösungsweg teilweise auf in eine bestimmte Richtung lenken. Hierzu will ich ein REGEX bei der Aufgabenerstellung definieren, welcher bei der Lösung erfüllt werden muss,
DAMIT	Studenten den gewünschten Ansatz/die gewünschte Technologie verwenden

Anhang F: Lizenzen

Frontend

Library	Lizenz
@angular/animations	MIT
@angular/common	MIT
@angular/compiler	MIT
@angular/core	MIT
@angular/forms	MIT
@angular/localize	MIT
@angular/platform-browser	MIT
@angular/platform-browser-dynamic	MIT
@angular/router	MIT
@ng-bootstrap/ng-bootstrap	MIT
bootstrap	MIT
rxjs	MIT
tslib	MIT
zone.js	MIT
ace-builds	BSD 3-Clause

Backend

Library	Lizenz
body-parser	MIT
debug	MIT
dotenv	BSD 2-Clause
express	MIT
express-handlebars	BSD 3-Clause
hbs	MIT
jsonwebtoken	MIT
morgan	MIT
node-fetch	MIT
rotating-file-stream	MIT

sequelize	MIT
sqlite3	BSD 3-Clause
winston	MIT

Runner

Library	Lizenz
canvas	MIT
commander	MIT
dotenv	BSD 2-Clause
fs-extra	MIT
jest	MIT
jest-html-reporter	MIT
jimp	MIT
merge-images	MIT
puppeteer	Apache-2.0
resemblejs	MIT

Lizenzvarianten

MIT

Eine kurze und einfache, freizügige Lizenz mit Bedingungen, die nur die Erhaltung der Urheberrechts- und Lizenzhinweise verlangen. Lizenzierte Werke, Modifikationen und größere Werke dürfen unter anderen Bedingungen und ohne Quellcode verbreitet werden. (MIT, 2020)

Berechtigungen	Einschränkungen
• Kommerzielle Benutzung • Änderung • Verteilung • Privater Gebrauch	• Haftung • Garantie

Apache License 2.0

Eine freizügige Lizenz, deren Hauptbedingungen die Beibehaltung von Urheberrechts- und Lizenzhinweisen erfordern. Die Mitwirkenden bieten eine ausdrückliche Gewährung von Patentrechten. Lizenzierte Werke, Modifikationen und größere Werke dürfen unter anderen Bedingungen und ohne Quellcode verbreitet werden. (Apache, 2020)

Berechtigungen	Einschränkungen
• Kommerzielle Benutzung • Änderung • Verteilung • Privater Gebrauch • Patentierung	• Haftung • Garantie • Verwendung als Marke

BSD Clause

Eine freizügige Lizenz, die in zwei Varianten, der BSD 2-Klausel und der BSD 3-Klausel, angeboten wird. Beide haben sehr kleine Unterschiede zur MIT-Lizenz. (Wikipedia, 2020)

Berechtigungen	Einschränkungen
• Kommerzielle Benutzung • Änderung • Verteilung • Privater Gebrauch	• Haftung • Garantie • Lizenz und Copyright muss erwähnt werden • Name der Inhaber darf nur mit einer schriftlichen Bestätigung vermerkt werden