technische universität
dortmund

Abschlussbericht

Projektgruppe 599

Sommersemester 2016 - Wintersemster 16/17

5. Juli 2017

---

in4all
Ein MooC für die informatische Grundbildung

Teilnehmer:
Can Celebi, Christian Everke, Jakob Knorr, Dmytro Marchenko, Fabian Pawlowski, Mesut Sahin, Julian Schilling, Tolgay Usul, Lara Waltermann, Roland Wyzgol

Betreuer:
Prof. Dr. Johannes Fischer
Dr. Arno Pasternak

Lehrstuhl Informatik XI
(Informatik und deren Vermittlung)
Technische Universität Dortmund

fakultät für
informatik

„Massive Open Online Courses, sogenannte MOOCs, bieten Hochschulausbildung von Hochschullehrern per Video für die breite Öffentlichkeit an. Es gibt keine formellen Zugangsvoraussetzungen, und die Studiengebühren sind denkbar gering, oft sogar kostenlos. Bezahlt wird meist nur für das Ablegen der Prüfung. Unternehmen wie Coursera oder Udacity in den USA oder iVeristy in Deutschland helfen Lernwilligen, ihren Rückstand aufzuholen. Jedes kluge Mädchen, jeder begabte Junge in den Slums von Rio oder Kapstadt kann Anschluss an den Weltmarkt finden. Alles, was man dazu braucht, ist Zugang zum Netz.“

Christoph Keese [Kee14]

Inhaltsverzeichnis

1 Ausgangssituation

Mit dem Beginn der massenhaften Verbreitung von Computern wurde in den 80er Jahren der Wandel zur Informationsgesellschaft eingeleitet. Seitdem hat sich in der Welt der Informationstechnologien viel getan und ihr Einfluss auf unser Leben ist unbestreitbar. Dieser rasante Wandel hat nicht nur bestehende Berufsbilder verändert, sondern auch eine Vielzahl neuer Perspektiven geschaffen. Obwohl diese Entwicklung schon wesentlich früher – nämlich Anfang der 70er Jahre – von Universitäten und Hochschulen aufgegriffen wurde, wurde erst 1996 die erste Forschungsgruppe zur „ Didaktik der Informatik“ gegründet [Hum06]. Aus diesem Grund waren die Schulen lange Zeit auf sich allein gestellt, wenn es um die Gestaltung des Informatikunterrichts ging. Seitdem sind weitere zwei Jahrzehnte vergangen und der Einfluss neuer Technologien und IT-Systeme hat – gefördert durch die Entwicklung mobiler Geräte und die weltweite Vernetzung – weiter zugenommen. Trotzdem hat es das Schulfach Informatik noch nicht zum Pflichtfach geschafft. Dadurch bleibt vielen Schülern eine dringend benötigte Grundlage für ihre Zukunft verwehrt.

Das Ziel des Projekts IN4ALL der Projektgruppe 599 ist, die Distanz zwischen informatischer Schulausbildung und Berufs- bzw. Studieneinstieg zu verringern. Mit Hilfe von neuen Technologien, einem weitreichenden Partner (Talentkolleg Ruhr) sowie der Unterstützung des Informatik-Lehrstuhls XI der TU Dortmund soll ein neuer Weg zur Informationsvermittlung etabliert werden.

1.1 Einführung: Massive open online Courses

von Roland Wyzgol

MooCs (kurz für: Massive Open Online Courses, auch unter dem Oberbegriff *E-Learning* verbreitet) stellen keine neue Entwicklung dar. Vielmehr ergeben sie sich aus der Kombination von klassischen Methoden der Wissensvermittlung (Problem-/Fragestellungen, Lese-/ Lernmaterial wie zum Beispiel an Schulen) und dem hohen Grad der Vernetzung über das Internet. Anstatt persönlich vor Ort zu sein, nimmt der Lehrende ein Video einer Präsentation auf und macht dieses online verfügbar. Durch Aufgaben und Tests können die Lernenden eingebunden und zum Mitdenken angeregt werden. Mit Hilfe von Chats, Foren und E-Mails können Lehrende und Lernende in Kontakt treten und eine Gemeinschaft bilden.

Durch eine Kooperation des *Massachusetts Institute of Technology* (MIT) mit der *Harvard University* schafften MooCs es in das Vorlesungsverzeichnis einer großen Universität und wurden weltweit bekannt (siehe [Mas17]). Die meist technischen Kurse wurden kostenlos im Internet angeboten und sorgten so für eine neue Form der Wissensvermittlung. Seitdem haben weitere Anbieter wie *Coursera* (USA, kostenpflichtig, [Cou17]), *Udacity* (USA, kostenpflichtig, [Uda17]), *Stanford Lagunita* (USA, Stanford University, kostenlos, [Sta17]), *iversity* (Deutschland, kostenlos, [ive17]) und viele andere das Angebot an Kursen erweitert (vgl. Kapitel 2.3.3). Die meisten dieser Plattformen behandeln vor allem naturwissenschaftliche Themen auf Universitätsniveau und sind auf Englisch. Die wenigen Angebote in deutscher Sprache richten sich ebenfalls an Studenten (zum Beispiel Angebote der FH Lübeck oder den bayrischen Hochschulen (siehe [OnC17] und [Bay17]) oder diejenigen, die berufliche Weiterbildung suchen (iversity).

Im Rahmen dieses Projekts werden MooCs als technisches Hilfsmittel zur Übermittlung von Themen der Schulinformatik produziert. Damit soll das Fundament einer einheitlichen Wissensbasis aufgebaut werden. Insbesondere soll den Schülern, die während der Schulausbildung keinen Zugang zum Fach Informatik hatten, eine Möglichkeit angeboten werden, bestehende Wissenslücken abzubauen und neues Wissen zu erlangen.

1.2 Das Talentkolleg Ruhr

von Roland Wyzgol

Das Projekt „Talentkolleg Ruhr“ [Tal17] wird von der Universität Duisburg-Essen (UDI), der Fachhochschule Dortmund sowie der Westfälischen Hochschule getragen und hat es sich zur Aufgabe gemacht, die Diskrepanz der Bildungschancen zwischen Schülern aus nicht-akademischen und akademischen Haushalten abzubauen. Mit Hilfe von Förderprogrammen in Herne, Dortmund, Duisburg und Essen sollen nicht nur Schüler, sondern auch Studenten in den ersten Semestern und Arbeitnehmer, die eine Weiterbildung anstreben, angesprochen werden. Durch individuelle Beratung, Coachings, Informationsveranstaltungen zu Ausbildungs- und Berufswegen, Workshops und Seminaren, sowie Assessments und Praktika sollen möglichst viele Möglichkeiten zur persönlichen Entwicklung aufgezeigt und vermittelt werden.

Das Projekt IN4ALL begleitet das unabhängige Projekt „Talentkolleg Ruhr“ und setzt, wie bereits angesprochen, bei der Förderung von Schülern an. In Kooperation

soll Schülern eine Lernplattform zur Verfügung gestellt werden, die Wissenslücken im Fach Informatik schließen und so bei der Suche nach Ausbildungs- und Studienplätze für bessere Chancen sorgen soll.

1.3 Projektgruppe 599

von Roland Wyzgol

Die Projektgruppe ist ein zentraler Bestandteil der Master-Studiengänge Informatik und Angewandte Informatik (sowie im Hauptstudium beider Diplomstudiengänge) und findet im Laufe von zwei Semestern statt. In diesen Semestern steht eine Gruppe von 10-12 Studierenden vor einer praktischen Problemstellung, die in kleineren Teams und hoher Autonomie gelöst werden soll. Im Sinne eines Softwareprojekts soll die Umsetzung geplant, implementiert und dokumentiert werden. Die Grundlage der Durchführung bilden die theoretischen und praktischen Kenntnisse des Bachelor-Studiengangs.

Die Projektgruppe 599 „IN4ALL – Ein MooC für die informatische Grundbildung“ begann ihre Arbeit im April 2016 mit einer Seminarphase. Im Rahmen verschiedene Vorträge wurden notwendige Themen und Konzepte vorgestellt und auf ihre Verwendbarkeit im Projekt untersucht. Dazu gehörte die Analyse verschiedene Technologien für Webentwicklung und bestehende Anbieter von MooCs (beispielsweise Udacity). Mit Hilfe der Projektmanagementmethode *Scrum* (vgl. Kapitel 3.5) wurde das Projekt zu Beginn der beiden Projektphasen, welche sich jeweils über ein Semester erstrecken, geplant und in Arbeitspakete verschiedener Größe unterteilt. Aufgeteilt in zwei Teams (Entwickler- und Inhaltsteam) begann die Arbeit nach einer Organisationsphase im Mai 2016. Die erste Arbeitsphase endete mit dem Sommersemester 2016 im Juli. Die Semesterferien wurden verwendet, um Arbeiten am Projekt durchzuführen, waren im Projektplan aber nicht mit Arbeitspaketen verplant. Die zweite Phase des Projekts begann im Oktober 2016 und endet mit dem Wintersemester 2016/2017 Mitte Februar 2017.

Dieser Abschlussbericht dient der Beschreibung der Arbeitsschritte, der verwendeten Methoden und Technologien sowie dem Stand des Projekts zur Abgabe.

2 Grundlagen

Das vorliegende Kapitel führt einige Grundlagen ein, zum Verständnis des Berichts erforderlich sind. Nach fachlichen Grundlagen aus dem Umfeld der Didaktik folgt eine allgemeine Beschreibung der verwendeten, fachlichen Komponenten.

2.1 Anforderungsanalyse

von Tolgay Usul

Die grundsätzlichen Aufgaben der Projektgruppe unterteilen sich grob in zwei Teilgebiete. In einem Teilgebiet galt es, exemplarische Lehrmaterialien für MooCs zu entwerfen, welche inhaltlich Informatikwissen auf Schulniveau (Sekundarstufe I und II) abdecken. Dazu muss zunächst erkannt werden, welche didaktischen Mittel und Methoden bei der Erstellung berücksichtigt werden müssen, um Inhalte auf digitale Art und Weise zu vermitteln. Hierzu zählt auch die Einhaltung von Bildungsstandards und die Erprobung, wie diese in MooCs umgesetzt werden. Die Aufgaben beschränken sich auf beispielhafte Inhalte. Wie das gesamte Schulwissen des Fachs Informatik umzusetzen ist, ist aus zeitlichen Gründen innerhalb der Projektgruppe nicht möglich. Das andere Teilgebiet befasst sich mit den technischen Aspekten für die Entwicklung der MooC-Software als Webapplikation.

Erste Anforderungen gehen aus der offiziellen Ausschreibung für die Projektgruppe [Fak16] hervor. Detailliertere Information bezüglich der konkreten Aufgabenstellung ergaben sich aus einer Seminarphase, die zu Beginn Projektgruppe stattfand und aus Gesprächen mit den Betreuern vom Lehrstuhl. Zusammengefasst wurde die Gesamtheit der Anforderungen in einem Pflichtenheft [Pro16a]. Dieses Kapitel zeigt die Anforderungen auf und vermittelt Grundlagen in fachlicher und technischer Hinsicht. Bei den Anforderungen wurde zwischen Hauptzielen und optionalen Zielen unterschieden.

Folgende Hauptziele galt es zu erfüllen:

1. Eine funktionierende und erweiterbare Lernsoftware, die eine grundlegende informatorische Bildung für Schüler/innen der Sekundarstufe I und höher ermöglicht.

2. Entsprechendes Lernmaterial von mindestens vierstündigen Aufwand für die implementierte Lernsoftware (MooC)

3. Mindestens ein Aufgabentyp (Multiple Choice) für das Stellen von Aufgaben

Folgende optionale Ziele galt es zu erfüllen

1. Produktion von Videos und Aufgaben nach einem fehlerfreien, geskripteten und pädagogisch angemessenen Muster

2. Weitere zusätzliche Aufgabentypen (wie Lückentexte)

3. Verschiedene Kurse mit eigenen Lerneinheiten zu erstellen

4. Speichern und Überwachung des Lernstandes von einzelnen Schülern

5. Weiterführendes Wissen der Informatik, das über die Grundbildung hinausgeht, vermitteln

Die Hauptziele, sowie die optionalen Ziele ging es im laufenden Projekt zu erfüllen. Folgende Use-Case Diagramme zeigen die möglichen Tätigkeiten, die die bestimmten Rollen in der Lernsoftware ausführen sollen.

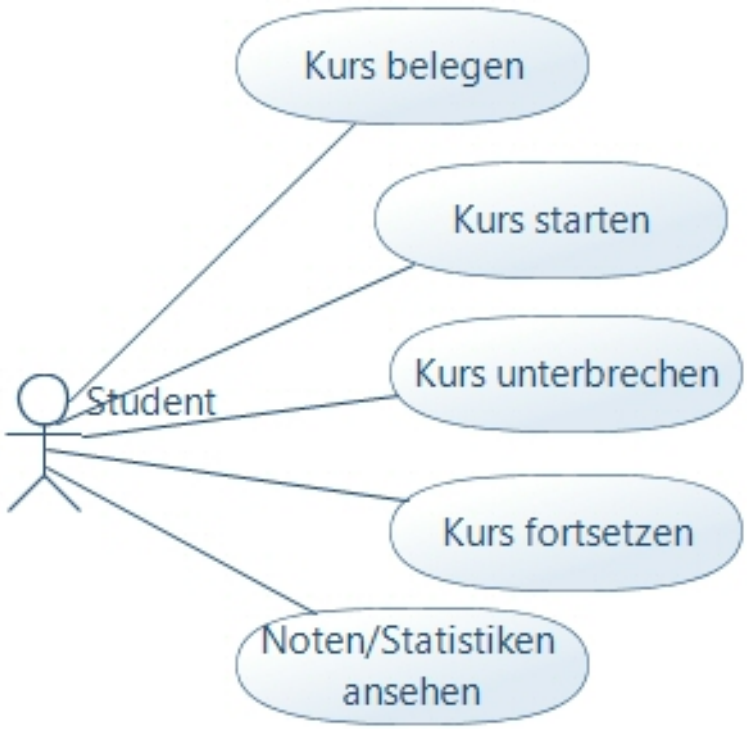


Abbildung 1: Use-Case Diagramm in der Rolle Student/Schüler

Abbildung 1 zeigt die geplanten Funktionen für den einzelnen Schüler/Studenten. Der Student ist in der Lage, vordefinierte und freigegebene Kurse zu belegen, zu starten, zu unterbrechen, fortzusetzen und Noten sowie Statisten einzusehen.

Abbildung 2 zeigt die geplanten Funktionen des Content-Managers. Der Content-Manager kann Kurse erstellen und bearbeiten. Die Kurse beinhalten Videos und Aufgaben zu einzelnen Lektionen.

2.2 Definition der Systemarchitektur

Aus der Erhebung der Anforderungen konnte schnell eine erste Systemarchitektur hergeleitet werden. Das zugehörige Datenmodell und die zu implementierenden Systemkomponenten werden im folgenden beschrieben.

2.2.1 Datenmodell

von Tolgay Usul

In diesem Unterkapitel werden die einzelnen Klassen mit ihren Funktionen dargestellt. Abbildung 3 zeigt ein Klassendiagramm, anhand dessen das Datenmodell implementiert worden ist. Im nachfolgenden werden die einzelnen Klassen mit ihren Eigenschaften näher erläutert:

Klassen In dem Projekt wurden die Klassen Group, User, Course, Unit, Lesson und Exercise implementiert. Die Entscheidung für diesen hierarchischen Aufbau der (Lehr-)Inhalte wurde aus zwei Gründen getroffen: Zum einen vereinfacht diese Struktur die Navigation auf der Webseite, sodass sich auch Studenten, die im Umgang mit

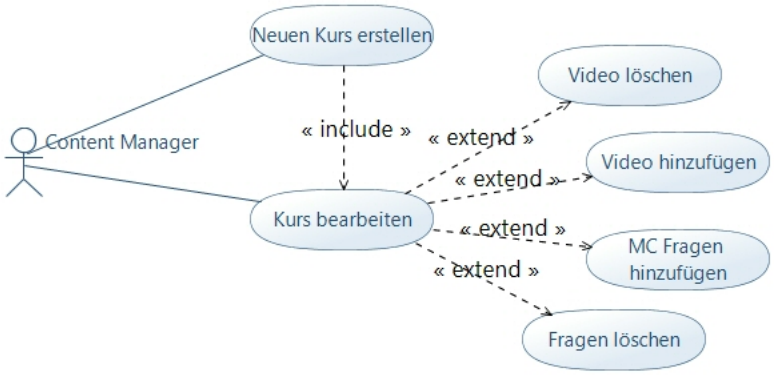


Abbildung 2: Use-Case Diagramm in der Rolle des Content-Managers

Abbildung 3: Klassendiagramm zur Visualisierung des Datenmodells des Projektes IN4ALL

derartigen Portalen wenig Erfahrung haben, schnell zurecht finden. Auf der anderen Seite zielt diese Struktur auf die Erweiterbarkeit des Systems ab. Seien es Manager, die neue Kurse und die dazu gehörenden Unterkomponenten anlegen, oder Entwickler, die das Klassensystem erweitern. Im Klassendiagramm (siehe Abbildung 3) sind die Abhängigkeiten, sowie die einzelnen Funktionen der Klassen abgebildet. Es handelt sich dabei um eine Vorabplanung des Projektes. Einige Funktionen können entweder nicht umgesetzt worden sein oder sind neu hinzugekommen.

Aufbau der Klassen Der Aufbau der unterschiedlichen Klassen ist in mehreren Stufen hierarchisch strukturiert.

Ein Kurs enthält mehrere Lerneinheiten. Diese Lerneinheiten sollen verschiedene Themenschwerpunkte eines Kurses charakterisieren. Beispielsweise sind für den Kurs die Lerneinheiten `Information und Daten`, `Einführung Datentypen` und `Verzeichnisbäume` zugeordnet, also die Themenschwerpunkte des Kurses.

Eine einzelne Lerneinheit beinhaltet mehrere Lektionen, also Teilgebiete eines Themenbereichs. In den Lektionen befinden sich die eigentlichen Lernmaterialien für das Themengebiet. Eine Lektion enthält jeweils ein zeitlich angemessenes Lernvideo sowie zugehörige Aufgaben zum Lernmaterial.

Die einzelnen Nutzer des MooCs werden in genau einer Gruppe eingeteilt. Die einzelnen Gruppen zeigen den Semester-Start der Nutzer an und dienen zu statistischen Zwecken sowie zur Übersicht. Ungebunden von der zugehörigen Gruppe kann ein Nutzer ein Kurs beitreten, sobald er vom zugewiesenen Verwalter veröffentlicht wird.

2.2.2 Systemkomponenten

von Christian Everke

Die Anwendung unterteilt sich insgesamt in drei grundlegende Komponenten: User-Frontend, CoMa-Frontend und Backend. Um ein erstes Verständnis für den Aufbau des Systems zu bekommen, werden die Komponenten und deren Aufgaben im folgenden kurz beschrieben. Technische Details und Beschreibungen zur Entwicklung der jeweiligen Komponenten sind in Kapitel 3.3 aufgeführt.

Das User-Frontend stellt die grafische Benutzeroberfläche (GUI) bereit, mit der Schülerinnen und Schüler interagieren können. Es ermöglicht, Kurse und Unterrichtseinheiten auszuwählen, die dazugehörigen Videos zu betrachten und Aufgaben

zu bearbeiten. Das User-Frontend ist eine nach Außen hin wichtige und sichtbare Komponente des Systems. Da es sich bei der Gesamtheit der Software um eine Client-Server-Applikation [Ben14] in Form einer Webanwendung handelt, ist das Frontend über einen Browser ausführbar.

Wie bei einer Client-Server-Architektur üblich, werden die Daten zentral verwaltet. Die Datenbank muss daher auf einem Webserver betrieben werden, damit die Informationen in die Benutzeroberfläche geladen werden können. Der Zugriff auf die Datenbank vom User-Frontend erfolgt über das Backend. Hierbei handelt es sich um eine serverseitige Software mit vielfältigen Aufgaben. Es stellt zunächst RESTful-Services [TESW15] bereit, über die das Frontend in der Lage ist, die benötigten Informationen abzufragen. Die Dokumentation dieser REST-Schnittstelle ist unter `http://in4all-pg.de/doc` oder als Zusammenfassung im Abschnitt A.1 dieses Dokumentes zu finden. Ferner stellt das Backend auch die Verbindung zur Datenbank her. Zwischen dem Bereitstellen und Persistieren von Daten kann das Backend fachliche Logik, auch als Business-Logik bezeichnet, implementieren. Es sei beispielsweise angenommen, dass ein Benutzer einen Kurs erst belegen kann, nachdem ein vorheriger Kurs erfolgreich abgeschlossen ist. Eine derartige, fachliche Anforderung wird beispielsweise als Business-Logik im Backend implementiert. Wird das System zu einem späteren Zeitpunkt um Konzepte wie Business-Process-Management (BPM) [Wes14] oder Business-Rule-Management [Noa14] erweitert, so erfolgt die Anbindung entsprechender Engines analog zur Datenbank ebenfalls an das Backend.

Dem User-Frontend werden die Inhalte der Datenbank, diese enthält unter anderem Kurse, Lerneinheiten, Lessons und Aufgaben, lediglich zur geeigneten Darstellung bereitgestellt. Für die inhaltliche Administration steht das Content-Management-Frontend, kurz CoMa oder CoMa-Frontend, bereit. Dieses Frontend funktioniert auf Komponentenebene betrachtet analog zum User-Frontend. Wesentlicher Unterschied ist, dass die Daten verändert werden können, wenn der Benutzer entsprechend autorisiert ist. Das CoMa verwendet die selbe RESTful-API wie das User-Frontend.

Abbildung 4 visualisiert den Aufbau und das Zusammenspiel der Systemkomponenten. Nicht aufgeführt ist ein notwendiger File-Server: Teile der Inhalte bestehen aus Videos, die im User-Frontend betrachtet werden können. Diese müssen entsprechend verfügbar gemacht werden, was einen File-Server erfordert. In der Datenbank werden zu den Lessons URLs gespeichert, die die Lokalisierung der Videos angeben. Diese Links werden dem Frontend verfügbar gemacht, so dass das Video über die URL in

das Frontend eingebunden werden kann. Das Hosting der Videos ist daher technisch und fachlich vollkommen unabhängig von den beschriebenen Systemkomponenten und muss daher nicht explizit aufgeführt werden.

2.3 Fachliche Aspekte

2.3.1 Bildungsstandards

von Lara Waltermann

Die Inhalte des in dieser Projektgruppe entwickelten MooCs basieren auf den *Grundsätzen und Standards für die Informatik in der Schule – Bildungsstandards Informatik in der Sekundarstufe I* der Gesellschaft für Informatik e.V. [BFF+08]. Der Bildungsstandard beschreibt Kompetenzen, die Schülerinnen und Schüler der Klassen 5 bis 10 im Rahmen des Informatikunterrichts erwerben sollen. Hierfür sind sowohl inhaltliche Aspekte formuliert als auch Prozesse zur Vermittlung der Inhalte.

Bei der Entwicklung des MooCs der PG599 wurden die vier Themengebiete *Informatiksysteme*, *Information und Daten*, *Sprachen und Automaten* sowie *Algorithmen* anhand der entsprechenden Inhaltsbereiche der Bildungsstandards aufgebaut.

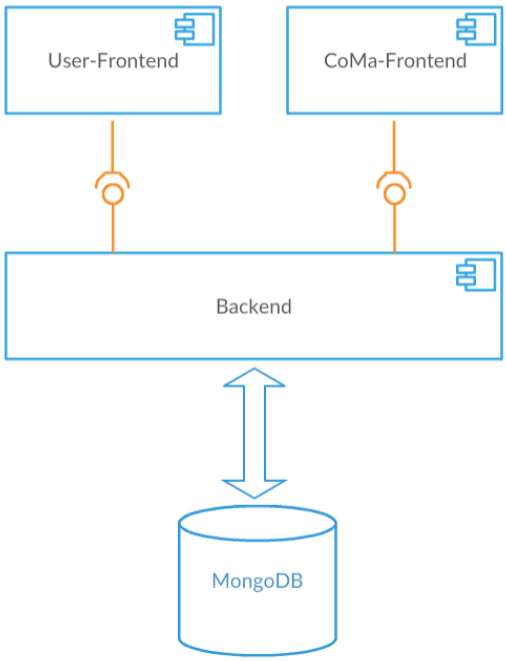


Abbildung 4: UML-Komponentendiagramm zur Visualisierung des Aufbaues des Systems. Die Schnittstellen sind über RESTful-Services realisiert.

Der Inhaltsbereich *Informatik, Mensch und Gesellschaft* wurde nicht als separates Themengebiet in den MooC aufgenommen, sondern an geeigneten Stellen im Zusammenhang mit den anderen Gebieten vermittelt. Die vorgeschlagenen Prozesse zur Vermittlung der Kompetenzen (*Modellieren und Implementieren*, *Begründen und Bewerten*, *Strukturieren und Vernetzen*, *Kommunizieren und Kooperieren* sowie *Darstellen und Interpretieren*) wurden für die Erstellung des MooCs und insbesondere im Rahmen der Übungsaufgaben berücksichtigt, jedoch konnte *Kommunizieren und Kooperieren* aufgrund des Online-Lernens keine Anwendung finden.

Die vier Inhaltsbereiche umfassen die folgenden Aspekte und Kompetenzen: [BFF+08]

1. Information und Daten

   Schülerinnen und Schüler

   - verstehen den Zusammenhang von Information und Daten sowie verschiedene Darstellungsformen für Daten,
   - verstehen Operationen auf Daten und interpretieren diese in Bezug auf die dargestellte Information, führen Operationen auf Daten sachgerecht durch.

2. Algorithmen

   Schülerinnen und Schüler

   - kennen Algorithmen zum Lösen von Aufgaben und Problemen aus verschiedenen Anwendungsgebieten und lesen und interpretieren gegebene Algorithmen,
   - entwerfen und realisieren Algorithmen mit den algorithmischen Grundbausteinen und stellen diese geeignet dar.

3. Sprachen und Automaten

   Schülerinnen und Schüler

   - nutzen formale Sprachen zur Interaktion mit Informatiksystemen und zum Problemlösen,
   - analysieren und modellieren Automaten.

4. Informatiksysteme

   Schülerinnen und Schüler

- verstehen die Grundlagen des Aufbaus von Informatiksystemen und deren Funktionsweise,
- wenden Informatiksysteme zielgerichtet an,
- erschließen sich weitere Informatiksysteme.

5. Informatik, Mensch und Gesellschaft

   Schülerinnen und Schüler

   - benennen Wechselwirkungen zwischen Informatiksystemen und ihrer gesellschaftlichen Einbettung,
   - nehmen Entscheidungsfreiheiten im Umgang mit Informatiksystemen wahr und handeln in Übereinstimmung mit gesellschaftlichen Normen,
   - reagieren angemessen auf Risiken bei der Nutzung von Informatiksystemen.

2.3.2 Didaktische Aspekte

von Lara Waltermann

Das Ziel der Projektgruppe ist die Entwicklung einer digitalen Lernplattform (MooC), die aus einer Vielzahl von *Kursen* bestehen soll. Die Lerninhalte sollen dabei anhand von Videos vermittelt werden. Die Nutzer können im Anschluss eines Lernvideos - z.T. interaktiv - die Lerninhalte durch Übungsaufgaben vertiefen und festigen. Die Zielgruppe des MooCs sind Schülerinnen und Schüler (junge Erwachsene, ggf. mit Migrationshintergrund), deren Bildungsstand in dem Schulfach Informatik sehr gering ist und deren Möglichkeit zum selbstständigen Aufarbeiten vorhandener Bildungslücken nicht oder nur gering vorhanden ist.

Innerhalb der Projektgruppe wurden daher die Inhaltsbereiche stärker konkretisiert und selektiert. Es war dabei nicht möglich, zu allen Unterbereichen vollwertige Lerneinheiten zu entwickeln, da dies den zeitlichen Rahmen gesprengt hätte. Hier bietet sich ein Ansatz für das Weiterarbeiten einer weiteren Projektgruppe.

Innerhalb des Projektes wurde ein Verlaufsplan erstellt, welcher die Inhalte in didaktisch sinnvoller Reihenfolge enthält. Anschließend wurden die Inhalte priorisiert und ausgewählt. Ziel war es, ein breites Spektrum an informatischer Grundbildung zu vermitteln.

Im weiteren Verlauf wurden die ausgewählten Inhalte und zugehörige Aufgaben geskriptet und anschließend als Video aufgenommen und bearbeitet.

Die Teilnehmer der MooCs sollten dabei nicht mit Lerninhalten überhäuft werden, sondern durch aktive Teilnahme und Anwendung von Inhalten lernen. Inhalte sollen nicht zusammenhangslos aneinander gereiht werden, stattdessen sollen in einer didaktisch sinnvollen Reihenfolge Kompetenzen vermittelt werden, welche mit Hilfe von beziehungsreichen Kontexten gefestigt werden sollen. Hierzu werden den Teilnehmern Aufgaben gestellt, bei denen Wissen aus verschiedenen Bereichen miteinander in Beziehung gesetzt werden muss. Dieses kennen sie zum Teil aus vergangenen Übungen und beinhaltet außerdem neue Inhalte, welche explizit erklärt und in Zusammenhang zu bereits Gelerntem gesetzt werden.

Im Anschluss an Lehrvideos finden Übungsaufgaben statt. Mögliche Aufgabenformen sind unter anderem Multiple-Choice-Fragen, Lückentexte, Rechen- oder Drag-and-Drop-Aufgaben.

2.3.3 Beispiele für existierende Systeme

von Lara Waltermann

Neben dem entwickelten MooC gibt es heutzutage bereits MooCs in verschiedenen Themenbereichen, zum Beispiel für den Englischunterricht. Im Bereich Informatik gibt es bisher jedoch wenig Auswahl. Die wenigen MooCs, die es hier gibt, werden auf Englisch angeboten und konzentrieren sich meist auf akademische (statt schulische) Inhalte.

Ein Beispiel für eine der größeren Plattformen ist Udacity [Uda17]. Hier können sich Nutzer in verschiedenen Themengebieten englischsprachige Videos anschauen und Kurse belegen, die sie mit einem Zertifikat abschließen können. Die Zertifikate von Udacity sind unter dem Namen *Nanodegree* markenrechtlich geschützt. Udacity wird u.a. von Partnerunternehmen unterstützt. Diese steuern z.B. Inhalte für die Kurse bei oder helfen bei der inhaltlichen Ausarbeitung von Kursen.

Ein weiteres Beispiel ist die Seite `mooc.org` [edX17b]. Sie ist eine Webseite der *edX* [edX17a], welche Online-Kurse bereitstellt und von führenden Universitäten (u.a. Massachusetts Institute of Technology, Harvard University, University of California at Berkeley) unterstützt wird. Bei Abschluss eines edX-Kurses können sogenannte *Micromaster* erworben werden, mit deren Hilfe z.B. eine Zulassung zu Master-Studiengängen an einigen Universitäten in den USA erreicht werden kann.

Die Themengebiete auf `mooc.org` umfassen neben Informatik weitere Bereiche, wie z.B. Sprachen oder Biologie.

Darüber hinaus gibt es einige weitere Beispiele für MooCs, wie z.B. MooCs der TU München [TU 17], des Deutschen Klima Konsortiums in Zusammenarbeit mit dem WWF [WWF17] [Deu17], und des Karlsruher Instituts für Technologie [Kar17].

2.4 Technische Aspekte

2.4.1 MEAN.JS - Stack

von Tolgay Usul

MEAN.JS ist eine JavaScript-Lösung, die eine schnelle robuste und wartbare Web-Anwendung mit Unterstützung von **M**ongoDB, **E**xpress.js, **A**ngularJS und **N**ode.js schaffen soll. Die ersten Buchstaben vor den einzelnen Komponenten geben den Namen wieder. Im nachfolgenden werden die Komponenten kurz beschrieben:

MongoDB ist eine Open-Source NoSQL Datenbank. Es setzt seine Prioritäten auf eine agile und hoch skalierbare Anwendungsentwicklung. Einige der komfortablen Funktionen sind das Modellieren der Dokumentdatenmodelle mit dynamischen Schemata, eine umfassende und flexible Indexunterstützung, eine erweiterbare Sicherheit (Beispielsweise mit Kerberos), das Auto-Sharding[1] für horizontale Skalierbarkeit, usw [Tre14].

Express.js ist eine serverseitiges Framework, welches Node.js erweitert. Das Framework stellt eine JavaScript Datei mit den Namen `app.js` in das Projektverzeichnis. Die Datei startet den Webserver und enthält mehrere Ereignishandler für die Kommunikation zwischen Client und Server. Möglich sind auch Middlewares,[2] die die Kommunikationsphasen unterteilen. Ein Beispiel hierfür wäre eine Authentifikationsschicht (Login) [Mar14].

AngularJS ist ein OpenSource JavaScript- Framework für client-seitige Webapplikationen. Es ist geeignet für Single-Page Anwendungen, d.h. die Logik einer Webseite besteht aus einer HTML-Datei. Das Programmieren wird durch viele Funktionen

[1] Ermöglicht gute Skalierbarkeit für einzelne und mehrere Server
[2] Verbirgt die untere Anwendungsschicht und zeigt nur einige Funktionen

vereinfacht. Weiterhin ermöglicht es eine Integration der Webapplikation zu Mobilen-Plattformen. Zudem standardisiert es ein modernes Design für einzelne Webkomponenten (Buttons, Tabs,...) [BT15].

Node.js ermöglicht die Ausführung von JavaScript auf serverseitigen Anwendungen. Durch die Implementierung in C/C++ wandelt es schnell JavaScript Code in kompilierbaren Maschinencode um und führt zu einer Verbesserung der Laufzeit. Es besitzt zudem viele integrierte Module, wie beispielsweise das Http-Modul, um einen Webserver zu hosten. Für die weitere Installation von Paketen muss im Projekt Verzeichnis der integrierte Paket-Manager mit dem Befehl `npm install [modulname]` ausgeführt werden [Day14].

Das Zusammenspiel dieser Komponenten ermöglicht es, eine solide Webanwendung zu schaffen. Nähere Erklärungen zu den einzelnen Komponenten sind im weiteren Verlauf des Berichts zu finden.

2.4.2 Präprozessoren

von Jakob Knorr

Webentwicklung besteht clientseitig stets aus einer Kombination von HTML (für die Struktur), CSS (für die Styledefinitionen) und JavaScript (für Skripte), da nur diese drei Sprachen von allen Browsern unterstützt wird. Doch nicht jeder Browser untersützt auch jedes Feature einer Sprache, sodass es sehr schwierig sein kann, bestehenden Code auch für sämtliche Browser zu optimieren. Damit ein derartiger Optimierungsprozess für die Browser nicht zuviel Zeit in Anspruch nimmt, wird die Unterstützung für sehr alte Browser mit niedrigem Marktanteil häufig in der Planungsphase eines Projekts ausgeschlossen. Dennoch entwickeln sich besagte Sprachen nur sehr langsam weiter und neue Features können erst sehr spät genutzt werden. Um neuere Funktionen (beispielsweise die neuen ECMAScript2015-Features) schon frühzeitig nutzen zu können, werden Polyfills[3] und Präprozessoren eingesetzt. Zudem ist beispielsweise CSS mit seiner deklarativen Syntax zwar sehr einfach gehalten, bietet jedoch keine Mittel um die Mengen von Styledefinitionen in großen Projekten wartbar und skalierbar zu halten, welche durch Präprozessoren eingeführt werden.

[3]Abgeleiteter Name von *Polyfilla*, einer englischen Spachtelmasse zum Füllen von Lücken in Mauern. Ist im JavaScript-Umfeld die Bezeichnung für Module, welche Funktionalitäten auch in Browsern bereitstellen, wenn diese Funktionen von Haus aus nicht unterstützt werden

SCSS SCSS [SCS17] (Sassy CSS) ist ein Präprozessor für CSS, welcher den beschränkten Funktionsumfang von CSS durch nützliche Features wie Variablen, Verschachtelung und Schleifen erweitert. Valides CSS ist auch immer valides SCSS, sodass ein Umstieg von CSS auf SCSS sehr leicht fällt. Das folgende Listing gibt zwei verschiedenen Kindern des *.wrapper*-Elements die gleiche Hintergrundfarbe:

```
$background-color: red;

.wrapper {
    .child1 {
        background-color: $background-color;
        font-size: 10px;
    }

    .child2 {
        background-color: $background-color;
        font-size: 12px;
    }
}
```

In CSS würde ein derartiges Listing wie folgt aussehen:

```
.wrapper .child1 {
    background-color: red;
    font-size: 10px;
}

.wrapper .child2 {
    background-color: red;
    font-size: 12px;
}
```

Die Vorteile hier sind, dass die Vaterklasse (*.wrapper*) nicht vor jede Kindklasse geschrieben werden muss und man durch die Einrückung die Vererbungshierarchie direkt erkennen kann. Zudem bedarf eine Änderung der Hintergrundfarbe im SCSS-Code lediglich einer Änderung, während im CSS-Code die Farbe an jeder Stelle, an der diese Verwendung findet, ausgetauscht werden muss. Bei großen Projekten kann dies sehr aufwendig sein. SCSS wird zur Entwicklungszeit in CSS übersetzt und dabei auch minifiziert (Zeilenumbrüche, Leerzeichen und Kommentare werden entfernt), was die Größe der Datei reduziert.

TypeScript TypeScript [TS17] ist ein JavaScript Präprozessor, der von Microsoft entwickelt wird. Durch die Unterstüzung von vielen gängigen Entwicklungsumgebung und die Verwendung in großen JavaScript-Projekten (beispielsweise Angular), findet TypeScript derzeit eine hohe Verbreitung bei JavaScript-Entwicklern. TypeScript ist eine Obermenge von JavaScript, sodass gültiges JavaScript auch valides TypeScript ist. Es werden lediglich einige Features aus den ECMAScript-Standards von 2015 und 2017 und Typsicherheit zur Übersetzungszeit hinzugefügt. Letzteres ist eine sinnvolle Ergänzung, um Entwicklern bessere Vervollständigungsvorschläge in Editoren anbieten zu können und um die Robustheit der Webapplikation zu erhöhen. TypeScript wird zur Entwicklungszeit in JavaScript übersetzt und minifiziert.

Jade/Pug Pug [PUG17] (ehemals Jade) ist ein Präprozessor für HTML. Dabei setzt Pug auf eine schlankere Syntax (wie z.B. das Weglassen von Klammern), bei der die korrekte Einrückung des Codes von großer Wichtigkeit ist. Auf einem NodeJS-Server eingesetzt, wird Pug zu einer *Rendering Engine*, bei dem einem Template Variablen übergeben werden. Dieses Template wird dann zur Laufzeit (etwa bei einer HTTP-Anfrage an den Server) in HTML übersetzt (wobei Variablen und Schleifen das generierte HTML beeinflussen). Eine Liste mit drei Listenelementen in HTML sieht wie folgt aus:

```
1 <ul>
2     <li>Element1</li>
3     <li>Element2</li>
4     <li>Element3</li>
5 </ul>
```

Die Einrückung dient hier lediglich als Hilfe für den Entwickler und hat keinen Einfluss auf das Verhalten im Browser. In Pug könnte eine solche Liste wie folgt ausgedrückt werden:

```
1 ul
2     each item in ["Element1","Element2","Element3"]
3         li= item
```

Die Nutzung einer *each*-Schleife über ein *Array* von Elementen erzeugt die drei Listeneinträge. Die Einrückung des Codes ist hierbei von semantischer Relevanz, da es keine schließenden Elemente wie in HTML gibt.

2.4.3 Erstellung von Videosequenzen für MooCs

von Fabian Pawlowski

Nachdem im vorherigen Kapitel die technischen Grundlagen für ein MooC erläutert wurden, ist noch die Frage zu klären, worauf bei der Produktion von Videos für ein MooC geachtet werden muss. Die Beantwortung dieser Frage verändert sich mit den Zielen, die mit dem MooC verfolgt werden. Im Folgenden soll eine Übersicht über mögliche Kriterien für die Videoproduktion gegeben werden.

Länge und Umfang Die Länge (und damit der Umfang) der Lehrvideos kann sehr variabel gewählt werden. Vorlesungen und Unterrichtstunden sind typischerweise zwischen 45 und 120 Minuten lang. Damit sind sie aber zu lang, wenn beispielsweise gewollt ist, dass die Lernenden sich diese auch „von unterwegs“ anschauen. Den starken Kontrast dazu stellen also Videos dar, die mit 3 bis 6 Minuten auskommen und so kurz und bündig das Thema erläutern. Es gibt natürlich auch solche Themen, die nicht in 5 Minuten erklärt werden können. Dann kann man - je nach Anwendungsfall - ein großes Video in mehrere kleinere Lektionen aufteilen. Des Weiteren spielt auch die Konzentrationsspanne der Zuschauer eine wichtige Rolle. Während eine Schulstunde oder eine Vorlesung die Konzentration durch Anwesenheit aufrecht hält, spielt die Konzentrationspanne bei einem MooC eine größere Rolle. Aus diesem Grund sollten Videos für ein MooC eine bestimmte Länger nicht überschreiten. Für das Projekt hat sich die Gruppe auf eine Länge von höchstens 7 Minuten geeinigt.

Stil Es gibt viele Möglichkeiten ein Video und damit eine Lektion aufzunehmen. Die am meisten vorkommende Variante ist die, dass der Lehrende auf einem Tablet die Inhalte der Lektion aufschreibt und diese mit Erklärungen vertieft. Es gibt als eine Bildschirm- und eine Tonaufnahme. Um die Aufnahme persönlicher zu gestalten, kann darüber hinaus auch die „schreibende Hand“ des Lehrenden mit einer Kamera erfasst und auf die Bildschirmaufnahme gelegt werden. Dann kann der Lernende beim Schreiben zuschauen. Daneben gibt es aber auch die Varianten, die an eine Vorlesung beziehungsweise Unterrichtsstunde erinnern. Der Lehrende hält einen Vortrag und interagiert mit den Zuschauern. In diesem Fall steht er also vor der Kamera und erklärt die Themen mit Augenkontakt. Eine dritte Variante ist die Aufnahme eines Dialogs zwischen zwei Personen, die sich einem Problem angenommen haben. Dieser wird dann aus einem bestimmten Blickwinkel aufgezeichnet, sodass der Zuschauer selbst das Gefühl bekommt, an dem Gespräch teilzuhaben.

Integration der Aufgaben Neben den Videos nehmen die Aufgaben in einem MooC eine zentrale Rolle ein. Sie stellen das Mittel dar, um mit dem Lernenden in Verbindung zu treten. Der Zeitpunkt und die ansprechende Gestaltung der Aufgaben entscheiden darüber, ob die Lektion ankommt und ein Lerneffekt eintritt. Im Wesentlichen gibt es zwei Möglichkeiten Aufgaben zu stellen: Im Verlauf des Videos und am Ende des Videos. Die Entscheidung darüber sollte in Abhängigkeit von der Länge des Videos getroffen werden. Kurze Videos eignen sich dazu, sie mit einer Aufgabe zu beenden und so gerade Gelerntes anzuwenden. Längere Videos können durch Aufgaben in Abschnitte geteilt werden und so dem Lernenden signalisieren, dass nun ein thematischer Abschnitt beendet wurde. Natürlich ist auch eine Kombination möglich.

Aus diesen Kriterien ergeben sich Rahmenbedingungen, die für eine technische Umsetzung erfüllt sein müssen. Seitens der Hardware wird ein Tablet, eine Kamera, ein Computer (für die spätere Nachbearbeitung der Aufnahmen) und Aufnahmeequipement (Beleuchtungsmittel, Mikrofon, ggf. Stative) benötigt. Auf Seiten der Software fallen Bild- und Videobearbeitungssoftware (zum Beispiel die „Adobe Creative Suite“), ein Notizprogramm mit TouchPen-Unterstützung (zum Schreiben auf dem Tablet) und ein Programm um eine Bildschirmaufnahme durchzuführen (zum Beispiel „Fast Stone Capture“) an.

3 Entwicklung

Dieser Abschnitt beschreibt die Entwicklung des konkreten MooC-Systems. Insbesondere die Umsetzung konkreter didaktischer Aspekte sowie die Verwendung der Elemente des vorgestellten MEAN.JS Stacks werden näher betrachtet.

3.1 Prototypen

von Christian Everke

Im Vorfeld der eigentlichen Implementierung wurden die im Abschnitt 2.2.2 beschriebenen Komponenten prototypisch entwickelt. Die Gründe für die Erstellung dieser Prototypen waren sehr vielfältig. Nach dem Kick-Off der Projektgruppe einigte sich das Entwicklerteam zeitnah, die Umsetzung auf Basis des MEAN-Stacks (vgl. Kapitel 2.4.1) umzusetzen. Es wurde festgestellt, dass nicht alle Mitglieder des Teams über die notwendigen Vorkenntnisse verfügten, so dass eine Einarbeitungszeit erforderlich war. Während der Prototyp-Entwicklung konnten erfahrenere Mitglieder ihre Kenntnisse bereits zügig umsetzen und an unerfahrene Mitglieder weitergeben. Für die „Neulinge“ im JavaScript-Umfeld war eine Möglichkeit gegeben, Kenntnisse und Wissen den Prototypen zu entnehmen und Experimente an diesen durchzuführen. Ebenso bildeten sich innerhalb des Entwicklerteams weitere Gruppierungen, jeweils für das User-Frontend, CoMa und Backend. Die einzelnen Komponenten wurden zunächst unabhängig voneinander entwickelt und später über Schnittstellen zusammengeführt, die gemeinsam in Teamarbeit spezifiziert wurden. Nach dem wirksamen Zusammenführen galt der „Proof of Concept“ als erfolgreich, so dass die Entwicklung des eigentlichen Systems begonnen wurde. Die durch die Prototypentwicklung erworbenen Erfahrungen waren für die Entwicklung der eigentlichen Anwendung sehr hilfreich. Insbesondere konnten einige Codefragmente wiederverwendet werden.

Neben der softwaretechnischen Erstellung von Prototypen wurden auch erste MooC-Sequenzen, also Lehrvideos entsprechend erster didaktischer Anforderungen, erstellt. Auch hier wurden erste Erfahrungen mit der eigens angeschafften Technik gemacht, auf die im weiteren Verlauf der Entwicklung zurückgegriffen wurde.

3.2 Didaktisches Vorgehen

In diesem Kapitel wird erläutert, wie die Projektgruppe bei der Erstellung der Skripte vorgegangen ist und welche Aufgabentypen für die Fragen nach den Videos implementiert worden sind. Abschließend werden die erstellten Skripte nach ihren Bereichen aufgelistet.

3.2.1 Aufgabentypen

von Roland Wyzgol

Das Ziel der Aufgaben zu den Videos ist, den Lerneffekt durch Wiederholung oder Weiterführung zu verbessern. Aus diesem Grund hat die Projektgruppe im Vorfeld einige Aufgabentypen geprüft, die im Laufe des Projekts implementiert werden sollten. Die folgenden Typen sind zum Abschluss des Projekts enthalten. Das Implementieren weiterer Aufgaben ist möglich. Dies erfolgt durch Anpassung und Erweiterung des Programmcodes, setzt also Kenntnisse über diesen voraus.

- **Single/Multiple Choice** Dieser gängige Typ wird verwendet, um wahlweise eine oder mehrere richtige Antworten zur Verfügung zu stellen. Der Anwender wählt diese dann durch einen Klick aus.

- **Drop Down** Mit diesem Typen kann die Antwort aus mehreren Möglichkeiten über ein Drop-Down Menü gewählt werden.

- **Lückentext** Dieser Typ ermöglicht es im Text ein Textfeld für eine Eingabe hinzuzufügen. Damit können Begrifflichkeiten abgefragt beziehungsweise eingeprägt werden. Alternativ kann das Eingabefeld des Lückentexts durch ein Drop-Down Menü ersetzt werden. Diese zweite Variante des Lückentextes verringert zwar die Freiheiten bei den Antworten, vereinfacht es dem Aufgabensteller aber konkrete Begriffe abzufragen und gegenüberzustellen.

Zusätzlich wurde eine Funktion realisiert, die die Darstellung von Bildern im Antwortbereich ermöglicht. Damit können Aufgabentypen (zum Beispiel Drop-Down Menü) um ein Bild erweitert werden. Auf diese Weise können Fragen lebendiger gestaltet werden.

3.2.2 Erstellung von Skriptvorlagen

von Lara Waltermann

Beim Erstellen von Skriptvorlagen für die spätere Umwandlung in Videos waren mehrere Aspekte zu beachten. Um diese Aspekte möglichst früh konkretisieren zu können, wurde zunächst ein Probeskript erstellt. Das Probeskript zum Thema *Umwandlung von Binärzahlen* war der erste Versuch ein Skript zu verfassen. Es war von vornherein nur für einen kurzen Probelauf geplant. Das Skript verfügte über eine eigene Übungseinheit, die für das Video aber nur schematisch umgesetzt wurde. Für die folgenden Skripte wurde sich an den Bildungsstandards orientiert. Das Thema zu dem Testskript wurde unabhängig von den Bildungsstandards gewählt, da es für die Probeaufnahme einen überschaubaren und gut umsetzbaren Rahmen darstellte.

Während des Schreibprozesses, der wider Erwarten aufgrund vieler zu berücksichtigenden Aspekte langwierig wurde, musste man sich zunächst mit der Form dieser Skripte auseinandersetzen. Ein einfaches „drauflos schreiben“ war nicht möglich, da man merkte, dass es mehrere Aspekte gibt, die man beachten und miteinander verbinden muss. Obwohl das Probeskript ein erster, wichtiger Versuch war und viele Probleme ersichtlich machte, konnte man zusätzlich einige Erfahrungen und Erkenntnisse beim Schreiben der darauffolgenden Skripte gewinnen. Es haben sich dabei einige Aspekte ergeben, die im Folgenden aufgegriffen und erläutert werden:

Die Dauer eines Videos bzw. die Länge eines Skripts So ist ein gewisser zeitlicher Rahmen einzuhalten. Die Länge des Probeskripts wurde geschätzt, indem die Aufnahme auf Papier geprobt und die Zeit dabei gemessen wurde. Das Schreiben auf dem Tablet, zusätzlich zu einigen Interaktionen mit dem Text (bei der Probeaufnahme wurde zunächst ein *Microsoft Surface Pro 3* verwendet), stellte sich dabei als komplizierter heraus als das Schreiben auf Papier.

Eine Referenz stellte die„Minute-Pro-Seite-Regel“ aus der Filmbranche dar, die sich auf das amerikanische Drehbuchformat bezieht [CH02]. Obwohl die Videos des MooCs nicht dem verbreiteten Format der Filmbranche entsprechen, wurde vermutet, dass der gesprochene Anteil dennoch vergleichbar sei, so dass zunächst davon ausgegangen wurde, dass die „Minute-Pro-Seite-Regel“ weitestgehend Anwendung finden könnte. Durch die Tests wurde diese Vermutung bestätigt. So entsprach eine DIN A4 Seite Skript mindestens einer Minute Video.

Wahl der Inhalte Unser Ziel war es, zunächst alle Themengebiete abzudecken. Dabei wurde sich an den Bildungsstandards orientiert. Zunächst wurden zu einigen Themenfeldern Grundlagen-Skripte entwickelt. Dabei fiel auf, dass an sehr vielen Stellen vertiefende Skripte möglich sind, da zu vielen Fachbegriffen weitere Informationen hilfreich sind. Es fällt an einigen Stellen schwer, an einem bestimmten Punkt den Informationsumfang einzuschätzen und abzuwägen, welche Informationen noch mit aufgenommen oder aufgrund des Umfangs weggelassen werden sollen. Deshalb wurden konkretere Themen und Begriffe gesucht, die weiter vertieft werden konnten. Dies führte zu einem weiteren Aspekt, den es zu beachten gilt, welcher im folgenden Abschnitt erläutert wird.

Im Rahmen der verwendeten Software-Beispiele wurde auf Hinweis der PG-Betreuer häufig OpenSource-Software genannt, um die Teilnehmer nicht zu sehr mit gängigen kommerziellen Produkten vertraut zu machen, und um sie in ihrer Wahl für Produkte nicht zu sehr von subjektiver Nutzung durch PG-Teilnehmer zu beeinflussen.

Die Verknüpfung der Skripte Eine Verknüpfung der Skripte ist sinnvoll und im oben genannten Fall sogar sehr gut umsetzbar, aber es hat sich herausgestellt, dass der umgekehrte Fall besonders zu beachten ist. In einem Skript dürfen nicht zu viele Fachwörter und Verweise fallen. Denn es ist kaum möglich, neben dem eigentlichen Thema des Skriptes zusätzlich weitere Definitionen und Erläuterung zu geben. Deshalb müssen beim Schreiben entweder zukünftige Skripte im Blick behalten oder versucht werden, zu viele Fachwörter und Verweise zu vermeiden. Der gesamte thematische Kontext darf dabei wiederum nicht außer Acht gelassen werden, denn die gezielte Nennung bestimmter Fachwörter kann möglicherweise ein elementarer Teil des Wissens sein, das vermittelt werden soll.

Daher ist bei einigen verwendeten Fachbegriffen die Absprache im Team besonders wichtig, um Begriffe nicht doppelt zu erklären, aber vor allem um darauf zu achten, dass sie in mindestens einem Skript tatsächlich erläutert werden. Auf Verweise wie „wurde bereits erklärt“ oder „wird in einer späteren Lektion erklärt“ wurde daher möglichst verzichtet, um die finale Reihenfolge der Videos und Lektionen zunächst noch offen und flexibel halten zu können. Hierzu hat das Team Content dann später einen roten Faden für alle Videos und Lektionen erarbeitet, um eine sinnvolle Reihe der Videos und Lektionen aufzustellen und in den Videos andere Themen referenzieren zu können.

Die Länge der Sätze und die Wahl der Sprache Die Skripte müssen eine Sprache enthalten, die einem Lehrvideo angemessen ist. Sätze, die verschriftlicht werden, sind oft zu verschachtelt und zu lang. Beim Schreiben der Skripte wurde also stets versucht, eine klare, einfache und direkte Sprache zu wählen, weshalb die Vertonung im Blick behalten werden musste. Beim Schreibprozess ist außerdem aufgefallen, dass eine einheitliche Ansprache für die Schüler gewählt werden musste. Es musste also geklärt werden, ob die „Sie"-oder „Du"-Form gewählt werden sollte. Es wurde sich schnell auf das Duzen geeinigt und die „Wir"-Form zur bevorzugten Ansprache gewählt, da sie am meisten zusagte.

Des Weiteren stellte sich u.a. die Frage, ob man wichtige Begriffe z.B. im Video an den Rand schreiben sollte, um im Laufe des Videos eine Liste mit diesen Begriffen aufzubauen und zu erarbeiten. Dies hätte den Vorteil, dass die Teilnehmer die Begriffe länger vor Augen hätten und sie sich besser einprägen können. Deshalb wurden im Rahmen von Begriffsdefinitionen in Videos zusätzlich häufig verwendete Tastenkombinationen notiert (z.B. `Strg+P` und `Strg+C`). So können die späteren Teilnehmer neben informatischer Grundbildung auch informationstechnische Grundlagen erwerben. Hierbei stellte sich die Frage, ob diese am Rand notiert werden sollten und/oder ein Glossar für Tastenkombinationen angelegt werden sollte.

Zu den vier Inhaltsbereichen der Bildungsstandards wurden folgende Skripte erstellt:

Informatiksysteme

1. Grundlagen Informatiksysteme
2. EVA-Prinzip 1
3. Grundlagen Datenspeicherung
4. Unterschied zwischen Betriebssystem und Anwendersoftware
5. Grafische Benutzeroberfläche
6. Dateiformate unterscheiden
7. EVA-Prinzip 2

Information und Daten

- Einführung Datentypen

- Verzeichnisbäume

Sprachen und Automaten

- Einführung Automaten
- Formale Sprachen
- Formale Grammatik
- Überführung von umgangssprachlichen Handlungsvorschriften in formale Darstellung

Algorithmen

- Grundlagen Algorithmen
- Grundlegende Bausteine von Algorithmen
- Sortieralgorithmen

3.3 Technische Umsetzung

3.3.1 Frontend-Templating: Layouts mit AngularJS

von Jakob Knorr

Das Frontend ist die Schnittstelle zum Benutzer der Anwendung und hat das Ziel, diesem die Navigation durch die Kurse, Einheiten und Übungen möglichst einfach zu bereiten. Um unter anderem die Notwendigkeit einer Installation der Anwendung zu vermeiden und um die Applikation unter jedem System zugänglich zu machen, setzen wir auf eine Weboberfläche, welche durch einen modernen Browser aufgerufen werden kann. Ein Browser ist heutzutage auf nahezu jedem PC vorinstalliert und da der Applikationscode von einem Webserver übermittelt wird und dann zur Ausführung interpretiert (und nicht vor der Übertragung in Maschinensprache compiliert) wird, kann unsere Anwendung von jedem gängigen System, das einen modernen Browser besitzt, verwendet werden.

Weboberflächen bestehen aus HTML (Hyptertext Markup Language) für den strukturellen Aufbau der Oberflächen und aus CSS (Cascading Stylesheets), mit welchen

die strukturellen Elemente in der Erscheinung verändert werden können. Da unsere Anwendung möglichst performant arbeiten soll, haben wir uns dazu entschlossen, nicht bei jeder Klickaktion auf einen Link oder ein anderes Seitenelement die gesamte Seite erneut vom Webserver anzufragen, sondern stattdessen die benötigten Inhalte dynamisch nachzuladen. Derartiges Nachladen von Inhalten wird durch AJAX (Asynchronous Javascript and XML) [Sel17] ermöglicht. Dies hat zur Folge, dass sämtliche Applikationslogik auf dem Client (im Browser) ausgeführt wird und nicht etwa der Webserver vor Auslieferung des HTML-Codes diesen dynamisch generiert, wie es bei einem Großteil der verbreiteten CMS (Content Managemenet Systems) der Fall ist (z.B. Wordpress, Joomla oder Drupal). Wir ersparen uns damit das ständige Nachladen von sich wiederholenden Inhalten und durch den Einsatz von Javascript manipulieren wir den Seitenbaum nur gezielt an den Stellen, wo sich Inhalte verändern müssen, sodass nicht die gesamte Seite vom Browser erneut gezeichnet werden muss. Das Resultat ist eine einzige Seite, die sich in Aussehen und Inhalt im Laufe des Betriebs verändert. Eine derartige Seite reagiert schneller auf Benutzereingaben (durch Wegfall von Lade- und Renderzeiten). Im Idealfall vermittelt die Weboberfläche dem Nutzer das Gefühl einer nativen Applikation, weshalb derartige Seiten auch SPA (Single Page Applications) genannt werden.

ModuleLoader und Packet Manager Seit jeher werden Javascript-Dateien durch das ⟨*script*⟩-Tag im HTML-Dokument geladen. Sobald der Browser bei der Analyse des Markups ein solches Tag findet und dieses mit einem *src*-Attribut versehen ist, wird versucht, die dort referenzierte Datei zu laden und im Anschluss auszuführen. Bei größeren Applikationen im Browser ist es jedoch sehr umständlich, jede Javascript-Datei im HTML-Dokument zu referenzieren. Zudem muss beachtet werden, dass das geladene Skript auch direkt ausgeführt wird, wodurch sich das Verhalten einer Seite ändern kann, je nachdem, in welcher Reihenfolge Skript-Verweise im HTML-Markup auftauchen. Um den Umgang mit Javascript-Dateien zu vereinfachen, wurde im ECMAScript-Standard von 2015 auch eine Syntax zum Importieren von Abhängigkeiten eingeführt:

```
1  import MeineKlasse from '/pfad/zu/meiner/klasse.js'
```

Dies hat zur Folge, dass die dort referenzierte Javascript-Datei geladen wird, sobald der Javascript-Interpreter diese Zeile analysiert. Innerhalb der geladenen Datei muss die importierte Klasse exportiert werden:

```
1  // '/pfad/zu/meiner/klasse.js'
2
```

```
3   export default MeineKlasse {
4   // Klassenimplemenetierung
5   }
```

Dies funktioniert nicht nur mit Klassen, sondern auch mit Variablen und Objekten. Da dieser Standard noch recht jung ist, fehlt jedoch die Unterstützung in den meisten gängigen Browsern. Um die Modulsyntax auch in solchen Browsern zu erhalten, laden wir zu Beginn ein Polyfill von dem *ModuleLoader* unserer Wahl *SystemJS* [Sys17].

Um externe Bibliotheken einfach zu verwalten, benutzen wir darüber hinaus den Paket Manager *JSPM (JavaScript Package Manager)* [JSP17]. Mit diesem können Abhängigkeiten einfach über die Kommandozeile installiert werden:

```
1   jspm install lodash
```

Obiger Befehl installiert die *lodash*-Bibliothek. Zur Verwendung im Code muss dieses Modul nur noch importiert werden:

```
1   import lodash from 'lodash'
```

AngularJS Im folgenden werden die von uns verwendeten Teile von AngularJS kurz erläutert, eine vollständige Dokumentation zu AngularJS würde hier jedoch nicht in den Rahmen passen. Für einen tieferen Einblick in AngularJS steht online eine ausführliche Dokumentation bereit. [Ang17a]

Da es sehr aufwendig sein kann, mit den zur Verfügung stehenden Browserschnittstellen die Daten nachzuladen und den Seitenbaum zu manipulieren, benutzen wir das Framework AngularJS. Dieses von Google entwickelte Framework vereinfacht das spezifische Austauschen von Daten im Seitenbaum, indem es ermöglicht, Variablen innerhalb von HTML-Dokumenten zu verwenden. Dabei prüft ein sich wiederholender Zyklus, ob Änderungen in einer Variable aufgetreten sind. Ist dies der Fall, so wird der Inhalt des HTML-Codes an den Stellen ausgetauscht, an denen ursprünglich die Variable stand. Es entsteht dabei eine doppelte Datenbindung zwischen den Variablen im Javascript-Code und denen im HTML-Code, da Änderungen an der Javascript-Variable im HTML gespiegelt werden und *vice versa* (etwa bei der Eingabe in ein Textfeld, wenn eine Variable an das Textfeld gebunden wurde). Um größere Applikationen wartbar zu halten, benutzt AngularJS das MVC (Model-View-Controller) Konzept.

Components So besteht eine AngularJS Applikation in der Regel aus einer Menge von Komponenten, von denen jede eine View (HTML-Datei) und eine Controller-Klasse beinhaltet. Die Controller-Klasse hält die Variablen für die View zur (doppelten) Bindung bereit. Komponenten lassen sich nutzen, um die Oberfläche in kleine, eigenständige und im Idealfall wiederverwendbare Stücke zu unterteilen. Eine Komponente erhält ein dem Namen entsprechendes HTML-Tag, mit dem sie im Rahmen der AngularJS-Applikation einfach über den HTML-Code aufgerufen werden kann.

Dependency Injection Mit *Dependency Injection* bietet AngularJS eine Implementierung des zur Auflösung von Abhängigkeiten eines Objekts verwendete Entwurfsmusters der *Inversion of Control* [Fow05]. So ist es vorgesehen, dass benötigte Abhängigkeiten (beispielsweise eine Service für Anfragen an die REST-Schnittstelle in einem Controller) nicht etwa im Konstruktor selbst initialisiert, sondern von einem zentralen *Injector* zur Verfügung gestellt werden. Wie dies in unserem Projekt aussehen kann, zeigt der folgende Auszug aus dem Controller für die Ansicht der Lesson-Seite:

```
1  export class lessonPageCtrl{
2
3  public static $inject : Array<string> = ['
       IExtendedDomainFactory', '$state'];
4
5    constructor(private DomainFactory : ExtendedDomainFactory,
       private $state: ng.ui.IStateService){
6    //DO SOMETHING
7    }
8  }
```

Dieser Codeauszug zeigt die Definition der Klasse *lessonPageCtrl*. In Zeile drei wird in dieser Klasse das statische Array *$inject* mit den Strings *IExtendedDomainFactory* und *$state* befüllt. Angular liest bei der Erzeugung dieses Controllers das Array aus und weiß nun, dass der Controller zwei Services erwartet, die dem Injector unter diesen beiden Namen zuvor bekannt gemacht wurden. Beim Aufruf des Konstruktors (Zeile fünf) werden diese beiden Services automatisch übergeben. Das $-Zeichen vor dem Namen des Services ist eine Konvention von angular und sagt aus, dass dies kein eigens erstellter Service ist, sondern dieser zum Umfang von AngularJS gehört. Die *IExtendedDomainFactory* hingegen ist ein von uns erstellter Service, der zur Entfaltungszeit der Applikation (*bootstrapping*) dem Injector bekannt gemacht wurde.

Services Ein AngularJS-Service ist eine Javascript-Klasse, die selbst andere Services als Abhängigkeiten haben kann. Sie sollten idealerweise dazu verwendet werden, die Applikationslogik (beispielsweise das Anfragen von Daten von einer REST-Schnittstelle oder das Speichern im *local storage* des Browsers) von den Controllern auszulagern und die Kommunikation zwischen Komponenten zu ermöglichen.

```
1  class MyService{
2
3    constructor(){
4    //DO SOME STUFF
5    }
6  }
7  angular.service('MyService', MyService);
```

Das obige Listing zeigt, wie eine Klasse mithilfe der globalen Variable *angular* dem *Injector* bekannt gemacht wird. In unserem Projekt genügt es, eine Datei mit der Endung *.service.ts* anzulegen. Das Gulp-Tooling 3.3.6 sorgt zur Entwicklungszeit dann dafür, dass dem *Injector* von AngularJS dieser Service mit dem Dateinamen bekannt gemacht wird. Befindet sich also im Frontendbereich der Projektstruktur eine Datei mit dem Namen *myService.service.ts*, so kann dieser Service unter dem Namen *IMyService* in einen Controller oder einem Service *injected*[4] werden.

Routes Routen in AngularJS sind Bereiche der Applikation und werden durch die URL im Browser erreicht. Dazu benutzen wir in unserem Projekt die Erweiterung *angular-ui-router* [Ang17d]. Routen werden durch ein JSON-Objekt definiert und können nach dem Bekanntmachen mit AngularJS über die Addressleiste im Browser aufgerufen werden. Beispielhaft betrachten wir die Routendefinition der Lesson-Seite:

```
1  export default {
2    name: 'lesson',
3    config: {
4      url: '/lesson/:lessonId',
5      template: '<ifa-lesson></ifa-lesson>'
6    }
7  }
```

Es wird ein Objekt definiert (und exportiert), welches die Attribute *name* und *config* aufweist. Ersteres ist dabei vom Typ *string*, während der *config*-Schlüssel ein weiteres

[4]Der *Injector* erzeugt eine Instanz von diesem Service. Diese Instanz wird allen Controllern und Services übergeben, welche diesen Service benötigen

Objekt mit den Attributen *url* und *template* beinhaltet, die ebenfalls vom Typ *string* sind. Der Name dient der eindeutigen Identifikation der Route und kann unter anderem zur Navigation benutzt werden. Die Url besteht aus dem festen String (*lesson*) und einer Variablen (*:lessonId*). Der variable Teil der Url kann vom Controller der Lesson-Seite ausgelesen werden, um die passende Lesson von der REST-Schnittstelle zu laden und anschließend anzuzeigen. Der Konstruktor des Lesson-Controllers sieht daher wie folgt aus:

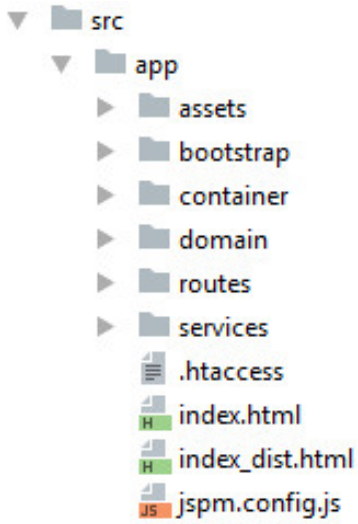
```
1     constructor(private DomainFactory : ExtendedDomainFactory,
          private $state: ng.ui.IStateService){
2
3       let lessonId = $state.params.lessonId;
4
5       DomainFactory.getLesson(lessonId)
6       .then((lesson : ifa.ILesson) => {
7         this.lessonVM = lesson.createViewModel();
8       })
9     }
```

In der dritten Zeile ist zu sehen, wie mithilfe des *$state*-Service die ID der anzufragenden Lesson ausgelesen wird. Im Anschluss wird diese ID benutzt, um von der *DomainFactory* die Lesson anzufragen (Zeile 5).

Im Template-Attribut ist ein HTML-String hinterlegt, welcher zwischen Kopf- und Fußbereich der Seite angezeigt wird. Hier wird durch das *ifa-lesson*-Tag die lesson Komponente aufgerufen.

Dateien mit der Endung *.route.ts* werden wie auch Services automatisch vom Gulp-Tooling beim Entfalten der AngularJS Applikation bekannt gemacht.

Die Projektstruktur Das Projekt hat in der obersten Ordnerstruktur mehrere Ordner und Dateien, die unter anderem die installierten Abhängigkeiten beinhalten und das Gulp-Tooling betreffen. Die Dateien, welche den Code für das Frontend beinhalten liegen unter *src/app*. Die dort enhaltene Ordner- und Dateistruktur kann folgendem Bild entnommen werden:



Assets In dem *assets*-Ordner liegen Bilder und Stylesheets, welche in die Seite eingebunden werden. Stylesheets haben in diesem Projekt die Dateiendung *scss*, weil sie zur Entwicklungszeit von dem Gulp-Tooling in gültige Stylesheets mit der Dateiendung *css* übersetzt werden.

Bootstrap Der *bootstrap*-Ordner enthält die Typescript-Dateien, die für den Start der AngularJS-Applikation im Browser notwendig sind. Hier werden Module von Drittanbietern, Routen und Komponenten bekannt gemacht und kleinere Konfigurationen vorgenommen (wie zum Beispiel, dass bei einem Routenwechsel immer zum Anfang der Seite gescrollt wird). Einstiegspunkt für die Applikation ist hier die *main.ts*.

Container Innerhalb des *container*-Ordners befinden sich globale Komponenten. Auf oberster Ebene befinden sich die Dateien, welche die Hauptkomponente, die *appComponent*, definieren. Diese Komponente wird aufgerufen, sobald das Laden der AngularJS-Applikation fertig ist. In dem Unterordner *components* befinden sich weitere Komponenten, welche routenübergreifend innerhalb der Applikation verwendet werden. Dies sind unter anderem Komponenten für den *header*, den *footer* oder auch für die *navigationBar*, welche innerhalb des Headers aufgerufen wird.

Domain Der *domain*-Ordner ist beim *Auschecken* des Projekts aus dem GIT-Repository erst einmal nicht vorhanden. Erst duch das Gulp-Tooling werden die

Dateien generiert, welche dort enthalten sind. Als Grundlage für die Generierung dient die Datei */templates/domain/domain.js*, in welcher deklarativ unser Datenmodel beschrieben ist. Hier werden alle Klassen mitsamt Attributen und zugehörigen Datentypen gelistet. Mithilfe von Gulp werden daraus alle Klassen und die zugeörigen Interfaces generiert. Zusätzlich wird eine *DomainFactory* erzeugt. Dies ist eine Javascript-Klasse, welche der AngularJS-Umgebung als Service bekannt gemacht und so den Komponenten bei Bedarf *injected* wird. Wichtig ist es, zu beachten, dass an diesen Dateien keine Änderungen vorgenommen werden können, da diese bei der nächsten Generierung wieder überschrieben werden. Änderungen am Datenmodel sollten ausschließlich in der Datei */templates/domain/domain.js* vorgenommen werden.

Services Ebenfalls auf oberster Ebene der Applikation befindet sich der *services*-Ordner. Dies ist der geeignete Ort, um AngularJS-Services zu implementieren, welche von allen Komponenten der Seite benutzt werden dürfen. Derzeit enthalten ist ein Service für die Authentifizierung und einer, der die *DomainFactory* um einige Funktionen erweitert. Ersterer kann jeder Komponente der Seite *injected* werden, um Anfragen wie *isLoggedIn()* zu stellen oder auch um mit *getToken()* Nutzerinformationen auszulesen (wie zum Beispiel den Namen der Benutzers). Auch das Einloggen wird hier mit den Methoden *login* und *logout* gehandhabt.

Routes Der letze Ordner in der Liste hat den Namen *routes*. Hier werden die Routen der AngularJS-Applikation bereit gestellt. Die für die Applikation relevanten Routen sind *home*, *login*, *imprint*, *user*, *courseCatalog*, *unitCatalog*, *lessonCatalog* und *lesson*. Jeder Ordner beinhaltet vier Dateien: eine für die Routendefinition (**.route.ts*) und drei weitere für die Definition der Komponente (**.component.ts, *.style.scss, *.view.html*). Diese Komponente wird angezeigt, sobald die zugehörige Route aktiv ist. Ein Klick auf eines der Menüpunkte in der Hauptnavigation (zum Beispiel *Katalog*) sorgt dafür, dass die Applikation die aktive Route wechselt (zum Beispiel zur *catalog*-Route). Dann wird zwischen Header und Footer der Seite die zugehörige Komponente angezeigt, welche in dem Ordner der Route definiert ist. Einige Routen haben weitere Unterordner. So hat die *lesson*-Route noch einen Ordner für *components* und *services*, welche nur innerhalb dieser Route benutzt werden. Die Komponenten dienen für die weitere Unterteilung der Seite (hier in *video*, *videoInfos*, *exercise* und *navigationSidebar*), um den Umfang einer Komponente nicht zu groß werden zu lassen. Die *lesson*-Route besitzt auch einen zusätzlichen Service,

den *exerciseProcessor*, welcher für das Aufbereiten und Lösen der Aufgaben, die zur Lesson gehören zuständig ist.

Exercise Processor Da die Fragen bzw. die Antworten einer Exercise durch eine vom Fragetyp abhängigen Syntax in der Datenbank gespeichert werden, muss ein Parser dieses Format verarbeiten, damit die Daten möglichst einfach in der Oberfläche ausgegeben werden können. Dafür gibt es den Service *exerciseProcessor*. Dieser bedient sich dem von der *Gang of Four* bekanntem Verhaltensmuster *Chain of Responsibility* [GHJV95, 223]. Auf dem Service können zwei Methoden aufgerufen werden, eine zum Parsen des Antwortstrings in ein Objekt und eine zum Lösen der Aufgabe (bzw. zum Überprüfen, ob die vom Nutzer gegebene Antwort korrekt ist). Bei beiden Methoden wird die übergebene *exercise* in einer Liste von *ExerciseWorkers* weitergereicht, bis einer von Ihnen erkennt, dass diese Art von Aufgabe zu ihm gehört. Für jeden Aufgabetyp (Multipe Choice, Lückentext, etc.) muss daher ein *ExerciseWorker* existieren, der die spezielle Syntax verarbeitet und in einem lesbaren Format zurückgibt bzw. überprüft, ob eine gegeben Antwort die Aufgabe korrekt löst. Möchte man den bisherigen Funktionsumfang des Frontends um einen neuen Aufgabetyp erweitern, so muss lediglich eine neue Klasse von der abstrakten Klasse *exerciseWorker* erben und die nötigen Methoden implementieren.
Ein Beispiel: Die zurzeit angezeigte Lesson hat eine Exercise, welche für die Frage *Mit welcher Tastenkombination kannst du auf einer Internetseite, ein PDF- oder ein Textdoument drucken?*. Die beiden angegebenen Antwortmöglichkeiten, aus denen der Nutzer auswählen kann, sind *Strg+P* und *Strg+C*. In der Datenbank liegen die Daten in etwa wie folgt vor:

```
1     answer: "YN:1_Strg+P,0_Strg+C"
2     created: "2017-02-10T12:07:39.054Z"
3     deleted: false
4     id: "589dad0bd730fd11ea25304f"
5     lastModified: "2017-02-10T12:07:39.054Z"
6     lesson: "5899b945cd79d90e5e2c7bf0"
7     level:1
8     orderId:1
9     question:"Mit welcher Tastenkombination kannst du auf einer
         Internetseite, ein PDF- oder ein Textdokument drucken?"
```

Während die Frage unter dem Attribut *question* direkt in einer Oberfläche ausgegeben werden kann, muss die *answer* erst noch für die Oberfläche aufbereitet werden. Dafür übergeben wir obige Daten an die Methode *processExercise* des *Exercise-*

Processors. Dieser geht die Liste der *ExerciseWorkers* durch, bis der *YesNoExerciseWorker* anhand des mit *YN* beginnenden Answer-String erkennt, dass er für das verarbeiten dieser Aufgabe zuständig ist. Im Anschluss liefert er an den Aufrufenden das verarbeitete Objekt zurück. Dieses sieht dann wie folgt aus:

```
{
  answers: ["Strg+P", "Strg+C"],
  givenAnswer: null,
  question: "Mit welcher Tastenkombination kannst du auf einer
     Internetseite, ein PDF- oder ein Textdokument drucken?",
  solution: "Strg+P",
  type:"radio",
  orderId:1
}
```

Die Syntax aus dem *answer*-Attribut ist nun verarbeitet, die Antwortmöglichkeiten stehen in einem Array, die Lösung wird explizit unter *solution* gemerkt und damit die Oberfläche auch weiß, welchen Aufgabentyp sie darstellen soll, gibt es ein zusätzliches Attribut *type*. In *givenAnswer* wird später vermerkt, welche Antwort der Benutzer ausgewählt hat, bevor das Objekt wieder dem *ExerciseProcessor* zum Lösen der Aufgabe übergeben wird.

Datenfluss im Frontend Nachdem nun die wichtigsten Bauteile von AngularJS bekannt sind und ein Überblick über die von uns verwendete Projektstruktur gegeben wurde, betrachten wir, beginnend bei der *index.html* unter *src/app*, das Zusammenspiel der einzelnen Bauteile, um zu verstehen, wie die Daten aus der REST-Schnittstelle bis zu Oberfläche gelangen:

```
<!DOCTYPE html>
<html>
<head>...</head>
  <body>

  <div id="app-container">
    <ifa-app></ifa-app>
  </div>

  <script src="jspm_packages/system.js"></script>
  <script src="jspm.config.js"></script>

  <script>
  System.import('./bootstrap/main')
```

```
15      .catch(console.error.bind(console));
16      </script>
17
18    </body>
19    </html>
```

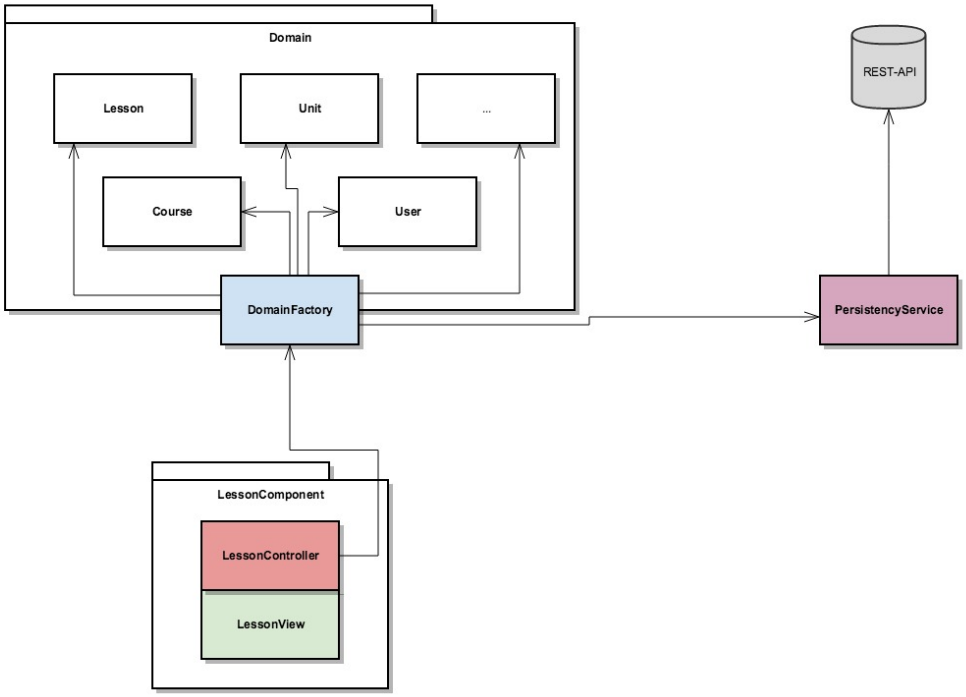
Der Header beinhaltet lediglich einige Meta-Informationen und Verlinkungen auf Stylesheets, weshalb dieser hier aus Gründen der Übersichtlichkeit ausgeblendet wurde. Im Body-Tag dieses HTML-Dokuments befindet sich ein Div-Tag, welches das Ifa-App-Tag umschließt. Dieses Ifa-App-Tag ist ursprünglich leer. Erst wenn SystemJS geladen und mit einer zweiten Javascript-Datei konfiguriert wurde, wird auf dem *bootstrap*-Ordner die Main-Datei geladen und ausgeführt. Dabei wird AngularJS entfaltet und das Ifa-App-Tag als Angular-Komponente erkannt. Diese Komponente ist unter *src/app/container* definiert. Es wird der zugehörige Controller (*src/app/-container/app.component.ts*) instanziiert und der Inhalt des Ifa-App-Tags durch das Markup der zugehörigen HTML-Datei ersetzt:

```
1     <ifa-header></ifa-header>
2     <ifa-mobile-menu></ifa-mobile-menu>
3     <ui-view></ui-view>
4     <ifa-footer></ifa-footer>
```

Hier werden drei weitere AngularJS-Komponenten benutzt: *header*, *mobileMenu* (welches nur auf mobilen Geräten sichtbar ist) und der *footer*. Zusätzlich gibt es hier ein Tag *ui-view*. Dies ist der Ort, an dem die Komponenten eingebunden werden, die zur aktuellen Route gehören. Befinden wir uns zum Beispiel derzeit in der *lesson*-Route, so wird hier das Template integriert, das in der Routendefinition angegeben ist. Dies ist in unserem Projekt stets eine einzelne Komponente, in diesem Beispiel die *lesson*-Komponente. Sind wir in der *lesson*-Route, so muss die URL in der Addressleiste des Browsers immer das Format *http://host/lesson/:lessonId* haben, wobei *:lessonId* ein Platzhalter für die eigentliche ID der aktuellen Lesson ist.

```
1     constructor(private DomainFactory : ExtendedDomainFactory,
          private $state: ng.ui.IStateService){
2
3     let lessonId = $state.params.lessonId;
4
5     DomainFactory.getLesson(lessonId)
6     .then((lesson : ifa.ILesson) => {
7       this.lessonVM = lesson.createViewModel();
8       })}
```

Diese ID wird vom Controller ausgelesen (Zeile drei) und anschließend an die *DomainFactory* übergeben. Die *DomainFactory* prüft nun, ob sie bereits ein Lesson-Objekt mit dieser ID im Speicher hat. Ist dies nicht der Fall, so wird der Aufruf an den *PersistencyService* übergeben.



Dieser Service stellt nun eine Anfrage an die REST-Schnittstelle des Servers, um die Lesson zu erhalten. Der Server überträgt die Daten als *LessonMemento* [GHJV95, 283]. Ein Memento ist dabei die rohe Form eines Objekts, welches einfach über HTTP übertragen werden kann:

```
1    interface ILessonMemento {
2      id: string
3      videoUrl: string
4      description: string
5      name: string
6      created: Date
7      lastModified: Date
8      previewImgUrl: string
9      level: number
10     duration: number
11     exercises: Array <string>
12     unit: string
13     orderId: number   }
```

Ist die Übertragung abgeschlossen, gibt der *PersistencyService* das Memento weiter an die anfragende *DomainFactory*. Diese nutzt das Memento, um daraus eine Instanz der Domänenklasse *Lesson* zu erzeugen. Die Instanz wird anschließend für weitere Anfragen unter der ID vermerkt und an den aufrufenden Controller der *lesson*-Komponente weitergegeben. Dies geschieht asynchron über sogenannte *Promises*. Ein Promise wird bei einer asynchronen Anfrage zurückgegeben und kann später *resolved* oder *rejected* werden. Hier liefert die *DomainFactory* bei der Anfrage *getLesson* ein Promise zurück (Zeile 5). Über die *then*-Methode des *Promise* übergibt der Controller (Zeile 6) eine Funktion, die ausgeführt wird, sobald der *Promise resolved* wird. Die übergebene Funktion erhält dabei die angefragte Lesson-Instanz als ersten Parameter. In Zeile 7 wird aus dieser Lesson-Instanz ein ViewModel erzeugt, welches anschließend im Attribut *lessonVM* des Controllers vermerkt wird. Nun kann das ViewModel der Lesson in der View benutzt werden. Ein ViewModel ist dabei dem LessonMemento sehr ähnlich. Anstelle von IDs als Verweise auf andere Objekte der Domäne hält ein ViewModel allerdings direkt eine ViewModel-Instanz dieser Referenz. Zusätzlich hat es Methoden zum Validieren und zum Speicher der Daten. So kann ein ViewModel an eine View gebunden werden. Dabei kann es vorkommen, dass die Daten des ViewModels (etwa durch ein Eingabefeld) manipuliert werden. Damit diese Änderungen nicht sofort in der Domäneninstanz vorgenommen werden, wird dies auf dem ViewModel gehandhabt. Erst nach erfolgreicher Validierung und dem Aufruf der Speichern-Methode befinden sich die Änderungen, welche an einem ViewModel vorgenommen wurden, auch in der Domäne wieder. Dabei kommuniziert die Domäneninstanz einer Klasse direkt mit dem *PersistencyService*, um Änderungen auch an den Server zu senden.

3.3.2 Backend: Bereitstellung von RESTful Services, Datenbankanbindung

von Dmytro Marchenko

Im Backend werden systemnahe Operationen, wie z.B. die Persistierung von Daten in die zugehörige nicht relationale Datenbank MongoDB vorgenommen sowie die Verwaltung der entsprechenden Daten umgesetzt und diese weiteren Anwendungen, wie etwa dem Frontend, bereitgestellt. (Eine detaillierte Beschreibung von Systemkomponenten ist im Abschnitt 2.2.2) Damit dem Backend Daten übermittelt bzw. Daten vom Backend abgefragt werden können, steht eine Schnittstelle bereit, die durch Representational State Transfer (abgekürzt REST) realisiert wurde. Die

Schnittstelle besteht aus mehreren Routen, die eine Antwort auf eine Clientanforderung an einem bestimmten Endpunkt bereitstellen. Ein Endpunkt ist eine URI und eine bestimmte HTTP-Anforderungsmethode (GET, POST usw.). Diese Routen ermöglichen die Durchführung von grundlegenden Funktionen wie Erstellung, Wiedergeben, Veränderung sowie Löschen der Daten. Sie wurden mittels Express.js für jede Klasse aus dem Klassendiagramm (siehe Abschnitt 2.2.1) erstellt. Die Dokumentation von REST-Schnittstelle ist im Abschnitt A.1 zu finden.

Die Bearbeitung einer Anfrage innerhalb einer Route besteht aus mehreren Schritten. Auf diesen Schritten werden verschiedene Services wie beispielsweise *ValidationService* verwendet. In diesem Abschnitt werden zunächst Bearbeitungsschritte anhand eines Beispiel von einer Route für die Veränderung eines Lesson-Objektes vorgestellt (siehe Listing 1). Im Anschluss werden die verwendeten Services detaillierter betrachtet.

Routen Im Laufe der Bearbeitung einer Anfrage wird zunächst geprüft, ob der Nutzer angemeldet ist (Listing 1, Zeile 1). Die Authentifizierung und ihre Überprüfung werden in Abschnitt 3.3.3 genauer betrachtet. Derzeit ist es wichtig, dass ein Zugriff auf die Route nur im Fall einer erfolgreichen Authentifizierung möglich ist, ansonsten wird eine Fehlermeldung 401 zurückgegeben. Im zweiten Schritt wird mit Hilfe von *SecurityServices* überprüft, ob der angemeldete Nutzer die benötigten Rechte für die Verwaltung eines angefragten Objekts hat (Listing 1, Zeile 2). Im dritten Schritt wird die Validierung der gesendeten Daten mit Hilfe von *ValidationService* durchgeführt (Listing 1, Zeile 4). Anschließend wird die Anfrage mit Hilfe von *PersistencyService* bearbeitet (Listing 1, Zeile 8). Dieser Service ist für die Verwaltung der Daten in der Datenbank verantwortlich. Falls in einem dieser Schritte ein Fehler auftritt, wird eine entsprechende Meldung zurückgegeben. Beispielsweise wird die Meldung mit dem Status 400 zurückgegeben, falls ein Fehler während der Validierung aufgetreten ist (Listing 1, Zeile 6). Im Falle einer erfolgreichen Bearbeitung wird die angefragte Information mit dem Status 200 zurückgeschickt (Listing 1, Zeile 9). Die Routen für die anderen grundlegenden Funktionen (Erstellung etc.) haben eine ähnliche Struktur und wurden mittels Templates für jede Klasse implementiert. Eines davon beschreibt die Interfaces und ein zweites Template die Implementierung der Routen. Im Folgenden werden die verwendeten Services genauer betrachtet.

```
1 router.put('/', security.jwtAuthenticationMiddleware, function (req, res) {
2   security.checkAccess('Lesson', req.payload.role, OperationTypeEnum.Update).then
      (function (hasAccess: boolean) {
```

```
3     if (hasAccess == true) {
4       var validationErrors = validationService.validateLesson(req.body)
5       if (validationErrors != null)
6         res.status(400).json(validationErrors);
7       else {
8         persistencyService.updateLesson(req.body).then(function (obj) {
9           res.status(200).json(obj);
10        }).catch(function (error) {
11            errorResponses.sendCustomError(res, error);
12          }
13        );
14      }
15    }
16    else
17      errorResponses.sendAccessError(res);
18  });
19 });
```

Listing 1: Eine Route für die Veränderung eines Lesson-Objekts

SecurityService Jeder Nutzer hat eine Rolle. Diese Rolle definiert für jede Klasse Operationen, die für diese Klasse durchgeführt werden können. Ein Nutzer mit der Rolle „Student“ darf beispielsweise Kurse anschauen, aber nicht verändern oder löschen. Die Informationen über die Rechte einer Rolle wird in der Datenbank gespeichert, was es ermöglicht, die Rechte zu verändern, ohne hierbei den Code verändern zu müssen. Die Struktur eines Dokumentes, das in der Datenbank gespeichert wird, wird mit Hilfe von *RightSchema* definiert. Der *SecurityService* sorgt dafür, dass ein Objekt nur von einem Nutzer bearbeitet werden kann, der über die entsprechenden Rechte verfügt (Listing 1, Zeile 2). Die grundlegenden Rollen wie „Student“ und „Manager“ sowie zu ihnen gehörigen Rechte werden beim Start des Servers erstellt, falls solche in der Datenbank noch nicht existieren.

Im Folgenden wird das *RightSchema* und die Methode zur Überprüfung der Rechte vorgestellt. Das *RightSchema* (Listing 2) definiert die Struktur des Dokuments in der MongoDB und enthält außer den trivialen Feldern, wie beispielsweise *ID*, drei weitere Felder. Sie dienen der Speicherung eines Bezeichners für die Rolle und das Schema, sowie erlaubter Operationen für diese Rolle. Es gibt vier mögliche Operationen, die auf den Objekten einer Klasse durchgeführt werden können. Dies sind Erstellen, Wiedergeben, Verändern sowie Löschen. Für jedes Feld wurden mögliche

Werte vorher definiert, um die Korrektheit der Daten zu garantieren. Beispielsweise kann das *schema*-Feld nur den Bezeichner von tatsächlich existieren Schemata erhalten (Listing 2, Zeile 10).

```
1  var RightSchema = new mongoose.Schema({
2    // "__v" (version)
3    // id
4    role: {
5      type: String,
6      enum: ['Student', 'Manager']
7    },
8    schema: {
9      type: String,
10     enum: ['Right', 'AttendCourse', 'Course', 'Exercise', 'Group', 'Lesson', '
           Unit', 'User', 'ExerciseAttempt']
11   },
12   permission: {type: Number, min: 0, max: 15},
13 }, customisationService.schemaOptions);
```

Listing 2: Das RightSchema

Erlaubte Rechte werden mittels einer Zahl definiert. Im Folgenden wird vorgestellt, wie diese Zahl berechnet wird. Die Erlaubnis für die Ausführung einer Operation wird über eine Dualzahl definiert. Dafür wird jeder Operation ein Bit zugeordnet (Erstellung – erstes Bit, Wiedergeben - zweites Bit etc.). Dieses Bit zeigt an, ob diese Operation durchgeführt werden darf oder nicht. Falls ein Bit einen positiven Wert (1) enthält, darf die Operation durchgeführt werden. Im alternativen Fall ist sie verboten. Die Rechte zum Wiedergeben und Verändern wird beispielsweise mittels der Zahl „0110“ dargestellt. Die Darstellung dieser Zahl im Dezimalsystem wird im *permission*-Feld gespeichert. Die Überprüfung der Rechte wird mittels einer *CheckAccess*-Funktion (Listing 3) innerhalb des Sicherheitsservices durchgeführt. Diese Funktion hat drei Parameter. Diese sind der Bezeichner eines Schemas, der Bezeichner einer Rolle und der Typ einer angeforderten Operation (Erstellung, Wiedergeben etc.). In der Funktion wird zunächst mittels des Bezeichners eines Schemas und des Bezeichners einer Rolle der Eintrag in der Datenbank (Listing 3, Zeile 2) herausgesucht. Dann wird überprüft, ob die angeforderte Operation durchgeführt werden darf (Listing 3, Zeile 6). Die Ausgabe wird mittels einer booleschen Variable definiert. Falls ein Eintrag in der Datenbank nicht gefunden wird oder die Rolle nicht über das Recht für die Ausführung der angefragten Operation verfügt, wird der Zugriff verweigert, ansonsten wird der Zugriff erlaubt.

```
export function checkAccess (schemaName: string, role: string, operationType:
    OperationTypeEnum) {
  return mongoose.model('Right').findOne({
    role: role,
    schemaName: schemaName
  }).then(function (obj: any) {
    return ((obj.permission & operationType) == operationType)
  }).catch(function (error) {
      return false;
    }
  );
}
```

Listing 3: Die Methode für die Überprüfung der Rechte

ValidationService Dieser Service ist für eine Prüfung auf Plausibilität von eingegebenen Werten (beispielsweise der Name eines Lessons) verantwortlich und wird in jeder Route aufgerufen, die für die Veränderung oder Erstellung der Objekte der Klassen (siehe Abschnitt 2.2.1) zuständig ist. Der Service enthält Validierungsmethoden für jede Klasse (*validateUnit*, *validateLesson* etc.). Um die Validierung zu vereinfachen, wird eine externe Bibliothek verwendet. Diese Bibliothek verfügt über eine Methode, die mittels der vordefinierten Regel ein Objekt validiert. Im Folgenden wird die Implementierung der Methode (siehe Listing 4) für die Validierung eines *Lesson*-Objekts vorgestellt.

Zunächst wird ein *constraints*-Array mit einer Liste von Validierungsregeln der *Lesson*-Klasse definiert, die überprüft werden müssen. Dabei wird für jedes Feld (zum Beispiel *videoUrl*, *name* etc.) mehrere Regeln bereitgestellt. Eine der Regeln fordert, dass der Name eines *Lesson*-Objekts definiert ist und hierbei nicht länger als 100 Zeichen ist. Eine weitere Regel bezieht sich auf das Format von *videoUrl*-Feldes. Diese Regel sogrt dafür, dass es tatsächlich eine URL-Adresse angegeben wird. Nachdem Regeln definiert wurden, wird die Validierungsmethode aufgerufen, die bestimmt, ob die Validierungsregeln erfüllt sind (Listing 4, Zeile 16). Für die Erzeugung von *ValidationService* werden auch Templates verwendet.

```
public validateLesson(obj) {
  var constraints = {
     ...
    "name": {
    "presence": {
      "message": "Bitte geben Sie einen Namen ein."
```

```
7       },
8       "length": {
9         "minimum": 1,
10        "maximum": 100,
11        "message": "Ein Name muss mindestens 1 Zeichen und Maximal 100 Zeichen
              enthalten."
12      }
13    },
14       // ...
15    };
16    return validate(obj, constraints);
17  }
```

Listing 4: Methode für die Validierung eines Lesson-Objekts

PersistencyService Dieser Service ist für die Kommunikation mit der Datenbank verantwortlich und dient als Zwischenstelle zwischen der Datenbank und der Anwendung. Für jede Klasse enthält dieser Service Methoden für die Speicherung, Veränderung, Wiedergabe sowie das Löschen der Daten. Falls eine Methode die Veränderung von mehreren Objekten verursacht, werden darunterliegende Operationen in einer Transaktion durchgeführt. Außerdem wird ein Objekt, das gelöscht werden soll, als solches mittels der *deleted*-Feldes markiert, aber nicht tatsächlich gelöscht. Dies ermöglicht eine Wiedererstellung im Fall eines unbeabsichtigten Löschens. Im Folgenden wird die Implementierung einer Methode für das Löschen eines *Lesson*-Objekts vorgestellt.

Beim Löschen eines *Lesson*-Objektes sollen mehrere Operationen durchgeführt werden, die innerhalb einer Transaktion ausgeführt werden. Zunächst muss die Referenz auf dieses Objekt aus einem *Unit*-Objekt gelöscht werden, zu dem dieses *Lesson*-Objekt gehört (Listing 5, Zeile 4). Außerdem sollen alle von diesem Objekt abhängigen Aufgaben gelöscht werden (Listing 5, Zeile 5). Zuletzt wird dieses *Lesson*-Objekt selbst gelöscht (Listing 5, Zeile 6).

```
1  public deleteLesson(lessonId: string): Promise {
2    return this.getLessonById(lessonId).then(function (obj: ifa.ILessonMemento) {
3      var transaction = Fawn.Task();
4      transaction.update('Unit', {_id: obj.unit}, {'$pull': {lessons: lessonId}});
5      transaction.update('Exercise', {lesson: lessonId}, {deleted: true}).options({
           multi: true});
6      obj.deleted = true;
7      transaction.save('Lesson', obj);
```

```
8     return transaction.run();
9   });
10 }
```

Listing 5: Methode für das Löschen eines *Lesson*-Objekts

Schema und Modell Die Struktur eines Dokuments, welches in MongoDB gespeichert wird, wurde aus dem Klassendiagramm (siehe Abbildung 3) übernommen und mittels eines Schemas deklariert. Ein Schema beschreibt, welchen Typ und welche Einschränkungen jedes Feld hat. Außerdem werden in einem Schema zusätzliche Methoden definiert, die ausgeführt werden müssen, nachdem die Operationen wie beispielsweise die Speicherung ausgeführt wurde. Dies sind die Methoden für die Validierung eines Objektes und Methoden für die Aktualisierung der Parent- und Kinder-Knoten nach der Erstellung oder Veränderung eines Objektes. Außerdem wurde auch die Methode für die automatisierte Ordnung der Elemente in einer Kollektion erstellt. Dies ist notwendig, um beispielsweise eine vordefinierte Reihenfolge der *Lesson*-Objekte in einem *Unit*-Objekt zu definieren. Anschließend wird für jedes Schema auch ein Model-Objekt erstellt, das die Verwaltung von Objekten einer Klasse mittels Anfangen ermöglicht. Solche Model-Objekte werden in dem *PersistencyService* verwendet.

3.3.3 Security: Authentifizierung im System

von Dmytro Marchenko

Die Authentifizierung wurde mittels JSON Web Token (JWT) implementiert. Dieses Token wird für die Sicherung der Routen verwendet (der Zugriff wird für einen nicht authentifizierten Nutzer verweigert), indem jede Anfrage mittels dieses Tokens authentifiziert wird. In diesem Abschnitt wird zunächst der Aufbau von JWT sowie ein typischer Ablauf einer Authentifizierung vorgestellt. Danach wird betrachtet, wie dies für das Frontend und Backend implementiert wurde.

Ablauf einer tokenbasierten Authentifizierung Die Authentifizierung beginnt mit einer Anfrage des Nutzers, welche ein Password und eine E-Mail-Adresse (oder Login) enthält. Auf dem Server werden diese Daten überprüft und im Falle einer erfolgreichen Authentifizierung wird ein JWT generiert und mittels Keyed-Hash Message Authentication Code (HMAC) [Wik17a] mit kryptologischen Hashfunktio-

nen SHA-256 [Wik17c] signiert. Danach wird dieser Token zum Nutzer zurückgeschickt. Das Token wird im Folgenden verwendet, um die Anfrage an gesicherten Routen durchzuführen. Bei einer solchen Anfrage wird das Token in dem Header übergeben. Auf dem Server wird überprüft, ob das Token valide ist. Im Falle einer erfolgreichen Validierung wird die angefragte Operation ausgeführt.

Die Verwendung eines Tokens ermöglicht somit eine Authentifizierung, ohne eine Session zu pflegen oder die für eine Authentifizierung notwendigen Daten, wie beispielsweise ein Password, jedes Mal zu versenden. Hierbei hat das Token eine Ablaufzeit, nach der das Token nicht mehr verwendet werden kann. Das Token selbst besteht aus drei Teilen, darunter sind der Header, der Payload und eine Signatur. Im Payload wird die individuelle Information, wie beispielsweise der Identifikator eines Nutzers, sowie die Ablaufzeit gespeichert. Diese Daten sind zwar nicht verschlüsselt, ihre Veränderung wird aber zu einer fehlerhaften Validierung auf dem Server führen, da die Signatur nicht übereinstimmen wird.

Frontend Für die Verwaltung einer Authentifizierung wurde im Frontend ein *AuthenticationService* entwickelt. Dieser Service beinhaltet verschiedene Methoden zur Verwaltung eines Nutzers, wie beispielsweise Einloggen, Abmeldung, Überprüfung der Anmeldung und einige mehr. Im Folgenden wird ein Ablauf für die Authentifizierung im Frontend vorgestellt.

Wie zuvor erwähnt, muss der Nutzer für das Einloggen Daten wie eine E-Mail-Adresse (oder Login) und ein Password eingeben. Dafür werden die Daten in einem HTML Formular eingetragen und mittels des Login-Controllers, der für diese Form verantwortlich ist, zu der *login*-Methode von *AuthenticationService* übergeben. In dieser Methode werden die Daten mit Hilfe von *RestService* an den Server geschickt (Listing 6, Zeile 4). War die Anmeldung erfolgreich, wird der Token mittels der *saveToken*-Methode von *AuthenticationService* in dem Local Storage des Browsers gespeichert (Listing 6, Zeile 6). Das Ergebnis der Authentifizierung wird zurück zum Login-Controller geschickt. Im Falle einer erfolgreichen Anmeldung wird eine Weiterleitung zur Home-Seite durchgeführt, ansonsten wird die entsprechende Fehlermeldung angezeigt.

```
1 public login(identifier: string, password: string) {
2   let deferred: ng.IDeferred = this.$q.defer();
3   var authService = this;
4   this.restService.login(identifier, password)
5     .then(response => {
```

```
6       authService.saveToken(response.token);
7       deferred.resolve();
8     }).catch(function (err) {
9     deferred.reject(err);
10  });
11  return deferred.promise;
12 }
```

Listing 6: Eine Methode für das Einloggen aus *AuthenticationService*

Wie in der Beschreibung des Ablaufs der tokenbasierten Authentifizierung erwähnt, muss bei jeder Anfrage an die gesicherten Routen des Servers das Token in dem Header der Anfrage übergeben werden. Dafür wird ein *$http*-Service von Angular [Ang17b] verwendet, der bei jeder Anfrage automatisch ausgeführt wird. Dieser Service wurde durch ein zusätzliches *Interceptor* [Ang17c] erweitert, in dem das Token aus dem Lokal Storage in Header eingefügt wird.

Backend Für eine Authentifizierung wird auf dem Server eine separate Route verwendet (Listing 7). In dieser Route wird zunächst überprüft, ob alle benötigen Daten eingetragen wurden (Listing 7, Zeile 2). Danach wird mittels eines regulären Ausdrucks (Listing 7, Zeile 6) bestimmt, ob es um eine Authentifizierung mittels einer E-Mail-Adresse oder eines Logins handelt. Danach werden die Daten zu einer *authenticate*-Methode des *Passport*-Services übergeben (Listing 7, Zeile 7). Anhand dieses Services können verschiedene Anmeldungsstrategien, wie beispielsweise die Anmeldung mittels Facebook oder eines Google Kontos durchgeführt werden. Derzeit werden nur zwei lokale Strategien verwendet. Eine ist für die Authentifizierung anhand des Logins und die andere ist für die Authentifizierung anhand der E-Mail-Adresse. In der Authentifizierungs-Methode der beiden lokalen Strategien werden die Anmeldungsdaten überprüft. Dafür wird der Nutzer anhand seiner E-Mail-Adresse (bzw. Login) gefunden und das Password mittels einer Validierungsfunktion der *User*-Klasse überprüft. Dabei ist ein direkter Vergleich des Passworts nicht möglich, da in der Datenbank ein tatsächliches Password nicht gespeichert wird, sondern ein Hashwert nach der Verwendung einer PBKDF2-Funktion (Password-Based Key Derivation Function 2) [Wik17b]. Somit wird während der Validierung des Passworts zunächst wieder ein Hashwert mittels Eingabedaten erzeugt und mit dem gespeicherten Wert verglichen. Im Fall einer fehlerhaften Authentifizierung oder falls der Nutzer nicht gefunden wurde, wird der Fehler 404 bzw. 401 zurückgegeben. Anderenfalls wird mittels der *generateJwt*-Methode der *User*-Klasse ein Token erstellt

und signiert (Listing 7, Zeile 13). Schließlich wird das erstellte Token des Nutzers mit dem Status 200 zurückgegeben.

Im Fall einer späteren Anfrage an die gesicherten Routen wird überprüft, ob die Signatur mit dem erhaltenen Token übereinstimmt und die Ablaufzeit noch nicht überschritten ist (Listing 1, Zeile 1). Dafür wird eine weitere Methode des Sicherheitsservices verwendet. Falls die Signatur nicht übereinstimmt oder ein Token abgelaufen ist, wird der entsprechende Fehler zurückgegeben. Ansonsten werden die Bearbeitungsschritte in der angefragten Route ausgeführt (sieh Abschnitt 3.3.2).

```
1  router.post('/login', function (req, res) {
2    if (!req.body.identifier || !req.body.password) {
3      errorResponses.sendNotAllFieldsSpesifiedSON(res);
4      return;
5    }
6    let authStrategy: string = (/\S+@\S+\.\S+/.test(req.body.identifier)) ? 'email-
       local' : 'login-local';
7    security.passport.authenticate('local', function (err, user, info) {
8      if (err) {
9        errorResponses.sendCustomError(res, err, 404);
10       return;
11     }
12     if (user) // User was found.
13       res.status(200).json({token: user.generateJwt()});
14     else
15       errorResponses.sendCustomError(res, info, 401);
16   })(req, res);
17 });
```

Listing 7: Eine Route für das Einloggen

3.3.4 Content-Management (CoMa)

von Mesut Sahin

In diesem Kapitel wird der Content-Management Bereich vorgestellt, welcher die Administrationsoberfläche des Projekts darstellt. Die gesamten Daten des Projekts werden im CoMa-Bereich verwaltet. Das CoMa wurde mit dem Ziel, eine strukturierte, erweiterbare und benutzerfreundliche Grundumgebung für die Verwaltung der Daten zu schaffen, erstellt. Bei der Entwicklung des CoMa-Bereichs sollte darauf geachtet werden, keine direkten Datenbankzugriffe zu verwenden. Aus diesem Grund greift das CoMa auf das Backend bzw. die API zu. Alle CRUD-Operationen

werden umgesetzt, indem die erhaltenen Daten aus dem angezeigten Template, seitens des Benutzers, an die API weitergeleitet werden und somit der Backend Teil die weiteren logischen Schritte und damit auch die Persistierung in der Datenbank übernimmt. Vereinfacht ausgedrückt ist der CoMa-Bereich im wesentlichen die Benutzeroberfläche der Create, Update und Delete Operationen der API. Die Read Operation ist wesentlicher Bestandteil des Frontends und spielt im CoMa-Bereich eine nebensächliche Rolle.

Der CoMa-Bereich wurde in *Gruppen*, *User*, *Kurse*, *Kapitel*, *Lektionen* und *Aufgaben* unterteilt. Dabei sind für die Benutzerverwaltung *Gruppen* und *User* und für die Lehrstoffverwaltung *Kurse*, *Kapitel*, *Lektionen* und *Aufgaben* zuständig.

Für die Benutzerverwaltung ist es vorgesehen, dass beim Erstellen eines Schülers eine Zuweisung in eine *Gruppe* erfolgen muss. Eine *Gruppe* besteht aus einem oder mehreren Schülern und kann ohne Einschränkungen erstellt werden.

Bei der Lehrstoffverwaltung werden *Kurse* bereitgestellt, welche *Einheiten* zugeordnet werden. Jeder *Kurs* besteht aus einer oder mehreren *Einheiten*. Auch *Einheiten* sind wiederum unterteilt in einer oder mehreren *Lektionen*, wobei eine *Lektion* ohne eine *Einheit* nicht bestehen kann. Zu jeder *Lektion* kann maximal ein Video hinzugefügt werden. Als letzter Punkt werden *Aufgaben* für die *Lektionen* zur Verfügung gestellt. Jede *Lektion* besitzt eine oder mehrere *Aufgaben* und jede *Aufgabe* muss einer *Lektion* zugeordnet sein.

Die Interaktion zwischen der API, Route und Template *von Can Celebi*
Die Zusammenarbeit zwischen der Route, Template und API wird durch Quelltextausschnitte zu den einzelnen Anwendungsfällen des CoMas näher erklärt. Die Stellen, die im Programmcode nicht berücksichtigt werden, sind mit einem [...] gekennzeichnet. Hierbei wird zwischen abhängigen und unabhängigen Anwendungsfällen unterschieden. Die abhängigen Anwendungsfälle können nur durchgeführt werden, wenn sie der jeweiligen und bestehenden Rest-Ressource (Datenobjekt) zugeordnet werden (siehe Abschnitt 3.5.9 User Zuweisung in eine Gruppe), wobei die unabhängigen Anwendungsfälle ohne eine bestehende Rest-Ressource ausgeführt werden. Die Anwendungsfälle für eine *Einheit* wären beispielsweise „Einheit erstellen“, „Einheit bearbeiten“, „Einheit anzeigen“, „Einheit löschen“ und „Kurs für die Einheit auswählen“, wobei „Einheit erstellen“ zu den abhängigen Anwendungsfällen gehört, da eine *Einheit* ohne eine Zuweisung zu einem Kurs nicht erstellt bzw. nicht existieren kann.

Für jede der CRUD-Operationen (Create, Read, Update, Delete), also für alle Anwendungsfälle, existiert ein Template und eine Route. Die Route ruft die Rest API des Backends auf, um von dem Client angeforderte Datenbankobjekt zu erhalten. Anschließend wird das Datenbankobjekt von der Route an das jeweilige Template zum Renderig übergeben.

Die HTTP Methode POST und das Template *new.jade* kann für das Anlegen neuer Rest-Ressourcen verwendet werden. Zum Editieren einer Rest-Ressource wird die HTTP Methode PUT und das Template *edit.jade* benutzt. Für das Anzeigen einer bestimmten Rest-Ressource oder Anzeigen aller Rest-Ressourcen wird die HTTP Methode GET und die Templates *show.jade* oder *index.jade* benutzt. Mit der folgenden HTTP Methode DELETE wird die bestehende Rest-Ressource von der Datenbank entfernt, die über das Template *show.jade* mithilfe des Löschen-Buttons erfolgen muss.

Mit Jade-Bootstrap wurde die Oberfläche des CoMa-Bereichs erarbeitet. Das Jade Bootstrap stellt die Mixins bereit, die notwendig sind für die Gestaltung der Oberfläche im Jade Template. Mixins können als Funktionen, wie aus anderen Programmiersprachen bekannt, verstanden werden. Diese stellen wiederverwertbare Programmbausteine dar, die mit Argumenten belegt werden können und an verschiedenen Stellen des Programmcodes aufgerufen und eingesetzt werden können.

Der CoMa-Bereich kann in einem mobilen Endgerät oder auch in einem Browser auf dem Desktop benutzt werden, da das Responsive Webdesign von Jade Bootstrap unterstützt wird.

Template new.jade am Beispiel „Kurs erstellen“ *von Mesut Sahin*

In diesem Abschnitt wird das Template *new.jade* für den Anwendungsfall „Kurs erstellen“ vorgestellt (siehe Listing 8). Das Basislayout (*layout.jade*) stellt die oberste Ebene der Jade Template-Engine dar, die in *new.jade* überschrieben wird. Das Template *new.jade* wurde in zwei Blöcke (navi und content) unterteilt.

Der Block navi stellt die Navigationsleiste des Coma Bereichs dar, und der Block content ist für das Formular zuständig. In Zeile 19 wird das Formular für „Kurs erstellen“ definiert. Der Parameter action enthält die Adresse, an der die Formulardaten übermittelt werden. Hierbei werden die Daten erst an `coma.in4all-pg.de/auth` überreicht, da zuerst eine Überprüfung stattfindet, ob der User im CoMa authentifiziert ist.

Im Formular sind zwei Inputfelder für Kursname und Kursbeschreibung, sowie eine Checkbox enthalten. Checkbox ist für die Veröffentlichung des Kurses gedacht. Die Mixins +inputValueReqL (Zeile 20) und +inputValue (Zeile 21) werden im Basislayout definiert. Der wesentliche Unterschied zwischen den beiden Mixins ist, dass bei +inputValueReqL das Inputfeld gesetzt sein muss. Somit kann das Formular ohne einen Kursnamen nicht abgeschickt werden. Die Mixins sind in den Templates mit einem plus gekennzeichnet.

Der erste Parameter von +inputValueReqL bzw. +inputValue setzt den Typen der Eingabe fest, der zweite Parameter steht für die id, der dritte Parameter steht für den Placeholder, der vierte Parameter ist der angezeigte Labelname, der fünfte Parameter ist der Name des Inputfeldes, welcher identisch mit dem jeweiligen Attribut des Model Schema sein muss und der letzte Parameter ist der Eingabe/Wert des Inputfeldes. Dieser Parameter ist beispielsweise bei *edit.jade* gesetzt, da schon Daten innerhalb der Datenbank existieren.

Das Mixin +Checkbox (Zeile 22) enthält drei Parameter. Der erste Parameter ist der angezeigte Labelname, der zweite Parameter ist der Name des Inputfeldes und der letzte Parameter steht für die id. Die Zeilen 27 bis 29 sind für das Anzeigen des Dialogfensters zuständig.

```
1  extends ../layout
2
3  block navi
4    +nav_item("/group") Gruppen
5    +nav_item("/user") User
6    +nav_item_dropdown("#")(label="Kurse")
7      +nav_item("#") Erstellen
8      +nav_divider
9      +nav_header Verwalten
10     +nav_item("/course/all") Kurs ausw\"ahlen
11   +nav_item("/unit") Einheiten
12   +nav_item("/lesson") Lektion
13   +nav_item("/exercise") Aufgabe
14
15 block content
16   h2.
17     Kurse
18   p
19     form#frmCourseAdd(action='/auth',method="post", enctype='application/x-www-
           form-urlencoded')
20       +inputValueReqL("text","inputName","Name eingeben","Name","name","")
```

```
21      +inputValue("text","inputDesc","Beschreibung eingeben","Beschreibung","
            description","")
22      +checkbox("Ver\"offentlichen", "isPublished", "cbPublished")
23      p
24      button.btn.btn-primary(data-toggle="modal",data-target="#myModal",onclick="
            return false;") Speichern
25      input(type='hidden',value=backendUrl + "/course",name='targetUrl')
26      input(type='hidden',value=csrf.csrfToken,name='_csrf')
27      +modal("Kurs speichern?","myModal")
28        p Wollen Sie den Kurs wirklich erstellen?
29        +submit-primary("Ja")
```

Listing 8: Ausschnitt des Quellcodes zur Kurserstellung (new.jade)

Template edit.jade am Beispiel „Kurs editieren“ *von Can Celebi*

Dieser Abschnitt stellt das Template *edit.jade* für den Anwendungsfall „Kurs editieren“ vor (siehe Listing 9). Da die beiden Templates *new.jade* und *edit.jade* sehr ähnlich aufgebaut sind, wird nur auf die Unterschiede der beiden Templates eingegangen. Der Aufbau der Navigationsleiste und des Formulars entspricht dem Template *new.jade*. Das Mixin +inputValueReqL (Zeile 7 und 8) stellt die Inputfelder für den Kursnamen und Kursbeschreibung bereit. Der Unterschied zu *new.jade* ist, dass der letzte Parameter gesetzt ist. Das Kursobjekt wird in der Route `courseComa.class.ts` mit dem Befehl *render* an das Template *edit.jade* übergeben. Anschließend werden im Formular die Platzhalter mit den übergebenen Kursdaten gefüllt. Falls zu dem Feld keine Informationen vorliegen, bleibt dieses Feld leer. Die Kursdaten werden durch die Änderung der Inputfelder im Formular aktualisiert.

```
1  [...]
2
3  block content
4    h2.
5      Kurs ID #{course._id}
6    form#frmUnitEdit(action='/auth',method='post',enctype='application/x-www-form-
         urlencoded')
7      +inputValueReqL("text","inputName","Name eingeben","Name","name",course.name)
8      +inputValueReqL("text","inputDesc","Beschreibung eingeben","Beschreibung","
           description",course.description)
9      +checkbox("Ver\"o\ffentlichen", "isPublished", "cbPublished")
10     p
11     button.btn.btn-primary(data-toggle="modal",data-target="#myModal",onclick="
           return false;") Speichern
```

```
12      input(type='hidden',value=backendUrl + '/course',name='targetUrl')
13      input(type='hidden',value=course.csrfToken,name='_csrf')
14      input(type='hidden',value=course._id,name='id')
15
16  [...]
```

Listing 9: Ausschnitt des Quellcodes zur Kursbearbeitung (edit.jade)

Template show.jade am Beispiel „Kurs anzeigen“ *von Mesut Sahin*

Im folgenden wird das Template *show.jade* für den Anwendungsfall „Kurs anzeigen“ vorgestellt (siehe Listing 10). Hierbei wird auch nur auf die Unterschiede der beiden vorgestellten Templates eingegangen.

In der Zeile 7 wird eine Tabelle definiert, die Informationen über den Kurs anzeigt. Der linke Index entspricht den Attributen des Model Schemas und der rechte Index weist diesen Attributen einen Wert zu. Des Weiteren kann mit den Löschen-Button der Kurs samt seiner gesamten Informationen aus der Datenbank entfernt werden (Zeile 9-16). Außerdem ist es möglich, mithilfe des Editieren-Buttons in den Anwendungsfall „Kurs editieren“ zu gelangen (Zeile 18-20). Als letztes bietet sich die Möglichkeit, eine Einheit zu diesen Kurs hinzuzufügen (Zeile 22-24).

```
1   [...]
2
3   block content
4     h2.
5       Kurs #{course.name}
6
7     +table(["Parameter", "Wert"],[["Name", course.name], ["Beschreibung", course.
          description], ["Stand", course.isPublished],["Einheit", course.units]],"
          table-striped table-bordered")
8
9     form(action='/auth',method='post',enctype='application/x-www-form-urlencoded',
          style="float:left;margin-right:10px;margin-top:25px")
10      input(type='hidden',value=backendUrl + '/course/#{course._id}',name='
            targetUrl')
11      input(type='hidden',value=course.csrfToken,name='_csrf')
12      input(type='hidden',value='DELETE',name='_method')
13      button.btn.btn-danger(data-toggle="modal",data-target="#myModal",onclick="
            return false;") L\"oschen
14      +modal("Kurs l\"oschen?","myModal")
15        p Wollen Sie den Kurs #{course.name} wirklich l\"oschen?
16        +submit-primary("Ja")
```

```
17
18    form(action='/course/edit/#{course._id}',method='post',enctype='application/x-
          www-form-urlencoded',style="float:left;margin-right:10px;margin-top:25px")
19      input(type='hidden',value='GET',name='_method')
20      +submit-primary("Editieren")
21
22    form(action='/unit/new/#{course._id}',method='post',enctype='application/x-www-
          form-urlencoded',style="float:left;margin-top:25px")
23      input(type='hidden',value='GET',name='_method')
24      +submit-primary("Einheit hinzuf\"ugen")
25
26  [...]
```

Listing 10: Ausschnitt des Quellcodes zur „Kurs anzeigen“ (show.jade)

Template index.jade am Beispiel „alle Kurse anzeigen“ *von Can Celebi*

In dem Template *index.jade* wird der Anwendungsfall „alle Kurse anzeigen“ dargestellt (siehe Listing 11).

Damit alle Kurse auf der Seite angezeigt werden können, erhält das Template von der Route `courseComa.class.ts` ein Kursobjekt. Das Kursobjekt, welches alle Kurse beinhaltet, besitzt den Namen „courses“. Anschließend wird dieses Kursobjekt in der foreach-Schleife verarbeitet und auf der Seite untereinander, mit der Form *Kurs: Kursname*, dargestellt (Zeile 7-9).

```
1  [...]
2
3  block content
4    h2.
5      Kurse verwalten
6    p
7    +list-group-custom()
8      each course in courses
9        a.list-group-item(href="/course/#{course._id}") Kurs: #{course.name}
```

Listing 11: Ausschnitt des Quellcodes zur „alle Kurse anzeigen“ (index.jade)

Template choose.jade am Beispiel „Gruppe für den User auswählen“ *von Mesut Sahin*

Die Templates, die in den vorherigen Abschnitten vorgestellt wurden, haben den gesamten CoMa-Bereich umfasst. Jedoch existiert das Template *choose.jade* nur

für die Bereiche *User*, *Einheiten*, *Lektionen* und *Aufgaben*, weil der Anwendungsfall „erstellen“ innerhalb dieser Bereiche nur durchgeführt werden kann, wenn sie der abhängigen Rest-Ressource zugewiesen werden können. In dem Template *choose.jade* wird der Anwendungsfall „Gruppe für den User auswählen“ dargestellt (siehe Listing 12).

Das Template *choose.jade* stellt für den *User* Bereich alle *Gruppen* zur Verfügung, da beim Erstellen eines Users eine Zuweisung in eine *Gruppe* erfolgen muss. Der Aufbau des Templates entspricht dem Template *index.jade*.

```
1 [...]
2
3 block content
4   h2.
5     Gruppe des Users ausw\"ahlen
6   p
7   +list-group-custom()
8     each group in groups
9       a.list-group-item(href="/user/new/#{group._id}") Gruppe: #{group.name}
```

Listing 12: Ausschnitt des Quellcodes zur „Gruppe für den User auswählen“ (choose.jade)

Beispielhafte Erklärung einer Route für das Template index.jade *von Can Celebi*

In diesem Abschnitt wird die all-Route für das Template *index.jade* erklärt. Da die gesamten Routen sehr ähnlich aufgebaut sind, wird nur beispielhaft auf diese Route eingegangen.

In der Klasse `courseComa.class.ts` wird die Route, mit der HTTP „GET“-Anfrage, definiert (siehe Listing 13). Die GET-Methode erhält als Parameter die Route `coma.in4all-pg.de/course/all` (Zeile 3). Diese Methode wird ausgeführt, wenn die angegebene Route aufgerufen wird.

Damit alle Kurse angezeigt werden können, müssen die Kurse zuerst aus der Datenbank geladen werden. Hierfür wird die Klasse `restService` aufgerufen mit der Methode *getObject* (Zeile 4). Die Methode *getObject* erhält als Parameter die API-URL `http://api.in4all-pg.de/course`. Anschließend wird in der API das Datenbankobjekt von der Datenbank angefordert und an die Route weitergeleitet. Diese werden in der Variabel obj gespeichert (Zeile 5).

Mithilfe der Funktion *render*, wird das Datenbankobjekt courses, an das Template *index.jade* übergeben (Zeile 9-10). In dem Datenobjekt courses befinden sich alle Kurse.

Die Zeilen 15 bis 18 stehen für die Fehlerbehandlung.

```
1  [...]
2
3  router.get('/all', function (req, res) {
4    restService.getObject(config.backendPath + '/course/', '', req.cookies,
5  res)
6    .then(function (obj) {
7
8     res.format({
9       html: function () {
10        res.render('courses/index', {
11          "courses": obj,
12           status: 200
13        });
14      }
15     });
16    }).catch(function (error) {
17        res.status(500).json(error);
18      }
19    );
20 });
21
22 [...]
```

Listing 13: Route-Datei course.Coma.class.ts - Erstellen eines neuen Kurses

Verzeichnisstruktur *von Mesut Sahin*

Die Verzeichnisstruktur des CoMa-Bereichs wird im Verzeichnisbaum, in Abbildung 5, dargestellt.

Im Ordner assets werden Bilder und CSS-Dateien für die Jade-Templates bereitgestellt (siehe Abschnitt 3.5.5.2.1). Die Jade-Templates werden im Ordner views abgelegt. Das Rendering der Jade Dateien findet in den Routen statt. Die Routen befinden sich im Ordner routes.

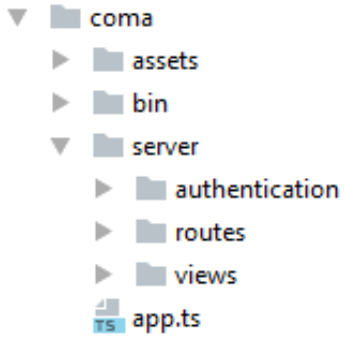


Abbildung 5: Verzeichnisbaum

3.3.5 Implementierung der Aufgabentypen

von Can Celebi

Dieses Kapitel beschreibt die Implementierung der Aufgabentypen. Dabei wird auf die Syntax der einzelnen Aufgabentypen, welche die Syntax verarbeitet, eingegangen. Die genaue programmtechnische Umsetzung wird nicht näher betrachtet, weil sie im Grunde auf reguläre Ausdrücke aufbaut und jedem Entwickler mit Grundverständnis der Informatik bekannt sein sollten.

Aufgabentypen und Syntax Das System unterstützt vier Aufgabentypen, wobei jeweils zwei Aufgabentypen Erweiterungen darstellen. Zum Einen existieren die Aufgabentypen Single Choice und Multiple Choice, welche die Einfach- bzw. Mehrfachauswahl von Antworten ermöglicht. Diese beiden Aufgabentypen werden in den folgenden Abschnitten auch als die Aufgabenklasse Auswahl zusammengefasst. Zum Anderen sind Aufgaben des Typus Lückentext als Eingabefeld und Lückentext als Dropdown umsetzbar, welche sich in der Darstellungsform unterscheiden. Diese Aufgabentypen werden im Folgenden auch unter der Aufgabenklasse Lückentexte geführt.

Die Syntax für beide Aufgabenklassen folgen dem gleichen Schema, wobei die Aufgabenklasse der Lückentexte eine Erweiterung der Syntax der Aufgabenklasse Auswahl darstellt.

Syntax Single Choice und Multiple Choice Die Syntax für eine Frage einer Single bzw. Multiple Choice Aufgabe bedarf keiner speziellen Anforderungen. Es kann jede erdenkliche Aufforderung oder Frage in Form eines freien Textes, ohne

einem bestimmten Muster zu folgen, gestellt werden.
Die Syntax für die Antwortmöglichkeiten dieser Aufgabenklasse ist wie folgt definiert:

Aufgabentyp Der Aufgabentyp für Single Choice ist YN(Yes/No) und der Aufgabentyp für Multiple Choice ist MC. Nach dem der Aufgabentyp benannt ist muss dieser mit einem Doppelpunkt ":" separiert werden.

Antwortmöglichkeiten Als nächstes müssen die Antwortmöglichkeiten genannt werden. Damit zwischen falschen und richtigen Antwortmöglichkeiten unterschieden werden kann, muss eine richtige Antwort mit einer "1" und dem Separator "_" und eine falsche Antwort mit einer "0" und dem Separator "_" vor dem Anzeigetext markiert werden. Die einzelnen Antworten wiederum müssen mit einem "," separiert werden, dabei ist zu beachten, dass kein Leerzeichen vor und nach dem "," eingegeben wurde.

Eine anschauliche und selbsterklärende Darstellung der Syntax und damit auch der Implementierung ist in Abbildung 6 zu sehen.

Syntax Lückentext und Lückentext Dropdown *von Mesut Sahin*
Die Syntax für eine Frage der Aufgabenklasse Lückentext muss im Gegensatz zur Aufgabenklasse Auswahl einem bestimmten Muster folgen. Als Frage muss für einen Lückentext der gesamte Text angegeben werden, wobei die Lücken mit einem doppelten Unterstrich und einem Index gekennzeichnet werden müssen. Als Index kann eine beliebige Zeichenkette ohne Leerzeichen gewählt werden. Die Lücken im Text, die über das Coma mit einem zusätzlichen Bild versehen werden sollen, müssen mit einem zusätzlichen i (für image) zwischen den beiden Unterstrichen, also Index_i_ versehen werden.

Die Syntax für die Antwortmöglichkeiten dieser Aufgabenklasse baut auf der Syntax der Aufgabenklasse Auswahl auf und erweitert diese lediglich um die folgenden Punkte:

Aufgabentyp Als Aufgabentyp kann zwischen LN(Lückentext als Eingabefeld) und LD (Lückentext als Dropdown) unterschieden werden. Auch hier muss wieder am Ende mit einem : separiert werden.

Antwortmöglichkeiten bzw. Antwortlisten Die Antwortmöglichkeiten müssen einem bestimmten Index zugeteilt werden, deshalb muss vor den Antwort-

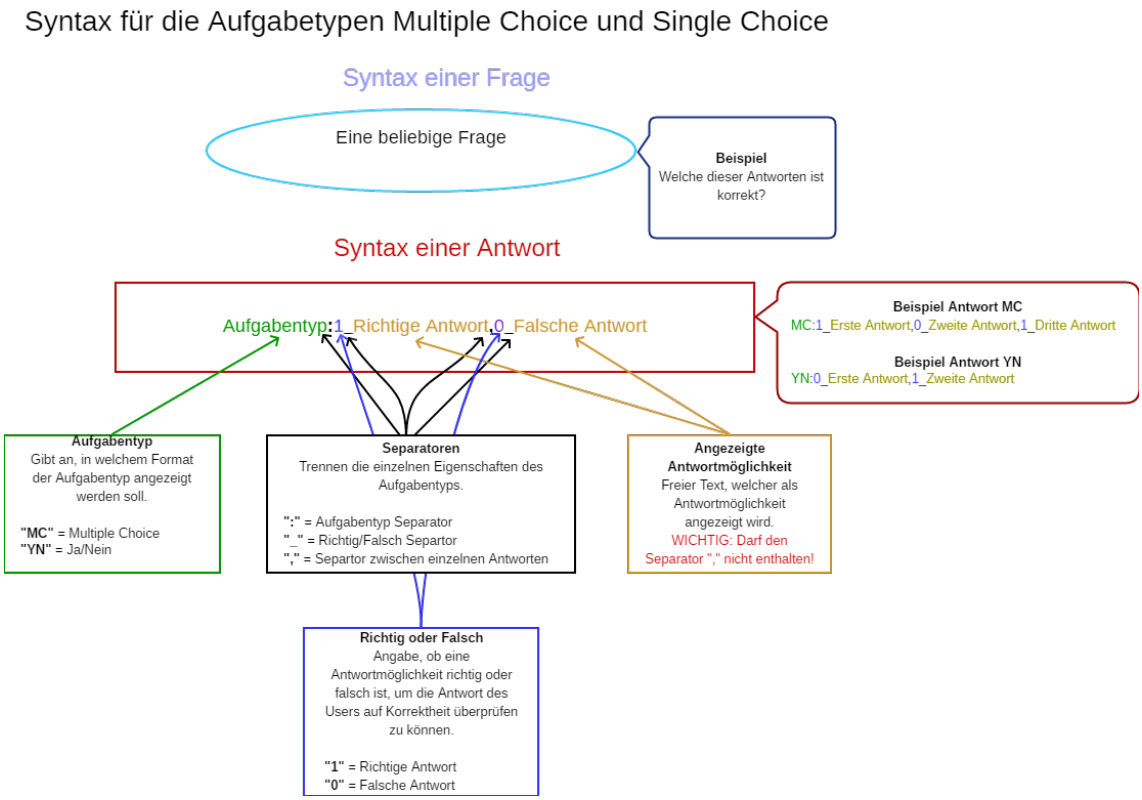


Abbildung 6: Syntax Aufgabenklasse Auswahl

möglichkeiten, der Index bzw. der Name der Lücke mit einem [voran gesetzt werden und mit einem] abgeschlossen werden. Mehrere Antwortlisten zu verschiedenen Lücken müssen mit einem ++ separiert werden.

Eine anschauliche und selbsterklärende Darstellung der Syntax und damit auch der Implementierung ist in Abbildung 7 zu sehen.

3.3.6 Buildmanagement

von Julian Schilling

Während der Entwicklung des Programmcodes fielen viele Aufgaben an, welche automatisiert werden konnten. Illustriert sei dies an einem Beispiel aus dem Bereich der Frontend-Entwicklung. Werden dort Änderungen im Code vorgenommen, müssen stets erneut einige oder alle der folgenden Operationen vorgenommen werden, um das Projekt zu aktualisieren:

- Übersetzung von TypeScript-Dateien nach JavaScript, da nur JavaScript von Browsern interpretiert werden kann (siehe 2.4.2)

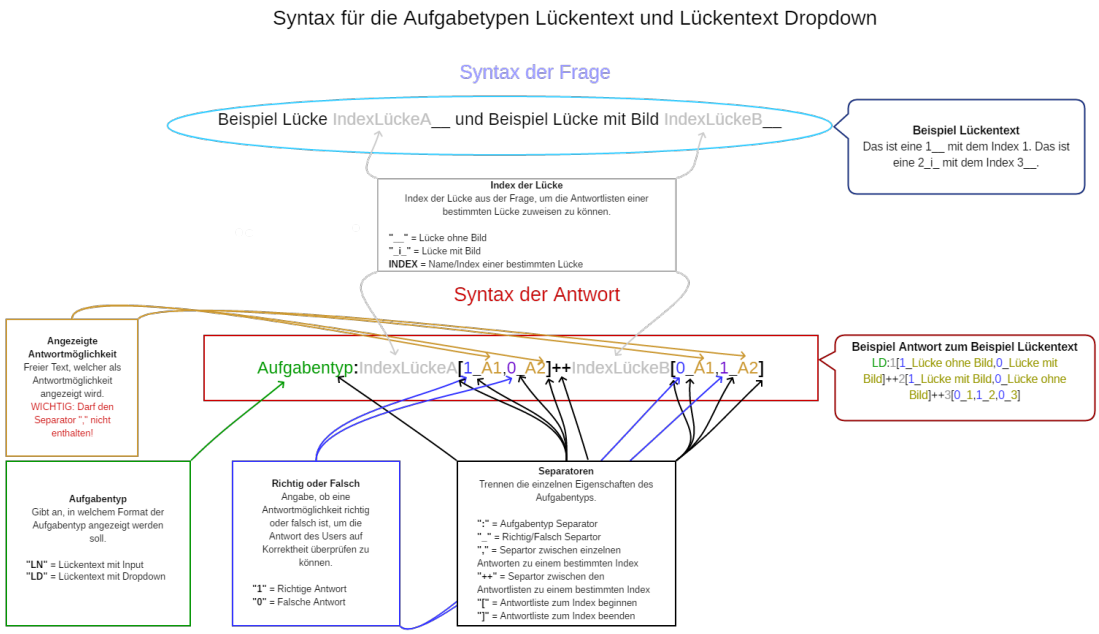


Abbildung 7: Syntax Aufgabenklasse Lückentext

- Übersetzung von SCSS-Dateien nach CSS, da nur CSS von Browsern interpretiert werden kann (siehe 2.4.2).
- Generierung von Klassen aus den zugehörigen Templates
- Lokales Hosten der Website zu Testzwecken
- Neuladen der Seite bei Änderungen, um diese anzuzeigen

Viele dieser Aufgaben lassen sich über Entwicklungsumgebungen automatisieren. Es sich jedoch an, das Verhalten auf allen Rechnern gleich zu gestalten, das Vorgen verschiedener Entwicklungsumgebungen voneinander abweichen kann. Hierzu wurde das Node-Paket **Gulp** genutzt. Gulp beschreibt sich selbst als „einen Werkzeugkasten, um schmerzhafte oder zeitraubende Aufgaben in Ihrem Entwicklungs-Workflow zu automatisieren, sodass Sie mit dem herumspielen aufhören und etwas bauen können“ [Gul17].

Um dies umzusetzen, bietet Gulp die Möglichkeit, Tasks („Aufgaben“) zu erstellen, welche Dateien unter anderem lesen, editieren, generieren und kopieren können. Ferner können sich Tasks gegenseitig aufrufen und asynchron mehrere Tasks nebeneinander abarbeiten.

Es wäre nicht zielführend, alle Tasks an dieser Stelle aufzulisten, auch weil einige primär von den größeren Tasks aufgerufen werden und für sich genommen nicht

sinnvoll nutzbar sind.. Es soll daher nur ein Überblick darüber gegeben werden, was der `!!SERVE!!`-Task tut, wenn er über IDE oder Terminal (mit dem Befehl `gulp !!SERVE!!`) ausgeführt wird. Es ist zu beachten, dass sämtliche kompilierten und kopierten Dateien als Zielverzeichnis den Ordner `.tmp` im Projektverzeichnis haben. Dies geschieht, um die zur Auslieferung gedachten Dateien möglichst von den Quelldateien des Ordners `src` getrennt zu halten und somit nicht nur Übersicht zu gewährleisten, sondern auch das spätere Ausliefern einfacher zu machen.

1. Das `.tmp`-Verzeichnis wird vollständig gelöscht, um einen sauberen Stand zu gewährleisten.

2. Sämtliche generierten Klassen werden neu generiert, um Änderungen am Template zu übernehmen und Änderungen an der Klasse selbst zu überschreiben.

3. Die SCSS-Dateien werden in CSS konvertiert.

4. HTML-, Jade- und CSS-Dateien sowie Bilder, die Konfigurations-Datei und die für das Frontend wichtigen JSPM-Pakete (siehe Abschnitt 3.3.1) werden in den `.tmp`-Ordner kopiert.

5. TypeScript-Dateien werden in JavaScript-Dateien übersetzt.

6. Die Komponenten Frontend, CoMa und Backend werden auf den in der Konfigurations-Datei festgelegten Ports gestartet.

7. Der Task läuft hiernach im Hintergrund weiter und überwacht sämtliche Dateien auf Änderungen. Geänderte Dateien werden sofort in den `.tmp`-Ordner kopiert/kompiliert und ein Browser, welcher gegenwärtig die Seite betrachtet, erhält die Anweisung, die Seite sofort neu zu laden, um Änderungen darzustellen.

Dies alles erleichtert die Entwicklung. Gerade das automatische erneute Laden der Webseite nach Änderungen nimmt einiges an Unbequemlichkeit.

3.3.7 Continuous Delivery

von Julian Schilling

Continuous Delivery wird auf der gleichnamigen Webseite wie folgt beschrieben:

> Continuous Delivery ist die Fähigkeit, Änderungen aller Art – einschließlich neuer Features, Änderungen der Konfiguration, Entfernen von Fehlern und Experimenten – in die Produktionsumgebung oder die Hände von Nutzern zu bringen, sicher, schnell und auf nachhaltige Weise. [Git17]

Vereinfacht gesagt bedeutet dies, dass es nicht gewünscht ist, Änderungen an einem Projekt erst auf umständliche Weise der Produktionsumgebung zuzufügen. Stattdessen soll dies automatisch und einheitlich erfolgen. Bei IN4ALLbedeutet dies, dass nicht jedesmal von Hand Änderungen an Frontend, CoMa und Backend auf den Server kopiert werden sollen, sondern die Änderungen automatisch vom Webauftritt reflektiert werden sollen. Bugfixes und neue Features sollen sich so schnellstmöglich und sicher auf den online verfügbaren Webseiten wiederfinden.

Hierzu wurde das Programm Bamboo genutzt, welches ebenso wie Bitbucket, Jira und Confluence von Atlassian entwickelt und vertrieben wird. Bamboo ermöglicht es, bei Änderungen auf dem Master-Branch diese zu erkennen und die aktuelle Version der Webseite auf den Webserver auszuliefern. Dies geschieht in zwei Schritten:

1. Zunächst erfolgt der **Build**-Prozess, also das „Zusammenbauen“ der auszuliefernden Elemente. Die Quelldateien des Projektes werden übersetzt und in eine Form gebracht, welche der Server an einen Webbrowser ausliefern kann.

2. Diese Elemente (von Bamboo „Artefakte“ genannt) werden dann im zweiten Schritt, dem **Deploy**-Prozess (das „Ausliefern“) über den Webserver ausgeliefert. Das Frontend-Artefakt wird beispielsweise in einen Ordner kopiert, welcher von einem Apache online zur Verfügung gestellt wird.

Beide Prozesse sollen hier kurz detaillierter dargestellt werden. Durch das Verknüpfen von Bamboo und Bitbucket war ein Einrichten eines Triggers („Auslösers“) kein Problem, der den Build-Prozess bei jedem neuen Push im Master-Branch auslöst. Dieser handelte dann nacheinander folgende Schritte ab:

1. Der aktuellste Stand aus dem Git-Repository wurde heruntergeladen. Hierbei handelte es sich nicht nur um eine Aktualisierung (Pull), sondern ein vollständig neues Herunterladen des aktuellsten Standes (Clone).

2. Bamboo installiert alle JavaScript-Pakete, welche CoMa und Backend später benötigen.[5]

[5] Bash-Befehl: `npm install -production`

3. Diese Pakete werden in einem zweiten Ordner per symbolischer Verknüpfung verlinkt.[6] Der Grund hierfür wird später noch deutlicher werden.

4. Bamboo installiert über den Paketmanager `npm` die restlichen benötigten NPM-Pakete von Backend und CoMa sowie die vom Frontend benötigten JSPM-Pakete.[7]

5. Die Datei `server_config.ts` überschreibt die Datei `config.ts`, da letztere nur die Konfiguration zum lokalen Testen enthält.[8]

6. Zuletzt wird der Gulp-Task `build-dist` ausgeführt. Dieser entspricht dem in Abschnitt 3.3.6 vorgestellten Task `!!SERVE!!`, startet jedoch weder die Server (Frontend, Backend und CoMa) noch überwacht es die Quelldateien auf Änderungen. Zusätzlich werden die erzeugten JavaScript-Dateien des Frontends zu einer einzigen Datei zusammengefasst. Dies erspart dem Browser im laufenden Betrieb eine Vielzahl von Requests an den Server, was die Last reduziert und die Geschwindigkeit der Webapplikation durch niedrigere Ladezeiten erhöht. Ferner ist die Datei von allem für Browser unwichtige Dinge (z.B. Leerzeichen und Zeilenumbrüche) bereinigt, was sie für Menschen fast unlesbar macht, jedoch die Datei zusätzlich verkleinert.

7. Es werden vier Artefakte erstellt:

   (a) **App** beinhaltet das Frontend.

   (b) **Backend** beinhaltet das Backend.

   (c) **CoMa** beinhaltet das Content Management.

   (d) **npm_modules** beinhaltet genau die NPM-Pakete, welche von Backend und CoMa benötigt werden.

Nach Abschluss des Build-Prozesses stehen Bamboo nun also vier Artefakte zur Verfügung, welche nicht mehr weiter verändert werden müssen. Dies löst automatisch den Deploy-Prozess aus, welcher die Artefakte an die richtigen Stellen kopiert und die nötigen Schritte vornimmt, damit Frontend, Backend und CoMa von Webbrowsern abrufbar sind.

[6] Bash-Befehl: `cp -al node_modules node_modules_hardlink`

[7] Der Bash-Befehl lautet `npm run install-server` Der Befehl `install-server` ist in der Datei `package.json` definiert und ruft `npm install` und `jspm install` auf.

[8] Bash-Befehl: `mv srcserver_config.ts srcconfig.ts`

Das hier beschriebene Vorgehen bezieht sich explizit auf den von der PG genutzten Server. Bei eigener Serverkonfiguration müssen gegebenenfalls die Ordnerpfade angepasst werden. Der Prozess führt folgende Schritte aus:

1. Der alte Inhalt des Frontend-Ordners wird gelöscht[9] und das in Schritt 7a des Build-Prozesses generierte Artefakt `App` im Ordner `/var/www/in4all-pg.de`[10] publiziert.

2. Anders als das Frontend werden Backend und CoMa nicht direkt über Apache ausgeliefert, sondern sind aktive Node-Prozesse. Diese Prozesse stellen Backend und CoMa an den vorkonfigurierten Ports zur Verfügung, welche von Apache aufgegriffen und weitergeleitet werden (nähere Informationen später in Kapitel 4.1.3). Diese müssen folglich gestoppt werden, ehe neue Versionen ausgeliefert werden können. Im Projekt wurde hierzu das selbstgeschriebene Bash-Script `forever_stopall` genutzt, welches die über das Node-Programm `forever` gesteuerten Node-Prozesse stoppt[11]. Das Script findet sich in Anhang A.3.2.

3. Die alten Ordner werden gelöscht[12] und das in Schritt 7d des Build-Prozesses generierte Artefakt `node_modules` nach `/opt/in4all` publiziert. Die in den Schritten 7b und 7c des Build-Prozesses generierten Artefakte `Backend` und `CoMa` werden in den Unterordner `/opt/in4all/server_applications` publiziert. Es ist zum Stand dieses Berichtes wichtig, dass diese sich genau eine Ordnerebene *unterhalb* von node_modules befinden. Darüber hinaus können die Pfade beliebig angepasst werden, solange sich dies in den Skripten wiederspiegelt, welche in den Schritten 2 und 4 genutzt werden.

4. Zuletzt müssen die `www.js`-Dateien von Backend und CoMa gestartet als Node-Prozesse gestartet werden. Auch dies geschieht über `forever` mithilfe eines Bash-Scripts `forever_startall`, welches sich im Anhang A.3.1 findet[13].

Nachdem der Deploy-Prozess abgeschlossen ist, findet sich die Webseite unter den im Apache konfigurierten Subdomains des Servers. Von der PG wurden `http://`

[9] `rm -r /var/www/in4all-pg.de/*`

[10] Der Ordner, welcher von Apache ausgeliefert wird. Bei abweichender Serverkonfiguration muss dieser Pfad angepasst werden.

[11] Sollten andere Pfade als die beispielhaft in Schritt 3 genannten genutzt werden, muss diese Änderung auch im Skript vorgenommen werden.

[12] `rm -r /opt/in4all/*`

[13] Sollten andere Pfade als die beispielhaft in Schritt 3 genannten genutzt werden, muss diese Änderung auch im Skript vorgenommen werden.

`in4all-pg.de` für das Frontend, `http://api.in4all-pg.de` für das Backend und `http://coma.in4all-pg.de` für das CoMa genutzt.

3.3.8 Produktion von Videosequenzen

von Fabian Pawlowski

Im folgenden Kapitel wird der Aufbau der Hardware und die eingesetzte Software, sowie der verwendete Workflow zum Erstellen der Videoinhalte beschrieben.

Hard- und Software Um die Inhalte bestmöglich zu vermitteln, hat sich die Projektgruppe an bestehenden MOOCs wie den bereits erwähnten Systemen von Udacity und dem MIT orientiert. Daraus ergibt sich ein bestimmtes Setup aus Hard- und Software, welches in diesem Kapitel dargelegt werden soll. Außerdem werden einige technische Aspekte – wie die Einstellungen der Effekt-Software zur Darstellung der Handaufnahme – beschrieben.

Das Ziel für die Videos war, dass zum einen der gesprochene Text auf dem Bildschirm verfolgt werden kann und zum anderen die Hand des Dozenten, der die Lektion aufschreibt, sichtbar ist. Die Kombination aus Text und sichtbarer Hand sollte die Wahrnehmung des Gesprochenen verbessern und gleichzeitig eine persönliche Verbindung zum Dozenten aufbauen. Daraus ergibt sich der folgende Hardware-Aufbau:

Grafiktablet Das Grafiktablett ist die Kernkomponente des Aufbaus. Es wird verwendet, um die Lektionen, welche von einer Software aufgenommen werden, zu verfassen. Die Projektgruppe hat sich für das Wacom Cintiq 27QHD entschieden. Die Entscheidung wurde durch zwei Faktoren beeinflusst: Zum einen liefert das Tablet eine Auflösung von 2560x1440 (QHD). Die Zielauflösung der Videos war mit 1920x1080 wesentlich geringer als die des Tablets. Dadurch ergaben sich für die PG viele Möglichkeiten, die Folien der Lektionen zu gestalten und in der Nachbearbeitungsphase den Anforderungen der Lektion anzupassen.

Zum anderen liefert der Eingabestift mit 2048 Druckpunkten ein sehr präzises Eingabetool. Diese erwies sich vor allem für Zeichnungen als Vorteil. Hinzukommend war die Einarbeitungszeit der verschiedenen Verfasser dadurch sehr gering. Außerdem biete ein großes Grafiktablett natürlich eine größere Arbeitsfläche, was bei späteren Videos dafür sorgt, dass das Verhältnis zwischen der aufgenommenen Handbewe-

gungen und der Präsentationsfläche für den Zuschauer angenehm ist und nicht zu viel Fläche durch die Hand des Dozenten verdeckt wird.

Kamera Um die Hand während des Schreibens aufzunehmen sollte eine Kamera über dem Tablet aufgehängt werden. Der Plan, diese an der Decke aufzuhängen, wurde auf Grund einer fehlenden Möglichkeit zur sicheren Befestigung an der Decke verworfen. Alternativ wurde die Kamera an einem Stativ befestigt und ca. eineinhalb Meter über dem Tablet aufgestellt.

Die Projektgruppe entschied sich für die Canon EOS700D. Zum Aufnehmen der Hand des Dozenten kann im Grunde jede Kamera verwendet werden, welche eine ausreichend hohe Auflösung bietet. Diese sollte jedoch mindestens 1920x1080 betragen, da beim späteren Schnitt Teile des Kamerabildes abgeschnitten werden müssen. Außerdem sollte eine Kamera gewählt werden, die eine zum Wacom-Tablet passende Bildwiederholfrequenz besitzt. Das bedeutet, dass die Bildwiederholfrequenz des Videos und der Bildschirmaufnahme ganzzahlige Vielfache voneinander sein müssen. Das ist wichtig, da die Aufnahmen später miteinander kombiniert („übereinander gelegt“) werden müssen. Haben die Videos dann nicht die gleiche Bildwiederholfrequenz (zum Beispiel beide 25 Bilder pro Sekunde) oder ein ganzzahliges Vielfaches der Bildwiederholfrequenz (zum Beispiel 25 und 50 Bilder pro Sekunde) ist das resultierende Video asynchron.

Computer In der Erprobungsphase des Projekts wurde festgestellt, dass die Nachbearbeitungszeit der Videos durch einen leistungsstarken Computer (Intel 4-Kernprozessor und 32GB RAM) drastisch reduziert werden kann. Auf Grund der Skalierungsoperationen (Veränderungen der Auflösung eines Videos) wurde vor allem eine leistungsstarke Grafikkarte (GTX970 mit 4GB RAM) erforderlich. Damit konnte die Zeit zum Bearbeiten eines Videos mit einer Länge von ungefähr 3 Minuten von fast 180 auf 30 Minuten reduziert werden.

Außerdem wurde für die Aufnahme eine spezielle PCI-Express Karte („Blackmagic Mini Recorder“) angeschafft. Die Verwendung der Karte musste nach ersten Tests eingestellt werden, da die verwendete Kamera über ihren HDMI-Ausgang das Bild nur in 1080i59,94 ausgeben kann und keine Software gefunden wurde, welche das Screen-Capturing des Grafiktablets ebenfalls in 59,94 oder 23,976 Bildern pro Sekunde gewährleisten konnte. Dadurch kam es zu der oben erwähnten Asynchronität der beiden Aufnahmen.

Mikrofon und Beleuchtung Des Weiteren wurde ein Mikrofon und Mittel zur Beleuchtung angeschafft. Die Beleuchtungsmittel sollten eine bessere Ausleuchtung der Handaufnahme gewährleisten. Das Mikrofon („Auna MIC-900 USB“) sollte eine möglichst gute Sprachqualität gewährleisten. Zusätzlich wurde ein Pop-Filter verwendet, um das Gesprochene möglichst gleichmäßig aufzunehmen.

Software Neben dem Hardware-Aufbau hatte in diesem Projekt auch die Software eine bedeutende Rolle. Die Handaufnahmen mit der Kamera und die Bildschirmaufnahme mussten zusammengeschnitten werden. Außerdem musste ihre Auflösung an die gewünschte Auflösung angepasst werden und die rohen Aufnahmen waren viel zu groß (10 Minuten Video haben ungefähr 50 GB), um sie auf einer Webseite zur Verfügung zu stellen.

Im Grunde gibt es bei der Wahl der Software keine Einschränkungen, solange sie die benötigten Funktionen zur Verfügung stellt. Die Projektgruppe hat sich für die Creative Cloud von Adobe entschieden. Die relevante Software aus diesem Paket sind: After Effect, Premiere Pro, Media Encoder und Photoshop, jeweils enthalten in der Version der Creative Cloud 2016 (CC2016). After Effect wurde verwendet, um die rohen Videodateien übereinander zu legen und den Bildausschnitt anzupassen. Mit Premiere Pro wurde das Video anschließend geschnitten (Fehler oder unnötige Sequenzen wurden entfernt) und vertont. Mit Hilfe von Photoshop wurden Grafiken erstellt. Der Media Encoder hat zwischen den einzelnen Programmen die Rendering-Tasks verwaltet und ausgeführt.

Des Weiteren wurde eine Software für die Bildschirmaufnahme benötigt. Die Projektgruppe hat Active Presenter 6.1.0 und FastStone Capture 5.3 verwendet.

Evaluation Zum Abschluss des Projekts gilt zu bewerten, ob die angeschafften Mittel optimal verwendet wurden oder Fehlinvestition getätigt wurden. Das Wacom-Tablet erwies sich nach den Erfahrungen der Gruppe als gute Wahl für die Aufgaben. Es stellt preislich zwar kein Einsteiger-Modell dar, bietet gegenüber der günstigeren Alternative aber vor allem den Vorteil der höheren Auflösung. Dies hat sich während der Aufnahmen und bei der Bearbeitung als sehr praktisch erwiesen. Die Anschaffung des Computers zur Bearbeitung der Videos war im Rahmen. Hier wurde mit einem akzeptablen Budget eine Rechenzeit von ungefähr 40 Minuten pro Lektion. Durch ein höheres Budget hätte die Rechenzeit verkürzt werden können, aber das Preis-/Leistungsverhältnis erwies sich als gut.

Die Anschaffung der Canon-Kamera erwies sich nicht als Fehler, führte aber in Kombination mit der Blackmagic Express Karte zu einer problematisch Konstellation. Im Vorfeld wurde bereits angesprochen, dass bei der Kombination von Video-Dateien die Framerate (Bildwiederholfrequenz) eine wichtige Rolle spielt. Werden zwei Videos mit unterschiedlichen Framerates (zum Beispiel 25 und 30 Bilder pro Sekunde) übereinander gelegt und abgespielt, entsteht pro Sekunden eine Diskrepanz von 5 Bildern. Bei kurzen Aufnahmen kann es sein, dass das gar nicht auffällt. Bei längeren Aufnahmen merkt man hingegen deutlich, dass diese asynchron sind. Diese Situation ist im Laufe des Projekts entstanden. Die Bildschirmaufnahme erfolgt mit 25 Bildern pro Sekunde. Das Signal der Canon-Kamera, welches von der Blackmagic Express Karte aufgenommen wurde, lieferte 30 Bilder pro Sekunde. Unglücklicherweise konnte die Gruppe im weiteren Verlauf des Projekts weder eine Software finden, die eine Bildschirmaufnahme mit 30 Bildern pro Sekunde durchführte, noch die Bildrate des HDMI-Ausgangs der Canon auf 25 (oder 50) Bilder pro Sekunde setzen. Aus diesem Grund wurde die Blackmagic Express Karte im weiteren Verlauf des Projekts nicht mehr verwendet und stattdessen direkt auf die SD-Karte der Canon-Kamera aufgenommen. Aus diesem Blickwinkel stellt die Anschaffung der Blackmagic Karte eine Fehlinvestition dar. Wären die Probleme mit der Bildwiederholrate behoben worden (durch eine geeignete Software oder einen andere Ausgabemodus der Kamera), wäre die Aufnahme über die Blackmagic Karte die bevorzugte Variante gewesen.

Workflow der Produktion In diesem Kapitel wird der Prozess der Aufnahme beschrieben. Das Hauptaugenmerk liegt auf den verwendeten Effekten bei der Nachbearbeitung.

Vor der Aufnahme Bevor mit den Aufnahmen begonnen wird, müssen einige Punkte sichergestellt werden. Zum Einen muss darauf geachtet werden, dass die aufzunehmende Szene, also die Aufnahme der Handeingabe des Dozenten gut ausgeleuchtet ist. Dazu bietet die interne Software der Kamera Möglichkeiten der Überprüfung, indem überbelichtete Bildbereiche eingefärbt werden. Somit lässt sich die Belichtung der Szene kontrollieren.

Außerdem muss sichergestellt werden, dass die Ausrichtung der Kamera etwa parallel zum Grafiktablett ist. Dafür hat die PG keine automatisierte Methode gefunden. Die Ausrichtung der Kamera wurde nach Augenmaß erledigt, da eine gewisse Verzerrung im Nachgang bei der Nachbearbeitung noch ausgeglichen werden kann. Um diesen Ausgleich zu erledigen, wird vor dem Beginn der Aufnahme auf dem Bildschirm des

Grafiktabletts ein Rasterbild eingeblendet, welches das Beheben der Verzerrung im Nachgang erst ermöglicht.

Ist die Szene ausreichend ausgeleuchtet, die Kamera etwa parallel des Grafiktabletts ausgerichtet und das Rasterbild angezeigt, kann die Aufnahme gestartet werden. Dazu wird jeweils eine Aufnahme der Bildschirmanzeige des Tabletts und eine Aufnahme der Kamera gestartet. Im Nachgang müssen die beiden getrennten Videos weiter bearbeitet werden, um nach dem Verschmelzen der Videos das gewünschte Resultat zu liefern.

Nachbereitung der Aufnahmen Das Ergebnis der Aufnahme sind zwei Video-Dateien. Eine Datei mit der Bildschirmaufnahme und die zweite Aufnahme mit dem Kamera-Bild, welches die Bewegungen der Hand beim Schreiben aufgenommen hat. Die ersten Schritte der Nachbearbeitung finden in After Effect (AE) statt. Beide Dateien werden in einem neuen Projekt zu einer Komposition zusammengefasst. Nachdem die Hand-Aufnahme um 180 Grad gedreht wurde (Effekt „Transformieren“ in AE) – dies ist auf Grund der Aufnahmekonstruktion erforderlich – werden Hand- und Bildschirm-Aufnahme „übereinander gelegt“. Der erste Schritt sollte das Angleichen des Bildes sein, sodass beide Aufnahmen synchron laufen. Besonders hilfreich ist dafür, das „Framelevel“ der Aufnahmen soweit zu erhöhen, dass in der Leiste jedes Bild einzeln angezeigt wird. Dadurch wird eine Bild-für-Bild Anpassung leichter. Mit Hilfe der Funktion „Eckpunkte verschieben“ werden anschließend die Aufnahmen verzerrt, bis die Bilder deckungsgleich sind. Am besten erkennt man das an der Schrift. Hat die Schrift einen Schatten oder ist sogar doppelt zu sehen, müssen die Aufnahmen weiter angeglichen werden. Dieser Schritt musste für jede Aufnahme durchgeführt werden und kostet viel Zeit.

Nun liegen die Aufnahmen übereinander und laufen synchron ab. Der nächste Schritt ist die Anwendung verschiedener Effekte, um die Hand-Aufnahme auf die Hand zu beschränken und sonst nur die Bildschirm-Aufnahme zu sehen.

Im Laufe des Projekts wurden zwei Methoden verwendet und optimiert, um die Hand bestmöglich darzustellen. Beide werden im Folgenden erklärt. Die zweite Methode liefert ein besseres Ergebnis (bessere Auflösung und Originaltreue der Hand) als die Erste, hat aber auch einen höheren Einstellungsaufwand. Dabei wird – ausgehend von dem bisherigen Vorgehen – wie folgt vorgegangen. Es ist zu beachten, dass die Hand-Aufnahme in der Video-Liste von After Effects über der Bildschirm-Aufnahme steht. (Die Hierarchie der Dateien spielt eine Rolle.) Wurden beide Videos

übereinander gelegt, wird die Hand-Aufnahme dupliziert (auswählen und „STRG-D“). Nun sind in der Liste zwei Versionen der Hand-Aufnahme enthalten. Auf die obere wird nun der Effekt „Master Sättigung“ auf -100 gesetzt (in AE: „Effekte → Farbton / Sättigung → Master-Sättigung“). Dadurch wird das Bild in Graustufen dargestellt. Anschließend wird eine Tonwertkorrektur (Effekt → Tonwertkorrektur) durchgeführt, um das Bild in schwarz-weiß darzustellen. Nun wird anhand eines Histogramms der Farbbereich angezeigt. Mit Hilfe eines Reglers unter dem Histogramm (links weiße (helle) und rechts schwarze (dunkle) Pixel) kann der schwarze Farbbereich verkleinert werden, bis nur noch die hellen Pixel angezeigt werden. Damit dies nun als Maske benutzt werden kann, muss ein weiterer Effekt („Umkehren“) angewendet werden. Wird eine Aufnahme als Maske verwendet, wird in der Regel der schwarze Bereich ausgeschnitten. Durch das „Umkehren“ wird der Bereich, der aus dem Video verwendet werden soll, umgekehrt. Der schwarze Anteil wird weiß, der weiße wird schwarz. Somit kann diese Aufnahme als Maske verwendet werden. Nun wird auf die zweite Hand-Aufnahme in der Video-Liste der Effekt „Bewegte Maske (Trackmatte) → Lunamaske mit Modus (Mode): Füllmethode Multiplizieren“ angewandt. Nun wird die erste Hand-Aufnahme als Maske für die zweite verwendet und die Anteile, die die bewegte Hand ausmachen, werden angezeigt.

Die zweite Methode ist einfacher einzustellen und erfordert weniger Effekte, führt aber auch zu einem unsauberen Ergebnis. Dafür wird – bevor die Videos übereinander gelegt werden – eine weitere Komposition (zum Beispiel mit dem Namen „Hand“) angelegt. In diese Komposition wird zwei Mal die Hand-Aufnahme eingefügt. Für das erste Video wird nun der Modus „Luminanzsilhouette“ und für das zweite Video der Modus „Normal“ ausgewählt. Nun wird diese Hand-Komposition zusammen mit der Bildschirm-Aufnahme in eine weitere Komposition eingefügt. Nun wird für die Bildschirm-Aufnahme der Modus „Multiplizieren“ ausgewählt. Damit wird der gewünschte Effekt, dass von der Hand-Aufnahme nur die Hand über der Bildschirm-Aufnahme angezeigt wird, erreicht. Es kann mit der oben beschriebenen Bearbeitung (Synchronisieren und Eckpunkte anpassen) fortgefahren werden.

Abschließend muss noch der Ausschnitt des finalen Videos angepasst werden. Durch die hohe Auflösung der Bildschirm-Aufnahme kann relativ viel vom Rand wegfallen. Der letzte Schritt ist das Rendern des Videos, um die bisher angewandten Effekte und Veränderungen umzusetzen und in einer Videodatei zu vereinen. Das Rendern eines Videos von 5 Minuten hat mit der eingangs beschriebenen Hardware ungefähr 35 Minuten gedauert.

Abschließend muss noch der Ausschnitt des finalen Videos angepasst werden. Durch die hohe Auflösung der Bildschirm-Aufnahme kann relativ viel vom Rand wegfallen. Der letzte Schritt ist das Rendern des Videos, um die bisher angewandten Effekte und Veränderungen umzusetzen und in einer Videodatei zu vereinen. Das Rendern eines Videos von 5 Minuten hat mit der eingangs beschriebenen Hardware ungefähr 35 Minuten gedauert.

Mit dem resultierenden Video, kann dann die Vertonung und der weitere Schnitt des Videos erfolgen. Größere Pausen die bei der Aufnahme des Inhaltes entstanden sind werden herausgeschnitten. Dafür wurde bei der Aufnahme darauf geachtet, dass zwischen dem Schreiben von Inhalten die komplette Hand des Erstellers aus dem Sichtbereich des Videos heraus- und hereingeführt wurde, damit an diesen Stellen die Pausen gekürzt werden können und trotzdem ein natürliches Aufbauen des Inhaltes entstehen kann.

Parallel zu diesem Schneiden des Videos wird die Vertonung in kleinen Schritten vorgenommen, um störende Pausen beim Sprechen zu reduzieren. Zum Vertonen wurden die eingebauten Funktionen von Adobe Premiere CC verwendet.

Nachdem das Video somit also fertig geschnitten und vertont war, musste es ein letztes Mal gerendert werden. Dabei hat sich die PG an den Rahmen für Webvideos wie sie auf YouTube zu finden sind orientiert, um möglichst kleine Videodateien zu erhalten, um den Bandbreitenbedarf beim Abspielen der Videos zu verringern.

Evaluation Abschließend kann – was die Aufnahme und Bearbeitung der Videodateien angeht – festgestellt werden, dass die Gruppe im Laufe des Projekts stetig Verbesserungen gefunden und in den Prozess eingearbeitet hat. Gleichzeitig muss auch festgehalten werden, dass dieser Prozess weiteres Potenzial für Optimierungen beinhaltet. Die Bearbeitung von Videodateien ist, durch die Fülle an hochgradig umfangreichen Bearbeitungsprogrammen, sehr komplex und Bedarf viel Einarbeitungszeit.

3.4 Funktionale Skalierung

In den vorherigen Abschnitten wurde der technische Entwicklungsprozess beschrieben. Natürlich ist die Anwendung skalierbar und kann um weitere Funktionalitäten erweitert werden. Dieser Abschnitt erläutert anhand eines Beispiels, wie weitere fachliche Konzepte implementiert werden. Leser mit fortgeschrittenen Kenntnissen über

den MEAN-Stack sollten in der Lage sein, anhand dieser Beschreibung das Beispiel zu programmieren. Weiterhin sind Kenntnisse der Sprache Typescript und Konzepte von RESTful-Services notwendig.

3.4.1 Fachliches Szenario

Anhand des folgenden Szenarios wird eine mögliche Erweiterung der Software erläutert. Die Anforderungen lassen sich durch User-Stories [Wir17] beschreiben. Zu beachten ist, dass die User-Stories lediglich die fachlichen Anforderungen aus Sicht des Benutzers darstellen; sie enthalten keinerlei Informationen über die konzeptionelle bzw. technische Umsetzung.

User-Story-01 Als Tutor möchte ich in der Management-Oberfläche alle Schüler angezeigt bekommen.

User-Story-02 Als Tutor möchte ich neue Gruppen von Schülern anlegen können und diesen Schüler hinzufügen.

User-Story-03 Als Tutor möchte ich digitale Übungsblätter erstellen und diesen den Schülern einer Gruppe zur Verfügung stellen.

User-Story-04 Als Schüler werden mir nach dem Login neue Übungsblätter angezeigt.

User-Story-05 Als Schüler möchte ich Übungsblätter bearbeiten können.

User-Story-06 Als Tutor sehe ich beim Login, dass ein Schüler Lösungen bearbeitet hat und kann die Ergebnisse einsehen.

3.4.2 Konzeptionelle Erweiterungen

von Christian Everke

Zunächst sind für die oben genannten User-Stories konzeptionelle Überlegungen erforderlich. Vorab sei festgestellt, dass in den User-Stories ein neuer User *Tutor* verwendet wird, der bislang nicht im System vorhanden ist. Das Hinzufügen dieser neuen Benutzerrolle muss zur Entwicklungszeit passieren; derartige Ergänzungen zur Laufzeit durchzuführen sind bislang nicht möglich bzw. Ansätze zur Realisierung während der Laufzeit werden erst in Abschnitt 3.4.3 angerissen.

Wie den User-Stories zu entnehmen ist, soll ein Tutor in der Lage sein, Gruppen von Schülern zu erstellen und diesen Gruppen Übungsblätter zuzuweisen. Eine entsprechende Anpassung des Datenmodells aus Abbildung 3 ist daher notwendig. Bevor Änderungen vorgenommen werden, sind unbedingt genaue fachliche Anforderungen zu analysieren, die nicht zwangsläufig nur aus den oben genannten User-Stories hervorgehen. Im Allgemeinen ist ein umfangreiches *Requirements Engineering* [PR15] für die Umsetzung derartiger neuer Funktionalitäten erforderlich, nicht zuletzt um Kosten für (fachlich betrachtet) falsche oder suboptimale Realisierungen zu vermeiden. Da die hier beschriebene funktionale Skalierung lediglich technische Aspekte fokussiert, genügt es an dieser Stelle, exemplarisch mögliche konzeptionelle Änderungen zu erwähnen. Es sei betont, dass es viele Möglichkeiten gibt, die User-Stories umzusetzen.

Eine Option ist es, das Datenmodell wie folgt zu modifizieren: Aus Abbildung 3 ist erkennbar, dass die Klassen *USER* und *GROUP* bereits vorhanden sind. Natürlich es möglich und auch sinnvoll, diese Klassen zur Umsetzung von User-Story-02 weiterhin zu verwenden, damit Tutoren Schülern als Instanzen von *USER* einer oder mehreren Gruppen zuzuordnen können. Es ist zu überprüfen, ob die Relationen zwischen den beiden Klassen *USER* und *GROUP* beibehalten werden können. Die aktuelle Modellierung sieht es vor, dass eine Gruppe mehrere Benutzer enthält, ein Benutzer aber nur einer Gruppe zugeordnet werden kann (Kardinalität $1 : n$). Ob diese Modellierung zielführend ist, ist Abhängig von den detaillierten fachlichen Anforderungen, die im Rahmen des oben genannten Requirements Engineering hervorgegangen sind. Betrachtet man eine Gruppe von Schülern als eine Art Schulklasse, denen verschiedene digitale Übungsblätter durch Tutoren bereitgestellt werden, so kann die bestehende Relation unverändert bleiben. Stellt die Relation den Sachverhalt an Hochschulen dar, wo mehrere Studenten mehreren Kursen (=Gruppe) zugeordnet sind, so ist eine Änderung zu einer $n : m$ Beziehung erforderlich. Ebenso ist es denkbar, eine Gruppenhierarchie abzubilden. In einem solchen Fall könnten Gruppen weitere Gruppen enthalten. Soll ein Übungsblatt mehreren Gruppen zugeordnet werden, kann dies über ein Superobjekt innerhalb der Gruppenhierarchie geschehen.

Das Hinzufügen von Übungsblättern ist auch auf verschiedene Weisen möglich. Es sei angenommen, dass ein Übungsblatt eine Sammlung bestehender Übungen (vgl. Klasse *EXERCISE*) ist, die auch entsprechend wiederverwertet werden. Da aber immer nur eine Teilmenge aller Übungen zu einem Übungsblatt gehören, ist eine neue Klasse zur Verwaltung der Übungsblätter notwendig. Das Modell wird um

die Klasse *WORKSHEET* erweitert und durch eine $n : m$-Beziehung in Relation zur Klasse *GROUP* gesetzt, damit ein Übungsblatt mehreren Gruppen und einer Gruppe mehrere Übungsblätter zugeordnet werden können. Attribute für die neue Klasse sind geeignet zu wählen. Neben den trivialen Attributen wie *ID*, *Name* und *Beschreibung* sind z.B. Attribute für Punkte oder Notenschema denkbar.

Die Implementierung der Änderungen des Datenmodells müssen in der Datei `Domain.js` vorgenommen werden. In dieser Datei ist das gesamte Schema durch die JavaScript-Object-Notation (JSON) einschließlich Validierungskriterien dargestellt. Nur hier sind Änderungen erforderlich, da auf Basis dieser Datei die konkreten Klassen auf Basis von Templates zur Kompilierungszeit erzeugt werden (vgl. 3.3.1). Damit eine Verbindung zu den API-Endpunkten (siehe Abschnitt 3.4.3) hergestellt werden kann, muss die Klasse `RestService.ts` modifiziert werden. Die Art und Weise, wie diese Änderungen durchzuführen sind, kann durch die bereits implementierten http-Anfragen für andere Klassen abgeleitet werden.

3.4.3 Erweiterungen im Backend

Dmytro Marchenko

In diesem Abschnitt werden die notwendigen Erweiterungen im Backend vorgestellt. Dafür wird zunächst erläutert, wie man allgemein ein neues Schema erstellen kann und welche Services dabei erweitert werden sollen. Im Anschluss werden auf der Basis von den konzeptionellen Überlegungen aus dem Abschnitt 3.4.2 die notwendigen Erweiterungen für die Anwendung der User-Stories vorgestellt.

Erstellung eines Schemas und Erweiterung der Services. Im Folgenden werden die typischen Schritte bei der Erstellung neuer Klassen vorgestellt. Es wird erläutert, wie ein neues Schema erstellt werden soll, sowie welche Erweiterung von grundlegenden Services dabei notwendig sind. Das Ziel dabei ist zu zeigen, wo man genau die Änderungen vornehmen muss. Die fortgeschrittenen Kenntnisse über den MEAN-Stack, die Sprache Typescript, Konzepte von RESTful-Services sind vorausgesetzt. Außerdem soll man darauf beachten, dass dieser Prozess wird zunächst allgemein beschrieben, um die überflüssigen Wiederholungen der Erstellungsschritte im Lauf des Abschnittes zu vermeiden.

Schritt 1: Ein Schema definieren. Im Ordner `model` (Abbildung 8) soll eine Datei für ein neues Schema erstellt werden (`*.ts`, beispielsweise `Worksheet.ts`).

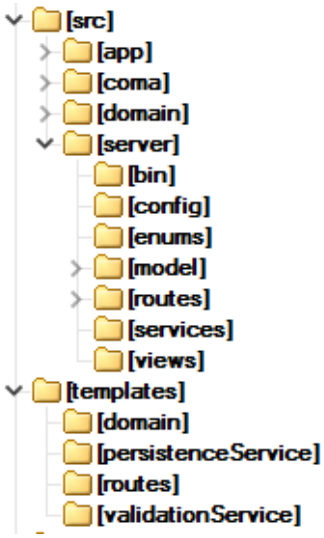


Abbildung 8: Verzeichnisbaum von Backend

In dieser Datei soll die Struktur des Schemas (notwendige Felder) gemäß des entwickelten Datenmodells definieren werden.

Zur Darstellung einer Beziehung zu den Objekten anderer Klasse soll zunächst in beiden Schemata der Beziehung ein neues Feld hinzugefügt werden. Hierbei kann dieses Feld entweder nur eine Referenz speichern (Kardinalität $1:1$) oder mehrere (Kardinalität $1:n$). Nachdem diese Felder festgelegt wurden, sollen die Methoden mittels der *pre* und *post* Middleware ([mon17b]) für die Aktualisierung dieser Referenzen hinzugefügt werden. Diese Methoden sorgen für die Konsistenz und sollen jedes Mal bei der Erstellung oder der Veränderung dieses Feldes aufgerufen werden.

Schritt 2: *PersistencyService* erweitern. Der *PersistencyService* soll durch die grundlegenden Operationen wie Erstellung (*createClassName*), Veränderung (*updateClassName*) und Löschen (*deleteClassName*) eines Objekts der neuen Klasse erweitert werden. Dafür sollen diese Methoden in der Datei `persistenceService.class.ts` (im Ordner `src/server/services`, Abbildung 8) implementiert werden. Dabei ist wichtig, eine Namenskonvention zu verfolgen, um die Verwendung der Templates bei den nächsten Schritten zu ermöglichen.

Schritt 3: Erlaubte Rechte definieren. Für jede Rolle sollen Operationen definiert werden, die für diese Klasse durchgeführt werden können (siehe Abschnitt 3.3.2). Dafür soll die Methode *createDefaultRights* (Datei `databaseInitialisationService.ts` im Ordner `src/server/model/services`) erweitert werden, indem den Aufruf einer weiteren Methode *createRightIfNotExist* für jede

Rolle angelegt wird. Als Paramether nimmt die Methode *createRightIfNotExist* den Name einer Rolle (derzeit 'Manager' oder 'Student'), den Name neuer Klasse und die erlaubten Rechte (siehe Abschnitt 3.3.2).

Schritt 4: *ValidationService* erweitern. Für die Erweiterung des Services soll man in der Datei `domain.js` (im Ordner `templates/domain`, Abbildung 8) eine Beschreibung der Regeln hinzufügen. Dann soll der Gulp-Task *generate:validationService* gestartet werden. Dieser Task erstellt mittels der Templates den *ValidationService*.

Schritt 5: Routen erstellen. Zuletzt sollen auch die Routen hinzugefügt werden. Dafür soll man in der Datei `routes.js` (im Ordner `templates/routes`, Abbildung 8) die notwendige Information wie den Namen der neuen Klasse und den Namen den neunen Route hinzufügen. Schließlich soll man den Gupl-Task *generate:routes* starten. Dadurch werden die Routen mittels schon vorhanden Templates erstellt.

In den nächsten zwei Abschnitten werden die notwendigen Erweiterungen (Aufgrund der konzeptionellen Überlegungen aus Abschnitt 3.4.2) für die Umsetzung der ersten vier User-Stories vorgestellt.

Anlegen einer neuen *Tutor*-Rolle. In diesem Abschnitt werden zwei mögliche Vorgehensweisen für die Erstellung der Rolle vorgestellt. Zunächst wird gezeigt, wie man derzeit eine Rolle durch Erweitung des Schemas erstellen kann. Im Anschluss wird erläutert, was gemacht werden soll, um die Erstellung der Rollen und Rechte mittels der Anfragen aus der Management-Oberfläche zu ermöglichen.

Die möglichen Rollen sind derzeit mittels eines Enumerationstyps im *User*-Schema definiert. Um eine neue *Tutor*-Rolle anzulegen, soll entsprechend dieser Enumerationstyp durch eine neue Rolle erweitert werden. Außerdem sollen die Rechte für die neue Rolle definiert werden. Beispielsweise soll die Verwaltung der Übungsblätter einer *Tutor*-Rolle erlaubt werden. Dafür kann man Methoden aus *DatabaseInitialisationService* verwenden (Datei `databaseInitialisationService.ts` im Ordner `src/server/model/services`). Nach der Implementierung dieser Veränderungen ist es möglich, neue Rollen mittels einer Management-Oberfläche zu einem Nutzer zuzuweisen.

Zuvor wurde gezeigt, wie man mittels der kleinen Änderungen in den Code eine neue Rolle erstellen kann. Um die Erstellung der Rollen und Rechte mittels der Anfragen aus der Management-Oberfläche zu ermöglichen, soll man die Struktur und

Services erweitern. Zunächst soll man ein neues Schema für die Rolle erstellen und die grundlegenden Services wie *PersistencyService* erweitern (siehe „Erstellung eines Schemas und Erweiterung der Services“ oben). Dann wird es notwendig, die Routen für die *Recht*-Klasse zu erstellen (siehe „Erstellung eines Schemas und Erweiterung der Services“, Schritt 5). Anschließend soll man den Typ des *role*-Felds im *User*-Schema verändern (eine Referenz auf *Role*-Klasse anstatt Enumerationstyps).

Zuordnung eines Nutzers zu den mehreren Gruppen. Um die Zuordnung eines Nutzers zu den mehreren Gruppen zu ermöglichen, soll das *User*-Schema geändert werden, indem die Kardinalität der Beziehung von $1 : n$ auf $n : m$ verändert wird. Dafür soll die Speicherung mehrerer Referenzen auf die Objekte der *Group*-Klasse ermöglicht werden. Außerdem soll man auch die Methoden für die Aktualisierung dieser Referenzen erweitern. Anschließend soll auch die Methode für das Löschen des Nutzers in *PersistenceSevice* angepasst werden, indem dieser Nutzer aus der Gruppe entfernt wird. Die Anpassungen der anderen Services, wie beispielsweise RESTful (Routen), ist dabei nicht notwendig.

Im weiteren Verlauf des Kapitels werden Erweiterungen erläutert, die eine Anwendung der Übungsblätter (*WORKSHEET*) ermöglichen (User-Stories 5-7).

Gruppenhierarchie. Eine Gruppenhierarchie ermöglicht eine Zuweisung der Übungsblätter zu mehreren Gruppen. Für ihre Implementierung soll das *GroupSchema* durch die $1 : n$-Beziehung erweitert werden. Dafür muss ein zusätzliches Feld hinzugefügt werden, das die Referenzen auf die Objekte der *Group*-Klasse speichert. Außerdem sollen die Middleware und *PersistencyService* angepasst werden, um eine Verwaltung der Referenzen sicherzustellen (siehe „Erstellung eines Schemas und Erweiterung der Services“).

Erstellung der Übungblätter. Um eine Anwendung der Übungsblätter (*WORKSHEET*) zu ermöglichen, soll zunächst das Schema für diese Klasse erstellt werden (siehe „Erstellung eines Schemas und Erweiterung der Services“). Hierbei sollen außer trivialen Attributen wie *ID*, auch die $n : m$-Beziehung zu der *Group*-Klasse und die $n : m$-Beziehung zu der *Exercise*-Klasse eingepflegt werden. Dadurch wird ermöglicht, die Übungsblätter (*WORKSHEET*) einer Gruppe zuzuweisen sowie mehrere Exercises einem Übungsblatt (*WORKSHEET*) zuzuordnen.

3.4.4 Erweiterungen im CoMa

von Mesut Sahin

In diesem Abschnitt wird, anhand der oben definierten User-Stories, die Erweiterbarkeit des CoMas aus Sicht eines Programmierers betrachtet. Die Kenntnisse, die in der vorherigen Abschnitten dieses Kapitels erwähnt wurden, sind auch nötig für die funktionale Skalierung des CoMa-Bereichs. Die Erweiterung und Änderungen des CoMa-Bereichs sind in den Ordnern, welche in der Abbildung 5 aufgelistet sind, vorzunehmen.
Bevor die User-Stories abgearbeitet werden, wird ein allgemeines Vorgehen für die Erweiterung des CoMa-Bereichs vorgestellt. Der CoMa-Bereich wurde in eine Navigation mit den Titeln *Gruppen*, *User*, *Kurse*, *Kapitel*, *Lektionen* und *Aufgaben* unterteilt. Die Funktionalität des Tutors lässt sich im bestehenden System entweder als eine Rolle der Rest-Ressource User oder als eigenständige Rest-Ressource umsetzen. Gleiches gilt für die Funktionalität der Übungsblätter, welche entweder durch Erweiterung der bestehenden Syntax der Rest-Ressource exercises umgesetzt werden kann oder als eigenständige Rest-Ressource implementiert werden kann. Zunächst einmal wird aufgezeigt wie die Funktionalität *Tutor* als eine eigenständige Rest-Ressource anzulegen ist. Die Umsetzung der Übungsblätter auf diese Weise ist analog zur Umsetzung der Rest-Ressource *Tutor*. Die folgende Auflistung enthält die Schritte, um diese Erweiterung zu ermöglichen.

Schritt 1: Templates erstellen Zunächst einmal wird in dem Ordner views (Abbildung 5) ein neuer Ordner (bsp. mit den Namen tutors) angelegt. Die Rest-Ressource *Tutor* kann ohne Abhängigkeiten ausgeführt werden, da die Anwendungsfälle der Ressource Tutor unabhängig von anderen Rest-Ressourcen sind (siehe Abschnitt 3.5.9.1). Aus diesem Grund wird auf das Template *choose.jade* verzichtet.
Anschließend werden die Jade-Dateien *edit.jade*, *index.jade*, *new.jade* und *show.jade*, innerhalb dieses Ordners, erstellt. Darauffolgend wird in der jeweiligen Jade-Datei das Template mit den benötigten Formularen gestaltet, welche den Paradigmen „Intuitivität“ und „Simplizität“ des CoMa-Bereichs folgen sollten. Sind alle aufgelisteten Grundtemplates erstellt, ist das Fundament für die Realisierung der CRUD-Operationen (Create, Read, Update, Delete) gelegt. Für eine detaillierte und weiterführende Erklärung der Templates kann auf das Kapitel 3.5.9.1 zugegriffen werden.

Schritt 2: Routen erstellen Als erstes wird unter dem Ordner routes (Abbildung 5) eine neue Klasse (bsp. mit den Namen `tutorComa.class.ts`) angelegt. Nachdem die Klasse angelegt worden ist, können die einzelnen Routen definiert werden. Jede dieser definierten Routen stehen für ein Template. Die Route für das Anzeigen aller Tutoren sieht folgendermaßen aus: `coma.in4all-pg.de/tutor/all`. Eine detaillierte Erklärung der Routen kann im Abschnitt 3.5.9.1.6 eingesehen werden.

von Can Celebi

Die vorherige Beschreibung bezog sich auf die Umsetzung als eigenständige Rest-Ressource. Die geforderten User-Stories sind, wie zuvor erwähnt, auch als Modifikation bzw. Erweiterung der bestehenden Rest-Ressourcen möglich, deshalb sind für den CoMa-Bereich nur die User-Stories 1-3 von Bedeutung.
Die Umsetzung der einzelnen Punkte ist wie folgt möglich:

User-Story-01 Alle User lassen sich bereits anzeigen. Wenn ein Tutor im CoMa-Bereich eingeloggt ist und die API die Anfrage aller Users erlaubt, dann ist die Funktion über die Navigation User auswählen bereits implementiert.

User-Story-02 Dieser Punkt bedarf auch keiner Erweiterung im CoMa, da diese Funktionalitäten für Administratoren bereits gegeben sind. Es ist lediglich eine Restriktion über die API von Nöten, um die Rechte einzelner Benutzergruppen zu verwalten.

User-Story-03 Die Übungsblätter können im CoMa als eine Sonderform von Aufgaben verstanden werden, somit reicht es aus ein Auswahlfeld, welcher den Typ angibt, bei den Templates *new.jade* und *edit.jade* in der Rest-Ressource Aufgabe zu erstellen. Der Name des Auswahlfeldes muss dem Namen des erweiterten Schema Feldes entsprechen, damit die API das Auswahlfeld dem passenden Schema-Feld zuweisen kann. Somit lässt sich eine Aufgabe als Übungsblatt definieren.

Damit der eingeloggte Tutor im CoMa einer Gruppe ein Übungsblatt zuweisen kann, müssen die Templates *new.jade* und *edit.jade* der Rest-Ressource Gruppe um ein Drop Down Element mit allen Aufgaben des Typus Übungsblatt erweitert werden. Das Drop Down Element muss den Namen des entsprechenden Feldes aus dem erweiterten Group-Schema tragen, damit die API die Referenz zum jeweiligen Feld zuweisen kann.

Falls die vorhandenen Aufgabentypen nicht ausreichen sollten für die Erstellung eines Übungsblattes, kann die Syntax wie im Kapitel Implementierung der Aufgabentypen gezeigt, entsprechend nach dem vorgestellten Muster, erweitert werden. Dies bedarf zusätzlich eine Erweiterung im Frontend zur korrekten Darstellung der Aufgabentypen.

3.4.5 Erweiterungen im Frontend

von Christian Everke

Neben den Änderungen für Logik und Persistenz im Backend sowie grafischen Anpassungen in der Content-Management-Oberfläche ist es auch erforderlich, die neuen Features im User-Frontend einzufügen. Konkret gilt es, die User-Stories 04 und 05 zu implementieren. User-Story-04 sieht vor, dass nach dem Login dem Benutzer neue Übungsblätter angezeigt werden. Ist ein User erfolgreich angemeldet, kann über die Implementierung des *AuthenticationService*-Interfaces, derzeit die Klasse `src/app/-services/authenticationService/authenticationService.service.ts`, abgefragt werden, mit welcher Identität der Benutzer angemeldet ist. Bereits im Abschnitt 3.4.2 wurde die Notwendigkeit beschrieben, den Benutzer im Datenmodell in Relation zu zugehörigen Gruppen (*GROUP*) und diese zu Übungsblättern (*WORKSHEET*) zu setzen. Eine Datenbankabfrage nach dem Login kann überprüfen, ob neue Übungsblätter für den Schüler hinterlegt sind. Die Umsetzung wird im Folgenden beschrieben:

Nach der Anmeldung wird die AngularJS-Komponente *HOME* (Verzeichnis `src/app/-routes/home`) dargestellt. In dem zugehörigen Controller muss über Dependency Injection die Klasse *DomainFactory* eingefügt werden. Die DomainFactory dient dazu, Instanzen des Datenmodells zu erstellen. Sind die konzeptionellen Änderungen aus Kapitel 3.4.2 in der Datei `Domain.js` richtig umgesetzt, können Instanzen für die neue Klasse *WORKSHEET* über die transitive Relation zum User bezogen werden. Durch die automatisierte Generierung des Datenmodells werden für die Klassen auch entsprechende View-Modell Klassen erstellt. Der Inhalt der Klassen wird im Frontend dargestellt. Die dortigen Methoden `save()` und `load()` dienen dem Speichern bzw. Laden neuer Daten aus der Datenbank. Die Verbindung zum API ist bereits realisiert, falls auch der Service `RestService.ts` korrekt angepasst worden ist. Um die Verfügbarkeit neuer Übungsblätter anzuzeigen, ist im Datenmodell ein Kontroll-Flag in Form eines boolschen Wertes notwendig. Dieses gibt an, ob die neuen Übungsblätter bereits angezeigt wurden. Nach dem Einfügen der neuen

Blätter durch einen Tutor ist der Wert per Default auf *FALSE* gesetzt. Wird nach dem Login und dem Ausführen der Datenbankanfrage über die DomainFactory festgestellt, dass der Wert auf *FALSE* gesetzt ist, werden die neuen Übungsblätter im Frontend angezeigt und der Wert auf *TRUE* gesetzt.

Für die Umsetzung von User-Story-05 ist es zunächst erforderlich, eine neue Route anzulegen. Eine Route beschreibt einen Pfad, über die eine bestimme AngularJS-Komponente verfügbar ist. Werden dem Benutzer, wie zu Beginn dieses Abschnitts beschrieben, die Übungsblätter angezeigt, so liegen auch die Primärschlüssel der neuen Übungsblätter vor. Ein konkreter Primärschlüssel ist Bestandteil der Route und gibt der Komponente Information darüber, welcher Übungszettel genau angezeigt werden soll. Ein Beispiel soll diesen Sachverhalt verdeutlichen: nach dem Login wird der Benutzer über die Verfügbarkeit des neuen Übungsblatts „Übung zum EVA Prinzip“ informiert. In der Datenbank hat diese Übung die ID 11223344. Der Benutzer wählt diese Übung zur Bearbeitung aus. Die Route `http://in4all-pg.de/worksheet/11223344` wird aktiv und ruft die entsprechende Komponente auf.

Um eine neue Route zu erstellen, wurde mit *PLOP* ein Tool durch die Projektgruppe entwickelt, um den Erstellungsvorgang zu vereinfachen. Dieses Tool kann über die Konsole ausgeführt werden, wie Abbildung 9 demonstriert.

Die neue Route muss namentlich benannt werden. Per Default wird der Pfad `app/src/routes` vorgeschlagen, welcher auch zu verwenden ist. Die neue Komponente wird, wie in Abbildung 10 dargestellt, dem Verzeichnisbaum hinzugefügt.

Mit der Generierung entstehen in dem neuen Verzeichnis vier neue Dateien: in `worksheet.component.ts` werden die AngularJS Komponente und der benötigte Controller definiert. In der Datei `worksheet.route.ts` wird die neue AngularJS Route definiert und hat zunächst den folgenden Inhalt:

```
1   name: 'worksheet',
2   config: {
3     url: '/worksheet',
4   template: '<ifa-worksheet></ifa-worksheet>'
5   }
6 }
```

Diese, durch PLOP generierte, Datei definiert die Route, die über die URL `http://in4all-pg.de/worksheet` erreichbar ist. Gewünscht ist jedoch, dass der Route die passende ID (Primärschlüssel) übergeben wird. Eine Anpassung der URL ist

```
C:\PG_in4all\in4all>plop
? [PLOP] Please choose a generator.
  component - erstellt eine neue angular component
  directive - erstellt eine neue angular directive
  service - erstellt einen neuen angular service
> route - erstellt eine neue angular route
```

Abbildung 9: Erstellung von Routen über PLOP

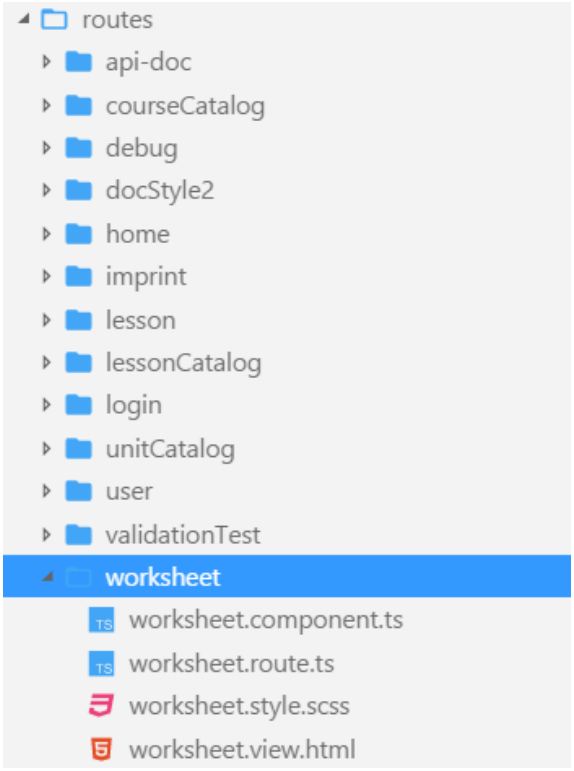


Abbildung 10: Verzeichnisbaum nach dem Anlegen einer neuen AngularJS-Route

erforderlich: `url: '/worksheet/:worksheedId'`. Die Variable *worksheetId* ist auch im Controller implizit vorhanden und kann von dort aus verwendet werden. Die Darstellung der Seite wird in der Datei `worksheet.view.ts` mit den zugehörigen Styles aus `worksheet.styles.scss` festgelegt.

Nun ist die neue Route vollständig und den Anforderungen entsprechend angelegt und kann mit Inhalten und Funktionalitäten gefüllt werden. Der Zugriff auf die Daten erfolgt, wie bereits beschrieben, über die Einbindung der *DomainFactory* in den Controller.

3.5 Projektmanagement

Mit Beginn der Projektgruppe wurde schnell klar, dass eine gute Organisation für den Erfolg der Projektgruppe unabdingbar ist. In diesem Kapitel wird beschrieben, wie das Projektmanagement umgesetzt wurde.

3.5.1 Allgemeines Vorgehen

von Christian Everke

Projektgruppen-Ordnung Damit das Projektvorgehen und die Teamarbeit für die gesamte Laufzeit der Projektgruppe einheitlich und verbindlich geregelt ist, wurde zunächst eine Projektgruppen-Ordnung gemeinsam durch alle Teilnehmer erarbeitet. Diese Ordnung umfasst Festlegungen wie bestimmte Rollen und Positionen und deren Aufgaben innerhalb der Projektgruppe, das Zeitintervall, der Ablauf und die Protokollierung regelmäßiger Treffen, die Beschlussfähigkeit für Entscheidungen und das Vorgehen bei Abwesenheit. Grundlage für die Aufstellung des Dokuments war eine ähnliche Ordnung einer anderen, erfolgreich beendeten Projektgruppe der Fakultät.

Folgende besondere Rollen wurden in der Projektgruppen-Ordnung festgelegt:

- Projektleiter
- Ticketmaster
- Berichtsmanager
- Sitzungsleiter
- Protokollant

Die folgenden Beschreibungen der Rollen, Aufgaben und Vorgehensweisen basieren auf der PG-Ordnung [Pro16b]. Einige Textfragmente wurden dabei vollständig entnommen. Aufgrund der besseren Lesbarkeit wurde immer die männliche Form für Rollenbezeichnungen verwendet. Selbstverständlich können alle Aufgaben auch durch Frauen ausgeführt werden. Hauptorgan des Teams ist der Projektleiter, der die Gesamtverantwortung für das Projekt trägt. Er ist dafür verantwortlich, dass die jeweiligen Aufgaben eines Bearbeitungsintervalls zu einem erfolgreichen Abschluss

kommen. Hierbei hat er nicht selbst die fehlenden Aufgaben zu erledigen, sondern die jeweiligen Ansprechpartner frühzeitig über den mangelnden Projektfortschritt zu informieren. Existiert eine Stimmgleichheit bei wichtigen Abstimmungen, so entscheidet ebenfalls der Projektleiter.

Die jeweiligen Arbeitspakete werden in Form von sogenannten Tickets in einer Projektmanagement-Software verwaltet (siehe auch Kapitel 3.5.2). Der Ticketmaster ist in erster Linie für die Administration der Ticket-Software verantwortlich. Ferner ist er zuständig für die Erstellung und Zuweisung von Aufgaben und den zugehörigen Tickets, sofern dies nicht in einzelnen Subteams intern gelöst wird. Werden durch die Mitglieder bereits zugeordnete Aufgaben untereinander getauscht, so kann dies erst nach Absprache mit dem Ticketmaster erfolgen.

Da es sich bei der Durchführung der Projektgruppe um eine wissenschaftliche Lehrveranstaltung der Universität handelt, sind akademische Berichte notwendig. So wurde neben dem vorliegenden Abschlussbericht nach Abschluss des ersten Semesters der Lehrveranstaltung ein Zwischenbericht angefertigt. Da für die Einreichung der Dokumente ggf. Fristen gesetzt werden können, ist der Berichtmanager für die Einhaltung dieser verantwortlich. Außerdem erstellt er Vorlagen für die Berichte und achtet auf die Einhaltung formaler Kriterien.

Die Projektgruppe hat sich darauf geeinigt, sich mindestens einmal wöchentlich in voller Besetzung zu treffen, um den allgemeinen Fortschritt zu besprechen. Zur Dokumentation des Meetings wurde die Funktion des Protokollanten festgelegt. Das Protokoll hat die Namen der Anwesenden, die Tagesordnungspunkte, die gefassten Beschlüsse mit jeweiligen Abstimmungsergebnissen und alle ausdrücklich zum Zwecke der Niederschrift abgegebenen Erklärungen zu enthalten. Das jeweils zuletzt erstellte Protokoll wird am Anfang einer jeden wöchentlichen Sitzung zur Abstimmung gestellt und (ggf. nach erfolgten Änderungswünschen) von den Anwesenden akzeptiert. Das Protokoll sollte am Tag nach der Sitzung im gemeinsamen Datenverzeichnis verfügbar sein [Pro16b]. Eine Änderung des Protokolls ist nicht mehr möglich. Der Protokollant ist für die Erstellung dieser Dokumentation verantwortlich.

Für das wöchentliche Treffen, auch Sitzung genannt, ist ein Vorsitz zu bestimmen, der sogenannte Sitzungsleiter. Die Aufgaben setzten sich wie folgt zusammen: zu Beginn jeder Sitzung ist die Anwesenheit der Teilnehmer und Beschlussfähigkeit der Projektgruppe festzustellen. Stimmberechtigt ist jeder PG-Teilnehmer, jedoch nicht die Betreuer vom zugehörigen Lehrstuhl. Die Projektgruppe ist beschlussfä-

hig, wenn mehr als 50% aller Teilnehmer anwesend sind. Beschlüsse durch Wahlen werden durch eine einfache Mehrheit getroffen (mehr „Ja“- als „Nein“-Stimmen). Die Abstimmungen erfolgen per Handzeichen. Ein Antrag ist abgelehnt, wenn die Anzahl der Enthaltungen größer ist als die Summe der „Ja“ und „Nein“-Stimmen. Die endgültige Tagesordnung ist einen Tag vorher im Dokumentenverzeichnis einsehbar. Weitere Tagesordnungspunkte werden am Ende der Sitzung in einem Tagesordnungspunkt „Verschiedenes“ besprochen. Vor dem Ende jedes Treffens wird die Aufgabenverteilung ggf. aktualisiert und dabei festgestellt, welche Tätigkeiten für die aktuelle Bearbeitungsphase geändert und bearbeitet werden müssen.

Die Rollen des Projektleiters, des Ticketmasters und des Berichtmanagers wurden fest vergeben. Die Sitzungsleiter wechselten im 14-tägigen Rhythmus nach lexikografischer Reihenfolge. Auch die Funktion des Protokollanten wurde nicht direkt vergeben, wurde aber jedoch in den meisten Fällen immer durch die selbe Person durchgeführt.

Projektmanagement-Workshop Das Management ist ein wichtiger Artefakt für die erfolgreiche Realisierung von Projekten. Insbesondere stehen Softwareentwicklungsprojekte vor der Herausforderung, das Spannungsfeld zwischen fachlichen und technologischen Anforderungen und verbundenen Risiken erfolgreich zu meistern [BVWS14]. Da zu Beginn alle Mitglieder keine bis nur wenig Erfahrung im Bereich Projektmanagement hatten, fand ein einführender Workshop in dieses Themenumfeld statt. Organisiert wurde dieser von erfahrenen Projektmanagern der Capgemini Deutschland GmbH. Vermittelt wurden Wissens- und Managementgrundlagen, wie etwa die Definition und Klassifikation des Projektbegriffs, Vorgehensmodelle und deren Elemente, Planung von Projekten und Überwachung des Projektfortschritts, Risikoanalyse und Bewertung sowie Führungsstiele, Teamentwicklung und Teamperformance.

Um die Artefakte des Projektmanagements auch in praktischer Weise zu trainieren, wurde durch die Dozenten von Capgemini ein *Projektmanagement-Board* ins Leben gerufen: in regelmäßigen Abständen trafen sich jeweils ein Vertreter mehrerer Projektgruppen, um formal über den Projektfortschritt zu berichten. Besprochen wurden die Themen *Status und Erreichtes*, *Nächste Schritte*, *Probleme und Risiken* und *Benötigte Entscheidungen*. Gemeinsam wurde über die Themen gesprochen und weitere wichtige Tipps zur weiteren Durchführung vermittelt.

Scrum als Vorgehensmodell Nach Erlangen des notwendigen Wissens über das Projektmanagement im Workshop wurden einigte sich die Projektgruppe schnell darauf, Scrum [Pic13] als Vorgehensmodell einzusetzen. Im folgenden wird Scrum kurz vorgestellt und Anpassungen des Modells an die Gegebenheiten der Projektgruppe erläutert.

Wichtig für das Verständnis von Scrum sind zunächst die Rollenverteilungen. Unterschieden wird hier zwischen dem sog. *Product Owner*, dem *Scrum Master* und dem *Entwicklerteam*.

Der Scrum Master ist verantwortlich, dass das Scrum-Vorgehensmodell richtig umgesetzt wird. Dies geschieht in enger Zusammenarbeit mit dem Entwicklungsteam. Bei richtiger Anwendung gehört der Scrum Master jedoch selbst nicht zu den Entwicklern selbst. Der Scrum Master dient dem Team als technischer Berater und wirkt als Führungskraft [Scr16].

Der Product Owner ist für den Gesamterfolg des Projektes bzw. des zu entwickelnden Produktes verantwortlich. Wesentliche Aufgaben sind die Erstellung von Produkteigenschaften und die Festlegung, welche dieser Eigenschaften zu welchem Zeitpunkt fertig gestellt werden sollen. Dazu wird das sog. *Product Backlog* verwendet, welches die Anforderungen an das Projekt bzw. Produkt beinhaltet.

Das Entwicklerteam sind alle Personen, die die im Backlog definierten Aufgaben nach vorgegebenen Plan entwickeln.

In Scrum sind folgende Aktivitäten vorgesehen: im *Sprint Planning* wird festgelegt, was und wie in einem bevorstehenden Sprit entwickelt wird. Ein *Sprint* meint im Sinne von Scrum einen festgelegten und zeitlichen Arbeitsabschnitt. Der Sprint beginnt klassischerweise mit dem Sprint Planning und endet mit den Aktivitäten *Sprint Review* und *Sprint-Retrospektive*, auf die im späteren Verlauf dieses Abschnitts noch eingegangen wird. Zum Informationsaustausch findet täglich der sogenannte *Daily Scrum* statt, ein kurzes, im stehen durchgeführtes Zusammenkommen aller Projektmitglieder, um den aktuellen Entwicklungsstand zu besprechen. Wie gerade kurz erwähnt, steht am Ende eines Sprints das Sprint Review an. Ziel ist die Überprüfung durch das Scrum Team, ob das Sprint Ziel erreicht wurde und ob das Product Backlog angepasst werden muss. Die vorzeigbaren, ggf. noch nicht fertigen Ergebnisse werden durch das Entwicklungsteam präsentiert. Gemeinsam wird die weitere Vorgehensweise, ggf. mit weiteren Stakeholdern wie Kunden, besprochen. Ob die Teamarbeit effizienter oder effektiver gestaltet werden kann, wird in der Sprint Re-

trospektive überprüft. Der Scrum Master unterstützt bei der Findung von Best-Practices, die im nächsten Sprint umgesetzt werden können.

Neben dem kurz erwähnten Product-Backlog, dass die Anforderungen an das Produkt enthält, gibt es noch das sog. *Sprint Backlog* und das *Product Increment*. Das Sprint Backlog ist eine Teilmenge des Product Backlogs und enthält alle Anforderungen, die in einem Sprint umgesetzt werden sollen. Das Inkrement ist die Zusammenfassung aller Product-Backlog-Einträge, in bislang allen durchgeführten Sprints erfolgreich abgearbeitet wurden.

Das Scrum-Modell ist in erster Linie für Projekte gedacht, die in Vollzeit bei Unternehmen umgesetzt wurden. Da die Projektgruppe nicht in Vollzeit durchgeführt wird und sich die einzelnen Teilgruppen nur zu unterschiedlichen Zeitpunkten zum Arbeiten zusammenfinden konnten, mussten einige Anpassungen des Scrum-Modells an die Gruppe vorgenommen werden: zunächst fand die Rollenverteilung statt. Es wurde festgelegt, dass die Rollen des Product Owners durch den in der PG-Ordnung definierten Projektleiter und der Scrum Master durch den festgelegten Ticketmaster vertreten werden. Die Aufgaben wurden, wie es in Scrum festlegt ist, wahrgenommen. Aufgrund des unregelmäßigen Zusammenkommens der Projektgruppe war das Praktizieren eines Daily Scrums nicht möglich. Als Ersatz wurde das „Weekly“ eingeführt, bei dem alle Mitglieder beim wöchentlichen Treffen über den aktuellen Stand berichteten. Die Sprintlänge war dynamisch und wurde nach jeder Planungsphase auf einen Zeitraum von zwei bis maximal vier Wochen festgelegt. Erst im Nachhinein zeigten sich einige Probleme bei der Praktizieren von Scrum. Diese sind in Kapitel 3.5.4 aufgeführt.

3.5.2 Tools

von Christian Everke

Im vorherigen Abschnitt wurde die Organisation des Projektmanagements beschrieben. Ohne funktionierende Software ist eine reibungslose Durchführung und Organisation schwierig. Der Einsatz entsprechender Tools war daher schon mit Beginn der Projektgruppe erkennbar. Es wurden zunächst verschiedene, mögliche Systeme in Betracht gezogen, die Anforderungen analysiert und abgeglichen. Letztendlich wurde eine finale Auswahl getroffen. Auf Details der Entscheidungsfindung sei an dieser Stelle verzichtet. Nach einer Abstimmung wurde entschieden, Produkte des

australischen Softwareunternehmens *Atlassian* zu verwenden, die im folgenden kurz beschrieben werden.

Die Webanwendung *Jira* dient dem klassischen Projektmanagement und wird primär in der Softwareentwicklung eingesetzt. Vorteile dieser Applikation liegen in der Anpassungsfähigkeit an existierende Vorgehensmodelle wie das zuvor beschriebene Scrum oder Kanban [And11]. Artefakte wie Sprint- und Product Backlog können definiert, eingesehen und der angegebene Entwicklungsstand verfolgt und grafisch visualisiert werden. Jira wurde auf Basis der service- und komponentenorientierten Softwareplattform OSGi [WHKL15] entwickelt. Erweiterungspakete, etwa zum Erstellen von UML-Diagrammen, können daher problemlos zur Laufzeit ergänzt werden.

Als Plattform zum Dokumenten- und Wissensaustausch wurde *Confluence* verwendet. Dokumente werden über dieses Tool verwaltet. Auch ist es möglich, über einfache WYSIWYG-Editoren neue Dokumente, etwa wie Sitzungsprotokolle, zu erstellen oder zu bearbeiten.

Zur grafischen Verwaltung und Zustandsvisualisierung der GIT-basierten Repositories wurde *Bitbucket* verwendet. Benutzer sind in der Lage, Repositories anzulegen, sog. *Branches* zu erstellen und diese wiederzuvereinigen (sog. *Mergen*). Alle drei Produkte, Jira, Confluence und Bitbucket sind miteinander verknüpft. So kann beispielsweise zu einem Jira-Ticket direkt ein Branch in einem entsprechenden Repository angelegt oder ein Dokument aus Confluence hinzugefügt werden.

Die Produkte unterliegen grundsätzlich einem kommerziellen und damit mit Kosten verbundenen Softwareüberlassungsvertrag. Atlassian stellt jedoch kostenfreie Lizenzen für Education-Bereich (sog. *Atlassian Academic Licence*) zur Verfügung, die im Zuge der Projektgruppe verwendet wurden[14].

Für die Einrichtung und Administration war die Projektgruppe selbst verantwortlich. Durch den Lehrstuhl wurden entsprechende Ressourcen zum Hosting zur Verfügung gestellt.

[14] `https://de.atlassian.com/licensing/purchase-licensing#licensing-2`

3.5.3 Rechtliche Aufgaben

von Christian Everke

Das Ergebnis der Projektgruppe dient neben Lehr- und Forschungszwecken auch dem Talentkolleg Ruhr zum Produktiveinsatz für die Weiterbildung von Schülerinnen und Schüler. Für die Herausgabe des fertigen Produktes sind rechtliche Überlegungen unabdingbar.

In der Lehrveranstaltung *IT-Recht — Grundlagen für Informatiker*, die auch von einigen Projektgruppenmitgliedern besucht wurde, wurden bereits erste Kenntnisse über rechtliche Grundlagen vermittelt. Die wohl einfachste Form, unbürokratisch Software zu verbreiten, ist über Lizenzen aus dem OpenSource-Umfeld [JM16]. Schnell wurde jedoch Unmut in der Projektgruppe deutlich, da eine derartige Software auch von industriellen Software- und Beratungsfirmen entwickelt werden kann und es die Aufgabe von Projektgruppen nicht sein soll, uneingeschränkt nutzbare und kostenlose Software für Dritte zu entwickeln. Zwar wurde festgehalten, das Programm kostenfrei zur Verfügung zu stellen, jedoch die Nutzungsrechte nach Möglichkeit einzuschränken. Als Einschränkungen kommen unter anderem in Frage, dass Dritte die Anwendung zwar nutzen, jedoch diese nicht weiter veräußern, verschenken oder vermieten dürfen. Auch ist eine Weiterentwicklung für nicht wissenschaftliche Zwecke nicht im Sinne der Projektgruppe. Neben der lauffähigen Software werden auch exemplarische Lerninhalte in Form von Videos und Aufgaben mitgeliefert. Auch hier soll eine Verbreitung durch Dritte ausgeschlossen werden.

Gemeinsam mit Herrn Rechtsanwalt Wolfgang Müller, der als externer Lehrbeauftragter für die Durchführung oben genannter Lehrveranstaltung verantwortlich ist, wurde der rechtliche Sachverhalt analysiert und gemeinsam ein Lösungsweg erarbeitet. Die Überlassung der Software an das Talentkolleg (ggf. an weitere Dritte) erfolgt im Rahmen eines Leihvertrages, der durch das bürgerliche Gesetzbuch in den Paragraphen 598-606 geregelt ist. Ein Leihvertrag liegt vor, wenn eine Sache (auch Software) unentgeltlich und befristet überlassen wird. Nach Ablauf der Frist ist eine Rückgabe oder Fristverlängerung möglich. Im Gegensatz zu einem Kauf- oder Werkvertrag kann der Vertragspartner gegenüber der Projektgruppe keine Haftungs- und Gewährleistungsansprüche stellen, die nach Beendigung der Projektgruppe in Anspruch genommen werden können. Ausgenommen hiervon sind Vorsatz und grobe Fahrlässigkeit.

Das Leihverhältnis hat eine Laufzeit von 24 Monaten und beginnt zum Zeitpunkt der Vertragsunterzeichnung, falls kein anderes Datum im Vertrag vereinbart ist. Es verlängert sich, sofern es nicht mit einer Frist von einem Monat gekündigt wird, automatisch um weitere 12 Monate. Die Projektgruppe selbst ist eine Gesellschaft bürgerlichen Rechts (GbR, auch BGB-Gesellschaft genannt), die durch den Paragraphen 705 des bürgerlichen Gesetzbuches geregelt ist. Diese BGB-Gesellschaft besteht weiter, auch wenn die Projektgruppe im Sinne der Lehrveranstaltung beendet ist. Dazu ist es unter Umständen erforderlich, dass zu weiteren Entscheidungsfindungen (z.B. Vertragsverlängerungen mit Dritten) alle Mitglieder zusammenkommen oder in einer alternativen Form an einer Entscheidungsfindung teilnehmen. Regelungen diesbezüglich sind bereits intern in der Projektgruppe festgesetzt worden.

3.5.4 Herausforderungen und Erfahrungen

von Christian Everke

Die Umsetzung eines korrekten und einwandfreien Projektmanagements brachte einige Herausforderungen mit sich, die im folgenden offen thematisiert werden sollen. Im Gegensatz zu anderen Projektgruppen oder Lehrveranstaltungen anderer Lehrstühle des Dortmunder Informatik Fachbereiches gab es keinerlei Vorgaben, mit welchen technologischen Mitteln die Entwicklung des Produktes durchgeführt werden soll. Die entsprechende Entscheidung lag somit bei den Mitgliedern der Projektgruppe selbst und zeichnete sich zunächst schwierig, da keine ausreichenden Erfahrungen über mögliche Technologien vorlagen. Ebenso konnte das Risiko, auch in Hinblick auf notwendige Einarbeitungszeit und technischen Anforderungen, aufgrund mangelnder Erfahrung nur sehr schwer eingeschätzt werden. Professionelle Unternehmen setzen für die Akquise und Planung von Projekten Mitarbeiter mit langjähriger Berufserfahrung ein. Die Zusammensetzung von Entwicklungsteams in Unternehmen ist nach Möglichkeit eine Kompositionen von erfahrenen Projektleitern, Softwarearchitekten und jungen Softwareingenieuren, die gerade erst ihre Hochschulausbildung absolviert haben. Natürlich ist eine derartige Zusammensetzung für die Projektgruppe nicht möglich. Unter allen Projektgruppenteilnehmern wurde die Unerfahrenheit und das damit verbundene Projektrisiko deutlich. Auf Unterstützung seitens der Betreuer musste ebenfalls verzichtet werden, da auch diese nicht über die notwendige, praktische Erfahrung im Bereich des Softwarengineerings verfügen.

Bei der Umsetzung des Vorgehensmodells waren auch einige Hürden zu verzeichnen. Wie bei der Vorstellung von Scrum in Abschnitt 3.5.1 bereits beschrieben,

ist Scrum für Vollzeitprojekte konzipiert. Im Falle der Projektgruppe galt es allerdings zu berücksichtigen, dass diese parallel zu weiteren Lehrveranstaltungen von den Studierenden durchgeführt wird und der wöchentliche Arbeitsaufwand pro Teilnehmer bei ca. 14-16 Stunden pro Woche liegt. Dies entspricht 35-40 Prozent einer wöchentlichen Arbeitszeit von fünf Personentagen (40 Stunden pro Woche)[15] in Unternehmen. Die Sprints im vorliegenden Projekt waren in den meisten Fällen zwei Wochen lang. Es ergibt sich somit für den zweiwöchigen Sprint eine Arbeitszeit von maximal 32 Stunden pro Studierenden, dass entspricht insgesamt vier Personentagen. Ein zweiwöchiger Sprint im Unternehmen dauert pro Kopf zehn Personentage. Dieser Umstand wurde zum jeweiligen Sprintende sehr deutlich, da nicht immer alle Aufgaben wie geplant durchgeführt werden konnten. Dies ist vermutlich zum einen damit verbunden, dass die tatsächlich verfügbare Arbeitszeit in einem Sprint deutlich weniger betrug, als unterbewusst angenommen. Auch wurde verhältnismäßig viel Zeit in die wöchentliche Organisation (Vorbereitung, Durchführung und Teilnahme von Besprechungen) investiert, die in die genannte Arbeitszeit einfließt. Weiterhin kam hinzu, dass ein gemeinsames Arbeiten der einzelnen Teams nicht immer möglich war. Die Gründe hierfür liegen unter anderem in Einschränkungen der zeitlichen Verfügbarkeit jedes Einzelnen, als auch in der Verfügbarkeit von räumlichen Ressourcen. Zwar wurde ein Labor des Lehrstuhl der Gruppe bereitgestellt, dieser konnte aber häufig aufgrund von Eigenbedarf des Lehrstuhls häufig nicht verwendet werden. Durch das häufige zeitliche und räumlich versetzte Arbeiten der Entwickler kam es teilweise zu Entwicklungsverzögerungen, falls Hilfestellungen oder allgemeine Fragen und Unterstützungen durch Kommilitonen notwendig war, diese aber zeitlich nicht greifbar waren. Ebenfalls führte ein falsches Verständnis von Sprints zu fehlerhaften Schätzungen: das Ziel eines Sprints nach Scrum ist, möglichst viele Aufgaben in einem Zeitraum abzuarbeiten. Auf Basis der abgearbeiteten Arbeitspakete und der gemachten zeitlichen Erfahrungen sollen die Arbeitsaufwände für Pakte der folgende Sprints besser geschätzt werden können. Leider war es teilweise der Fall, dass Sprintenden als Deadlines betrachtet worden sind und die Abarbeitung von Arbeitspaketen erst kurz vor Erreichen des Sprintendes begonnen wurde.

[15] 1PT = 8 h

3.6 Ausblick

von Christian Everke

3.6.1 Fachliche Weiterentwicklung

Der finale Stand, der zum Zeitpunkt des Abschlusses der Projektgruppe vorlag, entspricht im Wesentlichen den in der Projektgruppen-Ausschreibung [Fak16] festgelegten Minimalanforderungen. Selbstverständlich sind Erweiterungen um kundenindividuelle Funktionalitäten denkbar. Den technischen Ablauf der Skalierung beschreibt Kapitel 3.4. IN4ALL wurde entwickelt, um eine informatische Bildung auf Schulniveau zu unterstützen. Es stellt sich die berechtigte Frage, um welche didaktischen und technischen Mittel bzw. Methoden das Produkt erweitert werden kann. Aufgrund flächendeckender Verfügbarkeit von schnellem Internet und Speicherkapazitäten sind Überlegungen durchaus realistisch, Unterricht in Echtzeit und per „Remote“, also von unterschiedlichen Orten, durchzuführen. Unterricht von verschiedenen Standorten kann an andere Schulen und Bildungseinrichten live übertragen werden. Es bedarf keiner Aufzeichnung und Bearbeitung von Videosequenzen. Schüler interagieren untereinander und mit Lehrern über das Internet. Übungsblätter und Aufgaben werden digital zur Verfügung gestellt, bearbeitet und korrigiert. Erfasste Noten oder Leistungen können direkt mit Schülern, Lehrern und Eltern und zugehörige Stellen (z.B. Schulverwaltung) über das System ausgetauscht werden. Insgesamt wird die Kommunikation zwischen Schülern, Lehrern und Eltern verbessert. Dabei entfallen räumliche und unter Umständen auch zeitliche Grenzen. Der digitale Wandel findet also nicht nur im industriellen Umfeld Einzug, sondern auch im *Education Sector*.

Derartige Strategien betreffen Anwendungen nicht nur softwareseitig. Auch hardwareseitig müssen Lösungen geschaffen werden. Tafeln und Whiteboards müssen digitalisiert und angebunden werden, der Austausch und die Bearbeitung von Übungsblättern muss effizient und einfach durchführbar und eine funktionierende Interaktion zwischen Schülern, Lehrer und Computern muss gewährleistet sein.

Auf der CeBIT 2017 stellte der Chinesische Konzern *Huawei Technologies Co., Ltd.* beispielsweise mehrere Lösungen für den Bildungssektor vor[16]. Da sich IN4ALL bislang noch um einen anfänglichen Entwicklungstand befindet, sind derartige Technologien zwar noch weit entfernt, aber grundsätzlich nicht unmöglich.

[16] vgl. auch `http://e.huawei.com/de/solutions/industries/education`

3.6.2 Technische Weiterentwicklung

Fachliche Weiterentwicklungen wie zuvor beschrieben führen selbstverständlich zu zunehmender Komplexität der Software. Trotz wachsenden Code und größeren Datenmengen muss die Anwendung effizient und stabil sein. Eine vertikale Skalierung durch Hinzufügen von hardwareseitiger Rechenleistung kann Abhilfe schaffen. Doch nicht nur der Betrieb der Software muss effizient sein, auch die Entwicklung soll weiterhin einfach durchführbar sein. Eine große Projektstruktur kann schnell unübersichtlich werden. Die Einarbeitung durch neue Softwareingenieuere erfordert ein hohes Maß an Disziplin und fachlichem Verständnis. Durch wachsende Größe der Applikation werden auch Build- und Deployment-Prozesse umfangreicher: Kompilierzeiten steigen und die Auslieferung muss reibungslos, im besten Fall ohne Unterbrechung oder hohem Zeitaufwand erfolgen.

Abhilfe schafft das Konzept der Microservice-Architektur [Wol15]. Grundgedanke ist es, einen sogenannten Software-Monolithen in einzelne, kleinere Anwendungen zu zerlegen. Diese Zerlegung geschieht service-orientiert. Veranschaulicht sei dies am Beispiel von IN4ALL. Bislang sind alle Funktionalitäten in eine einzige Applikation gegossen. Wird nun eine Veränderung durchgeführt, muss die gesamte Anwendung neu gebaut und ausgeliefert werden. Handelt es sich um eine große Applikation, kann dieser Prozess lange dauern. Laufende Interaktionen oder Transaktionen werden unterbrochen und müssen ggf. zurückgerollt werden. Bei Verwendung einer Microservice-Architektur können spezifische Services, z.B. ein *LessonService* und ein *ExerciseService* vorhanden sein. Bei diesen beiden Services handelt es sich um eigenständige Applikationen, die vollkommen unabhängig auf unterschiedlichen Servern oder anderen Deployment-Einheiten (z.B. Docker Container [Mou16]) laufen. Beide Services betreiben eine eigene Datenbank, stellen eine grafische Benutzeroberfläche und eine REST-API bereit. Microservices können und müssen, ggf. unter Berücksichtigung eines vorgegebenen Ablaufes, miteinander kommunizieren. Durch die Tatsache, dass alle Services eigenständige Anwendungen sind, ist auch eine heterogene Technologielandschaft möglich: während beispielsweise der *LessonService* auf Basis des MEAN-Stacks entwickelt worden sein kann, kann der *ExerciseService* in Spring-Boot [Ant15] implementiert sein. Eine derartige Zerlegung des Monolithen hat mehrere Vorteile: neue Entwickler müssen sich nun nicht mehr in die gesamte monolithische Anwendung einarbeiten, sondern können sich technisch und fachlich direkt um den zugehörigen Microservice kümmern. Wird eine Änderung an einem der Serivces durchgeführt, wird lediglich diese eine Anwendung neu gebaut und ausgelie-

fert. Alle weiteren Microservices sind hiervon nicht betroffen, so dass die Gesamtheit der Plattform weiterlaufen kann. Durch den kleineren Umfang eines Microservices im Vergleich zum Monolithen reduzieren sich Build-Zeiten und Deployment-Prozesse. Insgesamt wird auch die Ausfallsicherheit erhöht. Es sei angenommen, dass in der Datenbank eines Monolithen ein Fehler auftritt und diese komplett ausfällt. Dieser Umstand führt dazu, dass der gesamten Software keine Daten zur Verfügung gestellt werden und somit die gesamte Applikation nicht funktionsfähig ist. Da in einer Microservice-Architektur jeder Service seine eigene Persistenz verwaltet, ist der Ausfall einer Datenbank eines Microservices zwar für diesen einen Service kritisch, alle weiteren Teile der Software laufen aber uneingeschränkt weiter. Fällt also die Datenbank des *ExerciseService* aus, können weiterhin Funktionalitäten des *LessonService* ausgeführt werden. Lediglich die Funktionen des *ExerciseService* sind nicht mehr verfügbar. Weitere Details und die Umsetzung wichtiger Eigenschaften dieser Architektur sind dem Buch [Wol15] zu entnehmen. Eine kürzer ausfallende Einführung gibt Martin Fowler in seinem Blogbeitrag [Fow14].

3.6.3 Teamorganisation

Für das Projekt IN4ALL teilten sich die Entwickler in insgesamt drei Subteams auf. Während zwei Subteams für die Umsetzung der beiden Frontends zuständig waren, war ein Teilteam für die Realisierung der Persistenz, der REST-API und der Logik im Backend zuständig. Eine derartige Aufteilung ist in der Softwareindustrie durchaus möglich und praktikabel. Es gibt jedoch auch Ansätze, bei denen das Team nach Fachlichkeiten aufgeteilt wird. Ist als Vorgehensmodell Scrum mit Hilfe von User-Stories [Wir17] vorgesehen, so können Teilteams gebildet werden, die für die Umsetzung der erfassten User-Stories verantwortlich sind. Hierbei agiert das Team auf dem gesamten Technologie-Stack, d.h. es ist sowohl für die grafische Oberfläche, für die Business-Logik als auch für die Persistenz der jeweiligen Funktionalität verantwortlich. Bei einem Einsatz der zuvor erwähnten Microservices ist es auch möglich, Teams zur Entwicklung dieser Services zu bilden. Auch bei diesem Ansatz ist das Team für den gesamten Technologiestack verantwortlich. Grundlage für eine optimale Teambildung ist neben der Anzahl der verfügbaren Entwickler auch die gesamte Projektgröße. In seinem Buch [Wol15] geht Eberhard Wolff neben dem Konzept der Microservices auch auf die Teamorganisation ein. Für Einsteiger in der Softwareentwicklung und für Projektgruppen die auf Basis dieses Berichts das IN4ALL-Projekt weiterentwickeln, ist das Buch sehr empfehlenswert.

4 Benutzerhandbuch

Dieser letzte inhaltliche Abschnitt führt die Endanwender in die Handhabung des Systems ein. Dazu gehört neben der allgemeinen Verwendung auch die Installation der Client-Server-Anwendung.

4.1 Systembeschreibung

Dieser Abschnitt wird sich mit dem fertigen Produkt IN4ALL befassen. Es wird beschrieben, welche Anforderungen zur Installation erfüllt werden müssen, wie diese vonstatten geht und worauf geachtet werden muss. Ferner wird der Funktionsumfang von IN4ALL beschrieben.

Es wird davon ausgegangen, dass grundlegende Kenntnisse im Umgang mit Apache und der Paketinstallation unter Linux vorhanden sind.

4.1.1 Systemanforderungen

von Julian Schilling

Um IN4ALL nutzen zu können, müssen verschiedene Voraussetzungen erfüllt werden, sowohl vom Server als auch abseits vom Server.

1. Es sind 4 verschiedene Domains nötig, jeweils eine für:

   - Frontend — Die Website an sich, auf welcher Videos abgerufen und Fragen beantwortet werden.
   - Backend — Der Zugangspunkt zur Datenbank.
   - Content Management — Das System, über welche neue Daten hinzugefügt und bearbeitet werden.
   - Assets — Der Ort, an welchem Bilder und Videos gehostet werden.

   Erfahrene Administratoren können durch geschickte Apache-Konfiguration möglicherweise mehrere dieser Domains zusammenfassen; dies wird zur Nutzung von IN4ALL jedoch ausdrücklich nicht empfohlen. Subdomains werden von IN4ALL unterstützt; es ist also beispielsweise möglich, die Domains `in4all.de`, `api.in4all.de` und `assets.in4all.de` zu nutzen.

2. Der Server muss eine MongoDB bereitstellen. Getestet wurde IN4ALL mit Version 3.4.0; es sollten jedoch auch andere, neuere Versionen funktionieren. Hierfür kann jedoch keine Garantie übernommen werden.

3. Der Server muss einen Apache bereitstellen. Bei diesem müssen die Module `proxy_module` und `proxy_http_module` aktiviert sein.

4. Der Server muss über eine NodeJS-Installation verfügen. Getestet wurde IN4ALL mit Version 6.9.1. Version 7.0.0 und aufwärts sollten jedoch ebenfalls nutzbar sein, hierfür kann jedoch nicht garantiert werden[17].

5. Zum Verwalten von CoMa und Backend empfiehlt es sich, das NPM-Paket `forever` zu installieren. Diese Anleitung geht im Weiteren davon aus, dass dieses genutzt wird.

6. Bis Stresstests vorgenommen wurden, können leider keine genaueren Aussagen über benötigte Prozessorleistung oder Arbeitsspeicher getroffen werden. Die getestete VM verfügt über 8 GB RAM, 4 GB Swap und die CPU „Intel Xeon E5-2620“. Tatsächliche Last entsteht primär, wenn Videos über das CoMa hochgeladen werden.

4.1.2 Funktionsumfang

von Tolgay Usul

In diesem Unterkapitel werden die verschiedenen Funktionen zum Frontend und zum Content Management Teil der Seite tabellarisch aufgelistet. Die einzelnen Methoden werden nach Typen und Objektgruppe geordnet und kurz beschrieben.

[17] Version 7.0.0 führte zu Problemen beim Kompiliervorgang; diese sollten sich jedoch nicht auf die Nutzung des ausgelieferten IN4ALL auswirken.

Frontend			
Typen	Methode	Beschreibung	Bemerkungen
Profil	Ansehen des Profils	Nutzer schaut auf seine Profildaten	Keine
Profil	Bearbeiten des Profils	Nutzer bearbeitet seine Profildaten	Keine
Katalog	Ansehen des Katalogs	Nutzer schaut auf verfügbare Kurse	Keine
Katalog	Auswahl des Kurses	Nutzer wählt Kurs aus	Keine
Katalog	Auswahl der (Lern)Einheit	Nutzer wählt (Lern)Einheit im Kurs aus	Keine
Katalog	Auswahl der Lektion	Nutzer wählt Lektion in der (Lern)Einheit aus	Keine
Lektion	Abspielen des Lernvideos	Nutzer schaut Lernmaterial an	Keine
Aufgaben	Erledigen der Aufgaben	Nutzer bearbeitet Aufgaben vom Lernvideo	Keine
Aufgaben	Korrektur der Aufgaben	Nutzer gibt Lösungen zur Korrektur ab und bekommt Feedback	Keine
Aufgaben	Einsehen der Lösungen	Nutzer sieht korrekte Lösungen an	Keine
Aufgaben	Zurück zur Lektion	Nutzer geht zurück zur gewählten Lektion	Keine
Typen	Methode	Beschreibung	Bemerkungen
Profil	Ansehen des Profils	Nutzer schaut auf seine Profildaten	Keine
Profil	Bearbeiten des Profils	Nutzer bearbeitet seine Profildaten	Keine
Katalog	Ansehen des Katalogs	Nutzer schaut auf verfügbare Kurse	Keine
Katalog	Auswahl des Kurses	Nutzer wählt Kurs aus	Keine
Katalog	Auswahl der (Lern)Einheit	Nutzer wählt (Lern)Einheit im Kurs aus	Keine
Katalog	Auswahl der Lektion	Nutzer wählt Lektion in der (Lern)Einheit aus	Keine
Lektion	Abspielen des Lernvideos	Nutzer schaut Lernmaterial an	Keine
Aufgaben	Erledigen der Aufgaben	Nutzer bearbeitet Aufgaben vom Lernvideo	Keine
Aufgaben	Korrektur der Aufgaben	Nutzer gibt Lösungen zur Korrektur ab und bekommt Feedback	Keine
Aufgaben	Einsehen der Lösungen	Nutzer sieht korrekte Lösungen an	Keine
Aufgaben	Zurück zur Lektion	Nutzer geht zurück zur gewählten Lektion	Keine

Content Managment			
Objektgruppe	Methode	Beschreibung	Bemerkungen
Gruppen	Erstellen	Das Erstellen einer Gruppe von Nutzern	Keine
Gruppen	Editieren	Das Editieren einer Gruppe von Nutzern	Keine
Gruppen	Löschen	Das Löschen einer Gruppe von Nutzern	Nutzer werden ebenfalls gelöscht
Gruppen	Nutzer hinzufügen	Das Hinzufügen eines Nutzers zur Gruppe	Keine
User(Nutzer)	Erstellen	Das Erstellen eines Nutzers	Keine
User(Nutzer)	Editieren	Das Editieren eines Nutzers	Keine
User(Nutzer)	Löschen	Das Löschen eines Nutzers	Keine
Kurse	Erstellen	Das Erstellen eines Kurses	Keine
Kurse	Editieren	Das Editieren eines Kurses	Keine
Kurse	Löschen	Das Löschen eines Kurses	Abhängige Einheiten, Lektionen, sowie Aufgaben werden ebenfalls gelöscht
(Lern)Einheiten	Erstellen	Das Erstellen einer (Lern)Einheit	Keine
(Lern)Einheiten	Editieren	Das Editieren einer (Lern)Einheit	Keine
(Lern)Einheiten	Löschen	Das Löschen einer (Lern)Einheit	Abhängige Lektionen, sowie Aufgaben werden ebenfalls gelöscht
(Lern)Einheiten	Lektion hinzufügen	Das Erstellen einer Lektion zur (Lern)Einheit	Keine
Lektionen	Erstellen	Das Erstellen einer Lektion	Keine
Lektionen	Editieren	Das Editieren einer Lektion	Keine
Lektionen	Löschen	Das Löschen einer Lektion	Aufgaben werden ebenfalls gelöscht
Aufgaben	Erstellen	Das Erstellen einer Aufgabe	Keine
Aufgaben	Editieren	Das Editieren einer Aufgabe	Keine
Aufgaben	Löschen	Das Löschen einer Aufgabe	Keine

4.1.3 Auslieferung und Installation

von Julian Schilling

Das System wird als ZIP-Archiv `in4all.zip` ausgeliefert werden. Im Folgenden wird beschrieben, wie die einzelnen Bestandteile genutzt werden. Das Projekt unterstützt die Option, die drei Bestandteile zu trennen; hochgeladene Videos und Bilder müssen bei Stand des Projektabschlusses jedoch stets auf dem Server abgelegt werden, auf welchem das CoMa ausgeführt wird. Ferner unterstützt das Backend zu diesem Zeitpunkt nicht, auf andere Weise als via `localhost` auf die MongoDB zuzugreifen.

1. Das Archiv ist zu entpacken. Dieses enthält die Ordner App, Coma, Backend und node_modules.

2. Die Konfigurations-Dateien `config.js` in den jeweiligen Ordnern sind an die vorhandenen Gegebenheiten anzupassen. Dies bedeutet:

   (a) `mongoPath` muss angepasst werden, sodass der Pfad dem korrekten Datenbanknamen entspricht. Wird MongoDB also wie empfohlen auf demselben Server wie das Backend gehostet und heisst die Datenbank `Talentkolleg`, muss `mongodb://localhost/Talentkolleg` eingetragen werden.

   (b) `backendPort` und `comaPort` müssen geändert werden, falls die Verwendung anderer Ports gewünscht wird (etwa, weil die dort eingetragenen Ports bereits anderweitig verwendet werden). Die Änderung muss sich in der Apache-Konfiguration widerspiegeln.

   (c) `backendPath` muss auf die URL eingestellt werden, unter welcher das Backend erreichbar sein wird, z.B. `http://api.in4all-pg.de`. Das führende `http://` ist hierbei unbedingt zu beachten!

   (d) `videoPath` und `picturePath` geben den Ort an, unter welchem das Coma hochgeladene Videos und Bilder ablegt. Der Nutzeraccount, welcher das Coma später ausführt, *muss* in diesen Ordnern Schreibrecht haben.

   (e) `videoUrl` und `pictureUrl` geben die URL an, unter welcher die hochgeladenen Videos und Bilder abgerufen werden können, beispielsweise `http://assets.in4all-pg.de/videos` und `http://assets.in4all-pg.de/images`. Das Projekt sieht momentan nicht vor, dass diese Daten vor fremden Zugriffen geschützt sind. Es kann also jeder mithilfe der vollständigen URL eines Bildes oder eines Videos auf diese zugreifen.

   Es ist anzumerken, dass *nur* der Inhalt der Variable `in4allConfig` verändert werden darf; der Rest der Datei ist *nicht* zu verändern!

3. Wurden diese Schritte vorgenommen, sind die einzelnen Bestandteile nun dazu bereit, online zur Verfügung gestellt zu werden. Hierzu sind folgende Schritte nötig:

   (a) **App** — Das Frontend ist der simpelste Teil. Es genügt, den Ordner von einem Apache-Server ausliefern zu lassen, welcher die Option `AllowOverride: All` gesetzt hat (da in der `.htaccess`-Datei zum Funktionieren unablässige Optionen definiert sind). Ein Beispiel für die Apache-Konfiguration findet sich im Anhang A.4.2, für die `.htaccess` in Anhang A.4.3.

   (b) **Coma** und **Backend** — CoMa und Backend werden über den installierten Node-Server an den Ports gestartet, die in Punkt 2b definiert wurden. Diese Ports müssen von Apache dann an die jeweils verwendeten URLs ausgeliefert werden; ein Beispiel für eine solche Konfiguration findet sich in Anhang A.4.1. Es ist zu beachten, dass die URL des Backends mit der in Punkt 2c festgelegten übereinstimmen muss. Zum bequemen Starten, Überwachen und Beenden empfiehlt diese PG das Programm `forever` und eine Einschränkung des Arbeitsspeichers, da Node ohne diese Einschränkung stets soviel Arbeitsspeicher zu belegen versucht wie es kann[18]. Dies kann zu ungewolltem Verhalten auf Servern führen. 4GB haben sich bei vorgenommenen Tests als ausreichend herausgestellt; wirklich genutzt wurden diese nur während im CoMa Dateien hochgeladen wurden. Ein Beispiel für kleine Skripte, welche `forever` zum Starten und Stoppen von CoMa und Backend nutzen, finden sich in Anhang A.3.

Ab diesem Zeitpunkt sind alle Bestandteile von IN4ALL installiert und einsatzbereit.

4.1.4 Import von Daten

von Julian Schilling

Nachdem das System gemäß Abschnitt 4.1.3 aufgesetzt wurde, ist es auf einem neuen System auch noch in einem „leeren“ Zustand. Um einen vorherigen Stand wiederherzustellen, müssen Datenbank, Bilder und Videos übernommen werden. Die Daten der PG werden hierbei als `database.zip` (Datenbank) und `assets.zip` (Bilder und Videos) übergeben.

Datenbank Das Archiv `database.zip` muss zunächst entpackt und ein Terminal im entpackten Ordner geöffnet werden. Danach kann mithilfe des Befehls `mongorestore -db InForAll InForAll/` die Datenbank wiederhergestellt werden. Hierbei ist `InForAll` der Name, welcher der importierten Datenbank in der lokalen MongoDB gegeben wird. `InForAll/` bezeichnet den Ordner, in welchem die Daten enthalten

[18] Installation mittels `npm install -g forever`

sind. Näheres ist der MongoDB-Dokumentation[Mon17a] unter dem Punkt „mongorestore“ zu entnehmen.

Bilder und Videos Das Archiv `assets.zip` enthält die Ordner `images` und `videos`. Diese müssen so entpackt werden, dass ihre Inhalte unter den in Schritt 2e von Abschnitt 4.1.3 definierten Urls abgerufen werden können. Im Falle der PG waren dies die Unterordner `videos` und `images` des Ordners `/var/www/assets.in4all-pg.de`. Es ist zu beachten, dass die Namen der Videos nicht geändert werden dürfen, da sie andernfalls nicht mehr mit der Datenbank übereinstimmen.

4.2 Produzierte Lehrinhalte

von Fabian Pawolowski

Zu den vier Inhaltsbereichen der Bildungsstandards wurden folgende Skripte erstellt:

Informatiksysteme

1. Grundlagen Informatiksysteme
2. EVA-Prinzip 1
3. Grundlagen Datenspeicherung
4. Unterschied zwischen Betriebssystem und Anwendersoftware
5. Grafische Benutzeroberfläche
6. Dateiformate unterscheiden
7. EVA-Prinzip 2

Information und Daten

- Einführung Datentypen
- Verzeichnisbäume

Sprachen und Automaten

- Einführung Automaten
- Formale Sprachen
- Formale Grammatik

- Überführung von umgangssprachlichen Handlungsvorschriften in formale Darstellung

Algorithmen

- Grundlagen Algorithmen
- Grundlegende Bausteine von Algorithmen
- Sortieralgorithmen

4.3 Verwendung des Systems

4.3.1 Frontend für Schülerinnen und Schüler

von Tolgay Usul

Im folgenden kommt eine Anleitung mit Bildern über die Verwendung des MooCs für Schüler und Schülerinnen:

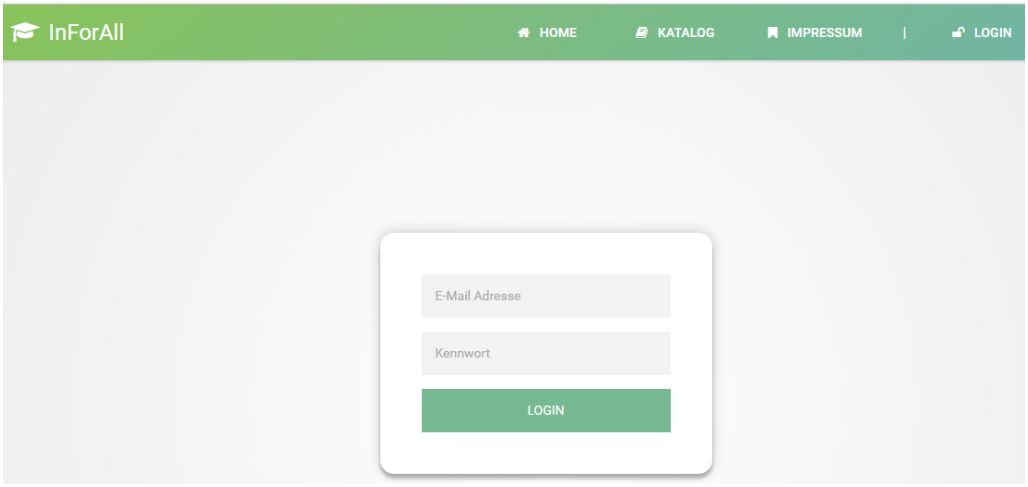


Abbildung 11: Anmeldung

In Abbildung 11 muss der Nutzer sich mit E-Mail und Passwort anmelden. Wenn die Login-Daten korrekt sind, gibt es eine Weiterleitung auf die MooC Hauptseite.

Nachdem Login ist es möglich auf der Hauptseite Hintergrundinformationen zum MooC, Zugriff zum Katalog und zum eigenen Profil zu erhalten (siehe Abbildung 12). Im Katalog ist es möglich freigegebene Kurse zu besuchen und abzuschließen.

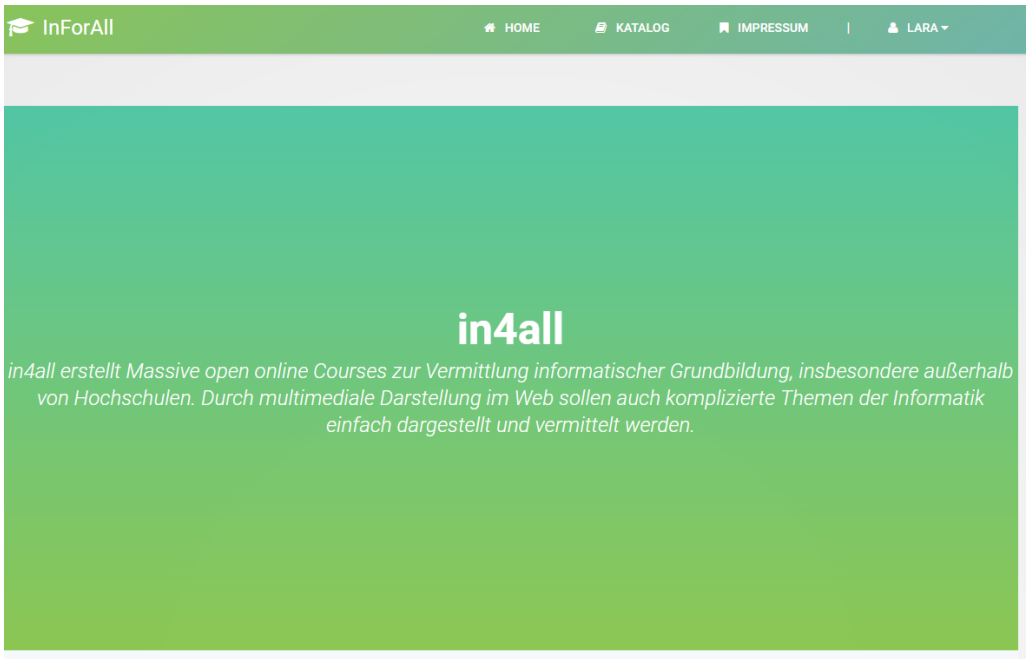


Abbildung 12: Hauptseite

Beim Klick auf Profil können die eigenen Nutzerinformationen geändert werden (siehe Abbildung 13).

Durch den Klick auf Katalog gelangt der Nutzer zur Kursauswahl (siehe Abbildung 14). Alle freigegebenen Kurse sind hier mit Anzahl der Lerneinheiten und jeweils einer Beschreibung aufgelistet.

Der Klick auf einen Kurs bringt den Nutzer zu einer Auflistung aller Lerneinheiten des Kurses (siehe Abbildung 15) . Alle freigegebenen Lerneinheiten sind hier mit Anzahl der Lektionen und jeweils einer Beschreibung aufgelistet.

Durch den Klick auf eine Lerneinheit öffnet sich ein c.a fünf minütiges Lernvideo zur Lektion (siehe Abbildung 16) .

Darunter befinden sich Aufgaben zur Lektion. Die Aufgaben beziehen sich auf das Lernvideo (siehe Abbildung 17). Der Schüler kann einzelne Aufgaben lösen oder überspringen. Am Ende der Aufgabe erhält der Schüler ein Feedback zu seinen Lösungen. Es werden die erreichten Punkte aufgelistet.

In diesem Modus kann der Schüler es noch einmal versuchen, die Lösungen ansehen, das Lernvideo erneut abspielen oder eine andere Lektion auswählen (siehe Abbildung 18).

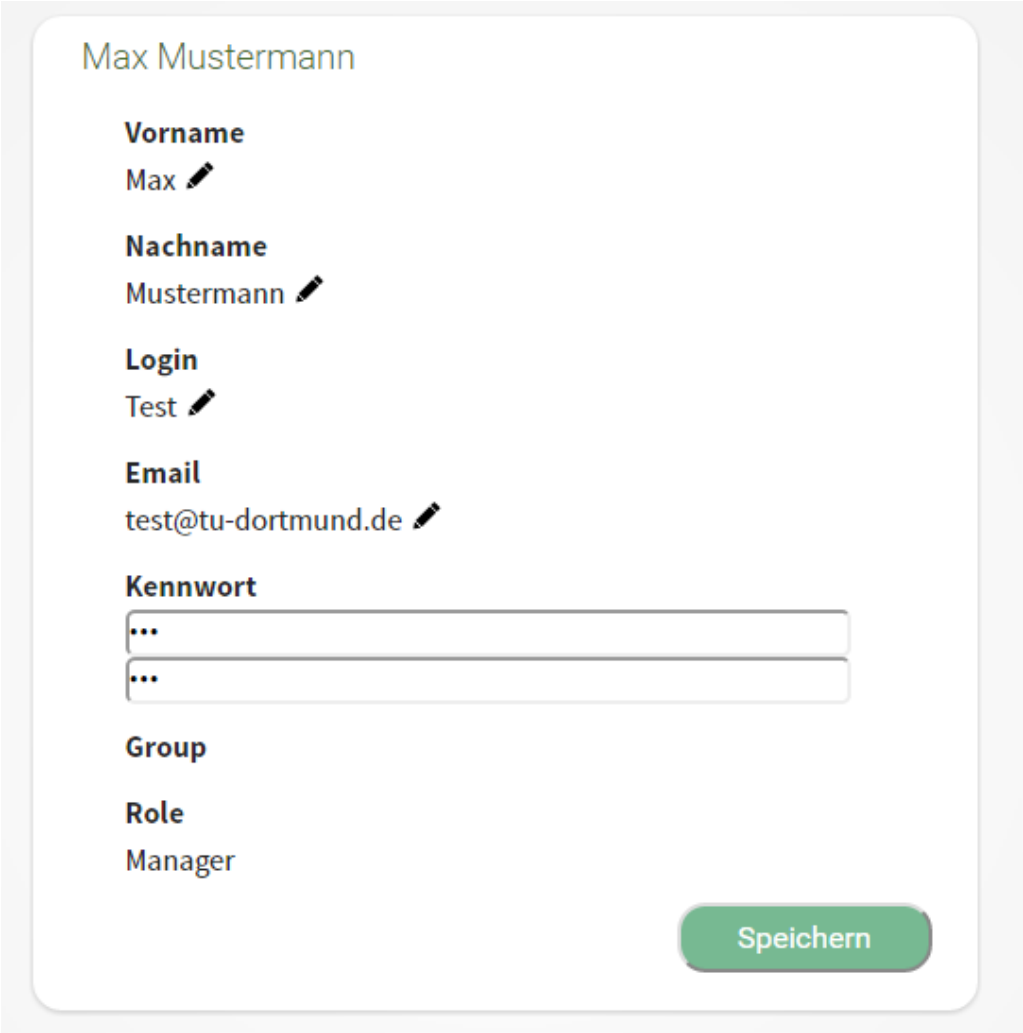


Abbildung 13: Profil des Nutzers

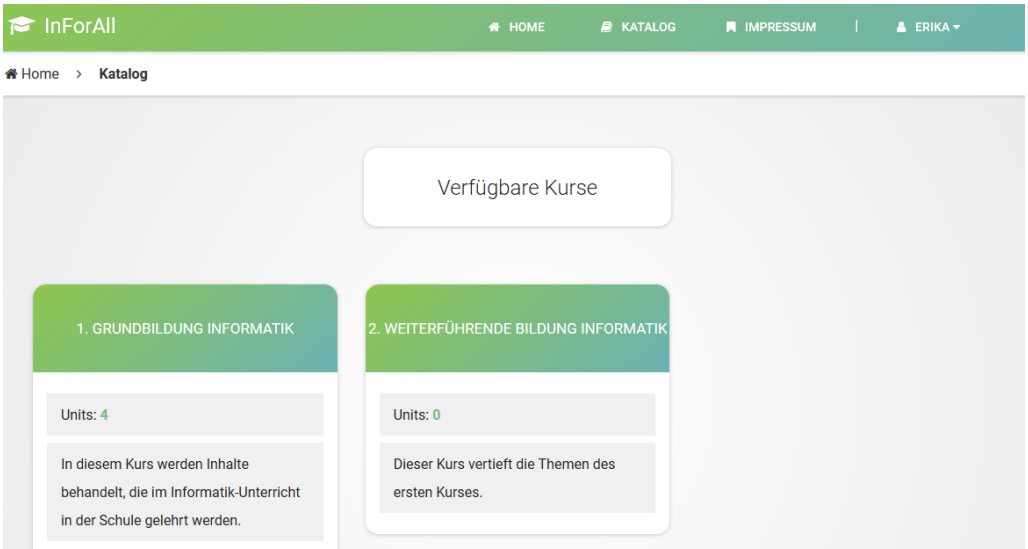


Abbildung 14: Auswahl des Kurses

Abbildung 15: Auswahl der Lerneinheit

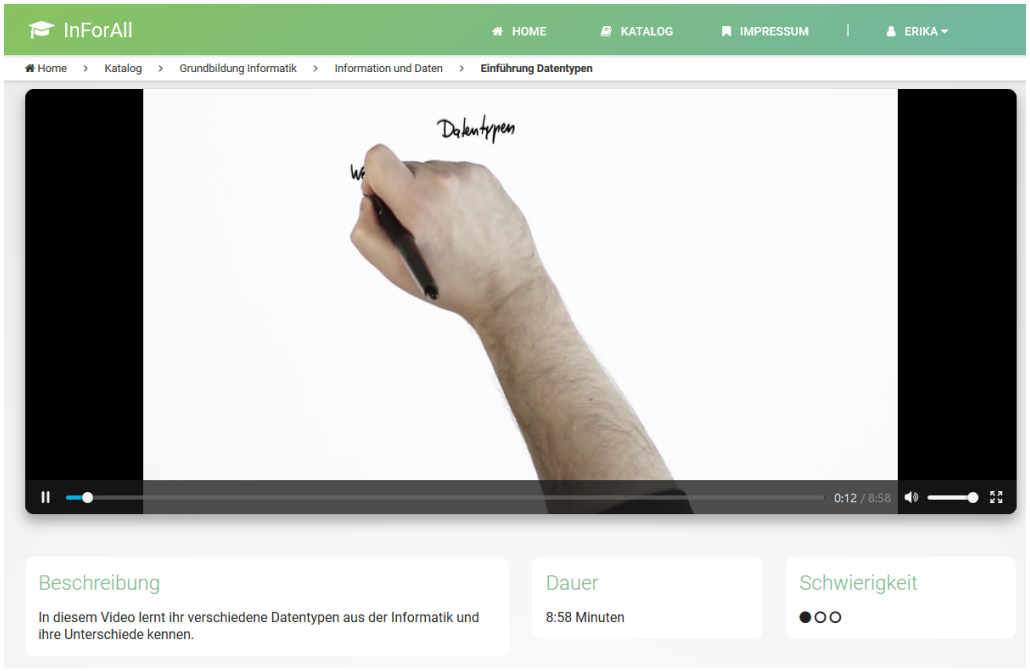


Abbildung 16: Lernvideo

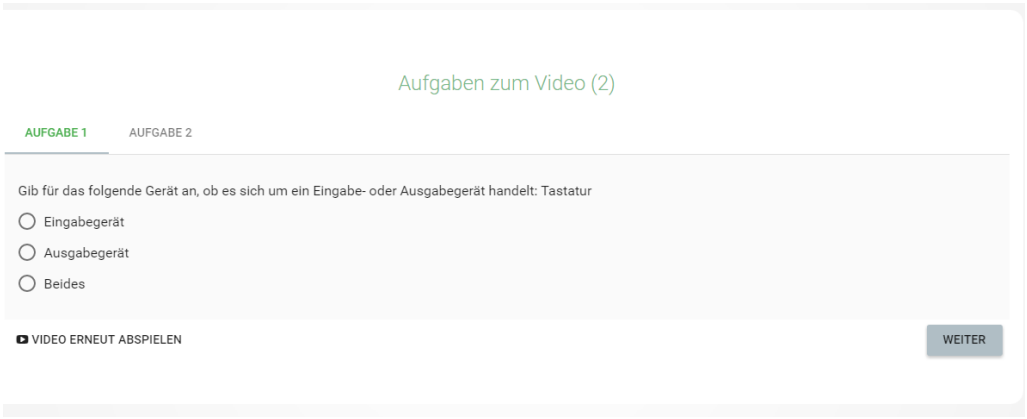


Abbildung 17: Aufgabe zur Lektion

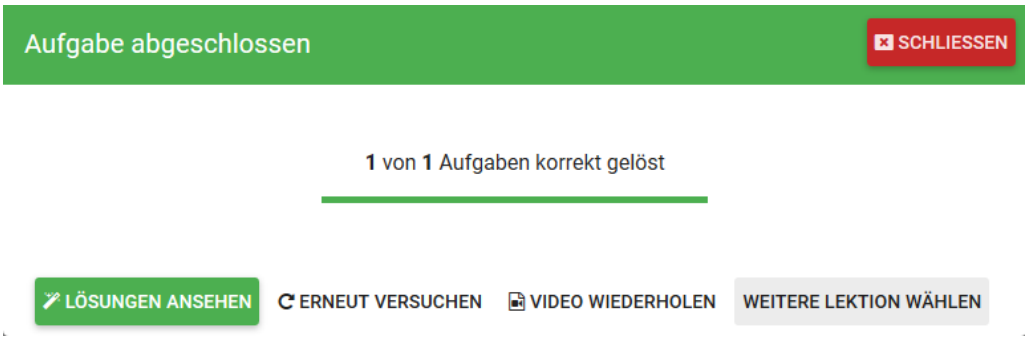


Abbildung 18: Fenster bei Aufgabenabgabe

4.3.2 Content Management für Lehrer und Administratoren

von Tolgay Usul

Im Folgenden eine Anleitung mit Bildern über die Verwendung des Content Managers (CoMa) für Lehrer und Administratoren:

Grundlagen Der Aufbau der Seiten für alle Objektgruppen ist sehr ähnlich, deshalb wird in diesem Paragraphen der allgemeine Aufbau der Seiten erläutert. Dieser Aufbau gilt für alle Objektgruppen. Auf Besonderheiten von einzelnen Objektgruppen wird später eingegangen.

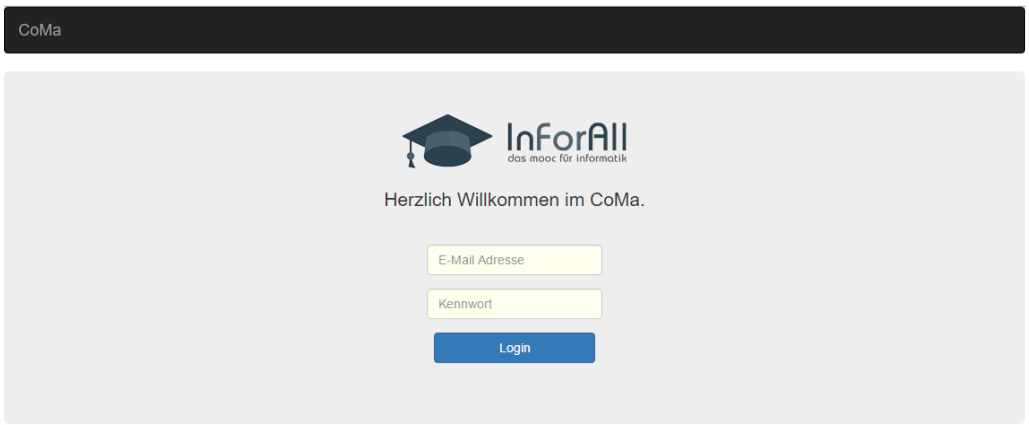


Abbildung 19: Loginseite

Zunächst muss der Administrator/Lehrer sich mit E-Mail und Passwort anmelden (siehe Abbildung 19) . Wenn die Login-Daten korrekt sind, gibt es eine Weiterleitung auf die CoMa Hauptseite.

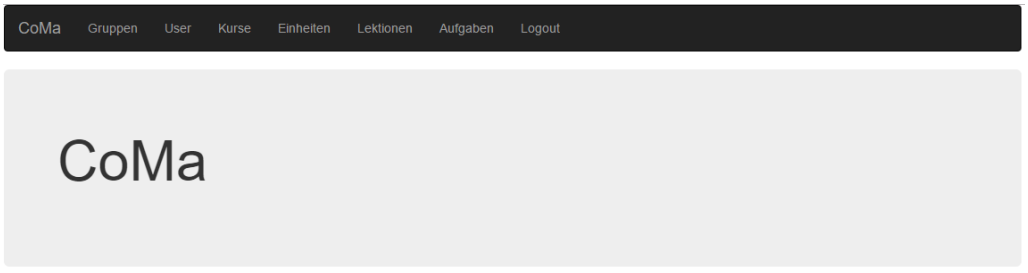


Abbildung 20: Hauptseite

Auf der Hauptseite hat der Administrator/Lehrer Zugriff auf alle vorhandenen Daten im MooC (siehe Abbildung 20). Er kann diverse Objekte (Gruppen, User(Nutzer), Kurse, Lerneinheiten, Lektionen und Aufgaben) erstellen, verändern oder löschen. Durch den Klick auf den entsprechenden Menüpunkt gelangt der Administrator/Lehrer zur gewünschten Objektgruppe und kann Veränderungen vornehmen.

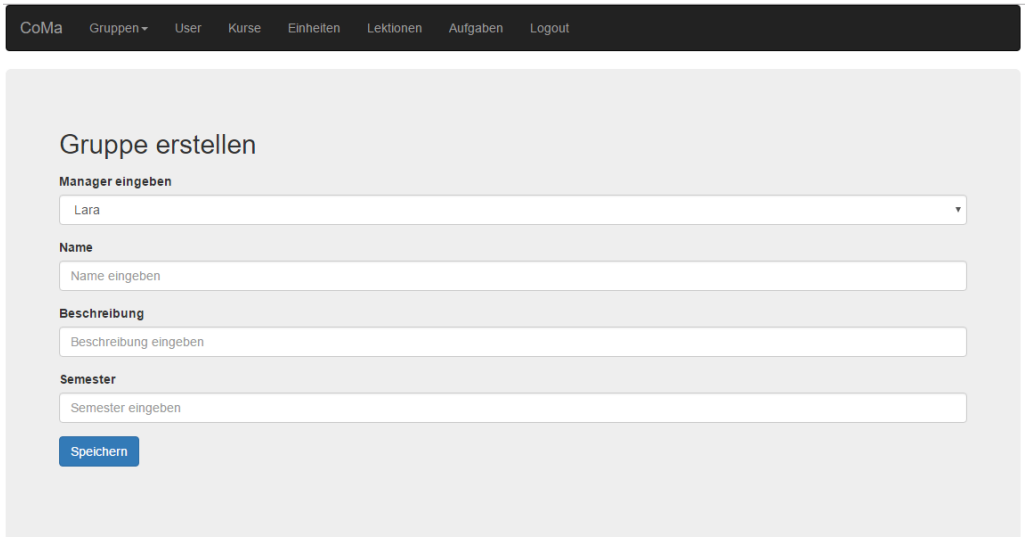


Abbildung 21: Gruppe erstellen

Bei der Auswahl einer Objektgruppe kommt eine Seite für das Erstellen eines Objektes zum Vorschein (siehe Abbildung 21). Auf dieser Seite kann der Administrator/Lehrer einen Objekt für die entsprechende Objektgruppe erstellen. Zum Speichern müssen alle notwendigen Informationen eingegeben werden.

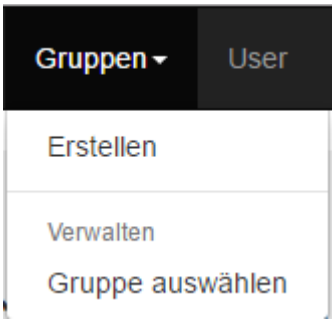


Abbildung 22: Dropdown Menü

Durch die Auswahl einer Objektgruppe erscheint bei einem weiteren Klick auf den Objektnamen ein Dropdown Menü (siehe Abbildung 22) , welches zusätzliche Optionen liefert, wie das Anzeigen aller Objekte.

Bei der Auswahl Objekt verwalten sieht man alle erstellten Objekte unter der jeweiligen Objektgruppe (siehe Abbildung 23). Die Auswahl eines Objektes, öffnet den Editiermodus des jeweiligen Objektes.

Im Editiermodus können die Informationen des jeweilige Objektes der Objektgruppe verändert werden (siehe Abbildung 24). Einzig die Identifikationsnummer ist unveränderbar. Zusätzlich ist es möglich das Objekt zu löschen. Bei der Löschung werden das Objekt und

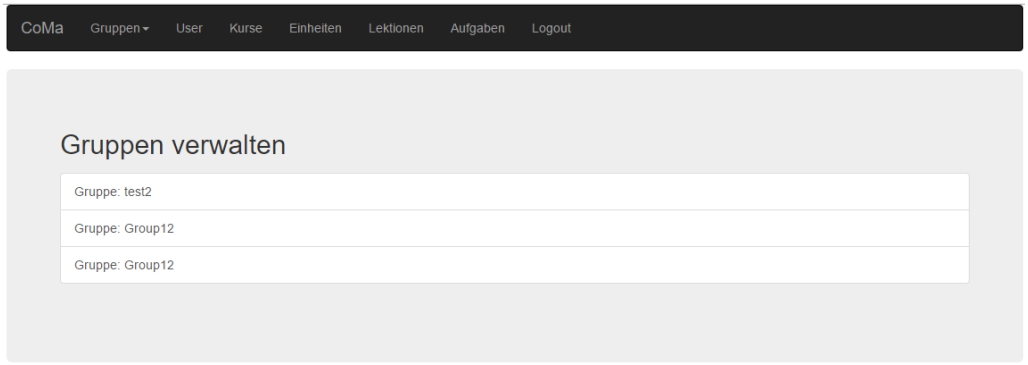


Abbildung 23: Gruppe auswählen

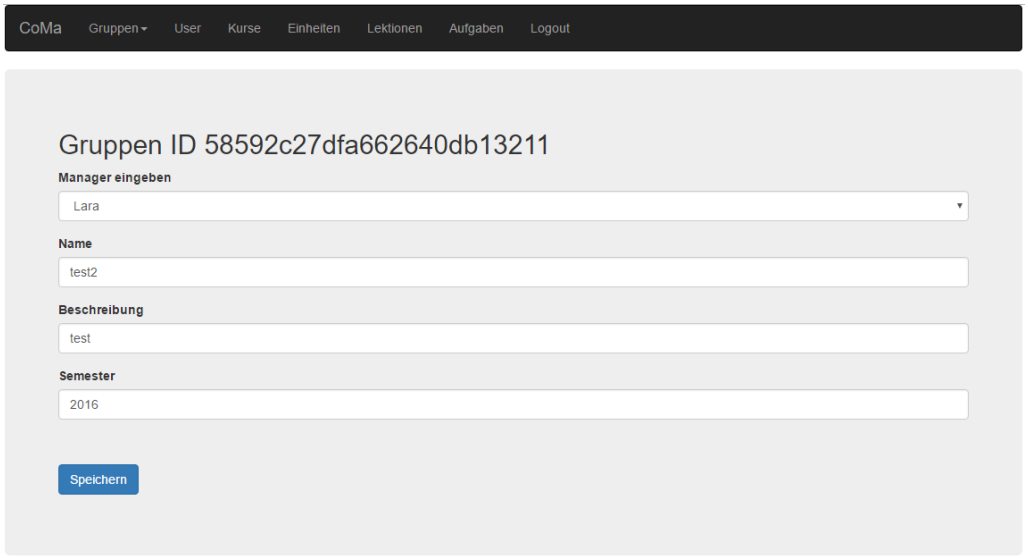


Abbildung 24: Gruppe editieren

seine Abhängigkeiten (alle Objekte aus anderen Objektgruppen, die keine Existenzgründe mehr haben,) entfernt. Beim Klicken des Löschen-Knopfes wird jeweils ein Warnhinweis mit Zwangsbestätigung eingeblendet.

Besonderheiten Einige der Objektgruppen haben, wegen ihrer Abhängigkeiten weitere Seiten. Sie unterschieden sich kaum von der Grundstruktur. Im nachfolgenden werden diese Besonderheiten behandelt.

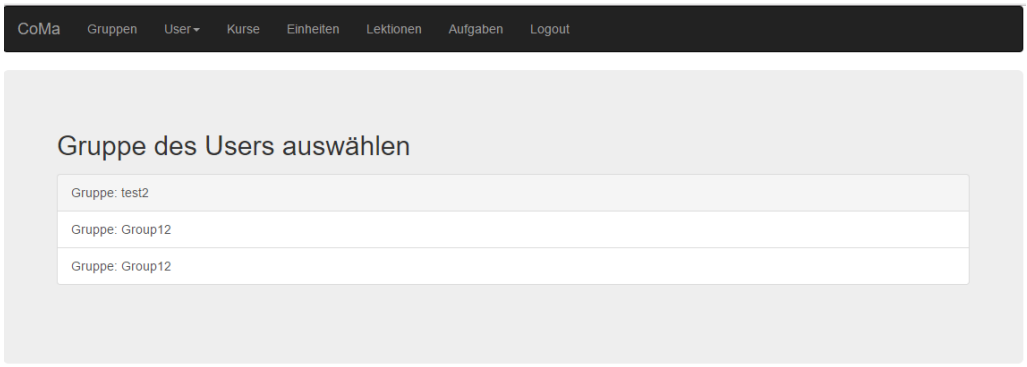


Abbildung 25: Abhängige Gruppe vom Nutzer auswählen

Auf der in Abbildung 25 abgebildeten Seite, muss der Administrator die jeweilige Abhängigkeit des Objektes auswählen. In der Abbildung wäre, dass die Auswahl der Gruppe zu dem der Nutzer hinzugefügt/verändert/gelöscht werden soll.
Wichtig ist, dass durch die Löschung der Abhängigkeit, das Objekt selbst ebenfalls gelöscht wird. Die Objektgruppen, die direkte Abhängigkeiten haben, sind User (zu Gruppen), Lektionen (zu Einheiten) und Aufgaben (zu Lektionen).

Das Hinzufügen/Verändern von Aufgaben ist ein wenig anders, als im Vergleich zu anderen Objektgruppen (siehe Abbildung 26). Zusätzlich zu den Aufgaben muss der Administrator/Lehrer eine passende Syntax für das Erstellen von Aufgabentypen eingeben. Die Syntax wird in Kapitel 3.3.5 näher erläutert. Eine Anleitung in Englisch wird auf der Seite abgebildet.

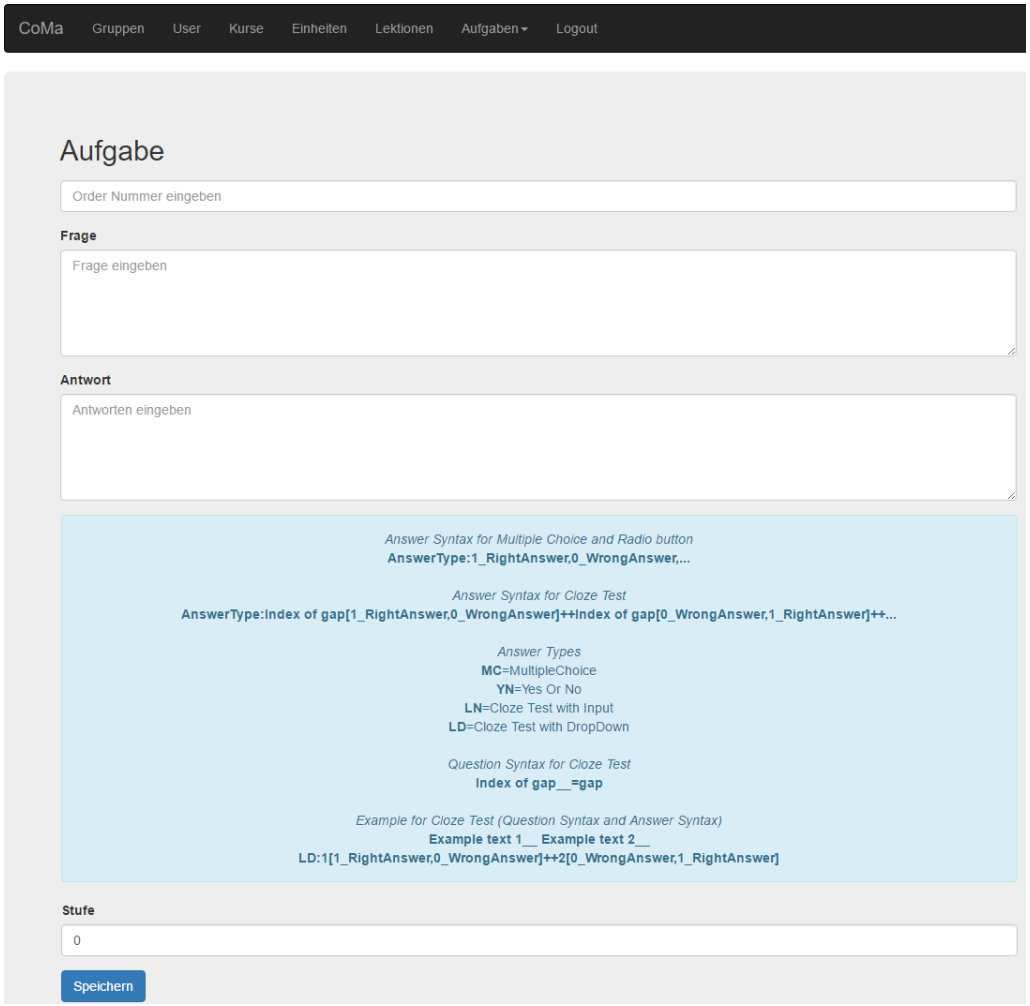


Abbildung 26: Hinzufügen von Aufgaben

A Anhang

A.1 REST API Dokumentation

von Christian Everke

Die Business-Logik Schicht des IN4ALL-Projektes, auch als Backend bezeichnet, ist, wie in Kapitel 2.2.2 beschrieben, jene Softwarekomponente, über die die Verbindung zur Datenbank hergestellt wird. Um die sogenannten CRUD-Operation, Create (Erstellen), Read (Lesen), Update (Aktualisieren) und Delete (Löschen), von anderen Komponenten (User Frontend, CoMa-Frontend oder für die Zukunft vorgesehene Komponenten wie z.B. Smartphone-Apps) zugänglich zu machen, stellt das Backend eine REST-Schnittstelle [TESW15] bereit. Die Dokumentation dieser Schnittstelle ist unter `http://in4all-pg.de/doc` erreichbar und wurde mit dem Software-Tool *Swagger*[19] erstellt. Eine Zusammenfassung dieser Dokumentation stellt dieses Kapitel dar.

Für die grundsätzliche Zurverfügungstellung der CRUD-Operationen der Dateninstanzen stehen für jede Klasse aus Abbildung 3 Endpunkte in Form von *URI*s zur Verfügung, die die HTTP-Methoden *GET, PUT, POST* und *DELETE* [TESW15] anbieten.

Im Folgenden sind die verfügbaren URIs aufgelistet. Als Basispfad ist `api.in4all-pg.de/` zu verwenden.

AttendCourse

- DELETE /attendCourse/$\langle attendCourseId \rangle$
- GET /attendCourse/$\langle attendCourseId \rangle$
- POST /attendcourse
- PUT /attendcourse

Course

- DELETE /course/$\langle courseId \rangle$
- GET /course/$\langle courseId \rangle$
- GET /course
- POST /course
- PUT /course

[19] `http://swagger.io/`

Exercise

- DELETE /exercise/⟨*exerciseId*⟩
- GET /exercise/⟨*exerciseId*⟩
- GET /exercise
- POST /exercise
- PUT /exercise

ExerciseAttempt

- DELETE /exerciseAttempt/⟨*exerciseAttemptId*⟩
- GET /exerciseAttempt/⟨*exerciseAttemptId*⟩
- GET /exerciseAttempt
- POST /exerciseAttempt
- PUT /exerciseAttempt

Group

- GET /group
- DELETE /group/⟨*groupId*⟩
- GET /group/⟨*groupId*⟩
- POST /group
- PUT /group

Lesson

- GET /lesson
- DELETE /lesson/⟨*lessonId*⟩
- GET /lesson/⟨*lessonId*⟩
- POST /lesson
- PUT /lesson

Unit

- GET /unit
- POST /unit
- PUT /unit
- DELETE /unit/$\langle unitId \rangle$
- GET /unit/$\langle unitId \rangle$

User

- GET /user
- POST /user
- PUT /user
- DELETE /user/$\langle userId \rangle$
- GET /user/$\langle userId \rangle$

Sollen einzelne, bestimmte Datensätze abgefragt oder gelöscht werden, so ist muss die benötigte Ressource über den Datenbank-Primärschlüssel (ID) identifiziert werden. Daher benötigen eine Vielzahl HTTP-GET und HTTP-DELETE Methoden der oben aufgeführten URIs die Angabe der entsprechenden ID im Pfad. HTTP-GET Anfragen ohne Angabe einer ID liefern alle Datensätze als Liste zurück, die zur Klasse in der Datenbank vorhanden sind. Zum Ändern oder Hinzufügen von Daten müssen diese im bekannten JSON-Format mit Hilfe des Request-Header des HTTP-Befehls übertragen werden. Nach Ausführung eines REST-Requests sendet der Server einen HTTP-Status Code, der über den Erfolg des Anfrage Auskünfte gibt. Im Projekt IN4ALL wurden unter anderem verwendet: 200 (*OK*), 401 (*Unauthorized*), 404 (*Not found*).

A.2 Softwareüberlassungsvertrag

Der folgende Softwareüberlassungsvertrag wurde von Herrn Rechtsanwalt Wolfgang Müller freundlicherweise entworfen und der Projektgruppe zur Verfügung gestellt. Er regelt die Überlassung der Software an das Talentkolleg Ruhr und ist eigens für dieses Vertragsverhältnis entworfen worden. Es ist grundsätzlich nicht davon auszugehen, dass dieser Vertrag sich auf andere Situationen anwenden lässt. Es ist weiterhin nicht erwünscht, den Vertrag ohne Rücksprache mit Herrn Müller für weitere Zwecke zu verwenden.

Software-Überlassungsvertrag

Zwischen

der Projektgruppe 599 „*in4all*“ der Fakultät für Informatik der Technischen Universität Dortmund, diese bestehend aus den Mitgliedern ...

diese vertreten durch den Projektgruppenleiter Christian Everke

- im Folgenden „*Anbieter*“ genannt -

und

dem Talentkolleg Ruhr,

dieses vertreten durch

- im Folgenden „*Kunde*“ genannt -

wird der folgende Software-Überlassungsvertrag geschlossen:

1 Vertragsgegenstand

1.1 Der Anbieter überlässt dem Kunden für die Laufzeit dieses Vertrages unentgeltlich (Leihe) das Standard-Client-Server-Computerprogramm „.....“ in der Version Nr. ..... mit Stand vom 00.00.2017. Eine Überlassung von Aktualisierungen des Computerprogrammes erfolgt – außer zum Zwecke der Mängelbeseitigung – nur auf Grund gesonderter Vereinbarung.

1.2 Bei dem Computerprogramm handelt es sich inhaltlich um ein Programm, mit dessen Hilfe vom Kunden Dritten digitale Lernkurse in Form von sog. MOOC (Massive Open Online Courses) angeboten werden können. Hierbei können vom lernenden Dritten die Lerninhalte in individuellem Tempo abgerufen und der Lernerfolg des Kurses mittels darin enthaltener Aufgaben überprüft werden. Dadurch soll versucht werden, eine ausschließlich passive Haltung (wie sie z.B. beim reinen Konsumieren einer gefilmten Vorlesung auftreten könnte) des Lernenden zu vermeiden, indem nach oft sehr kurzen Vermittlungssequenzen, eine Form der aktiven Wissenssicherung (z.B. durch eine kurze Rechen- oder Multiple-Choice-Aufgabe) erfolgt. So kann entsprechender Kurs mit bis zu 30 dieser Aufgaben zur Lernkontrolle unterbrochen sein.
Der Kunde ist hierbei insbesondere auch in der Lage über eine separate Client-Oberfläche (sog. Content-Management) die vorhandenen Lerninhalte zu bearbeiten und zu verändern, so insbesondere neue Inhalte und Aufgaben hinzuzufügen oder zu löschen.

1.3 Bei dem Computerprogramm handelt es sich technisch um eine Client-Server-Anwendung die dem Kunden zur Verfügung gestellt wird. Die teilnehmenden Dritten sind über Web-/Browser-basierte Client-Applikationen in der Lage auf das Programm d.h. seinen Inhalt Zugriff zu nehmen und die Übungen auszuführen.

1.4 Der Kunde erhält das Computerprogramm installationsbereit im Objektcode nebst einer Anwendungsdokumentation (Installationsanleitung und Benutzerhandbuch).

1.5 Der Funktionsumfang des Computerprogramms sowie die Hard- und Software-Einsatzbedingungen ergeben sich aus **Anlage 1** zu diesem Vertrag. In **Anlage 1** ist außerdem die Systemumgebung (Client, Server und Netzwerk) beschrieben, in der das Programm genutzt werden darf.

Anlage 1 enthält auch eine Auflistung und Beschreibung von Lerninhalten und Aufgaben, die vom Anbieter exemplarisch im Computerprogramm vor Überlassung desselben zu Demonstrationszwecken hinterlegt worden sind. **Anlage 1** ist Vertragsbestandteil.

2 Überlassung, Installation, Beratung

2.1 Der Anbieter überlässt dem Kunden das Computerprogramm einschließlich der Anwendungsdokumentation ab dem **00.00.2017** für die Laufzeit des Vertrages unentgeltlich.

2.2 Der Kunde installiert die Software selbst.

2.3 Der Anbieter schuldet Beratungsleistungen nur, sofern dies ausdrücklich und gesondert vereinbart wird.

2.4 Anpassungen bzw. Änderungen des Computerprogramms sowie die Erstellung von Schnittstellen zu Dritt-Programmen durch den Anbieter sind nur geschuldet, soweit diese zur Instandhaltung bzw. Instandsetzung des Vertragsgegenstandes bzw. zur Sicherung des vertragsgemäßen Gebrauchs erforderlich sind.

3 Nutzungsrechte an der Software, Nutzung im Netzwerk

3.1 Der Anbieter räumt dem Kunden das einfache, nicht übertragbare Recht ein, das überlassene Computerprogramm im Objektcode sowie die sonstigen Komponenten zum vorstehend beschriebenen vertraglichen Zweck am und für den Standort Dortmund und nach Maßgabe der nachfolgenden Bestimmungen der **Ziffer 4** und **Ziffer 5**, befristet für die Dauer der Laufzeit dieses Vertrages zu nutzen.

3.2 Ist die Nutzung des Computerprogramms auf dem hierfür bezeichneten Rechner des Kunden zeitweise, insbesondere wegen Störungen oder wegen Reparatur- bzw. Wartungsarbeiten nicht oder nur eingeschränkt möglich, so ist er berechtigt, das Computerprogramm übergangsweise auf einem Austausch-Rechner zu nutzen. Bei einem dauerhaften Wechsel des Rechners ist die Nutzung des Computerprogramms auf dem neu eingesetzten Rechner; das Computerprogramm ist auf dem zuvor eingesetzten Rechner vollständig zu löschen. Der Kunde ist verpflichtet dem Anbieter den dauerhaften Einsatz auf einem anderen Rechner und die Löschung des Computerprogramms auf dem ursprünglichen Rechner unverzüglich nach dem dauerhaften Wechsel schriftlich anzuzeigen.

4 Vervielfältigung des Computerprogramms

Der Kunde ist zur Vervielfältigung des Computerprogramms sowie der Dokumentation berechtigt, wenn und soweit dies für die bestimmungsgemäße Nutzung notwendig oder gesetzlich erlaubt ist. Der Kunde ist insbesondere berechtigt, Kopien des Programms zu erstellen, soweit diese zur Sicherheit der künftigen Nutzung des Programms sowie zu Zwecken einer den betrieblichen Anforderungen des Kunden entsprechenden Datensicherung und Archivierung erforderlich sind.

5 Überlassung der Software an Dritte

5.1 Der Kunde ist ohne Erlaubnis des Anbieters nicht berechtigt, die Software Dritten zu überlassen, insbesondere diese zu veräußern oder zu vermieten.

5.2 Die unselbständige Nutzung durch die Arbeitnehmer des Kunden bzw. sonstige dem Weisungsrecht des Kunden unterliegende Dritte im Rahmen des bestimmungsgemäßen Gebrauch ist zulässig.

6 Vertragslaufzeit, Beendigung des Leihverhältnisses

6.1 Das Leihverhältnis beginnt zum 00.00.0000 und hat eine Laufzeit von 24 Monaten. Es verlängert sich, sofern es nicht mit einer Frist von einem Monat gekündigt wird, automatisch um weitere 12 Monate.

6.2 Das Recht jeder Partei zur außerordentlichen Kündigung aus wichtigem Grund bleibt unberührt.

6.3 Eine Kündigung bedarf zu ihrer Wirksamkeit der Schriftform.

7 Rückgabe

8.1 Bei Beendigung des Vertragsverhältnisses hat der Kunde dem Anbieter das Computerprogramm auf den Originaldatenträgern einschließlich Handbüchern und Dokumentation

zurückzugeben. Gegebenenfalls erstellte Kopien des vom Anbieters überlassenen Programms sind vollständig und endgültig zu löschen.

8.2 Jede Nutzung des Computerprogramms nach Beendigung des Vertragsverhältnisses ist unzulässig.

9 Sonstige Vereinbarungen

9.1 Änderungen und Ergänzungen sämtlicher zwischen den Parteien geschlossener Verträge sollen nur schriftlich vereinbart werden. Textform (126b BGB) genügt diesem Schriftformerfordernis. Soweit vertraglich ausdrücklich Schriftform vereinbart worden ist (z.B. für eine Vertragsänderung oder eine Kündigung) genügt Textform nicht. Mündliche Absprachen gelten nur, wenn sie binnen sieben Tagen in Textform durch den Anbieter bestätigt werden; ein Fax bzw. eine E-Mail genügt dem Schriftformerfordernis.

9.2 Der Anbieter und der Kunde sind verpflichtet, über Geschäfts- und Betriebsgeheimnisse sowie über sonstige als vertraulich bezeichnete Informationen, die im Zusammenhang mit ihrem Vertragsverhältnis bzw. der daraus resultierenden Vertragsbeziehung bekannt werden, Stillschweigen zu wahren. Die Weitergabe solcher Informationen an Personen, die nicht an dem Abschluss, der Durchführung oder der Abwicklung des Vertragsverhältnisses beteiligt sind, darf – soweit nicht eine gesetzliche Verpflichtung besteht - nur mit ausdrücklicher schriftlicher Einwilligung des Vertragspartners erfolgen. Soweit nichts anderes vereinbart ist, endet diese Verpflichtung nach Ablauf von fünf Jahren ab Bekanntwerden der jeweiligen Information, nicht jedoch vor Beendigung des zwischen dem Anbieter und dem Kunden bestehenden Vertragsverhältnisses.

9.3 Dem Anbieter und dem Kunden ist bekannt, dass eine elektronische und unverschlüsselte Kommunikation (z. B. per E-Mail) mit Sicherheitsrisiken behaftet ist. Bei dieser Art der Kommunikation werden weder der Anbieter noch der Kunde daher Ansprüche geltend machen, die durch das Fehlen einer Verschlüsselung begründet sind, außer soweit zuvor eine Verschlüsselung vereinbart worden ist.

9.4 Sämtliche Vertragsverhältnisse der Parteien unterliegen ausschließlich dem Recht der Bundesrepublik Deutschland.

10 Erfüllungsort und Gerichtsstand

10.1 Erfüllungsort für alle Verpflichtungen aus den Vertragsverhältnissen der Parteien ist der Sitz des Anbieters.

10.2 Gerichtsstand für alle Rechtsstreitigkeiten aus den Vertragsverhältnissen der Parteien sowie für Streitigkeiten in Bezug auf das Entstehen und die Wirksamkeit dieser Vertragsverhältnisse ist gegenüber Kaufleuten, einer juristischen Person des öffentlichen Rechts oder einem öffentlich-rechtlichen Sondervermögen der Sitz des Anbieters. Der Anbieter ist jedoch berechtigt, den Kunden an seinem Sitz zu verklagen.

A.3 Server-Bashscripts

von Julian Schilling

Anmerkung: Das Symbol ↪ zeigt an, dass hier kein echter Zeilenumbruch vorliegt, sondern zwecks Lesbarkeit umgebrochen wurde.

A.3.1 forever_startall

```
1 #!/bin/bash
2 forever start -c "/usr/bin/nodejs -1-max-old-space-size=4096" /opt/
    ↪ in4all/server_applications/coma/bin/www.js
3 forever start /opt/in4all/server_applications/server/bin/www.js
4 exit 0
```

A.3.2 forever_stopall

```
1 #!/bin/bash
2 forever stop /opt/in4all/server_applications/coma/bin/www.js
3 forever stop /opt/in4all/server_applications/server/bin/www.js
4 exit 0
```

A.4 Apache-Konfigurationen

von Julian Schilling

Anmerkung: Das Symbol ↪ zeigt an, dass hier kein echter Zeilenumbruch vorliegt, sondern zwecks Lesbarkeit umgebrochen wurde.

A.4.1 Backend-Konfiguration

```
1  LoadModule proxy_module modules/mod_proxy.so
2  LoadModule proxy_http_module modules/mod_proxy_http.so
3
4  <VirtualHost *:80>
5      ServerName api.in4all-pg.de
6
7
8      ProxyRequests Off
9      ProxyPreserveHost On
10     <Proxy *>
11         Order deny,allow
12         Allow from all
13     </Proxy>
14     ProxyPass / http://localhost:3005/
15     ProxyPassReverse / http://localhost:3005/
16 </VirtualHost>
```

A.4.2 Frontend-Konfiguration

```
1  <VirtualHost *:80>
2    DocumentRoot /var/www/in4all-pg.de
3    <Directory /var/www/in4all-pg.de>
4      AllowOverride All
5      Order allow,deny
6      allow from all
7    </Directory>
8    ServerName in4all-pg.de
9  </VirtualHost>
```

A.4.3 Frontend-htaccess

```
1  #Rewriting On
2  RewriteEngine On
3  #Rewrite all URLs except those ending in .html, .png, .jpg, .gif, .
       ↪ jpeg, .bmp, .js or .css
```

```
4 RewriteCond %{REQUEST_URI} !(\.html|\.png|\.jpg|\.gif|\.jpeg|\.bmp
    ↪ |\.js|\.css)$
5 RewriteRule (.*) index.html
6 #[QSA]
```

Literatur

[And11] Anderson, David J.: *Kanban - Evolutionäres Change Management für IT-Organisationen.* Heidelberg : dpunkt.verlag, 2011

[Ang17a] AngularJS: *AngularJS API Docs.* Online: `https://docs.angularjs.org/api`, 2017

[Ang17b] AngularJS: *$http-Service.* Online: `https://docs.angularjs.org/api/ng/service/$http`, 2017

[Ang17c] AngularJS: *Interceptor.* Online: `https://docs.angularjs.org/api/ng/service/$http#interceptors`, 2017

[Ang17d] AngularUI: *UI Router for Angular 1.* Online: `https://ui-router.github.io/ng1/docs/latest/`, 2017

[Ant15] Antonov, Alex: *Spring Boot Cookbook.* Birmingham : Packt Publishing Ltd, 2015

[Bay17] Bayerischen Hochschulen: *Virtuelle Hochschule Bayern.* Online: `https://www.vhb.org/startseite/`, 2017

[Ben14] Bengel, Günther: *Grundkurs Verteilte Systeme - Grundlagen und Praxis des Client-Server und Distributed Computing.* Wiesbaden : Springer Fachmedien, 2014

[BFF+08] Brinda, T. ; Fothe, M. ; Friedrich, S. ; Koerber, B. ; Puhlmann, H. ; Röhner, G. ; Schulte, C.: *Grundsätze und Standards für die Informatik in der Schule – Bildungsstandards Informatik für die Sekundarstufe I.* 2008

[BT15] Böhm, Robin ; Tarasiewicz, Philipp: *AngularJS: Eine praktische Einführung in das JavaScript-Framework.* Paderborn : dpunkt.verlag, 2015

[BVWS14] Brisinski, Niklas Spitczok v. ; Vollmer, Guy ; Weber-Schäfer, Ute: *Pragmatisches IT-Projektmanagement - Softwareentwicklungsprojekte auf Basis des PMBOK® Guide führen.* Heidelberg : dpunkt.verlag, 2014

[CH02] Cole, Hillis R. ; Haag, Judith H.: *The Complete Guide to Standard Script Formats: Screenplays.* CMC Pub., 2002

[Cou17] Coursera Inc.: *Coursera.* Online: `https://www.coursera.org/`, 2017

[Day14] Dayley, Brad: *Node.js, MongoDB, and AngularJS Web Development.* USA : Addison Wesley, 2014

[Deu17] Deutsches Klima Konsortium: *MOOC zum Klimawandel*. Online: http://www.deutsches-klima-konsortium.de/de/bildung/online-vorlesung-mooc/mooc-zum-klimawandel.html, 2017

[edX17a] edX: *edX*. Online: https://www.edx.org/, 2017

[edX17b] edX: *mooc.Org*. Online: http://mooc.org/, 2017

[Fak16] Fakultät für Informatik der Technischen Universität Dortmund: *INFO zu den Projektgruppen der Fakultät für Informatik mit Beginn im Sommersemester 2016*. Interne Dokumente, 2016

[Fow05] Fowler, Martin: *InversionOfControl*. Online: https://martinfowler.com/bliki/InversionOfControl.html, 2005

[Fow14] Fowler, Martin: *Microservices — a definition of this new architectural term*. Online: https://martinfowler.com/articles/microservices.html, 2014

[GHJV95] Gamma, Erich ; Helm, Richard ; Johnson, Ralph ; Vlissides, John: *Design Patterns - Elements of Reusable Object-Oriented Software*. II. Series. Addison-Wesley, 1995

[Git17] Git: *Git*. Online: http://git-scm.com/, 2017

[Gul17] GulpJS: *gulp.js*. Online: http://gulpjs.com/, 2017

[Hum06] Humbert, Ludger: *Didaktik der Informatik*. Wiesbaden : B. G. Teubner Verlag, 2006

[ive17] iversity Learning Solutions: *iversity*. Online: https://iversity.org/de, 2017

[JM16] Jaeger, Till ; Metzger, Axel: *Open source software*. München : Beck, 2016

[JSP17] JSPM: *JSPM*. Online: http://jspm.io/, 2017

[Kar17] Karlsruher Institut für Technologie: *MOOCs (Massive Open Online Courses): Mediales Lernen im 21. Jahrhundert*. Online: http://www.zml.kit.edu/moocs.php, 2017

[Kee14] Keese, Christoph: *Silicon Valley - Was aus dem mächtigsten Tal der Welt auf uns zukommt*. München : Albrecht Knaus Verlag, 2014

[Mar14] Mardan, Azat: *Express.js Guide: The Comprehensive Book on Express.js*. USA : CreateSpace Independent Publishing Platform, 2014

[Mas17] Massachusetts Institute of Technology (MIT: *MIT Opencourseware*. Online: https://ocw.mit.edu/index.htm, 2017

[Mon17a] MongoDB, Inc.: *MongoDB Documentation, Version 3.4*. Online: https://docs.mongodb.com/manual/, 2017

[mon17b] mongoose: *mongoose Middleware*. Online: http://mongoosejs.com/docs/middleware.html, 2017

[Mou16] Mouat, Adrian: *Docker - Software entwickeln und deployen mit Containern*. Heidelberg : dpunkt.verlag, 2016

[Noa14] Noak, Andreas: *Business Rules – Geschäftsregeln: Konzepte, Modellierungsansätze, Softwaresysteme* -. Hamburg : disserta Verlag, 2014

[OnC17] OnCampus GmbH: *OnCampus*. Online: https://www.oncampus.de/, 2017

[Pic13] Pichler, Roman: *Scrum - Agiles Projektmanagement erfolgreich einsetzen*. Heidelberg : dpunkt.verlag, 2013

[PR15] Pohl, Klaus ; Rupp, Chris: *Basiswissen Requirements Engineering - Aus- und Weiterbildung nach IREB-Standard zum Certified Professional for Requirements Engineering Foundation Level*. Heidelberg : dpunkt.verlag, 2015

[Pro16a] Projektgruppe 599, Fakultät für Informatik der Technischen Universität D.: *Pflichtenheft zum Projekt in4all*. Interne Dokumente, 2016

[Pro16b] Projektgruppe 599, Fakultät für Informatik der Technischen Universität D.: *Projektgruppenordnung*. Interne Dokumente, 2016

[PUG17] PUG: *Pug*. Online: https://pugjs.org/api/getting-started.html, 2017

[Scr16] Scrum Alliance: *Agile Atlas*. Online: https://www.scrumalliance.org/why-scrum/agile-atlas, 2016

[SCS17] SCSS: *SCSS Sprache und Dokumentation*. Online: http://sass-lang.com/, 2017

[Sel17] Selfhtml: *JavaScript/Ajax*. Online: https://wiki.selfhtml.org/wiki/JavaScript/Ajax, 2017

[Sta17] Stanford University: *Stanford Online - Lagunita*. Online: https://lagunita.stanford.edu/, 2017

[Sys17] SystemJS: *SystemJS*. Online: https://github.com/systemjs/systemjs, 2017

[Tal17] Talentkolleg Ruhr: *Talentkolleg Ruhr*. Online: http://www.talentkolleg.ruhr, 2017

[TESW15] Tilkov, Stefan ; Eigenbrodt, Martin ; Schreier, Silvia ; Wolf, Oliver: *REST und HTTP - Entwicklung und Integration nach dem Architekturstil des Web*. Heidelberg : dpunkt, 2015

[Tre14] Trelle, Tobias: *MongoDB: Der praktische Einstieg*. Düsseldorf : dpunkt.verlag, 2014

[TS17] TS: *TypeScript - Javascript that scales*. Online: https://www.typescriptlang.org/, 2017

[TU 17] TU München: *MooCs - TUM*. Online: https://www.tum.de/studium/weiterbildung/oeffentlichkeit/moocs/, 2017

[Uda17] Udacity Inc.: *Udacity*. Online: https://de.udacity.com/, 2017

[Wes14] Weske, Mathias: *Business Process Management - Concepts, Languages, Architectures*. Wiesbaden : Springer Berlin Heidelberg, 2014

[WHKL15] Wütherich, Gerd ; Hartmann, Nils ; Kolb, Bernd J. ; Lübken, Matthias: *Die OSGi Service Platform - Eine Einführung mit Eclipse Equinox*. Heidelberg : dpunkt.verlag, 2015

[Wik17a] Wikipedia: *Keyed-Hash Message Authentication Code (HMAC)*. Online: https://de.wikipedia.org/wiki/Keyed-Hash_Message_Authentication_Code, 2017

[Wik17b] Wikipedia: *PBKDF2-Funktion*. Online: https://de.wikipedia.org/wiki/PBKDF2, 2017

[Wik17c] Wikipedia: *Secure Hash Algorithm (SHA)*. Online: https://de.wikipedia.org/wiki/Secure_Hash_Algorithm, 2017

[Wir17] Wirdemann, Ralf: *Scrum mit User Stories*. München : Carl Hanser Verlag GmbH Co KG, 2017

[Wol15] Wolff, Eberhard: *Microservices - Grundlagen flexibler Softwarearchitekturen*. Heidelberg : dpunkt.verlag, 2015

[WWF17] WWF: *MOOC - Online-Vorlesung zu „Klimawandel und seinen Folgen“*. Online: http://www.wwf.de/aktiv-werden/bildungsarbeit-lehrerservice/mooc-online-vorlesung/inhalte-des-mooc/, 2017