HOCHSCHULE
MITTWEIDA
University of Applied Sciences

Fakultät Angewandte Computer- und Biowissenschaften

Professur Digitale Transformation und angewandte Medieninformatik

Bachelorarbeit

Entwicklung einer Servicestruktur zum skalierbaren Einsatz im E-Learning

Max Schlosser

Mittweida, den 29. Juli 2022

Erstprüfer: Prof. Dr.-Ing. Christian Roschke
Zweitprüfer: Susan Labude B.Sc.

Schlosser, Max
Entwicklung einer Servicestruktur zum skalierbaren Einsatz im E-Learning
Bachelorarbeit, Fakultät Angewandte Computer- und Biowissenschaften
Hochschule Mittweida — University of Applied Sciences, Juli 2022

Referat

Im Kontext der Wissensvermittlung existieren vielseitige E-Learning Tools, die oft ein dediziertes Lehr-Lern-Szenario betrachten. Die zugrundeliegende Software konzentriert sich somit häufig auf einen spezifischen Bereich oder eine gesonderte Fragestellung, sodass eine Erweiterung des Angebots nicht ohne weiteres möglich ist. Es soll untersucht werden, welche serverseitigen Anforderungen ein System für die E-Learning-Domäne erfüllen muss, um keine derartigen Restriktionen aufzuweisen und gleichzeitig für den Einsatz verschiedener Szenarien der digitalen Lehre geeignet zu sein.

Name: Schlosser, Max
Studiengang: Medieninformatik und interaktives Entertainment
Seminargruppe: MI19-w1-B
English Title: Development of a service structure for scalable deployment in the e-learning domain

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

ANSI American National Standard Institut

C3MS Community, Content, and Collaboration Management System

CD Continuous Delivery

CI Continuous Integration

CI/CD Continuous Integration und Continuous Deployment

HTTP Hypertext Transfer Protocol

IEEE Institute of Electrical and Electronics Engineers

ID Instuctional Design

JWT JSON Web Token

LMS Lern Management System

LOM Learning Object Metadata

LRS Learning Record Store

MOOC Massive Open Online Course

NPM Node Package Manager

PLE Personal Learning Environment

REST Representational State Transfer

RE Requirements Engineering

SCORM Sharable Content Object Reference Model

SQL Structured Query Language

URL Uniform Resource Locator

VLE Virtual Learning Environment

VM Virtual Machine

WBT Web Based Training

1. Einleitung

Digitale Lehr-Lern-Szenarien haben im Zuge der Digitalisierung bedeutend an Einfluss gewonnen. Besonders während der Corona-Pandemie wurde deutlich, dass digitales Lernen eine bereichernde Ergänzung oder sogar alleiniger Ersatz für klassische, analoge Lehrverfahren sein kann. Auch im internationalen Gesamtbild zeigt sich, dass zahlreiche Lehr- und Lernszenarien neben analogen Methoden im Allgemeinen digitale Strukturen nutzen, um den Prozess der Wissensvermittlung proaktiv zu unterstützen. Allein aus diesem Gedanken heraus erschließt sich, dass es einen steigenden Bedarf an Software, die sich für den Einsatz in der Bildungsdomäne eignet, gibt. Die Vielfältigkeit der vorhandenen E-Learning-Systeme lässt erahnen, dass sich die Entwicklung einer Grundstruktur für eine neue Lernplattform meist an spezifischen Kriterien orientiert. Zu diesem Zwecke entwerfen Anbieter häufig eigens entwickelte Systeme, welche auf den Kontext des zu entwickelnden Angebots abgestimmt sind. Eine Wiederverwendbarkeit der Grundstruktur ist somit nicht gegeben.

Diese nicht vorhandene erneute Verwertbarkeit eines Systems zieht im Kontext des E-Learnings mehrere Nachteile nach sich. Erfolgt die Implementierung eines neuen Backends für einen spezifischen Zweck, so realisiert dieses meist keine oder nur eingeschränkte Wege, um mit vorhandenen Plattformen zu interagieren. Lehrende müssen sich somit mit unterschiedlichen Vorgehensweisen der Erstellung und Repräsentation von Inhalten auseinandersetzen, während Lernende selbst mit wesentlich voneinander abweichenden Systemen umgehen müssen. Derartige Umstellungen können einen zeitlichen und gedanklichen Mehraufwand nach sich ziehen und den Lehr- und Lernprozess somit grundlegend einschränken.

1.1. Zielstellung

Die vorliegende wissenschaftliche Arbeit greift die Problemstellung einer möglichst generischen Grundlage für die Entwicklung einer E-Learning-Plattform auf. Um eine derartig universelle Basis zu konzipieren, müssen existierende Angebote, ihre Anforderungen und zugrundeliegende Standards systematisch analysiert werden. Es ist somit die zentrale Frage, wie Grundlagen des E-Learnings in einem abstrakten Kontext eingesetzt werden können, um eine möglichst allgemein einsetzbare Servicestruktur aufzubauen, welche mit bestehenden Standards kompatibel ist.

Unter der Verwendung von softwaretechnischen Ansätzen und vorherrschenden Standards ist es dabei die Aufgabe, eine Serverstruktur prototypisch zu implementieren und für eine realitätsnahe Verteilung anzuwenden. Es sollen hierbei client- und serverspezifische Anforderungen erfüllt werden, um die verwendeten Ansätze zu validieren. Anhand der vorgenommenen Evaluation der ausgewählten Kriterien soll es schlussendlich möglich sein, die implementierte Anwendung hinsichtlich ihres abstrakten Einsatzes einzuschätzen und potenzielle Verbesserungsansätze zu skizzieren.

Das entwickelte Konzept soll hauptsächlich als Basis für die gezielte Einführung neuer Lehr-Lernangebote dienen. Gleichzeitig kann dieser Ansatz genutzt werden, um tiefer greifende Frameworks und Entwicklungsmethoden zu konzipieren, die sich auf den Bereich des digitalen Lernens fokussieren. Die Arbeit soll zudem als Grundlage für weiterführende Forschungen im Bereich der E-Learning-Anwendungsentwicklung dienen. In Zukunft kann Bezug auf den aufgebauten Service-Prototyp genommen und Anforderungen anhand der aufgebauten Metriken neu strukturiert oder erweitert werden.

1.2. Aufbau der Arbeit

Die Arbeit besteht neben dem ersten Kapitel, welches die Einleitung, die Aufgabenbeschreibung und Ziele der Arbeit darlegt, aus fünf weiteren Kapiteln. Inhalt von Kapitel zwei sind grundlegende theoretische Grundlagen, welche für die weiteren Inhalte bedeutsam sind. Kapitel drei analysiert vorhandene Plattformen und legt die daraus resultierenden Anforderungen an die Anwendung dar. Dieser Abschnitt

skizziert auch ein Konzept für die Entwicklung und die darauffolgende Evaluation des Prototyps. Das vierte Kapitel thematisiert die Implementierung der Servicestruktur. Neben verschiedenen Entwicklungsschritten beschreibt dieser Abschnitt Schnittstellen zur Nutzung und Weiterentwicklung des Service sowie das Deployment der Anwendung. Kapitel fünf behandelt die Evaluation des Prototypens. Es erfolgt die Durchführung aller zuvor aufgestellten Testfälle, um eine Aussage über die Erfüllung der Anforderungen treffen zu können. Das abschließende sechste Kapitel zieht ein Fazit der wissenschaftlichen Arbeit. Außerdem gibt es einen Ausblick auf potenzielle Erweiterungen und mögliche Nutzungsszenarien für zukünftige Forschungen.

2. Grundlagen

Vor Beginn der Konzeptionierung und Implementierung einer skalierbaren Servicestruktur für die E-Learning Domäne sind diverse Grundlagen zu zentralen Themen der wissenschaftlichen Arbeit unverzichtbar. Dieses Kapitel soll deshalb die bedeutendsten Grundlagen zusammenfassend darlegen. Der Beginn thematisiert den Teilbereich der Software-Anforderungsanalyse. Anschließend werden Anforderungen an digitale, didaktische Konzepte dargestellt, wobei der Fokus besonders auf generellen Rahmenbedingungen, Standards und aktuell vorherrschenden Plattformen liegt. Weiterführend wird die Persistenz von Datenstrukturen behandelt. Dabei stehen Datenbanktechnologien und verschiedene Anfrage- und Responsemodelle im Vordergrund. Im Anschluss erfolgt eine Erläuterung der Grundidee von verteilten Systemen mitsamt ihrer Ziele, wobei der Aspekt der Skalierbarkeit gesondert betrachtet wird. Abschließend wird der Bereich des DevOps beleuchtet, wobei das Kapitel verstärkt auf die Teilaspekte Continuous Integration und Continuous Deployment eingeht. Auch nutzbare Technologien des DevOps-Bereiches werden thematisiert.

2.1. Software-Anforderungsanalyse

Um während der Entwicklung eines Produktes aus grundlegenden Ideen einen möglichst hohen Grad von Umsetzbarkeit und somit weniger Verwerfungen von Vorstellungen zu erzielen, wird der Entwicklung oft eine Phase der Planung und Priorisierung von Ideen vorangestellt. Dieser als Anforderungsanalyse, oder im Englischen Requirements Engineering (RE), bekannte Prozess kann sequenziell in den Gesamtprozess eingepflegt werden, sodass eine Entwurfsphase erst nach Fertigstellung der Anforderungsanalyse erfolgt, oder sich nebenläufig eingliedern, womit der Entwurf bereits während des RE begonnen wird. [Bal09, S. 449 ff.]

Aus den fachlichen Anforderungen entstehen schriftliche Spezifikationen. Aus der Sicht des Kunden geht das Lastenheft hervor, das dem Produzenten bei der Auffassung des Auftrages helfen soll. Darauf aufbauend kann das so genannte Pflichtenheft erstellt werden, das die Anforderungen aus der Perspektive des Produzenten spezifiziert. In diesem werden alle für den Auftraggeber wichtigen fachlichen Anforderungen beschrieben. Im Gegensatz zum Lastenheft ist dieses Dokument somit für den Kunden verfasst. [Met20, S. 62 f.]

Zu den zentralen Tätigkeiten des RE gehören das Ermitteln, Dokumentieren, Prüfen und Abstimmen sowie das Verwalten von Anforderungen. Auch wenn die Reihenfolge zur Durchführung dieser Tätigkeiten flexibel auf das Projekt angepasst werden kann, muss beachtet werden, dass diese teils in einer logischen Abhängigkeit zueinander stehen. Aus der Vielgestaltigkeit der Prozessteile wird gleichzeitig deutlich, dass zahlreiche Tätigkeitsfelder in die Entstehung der Anforderungsanalyse einfließen. So können etwa Kommunikationsfähigkeit und Konfliktlösung, Beherrschung von Standards oder analytische Fähigkeiten unabdingbar für die Durchführung einer Anforderungsanalyse sein. [Her22, S. 5]

2.1.1. Anforderungen

Um genauer auf den Einsatz von Anforderungen eingehen zu können, bietet sich zunächst eine Definition an. Anforderungen legen demnach fest, welche Eigenschaften von einem Softwaresystem erwartet werden. Das Spezifizieren der Anforderungen wird dabei meist von Akteuren übernommen, die ein Interesse an dem zu entwickelnden Produkt haben oder in irgendeiner Form von dessen Einführung betroffen sind. Diese werden auch als *Stakeholder* bezeichnet. Anforderungen selbst besitzen Ansprüche, die sich auf zahlreiche Aspekte, wie ihre Eindeutigkeit, Klassifizierbarkeit oder Validierbarkeit beziehen. Derartige Anforderungen ergeben sich überwiegend aus vorliegenden Standards, die an ausgewählten Beispielen näher im Kapitel 2.1.2 erläutert werden. [Bal09, S. 455, S. 475 ff.]

2.1.2. Standards

Das ISO/IEC 25010 Modell ist besonders für die Ermittlung von Qualitätsanforderungen geeignet. Wie in Abbildung 2.1 sichtbar, wird zwischen mehreren Qualitätsklassen unterschieden. Es werden verschiedene Anforderungskriterien für Funktionalität, Zuverlässigkeit, Effizienz, Benutzerfreundlichkeit, Sicherheit, Kompatibilität, Wartbarkeit und Übertragbarkeit des Produktes gestellt. In seiner Gesamtheit kann dieser Standard als stichhaltige Überprüfungsliste für Qualitätsanforderungen genutzt werden. [Her22, S. 41 ff.]

Abbildung 2.1.: ISO/IEC 25010 Qualitätsmodell [ISO22]

Ein weiterer Standard wird durch eine im IEEE Standard 830-1998 verankerte Anforderungsschablone realisiert, in welcher entsprechende Gliederungsschemata für Spezifikationen beschrieben werden. Auch wenn jegliche Gliederungspunkte beschrieben sind, bieten sich in der Praxis neben verbalen Beschreibungen weitere externe Konzepte zur Niederschrift an. Die Schablone umfasst dabei ein Einleitungskapitel, welches unter anderem Ziele, Definitionen, Referenzen sowie einen Gesamtüberblick beinhaltet. Im Anschluss werden in einer Übersichtsbeschreibung Umgebung und Funktion des Produktes, Eigenschaften des Nutzers sowie mögliche Restriktionen und Annahmen zusammengefasst. Abschließend werden spezifische Anforderungen des Produktes aufgezeigt, wobei die Art der Konzeptionierung und Beschreibung stark von den Rahmenbedingungen der Anwendung abhängt. [Bal09, S. 485 f.]

2.2. Anforderungen an digitale Didaktikkonzepte

Mit dem Wunsch nach einer digitalen Lehr-Lern-Umgebung geht die Frage nach zu erfüllenden Anforderungen einher. Oft sind Ansätze zur Abbildung digitaler Lerninhalte nicht universell anwendbar und benötigen eine enge Abstimmung auf den Lehrzweck. Im Folgenden werden diese Ansätze beispielhaft thematisiert. Im Vordergrund steht jedoch die anschließende Gesamteinordnung des E-Learnings anhand von Rahmenbedingungen und vorhandenen Standards sowie die Hervorhebung aktuell dominierender E-Learning Plattformen.

2.2.1. Methoden

Eine als Instuctional Design (ID) bekannte Kerndisziplin befasst sich mit dem fundierten bildungswissenschaftlichen Ansatz, durch den Lernangebote so konzipiert werden, dass bestimmte Ziele erreicht werden können. Die ursprünglich auf Robert M. Gagné zurückzuführende Grundidee zieht dabei die Berücksichtigung der Eigenschaften von Lernstoff und -umgebung ebenso wie die Merkmale Lernender in Betracht. Beides soll eine situativ auf die Eigenschaften abgestimmte Vorgehensweise bei der Vermittlung des Lernstoffs begünstigen. Gleichzeitig soll die hierarchische Abstimmung aufeinander aufbauender Lerninhalte sichergestellt werden. In den Folgejahren entstanden weitere an dieses Grundmodell des ID angelehnte Modelle und Frameworks, die primär als Erweiterung und Spezifikation der zugrundeliegenden Idee aufgefasst werden können. Als ein Vertreter ist dabei das Rahmenmodell *ADDIE* hervorzuheben, das als Akronym die jeweils englischen Übersetzungen der Unterpunkte Analyse, Konzeption, Entwicklung, Implementierung und Evaluation zusammenfasst. Innerhalb des sehr allgemein gehaltenen Aufbaus wird im Laufe der Planung die Entscheidung für das Lernformat sowie die Lernziele getroffen. Die Entwicklungsphase berücksichtigt diese Kriterien in der Ausarbeitung von notwendigen Materialien und Prozessen, die im Laufe der Implementierung in die Lehr-Lern-Aktivität integriert werden. Die meist zu Iteration führenden Evaluationsphase beinhaltet formative und summative Kriterien. [Nie20, S. 96 ff.]

Im Bereich der digitalen Didaktik gestaltet sich die Transformation von Inhalt zu Lehrinhalt im Rahmen des ursprünglichen ID schwierig, sodass adäquate Ansätze, die neue Kriterien innerhalb des *ADDIE*-Prozesses einbeziehen, in den Vordergrund rücken. Dabei ist die Theorie des Konstruktivismus relevant, laut welcher der Erwerb

von Wissen weniger auf der objektiven Vermittlung von diesem aufbaut, sondern im sozialen Kontext durch den Lernenden selbst konstruiert wird. Konstruktivistische Kriterien lassen sich im digitalen Raum abbilden. Die Strukturierung eines Lernraumes, in dem neben der Gelegenheit zum kollaborativen Lernen auch Möglichkeit für eine individuelle Entfaltung gegeben werden sollte, ist ebenso wichtig wie die Anpassung der Lerninhalte auf authentische Szenarien oder die Einrichtung von Elementen, die dem Lernenden Möglichkeiten zur Reflexion erlernter Fähigkeiten bieten. Auch die Einbettung der Selbststeuerung des Lernprozesses durch den Lernenden ist zentraler Teil des Konstruktivismus: das aktive Einwirken auf Lernsituationen, das Herstellen von Produkten aus dem Lernprozess oder das selbstständige Erforschen von Lernwegen mit paralleler Hilfe durch Lehrende in Fehlersituationen wirken unterstützend auf den Lernprozess. Nicht zuletzt ist auch der dialogische Austausch, der gegebenfalls im Rahmen eines spezifizierten Regelwerks stattfindet, wichtig für den Lernaustausch. Weitere Aspekte, die neben konstruktivisitischen Anforderungen beachtet werden sollten, umfassen etwa das Bereitstellen verschiedener kollaborativer Lernressourcen und -kanäle, das Schaffen eines herausfordernden, transparenten und begleitend im Kontext eingebetteten Lernrahmens sowie das Adressieren der intrinsischen Explorationsneugier Lernender. [Ker20, S. 10 f., S. 20 ff.]

2.2.2. E-Learning Anwendungen und ihre Arten

Der Begriff des E-Learnings hat sich in den letzten Jahren als vorherrschender Standard und gleichzeitig Synonymbegriff für verschiedene Lehr-Lern-Szenarien entwickelt. Überschneidungen derartiger Begriffe sind missverständlich, sodass eine klare Trennung erfolgen sollte. E-Learning selbst beschreibt die Vielzahl vorhandener "gegenständlicher und organisatorischer Arrangements", die mithilfe von digitalen Medien zum zeitunabhängigen Lernen genutzt werden kann. Als Hybridlösung gilt der Begriff des *Blended Learnings*, der das digitale Lernen um zeitlich gebundene Einheiten, welche ebenfalls in einem digitalen Rahmen stattfinden können, erweitert. [Arn18, S. 22 f.]

Die Nutzung digitaler Lernstrukturen unterlag in den letzten Jahren einem stetigen Wandel. Die Grundidee der Einbettung von digitalen Lern-Arrangements beruht auf einem hohen Grad von Selbstorganisation, da Lernende das zeitliche und örtliche Umfeld sowie die Lernform frei bestimmen können. Wie in Abbildung 2.2 dargestellt, hat sich der besonders durch internetbasierte Lernprogramme, auch als Web Based

Trainings (WBT) bezeichnet, geprägte, primär selbstgesteuerte Lernprozess nach und nach zu einem selbstorganisiertem Lernen gewandelt. Auch wenn dabei auf der Seite Lernender der Bedarf nach jederzeit abrufbarem Wissen auch weiterhin hoch ist, haben Blended Learning Szenarien zunehmend an Einfluss gewonnen. Für die Ausarbeitung von möglichst selbstorganisierten Lern-Arragements ist deshalb eine komplexe Abstimmung von Didaktik und Methodik notwendig. [Erp15, S. 1 ff.]

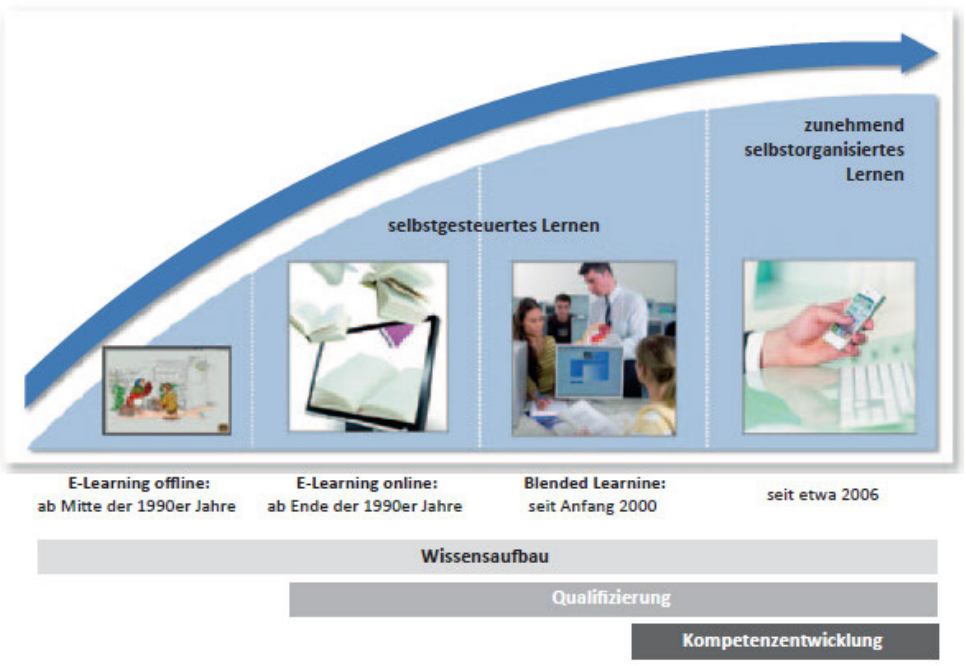

Abbildung 2.2.: Entwicklungsstufen des E-Learning [Erp15, S. 2]

Die Konzeptionierung von digitalen Lerninhalten bringt somit neue Herausforderungen mit sich, die in analogen Szenarien nicht auftreten. So ist es beispielsweise in einer digitalen Umgebung nicht ohne weiteres möglich, Lernende zielgerichtet auf Änderungen eines Angebots aufmerksam zu machen, während in einer klassischen Lernumgebung derartige Überarbeitungen und eventuelle Einzelheiten auf verbalem Weg kommunizierbar sind. Auch erfordert das Entwerfen und Implementieren einer digitalen Lernanwendung meist interdisziplinäre Zusammenarbeit mitsamt einer zeitlichen und inhaltlichen Abstimmung zwischen Bereichen der Didaktik, Informatik und des Projektmanagements. Das Einpflegen und Etablieren eines neu entworfenen Lehr-Lern-Szenarios muss zur bestmöglichen Unterstützung der Lernerfahrung und -motivation eng in die Modulplanung einbezogen werden. Dabei sollten Lernen-

de umfangreich über zentrale Konzeptpunkte informiert werden. Dazu zählt etwa der Modulablauf, die Taktung der Lerneinheiten und -materialien, die Planung von digitalen und analogen Phasen, genutzte Sozial- und Betreuungsformen sowie die Durchführung eventueller Prüfungen. [Arn18, S. 118 f., S. 139 f.]

In den letzten Jahren haben sich verschiedene Arten des E-Learnings etabliert. Unter Massive Open Online Courses (MOOCs) werden freie, im Internet angebotene Kurse verstanden, die Nutzern unabhängig von ihren Hintergründen und Erfahrungen frei zur Verfügung stehen. Inhalte werden meist von einem oder mehreren Experten gestaltet und können Lerneinheiten oder Diskussionsplattformen umfassen. Der Begriff Personal Learning Environment (PLE) meint eine Lernumgebung oder eine Sammlung an Tools, welche Lernenden beim Aufbauen und Organisieren von Lernstrukturen helfen sollen. Der Lernende besitzt dabei selbstständig die Kontrolle über Lernprozess, -umgebung, -ressourcen und -partner. Dagegen werden Plattformen, die normalerweise von der Seite von Bildungseinrichtungen bereitgestellt werden und Nutzern Gruppen, Ressourcen und andere Aktivitäten zuteilen, als Virtual Learning Environments (VLEs) klassifiziert. Es werden dabei wichtige Teilbereiche wie Content Management, Kursplanung, die gezielte Bereitstellung von Inhalten oder Kommunikation realisiert. Des weiteren können Webplattfomen, welche in Form einzelner Seiten bestimmte Ressourcen zusammenstellen, als Community, Content, and Collaboration Management Systems (C3MS) bezeichnet werden. Diese erlauben das Verwalten einer Community sowie der Zusammenarbeit zwischen ihren Mitgliedern und Inhalten. Oft wird zusätzlich das Einpflegen von Erweiterungen, die das C3MS um neue Funktionen erweitern, ermöglicht. [ORC21, S. 65 ff.]

2.2.3. Technische Standards des E-Learnings

Die Wichtigkeit der Standardisierung von digitalen Lerninhalten hat sich in der Vergangenheit in der schlechten Wiederverwendbarkeit entworfener Inhalte gezeigt. Verschiedene internationale Spezifikationen wurden zu einem Standard mit dem Namen Sharable Content Object Reference Model (SCORM) vereint, der Regeln für das Entwickeln, die Präsentation und Produktion von Anwendungen und Materialien festlegt. Der Hauptfokus liegt dabei auf der bestmöglichen Wiederverwendbarkeit, einem einfachen Zugriff für Lernende und Lehrende, einer breiten Kompatibilität

für verschiedenste Systeme sowie der unkomplizierten Anpassbarkeit im Falle eines Softwareupdates. Auch werden feste Formate für die Komprimierung von Inhalten, der Kommunikation von Softwarekomponenten zur Laufzeit und für Metadaten von Kursen festgelegt. [Jon22, S. 4]

Ein weiterer Standard, dessen Fokus auf Datentransport und -speicherung liegt, trägt den Namen *Experience API*, oder kurz *xAPI*. Statt der Fokussierung auf ein vollwertiges Lern Management System (LMS), wird durch xAPI lediglich die Kommunikation der Lernerfahrung zu einer Speicherinstanz für diverse Lerneinträge, einem so genannten Learning Record Store (LRS), geregelt. Damit xAPI, das nie als Ersatz für SCORM und anderweitige Managementzwecke konzipiert wurde, plattformübergreifende Standards realisiert und somit aufbauend auf SCORM nutzbar wird, wurde der *cmi5*-Standard auf drei Hauptzielen aufbauend entworfen. Als besonders wichtig gilt die Interoperabilität, die sicherstellt, dass cmi5-kompatible Komponenten auf verschiedenen LMS gleich funktionieren, solange die Spezifikation erfüllt wird. Außerdem sollen unter cmi5 erweiterte Datenformate, wie Bild-, Video- oder Audiodateien, nutzbar sein. Zuletzt soll der Standard auch unter Mobilgeräten verfügbar sein. Innerhalb der Regeln des Standards definiert cmi5 so genannte Verben, die für festgelegte Lernaktivitäten verschiedene Statusmeldungen wie *Launched*, *Completed* oder *Failed* übermitteln können. [VW16, S. 10 ff.]

Für die Beschreibung von Lernobjekten, also jeglichen Entitäten, die in irgendeiner Form zum Lehren oder Lernen genutzt werden, wurde vom Institute of Electrical and Electronics Engineers (IEEE) der Standard Learning Object Metadata (LOM) entworfen. Die Schaffung einheitlicher, strukturierter Beschreibungen soll beim Finden, Evaluieren und Anschaffen von Inhalten helfen. Das Teilen von Informationen zwischen Ressourcen soll ebenso erleichtert werden wie das Verknüpfen und Anpassen der Beschreibung an die eigentlichen Ressource. LOM arbeitet mit einem hierarchischen Modell, das aus neun Hauptkategorien sowie verschachtelten Unterkategorien mit jeweils verschiedenen Arten von Daten besteht. [Bar05, S. 1 f.]

2.2.4. Aktueller E-Learning Markt

Auf dem internationalen Markt existieren mehr als 600 verschiedene Plattformen zum Distanzlernen, davon besitzen etwa 40, wie etwa die populäre Plattform Moodle, eine freie Lizenz. Viele LMS erlauben dabei die Einbindung und Übertragung von

Inhalten über Standards, wie sie beispielhaft unter dem Kapitel 2.2.3 beschrieben wurden. Plattformen realisieren dabei in den meisten Fällen Funktionen für Lehrende, Lernende und Administratoren der Seite. Dazu zählen vielfältige Inhalte, wie sie etwa an den Plattformarten in Kapitel 2.2.2 aufgezeigt wurden, aber auch weitere Punkte wie die Verwaltung von Tests, Lernfortschritten oder kollaborativen Lernumgebungen. [ORC21, S. 54 f.]

Durch die Veränderung demographischer Bedingungen, wie beispielsweise dem rasanten Bevölkerungszuwachs in asiatischen Ländern wie Indien, ist der Bedarf an flächendeckenden Lernmöglichkeiten gewachsen. Besonders für das Distanzlernen, welches vor allem durch den einschneidenden Einfluss der Covid-19 Pandemie an Bedeutung gewonnen hat, ist die Relevanz von MOOCs gewachsen. Wenn auch speziell zugeschnittene LMS für viele Bildungseinrichtungen ebenso unverzichtbar sind, so sind die durch MOOCs geschaffenen Möglichkeiten der Bildung in der Breite der Gesellschaft im internationalen Rahmen nicht zu unterschätzen. [PLP21, S. 59]

Anhand von aktuellen Entwicklungen des digitalen Lernens sowie der sich deckenden Ergebnissen verschiedener Studien ist es möglich, weitere für den aktuellen Markt wichtige Aspekte herauszustellen. Die Möglichkeit der sozialen Interaktion, die von einer Plattform bereitgestellt wird, hat einen entscheidenden Einfluss auf die Lernatmosphäre und -motivation und sollte deshalb qualitativ gepflegt werden. Auch der Einfluss der Lehrpersonen auf die Motivation der Lernenden und die Verantwortung für die Strukturierung von Inhalten ist von großer Bedeutung. Besonders deshalb sollten Plattformen nicht nur für die Lernenden, sondern auch für die Lehrpersonen einfach zugänglich sein. Für die Evaluierung der genannten Kriterien existieren vielfältige Modelle, wie etwa das *Information System Success Model*, welches verschiedene Kriterien aus der Lehrer-Nutzer-Sicht einbezieht, um Systemqualität und -erfolg zu beurteilen. Derartige Frameworks sind für das Entwerfen neuer Plattformen nützlich. [SP21, S. 2 f.]

2.3. Persistenz von Datenstrukturen

Für Anwendungen, welche Daten abrufen, speichern und darstellen, sind verschiedene Speicherstrukturen notwendig. In diesem Kapitel werden Grundlagen und Technologien zur permanenten Speicherung, der so genannten Persistenz, vorgestellt. Außerdem werden Möglichkeiten zur Anfrage und Rückgabe entsprechender Daten beschrieben.

Um für einen schnellen Zugriff bereit zu stehen, werden Dateien temporär im so genannten *Internspeicher* aufbewahrt. Neben einer limitierten Größe besitzt dieser allerdings den Nachteil, dass mit dem Beenden eines Systems alle Dateien verloren gehen. Aus diesem Grund liegt der Fokus bei der Persistenz auf dem *Externspeicher*, der eine Erhaltung der Dateien auch über die Systemzeit hinaus ermöglicht. Zur schnellen Verarbeitung werden Dateien aus dem Externspeicher in so genannte Dateipuffer des Internspeichers geladen. [Lus03, S. 20]

2.3.1. Datenbanken

Datenbanken realisieren eine besonders strukturierte und effiziente Möglichkeit zur Speicherung von Datenstrukturen. Beinhaltete Daten sollen dabei verwaltet werden können - das heißt hinzufügbar, bearbeitbar und löschbar sein. Gleichzeitig ist der Schutz der Daten vor unberechtigten Zugriffen und eine Toleranz gegenüber Manipulationen, die Datensätze oder ganze Strukturen nicht mehr nutzbar machen, bedeutend. Der Zugriff auf die Daten soll dabei unabhängig von der zugrundeliegenden Organisation möglich sein. Gleichzeitig muss dieser Aufbau änderbar sein, ohne dass zugreifende Programme angepasst werden müssen. [Ste21, S. 5 f.]

Es gibt eine Vielzahl von Ansätzen, die bei der Modellierung von Datenbanken zum Einsatz kommen können [Mei10, vgl. S. 159-189]. Die im Folgenden dargestellten Ansätze gelten beispielhaft für die breite Palette an Technologien, die zur Entwicklung von Datenbanken eingesetzt werden kann.

Die Anfänge der strukturierten Datenspeicherung wurden durch hierarchische Datenbanken geschaffen. Dabei wurden sämtliche Daten in einer einzelnen Datei gespeichert, die in zwei Ebenen jeweils eine Hierarchiestufe und alle zugehörigen Daten abgelegt hat. Den heute populärsten Ansatz für die Datenbankmodellierung bilden relationale Datenbanken, in welchen Daten als so genannte *Entitäten* thematisch

getrennt in Tabellenform gespeichert werden. Dieser realitätsnahe Ansatz bietet im Gegensatz zur hierarchischen Modellierung eine deutlich höhere Flexibilität, da für neue Strukturen im Optimalfall lediglich neue Tabellen, die die bereits existierende Grundstruktur nicht beeinflussen, notwendig werden. Diese Aufteilung auf verschiedene Tabellen ermöglicht somit die Auslagerung der Datensätze unabhängig von ihren Verknüpfungen untereinander. Durch die gegenseitigen Verbindungen der Datentabellen werden die Datenstrukturen mit steigender Komplexität allerdings schwieriger überschaubar und benötigen bei der Datenanfrage durch das Zusammenführen verschiedener Tabellen unter Umständen mehr Zeit. [Ste21, S. 9 ff.]

Aufbauend auf klassischen relationalen Datenbankmodellen existieren so genannte objektrelationale Ansätze. Diese versuchen Eigenschaften eines Objektes, welche in relationalen Modellen unpraktischerweise meist über mehrere Tabellen verteilt sind, an ein zentrales Objekt zu binden. Zugleich ist es möglich, nach Vorbild der objektorientierten Programmierung neue Struktureigenschaften, Objekttypen und Operationen anzulegen, die zudem vererbt werden können. [Mei10, S. 169 ff.]

Ein Datenbanksystem, welches sich auf einen nicht-relationalen Ansatz fokussiert, ist MongoDB. Als Hauptziele gelten eine hohe Geschwindigkeit, Skalierbarkeit und Flexibilität des Systems. Für die Speicherung von Datensätzen werden so genannte *Documents* verwendet, welche eine auf JSON basierende Struktur aufweisen und gesammelt in einer *Collection* aufbewahrt werden. Im Gegensatz zu klassischen Tabellen ist die Datenspeicherung an kein festes Schema gebunden und daher dynamisch einsetzbar. Das Anfragen von Daten ist dabei an die Schlüssel der einzelnen Documents gebunden, sodass eine Verteilung der Datenbank über mehrere Server möglich wird. [Edw15, S. 26 f.]

Hierarchische Modelle sind heute weitestgehend von relationalen Ansätzen ersetzt worden. Es gilt als wahrscheinlich, dass objektrelationale Ansätze relationale Datenbanken nicht ersetzen werden, sondern lediglich situativ zum Einsatz kommen, wenn ihre Vorteile überwiegen. [Ste21, S. 11]

2.3.2. Anfrage- und Responsemodelle

Auf der Ebene der Datenbanken existieren verschiedene Sprachen, die das Anfragen und Modifizieren von Daten ermöglichen. Der dabei gängigste Vertreter ist die Structured Query Language (SQL), die als Kommunikationssprache zur Abfrage,

Manipulation und auch zum Schützen der zugrundeliegenden Daten dient. Auch wenn SQL vom American National Standard Institut (ANSI) als Standard für die in Kapitel 2.3.1 erläuterten relationalen Datenbanksysteme gilt, haben viele Hersteller ihre Systeme so erweitert, dass Abweichungen auftreten. [Ste21, S. 143]

Auf der Seite der Anwendungen werden jedoch oft Zwischenschichten benötigt, die das Anfragen von Daten, beispielsweise aus Datenbanken, für Entwickler und Nutzer deutlich erleichtern sollen. Ein Standard, der sich in den letzten Jahren als bedeutend erwiesen hat, sind so genannte Representational State Transfer (REST) Schnittstellen. Hauptbestandteil sind dabei eindeutige Ressourcen, die über das Hypertext Transfer Protocol (HTTP) geladen, erzeugt, modifiziert oder gelöscht werden können. Dafür werden von HTTP standardisierte Methoden, wie *GET*, *POST*, *PUT* oder *DELETE*, in die Schnittstelle integriert. Wichtig ist zudem die Repräsentierbarkeit von Ressourcen. Für Ressourcen sollen ein oder mehrere Transportformate angegeben werden, die je nach Kontext von der Schnittstelle angefragt werden können. Die Verbindung der einzelnen Ressourcen, die etwa durch das Verlinken anderer Ressourcen oder zugehörige Aktionen geschehen kann, spielt dabei ebenfalls eine zentrale Rolle. Weitere wichtige Grundsätze, die REST verfolgt, sind etwa ein Entwurf nach einer Client-Server-Logik, die Möglichkeit erhaltene Antworten zu cachen, die Nutzung einheitlicher Schnittstellen-Standards oder die Option, Zwischenschichten in die Schnittstelle zu integrieren. [Wir19, S. 27 ff.]

Bei der Nutzung von REST fallen oft zahlreiche *Endpoints* - daher Uniform Resource Locators (URLs), über welche die Anfragen gestellt werden - an. Im Gegensatz dazu definiert die Technologie GraphQL lediglich einen Endpoint, welcher je nach Anfrage nur Daten zurückgibt, die vom Client angefordert werden. In der Anfrage wird dabei ein so genanntes *GraphQL Schema* übermittelt. Dieses gibt die genaue Struktur der angefragten Datenmodelle an und erlaubt auch die Übergabe von Funktionen zur Datenmanipulation. Es wird dabei semantisch zwischen Schemas zur Datenanfrage (*Queries*) und Schemas zum Hinzufügen, Verändern oder Löschen von Daten (*Mutations*) unterschieden. GraphQL unterstützt zahlreiche andere Datenkonzepte und ist in der Anfrage deutlich flexibler und selektiver als REST. [Fri18, S. 1 ff., S. 9]

2.4. Verteilte Systeme

Verteilte Systeme sind ein zusammenfassender Begriff, der im Rahmen komplexer Anwendungen zunehmend an Bedeutung gewonnen hat. Durch die Vielzahl unterschiedlicher Ausprägungen bietet sich nach Tanenbaum eine grobe Annäherung an die Thematik durch folgende Definition an:

> "Ein verteiltes System ist eine Ansammlung unabhängiger Computer, die den Benutzern wie ein einzelnes kohärentes System erscheinen."
> [Tan08, S. 19]

Im Folgenden sollen Kernaspekte, die ein verteiltes System ausmachen, genauer herausgearbeitet und durch ihre grundlegenden Konzepte und Ziele unterstrichen werden.

2.4.1. Grundideen und Ziele

Wie bereits in der unter 2.4 gegebenen Definition deutlich wird, kann ein verteiltes System nur durch die Kooperation mehrerer autonomer Komponenten existieren, die in ihrer Gesamtheit die Funktionalität realisieren. Eine Koppelung der einzelnen Elemente auch über räumliche Grenzen hinweg ermöglicht somit eine Kooperation oder Integration innerhalb des Gesamtsystems. Dabei ergeben sich mehrere Ziele, die bei der Realisierung eines solchen Systems im Vordergrund stehen. [SS12, S. 5]

Das wichtigste Ziel besteht in der Vereinfachung des Zugriffs auf entfernte Ressourcen, sodass eine gegenseitige Kontrolle und effiziente gemeinsame Nutzung möglich werden. Neben den offensichtlichen Einsparungen, die sich durch eine geteilte Nutzung ergeben, wird durch eine Verbindung auch die gegenseitige Zusammenarbeit und der Informationsaustausch vereinfacht. Durch die starke Vernetzung, wie sie beispielsweise aus dem Rahmen des Internets bekannt ist, ergeben sich gleichzeitig zahlreiche Sicherheitsansprüche, die auch in verteilten Systemen relevant sind. Noch heute bestehen neben offensichtlichen Bedenken, beispielsweise bei der sicheren Übertragung und Speicherung sensibler Daten, wie etwa bei Passwörtern, oft Lücken bei der Authentifizierung von Nutzern, der ungefragten Erstellung von Werbeprofilen oder der unerwünschten Kommunikationsaufnahme. [Tan08, S. 20 f.]

Ein weiteres Ziel besteht in der Herstellung verschiedener *Transparenzen* des verteilten Systems. Darunter werden Vorteile zusammengefasst, die dem Nutzer des Systems allerdings gezielt verborgen werden sollen. Unter einer *Ortstransparenz* wird das Verbergen des genauen Aufenthaltes einer Ressource oder Anwendung verstanden. Der Nutzer greift somit lediglich über einen Namen zu, der keinen Rückschluss auf den eigentlichen Aufenthaltsort zulässt. *Zugriffstransparenz* meint eine identische Form des Zugangs auf alle Ressourcenarten, unabhängig von ihrem eigentlichen Aufenthaltsort. Greifen mehrere Nutzer parallel auf ein System oder eine Ressource zu, so soll die *Nebenläufigkeitstransparenz* garantieren, dass ein exklusiver Zugriff möglich wird und eventuelle Änderungen versteckt für andere Benutzer synchronisiert werden. *Skalierungstransparenz* beschreibt, dass auf eventuelle Änderungen und Erweiterungen von Hard- oder Software flexibel reagiert wird und sich für Nutzer nur minimale Unterbrechungen ergeben. Wird eine Ressource oder ein Prozess auf ein anderes System migriert, soll dieser Prozess im Rahmen der *Migrationstransparenz* verdeckt geschehen. Unter *Leistungstransparenz* wird die automatisierte gleichmäßige Verteilung der gesamten Arbeitslast auf verschiedene Komponenten verstanden, um so die Performanz des Gesamtsystems zu erhöhen. Beim Vorliegen von Kopien von Ressourcen soll die *Replikationstransparenz* sicherstellen, dass Kopien konsistent bleiben und sich für den Nutzer somit der Eindruck lediglich einer Instanz ergibt. Maßnahmen, um Systemfehler und -ausfälle auszugleichen und für den Nutzer somit weiterhin eine Nutzung des Gesamtsystems zu ermöglichen, werden unter *Fehler-* und *Ausfalltransparenz* klassifiziert. [Ben14, S. 10 ff.]

Durch die Einbindung von Diensten nach vorgefertigten Kriterien, wie etwa Syntax und Semantik, kann das Ziel eines offenen verteilten Systems realisiert werden. Durch das Spezifizieren von Schnittstellen wird das Kommunizieren verschiedener Prozesse ermöglicht und innerhalb des Systems standardisiert. Gleichzeitig ist das unterschiedliche Implementieren dieser Schnittstellen möglich, wodurch verschiedene Implementierungen den gleichen Stellenwert innerhalb des verteilten Systems einnehmen können. Die Offenheit umfasst zudem eine einfache Erweiterbarkeit durch austauschbare Komponenten, die von verschiedenen Entwicklern in das System eingepflegt werden können. Derartige Komponenten sollen hinzufüg- oder entfernbar sein, ohne bereits vorhandene Funktionen zu beeinträchtigen. [Tan08, S. 25]

2.4.2. Skalierbarkeit

Die Skalierbarkeit ist ein Ziel, welches implizit in diversen Konzepten des vorhergegangenen Kapitels 2.4.1 erwähnt wurde. Skalierbarkeit beschreibt im Allgemeinen die Fähigkeit eines Systems, eine steigende Anforderungslast durch hinzufügbare Ressourcen auszugleichen. Das Hinzufügen sollte dabei keine Änderung von Komponenten des Systems nach sich ziehen. Teilkonzepte der Skalierbarkeit können je nach Möglichkeit ebenfalls auf die Implementierungen der Prozessanwendungen ausgelagert werden. [SS12, S. 6]

Die Skalierung eines Systems ist in verschiedenen Dimensionen umsetzbar, wobei jeweils bestimmte Probleme zu lösen sind. Im Folgenden soll der Punkt der Größenskalierung betrachtet werden, in dem ein entscheidendes Problem in der Nutzung von zentralisierten Diensten, Daten oder Algorithmen besteht. Auch wenn sich ein zentralisierter Ansatz bei bestimmten Anwendungsklassen, wie etwa hochsensiblen Daten, nicht vermeiden lässt, kann es bei einer hohen Nutzungslast zur Überlastung des Servers kommen. Bei dezentralisierten Systemen sollten verschiedene Eigenschaften beachtet werden, um sie von ihren zentralisierten Pendants zu differenzieren. So sollte kein Computer vollständige Informationen über den Systemstatus abrufen können und somit nur auf Basis lokaler Informationen entscheiden. Wichtig ist dabei auch die Annahme, dass es keine globale Uhr gibt. Zudem sollte der Ausfall eines Computers den Algorithmus nicht schädigen. [Tan08, S. 27 f.]

Um den Ansatz der Skalierbarkeit zu verfolgen, können mehrere Techniken verwendet werden. Eine Technik besteht dabei in dem bestmöglichen Verstecken von auftretenden Latenzzeiten innerhalb der Kommunikation verschiedener Systemkomponenten. Ein anderer Ansatz sieht die asynchrone Auslagerung von Prozessen in nebenläufige Strukturen vor, wenn Aufgaben unabhängig vom eigentlichen Ablauf der Anwendung ausführbar sind. Zudem ist die Kommunikation reduzierbar, wenn bestimmte Aufgaben beim Client statt beim Server durchgeführt werden. Wie es beispielsweise in Abbildung 2.3 gezeigt wird, ist es deutlich effizienter einen Formularinhalt nach Abschluss auf der Seite des Clients zu validieren, statt nach jeder Eingabe eine verzögerte Validierung beim Server durchzuführen. [Tan08, S. 30 f.]

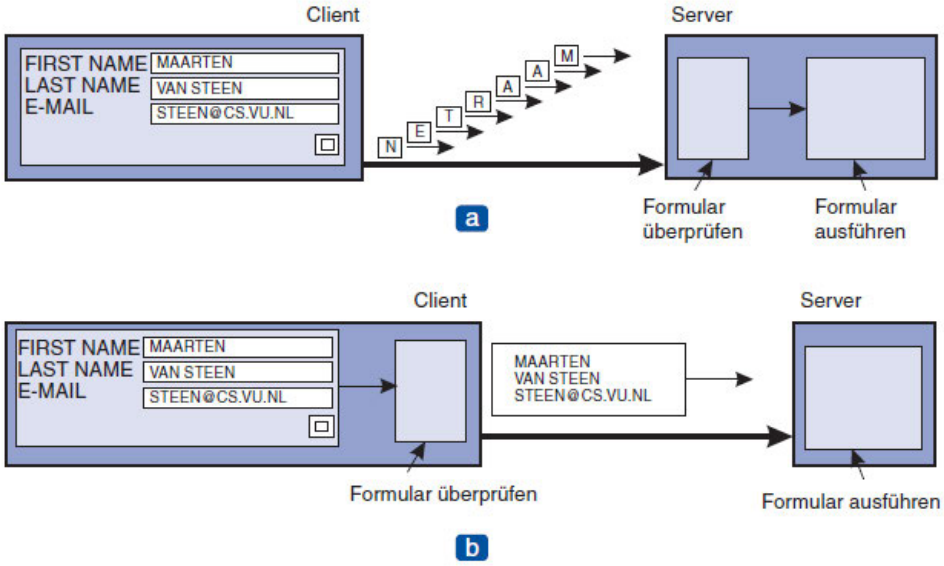

Abbildung 2.3.: Formularüberprüfung auf dem Server (a) und beim Client (b) [Tan08, S. 30]

Wie es der Name der verteilten Systeme impliziert, ist die Verteilung selbst ein grundlegender Ansatz. Dabei wird ausgehend von einer Komponente versucht, diese in kleinere Teilkomponenten zu zerlegen und anschließend über das Gesamtsystem zu verteilen. Eine weitere Herangehensweise besteht in der Replikation von Systemkomponenten, was neben einer stabileren Systemverfügbarkeit auch zu einem verbesserten Lastenausgleich beiträgt. [Tan08, S. 31]

2.5. DevOps

Der Begriff DevOps wird heute für eine Vielzahl verschiedener Szenarien und Inhalte genutzt, sodass eine allumfassende Definition kaum festzulegen ist. Der Begriff kommt aus einer Kombination der englischen Worte “development“ und “operations“ und ist somit als Schnittmenge zwischen der Softwareentwicklung und Operationen, die zur Produktion und Verwaltung der Software und ihrer Infrastruktur gehören, zu verstehen. Es werden dabei verschiedene Grundsätze in den Vordergrund gestellt, so etwa die Zentrierung von DevOps um den Menschen, eine Automatisierung von

Prozessen oder die Messbarkeit möglichst vieler Kriterien. Insgesamt soll durch die Integration von DevOps nicht nur die Produktionszeit, sondern auch die Qualität der Software verbessert werden. [Hü12, S. 3 f.]

Neben den bereits erläuterten Vorteilen einer Automatisierung kann eine Vielzahl von Motivationen dazu führen, dass DevOps in den Entwicklungsprozess integriert wird. Durch die Einrichtung einer Produktions-Pipeline bereits während der Entwicklungszeit ist zudem besser feststellbar, wie nah der Funktionsumfang und die Lieferzeiten des Produktes bereits an den Realanforderungen sind. Auch die eventuelle Virtualisierung der Software, wie sie etwa durch Container im Kapitel 2.5.2 genauer erläutert wird, kann eine besondere Rolle spielen, um die spätere Portierbarkeit der Anwendung sicherzustellen. Für die Entwicklung kann außerdem der Review-Prozess des Quellcodes verbessert werden, sodass verschiedene Versionen der Codebasis besser unterscheidbar sind und Missverständnisse vermieden werden. [Vad18, S. 3 ff.]

2.5.1. Continuous Integration, Delivery und Deployment

Continuous Integration (CI) beschreibt den Prozess der Softwareentwicklung, bei dem alle beteiligten Entwickler agile Methoden und Grundsätze einhalten. Im Rahmen des Softwareprojekts, das zwingend in ein Versionskontrollsystem eingepflegt sein sollte, sollen so beispielsweise Code Reviews durchführbar sein. Zudem sollten Änderungen automatisiert integriert, getestet und abschließend zu einer Anwendung gebaut werden. Der Bauprozess soll überwacht und mit Metriken auswertbar sein, eventuelle Fehler sollen dabei schnell einsehbar sein. Continuous Delivery (CD) realisiert aufbauend auf der Continuous Integration die manuelle Möglichkeit, die neueste, ohne Fehler gebaute Version der Software automatisiert in den Veröffentlichungsprozess zu überführen.[Vad18, S. 40 ff.]

Continuous Deployment ähnelt dem Ansatz der Continuous Delivery und benötigt eine bereits aufgebaute Delivery-Pipeline. Der Hauptunterschied besteht in der Automatisierung des Deployments. Statt der manuellen Einleitung der Veröffentlichung der Produktion werden die Schritte zur Akzeptierung einer neuen Version automatisiert. [Cha21, S. 3]

Zur Vereinfachung wird im weiteren Verlauf der wissenschaftlichen Arbeit lediglich von der Kombination aus Continuous Integration und Continuous Deployment (CI/CD) gesprochen, ohne den Zwischenschritt der Continuous Delivery explizit zu erwähnen.

2.5.2. DevOps-Technologien

Für die Integration der erläuterten DevOps-Prinzipien existieren zahlreiche Technologien, welche verwendet werden können. Zwei bedeutende Möglichkeiten sollen im Folgenden genauer erläutert werden.

Eine Lösung für CI/CD lässt sich in die populäre Versionskontroll-Hostinglösung *GitHub* integrieren. Die als *GitHub Actions* bekannte Technologie erlaubt das Anlegen von mehreren *Jobs*, die in ihrer Gesamtheit einen *Workflow* ergeben und beim Ausführen sowohl parallel als auch sequenziell durchlaufen werden können. Ein Job besteht dabei wiederum aus mehreren *Actions*, die lediglich eine individuelle Aufgabe ausführen und somit die kleinsten Komponenten des Gesamtworkflows darstellen. GitHub bietet über einen Marktplatz eine Vielzahl vorhandener Actions an, allerdings ist es ebenso möglich, eigene Actions zu erstellen. Alle Dateien, die während der Ausführung eines Workflows generiert werden und weiter genutzt werden können, werden als *Artifacts* bezeichnet. Für die automatisierte Ausführung eines Workflows erlaubt GitHub Actions das Anlegen von so genannten *Events*, die bei bestimmten Aktionen den Workflow starten. Jeder Job im Workflow wird dabei auf einem so genannten *Runner* ausgeführt. Es gibt unter GitHub freie Runner für mehrere virtualisierte Betriebssysteme. Es ist aber genauso möglich, eine eigene Maschine mit GitHub Actions zu verknüpfen. [Cha21, S. 5 ff.]

Um eine plattformübergreifende Nutzung der Anwendung zu ermöglichen, besitzt die Virtualisierung der Anwendung über Container einen besonderen Stellenwert. Container besitzen im Gegensatz zu einer Virtual Machine (VM) kein eigenes Betriebssystem, sondern teilen sich ein Betriebssystem des Hosts. Neben Leistungseinsparungen im Vergleich zur Nutzung einer VM wird damit auch die Verteilung einer Anwendung auf verschiedene Systeme erheblich erleichtert. Im Folgenden wird die Technologie *Docker* betrachtet, die das Erstellen, Verwalten und Orchestrieren von Containern ermöglicht. Docker setzt sich im Wesentlichen aus drei Komponenten zusammen. Die niedrigste Ebene der Runtime ist hauptsächlich für das Star-

ten und Stoppen von Containern verantwortlich. Die Docker Engine (auch Docker Daemon) übernimmt höherwertige Aufgaben, wie das Bereitstellen der Docker Remote API oder das Verwalten diverser Docker-Komponenten. Die Orchestrierungsebene stellt Cluster über Docker bereit. Einem Container in Docker liegt ein so genanntes *Image* zugrunde. Das Image enthält alle zum Starten eines Containers notwendigen Bestandteile, wie den Anwendungscode, notwendige Abhängigkeiten oder Betriebssystem-Grundlagen. [Pou20, S. 8 ff., S. 51]

3. Anforderungsanalyse und Konzeptionierung

Aus den unter Kapitel 2 dargestellten Grundlagen sollen zunächst ausgewählte Punkte, die für die Entwicklung der Servicestruktur von besonderer Bedeutung sind, genauer analysiert werden. Im Anschluss sollen aus den gewonnenen Erkenntnissen Anforderungen für die Entwicklung der Anwendung abgeleitet werden. Darauf aufbauend wird das Konzept für die zu entwickelnde Anwendung anhand der Systemarchitektur und zu nutzenden Technologien konzeptioniert. Abschließend wird das Konzept der Evaluation, welche in Kapitel 5 durchgeführt wird, beschrieben.

3.1. Analyse von Plattformen und Standards

Durch eine tiefere Einordnung von bereits vorhandenen Plattformen und Standards soll eine bessere Abschätzung relevanter Entwicklungskriterien möglich werden. Im Fokus der Analyse sollen Bedürfnisse stehen, die im Generellen durch digitale Lehr-Lernangebote befriedigt werden sollen. Es werden dabei Entwicklungen der letzten Jahre sowie die Relevanz verschiedener Formate des E-Learnings berücksichtigt.

Aus allen dargestellten Lehr-Lernformen wird deutlich, dass die Einbindung von Ressourcen verschiedenster Art für nahezu alle Einsatzzwecke einen hohen Stellenwert besitzt. Als Ressourcen können neben einfachen Textinhalten auch Dateien oder Inhaltsstrukturen für die Plattform notwendig werden. Auch die Möglichkeit zum Anlegen und Verwalten von Nutzerprofilen ist bei nahezu allen Plattformarten unverzichtbar, um beispielsweise eine gezielte Distribution von Inhalten oder Austausch zwischen Nutzern zu ermöglichen. Einhergehend mit der Erstellung von Nutzerprofilen ist zudem eine Einrichtung personalisierter Inhalte möglich. Nutzer können damit Daten und Einstellungen für die Lernplattform hinterlegen, die nur für sie gültig oder einsehbar sind.

Mit der Komplexität einer Plattform nimmt meist auch die Quantität und Diversität der zugehörigen Inhalte zu. Besonders im Rahmen von MOOCs können vielseitige Kurse und Materialien auf einer Plattform bereitgestellt werden. Vor allem dann wird eine Kategorisierung entsprechender Ressourcen wichtig, damit Lernende und Lehrende die Übersicht bewahren und Inhalte einfacher einordnen können. Damit verbunden steigt die Nachfrage des gezielten Suchens nach plattformspezifischen Daten. Derartige Suchfunktionen können für simple Zwecke durch inhaltliche Kennzeichnungen, wie so genannte Tags, realisiert werden. Für komplexere Zwecke sind aber auch tiefergreifende Verknüpfungen, wie die Assoziation von Nutzern oder Nutzergruppen mit der Erstellung und Bearbeitung von Ressourcen, möglich.

Auch wenn die oft fehlende Interoperabilität zwischen Plattformen der Kern des betrachteten Problems der vorliegenden wissenschaftlichen Arbeit ist, existieren einige Standards für die Domäne. Plattformen, welche derartige Standards implementieren, erlauben Wege zum Datenaustausch oder der Darstellung entsprechender Daten. Eine Umsetzung dieser Standards innerhalb der generischen Servicestruktur besitzt somit eine hohe Priorität, um eine Integration des Service in bereits existierende Schnittstellen, welche eine Einbindung entsprechender Standards erlauben, zu ermöglichen.

Die Entwicklungen, die sich beispielsweise aus der Veränderung demographischer Bedingungen oder der Covid-19 Pandemie ergeben haben, zeigen die Vielfältigkeit der Relevanz verschiedener Plattformarten. Auch wenn diese neben der Optimierung für bestimmte Lehrinhalte teils an sozio-kulturelle Kriterien gebunden sind, ist eine Rekombination verschiedener Grundkonzepte nicht ausgeschlossen. So wäre es beispielsweise möglich, innerhalb eines MOOCs erweiterte Möglichkeiten einzubinden, die einen Austausch Lehrender und Lernender ermöglichen. Umgedreht können Elemente, wie zum Beispiel die Bewertung eines Kurses, die eher einem MOOC zugeschrieben werden, in anderen Lernmodellen verwirklicht werden. Auch die Integration von Elementen des PLE innerhalb eines VLE oder anderen Arten ist heute keine Seltenheit. So bieten Lernplattformen beispielsweise eigene Kalender, die das Organisieren von Aufgaben oder Terminen innerhalb der Lernumgebung ermöglichen oder sogar automatisieren. Entsprechende Angebote sind oft auch als mobile Anwendungen vorhanden, sodass eine optimale Verfügbarkeit und Zugänglichkeit der Plattform angeboten werden können. Kriterien dieser Art sollten bei der Entwicklung eines neuen Angebotes und der zugrundeliegenden Servicestruktur dringend berücksichtigt werden.

3.2. Anforderungen

Die Konzeptionierung und anschließende Implementierung der Anwendung soll von diversen Anforderungen abhängig sein. Diese umfassen den grundlegenden Aufbau des Service, ebenso wie die Anpassung an spezifische Lernformen und deren Erweiterbarkeit in der Codestruktur.

Anhand der unter Kapitel 3.1 analysierten Gesichtspunkte wird klar, dass die Anforderungen an ein Backend innerhalb der E-Learning Domäne teils generisch gehalten sein müssen, um eine abstrakte Anpassung an ein breites Anwendungsfeld zu ermöglichen. Insgesamt können jedoch Anforderungen formuliert werden, die eine möglichst breite Palette verschiedener Plattformarten und zugehörige Werkzeuge realisierbar machen. Dabei besitzt die Möglichkeit, verschiedene Grundkonzepte miteinander zu kombinieren, einen besonders hohen Stellenwert.

Es soll eine Einteilung zwischen funktionalen und nicht-funktionalen Anforderungen, wie sie genauer unter Kapitel 2.1.1 erläutert wurde, vorgenommen werden. Unter der Maßgabe, dass der Service neben einer einfachen Nutzung auch für eine Weiterentwicklung geeignet sein soll, spaltet sich das RE in Anforderungen für potenzielle Clients und Anforderungen für die Weiterentwicklung des Servers. Als Client werden dabei alle Zwecke klassifiziert, die die grundlegende Servicestruktur verwenden, beispielsweise für die Bereitstellung einer Benutzeroberfläche. Die Weiterentwicklung des Servers meint dagegen eine Erweiterung der bereitgestellten Codebasis um neue Funktionen, Inhalte oder Schnittstellen.

	ID	Beschreibung
Funktionale Anforderungen	**FA-10**	**Authentifizierung**
	FA-11	Zugänglichkeit nach Nutzergruppen
	FA-12	Verwaltung von Nutzergruppen und -berechtigungen
	FA-20	**Selektive Anfrage von Daten**
	FA-30	**Wiederverwendbarkeit von Kurs-Komponenten**
	FA-31	Möglichkeit zum Anlegen von Übungen verschiedener E-Learning-Systeme
	FA-32	Modularisierung von Teilen des Service
	FA-40	**Verwaltbarkeit verschiedener Kursstrukturen**
	FA-41	Kategorisierung von Kursen und Ressourcen nach ihrem Kontext und Einsatz
	FA-42	Realisierung von Suchanfragen
	FA-50	**Implementierung des xAPI-Standards**
	FA-60	**Möglichkeit zum einfachen Deployment der Gesamtstruktur**
	FA-61	Verfügbarkeit unter Windows, Linux und Mac
Nicht-funktionale Anforderungen	**NA-10**	**Performanz der Datenanfrage**

Tabelle 3.1.: Anforderungen für die Nutzung auf der Clientseite

Die funktionalen Anforderungen auf der Clientseite fokussieren sich im Wesentlichen auf die Rahmenbedingungen der Datenanfrage und -restrukturierung. Durch die Möglichkeit der Authentifizierung soll sichergestellt werden, dass bestimmte Daten nur von Nutzern abgerufen und bearbeitet werden können, die dazu berechtigt sind. Zusätzlich soll der Server eine Verwaltung entsprechender Gruppen und Berechtigungen möglich machen. Daten, die auf der Seite der Plattform und dessen LMS zur Verfügung stehen, müssen vom Client selektiv abrufbar sein. Damit soll garantiert werden, dass keine überflüssigen Informationen an den Client übermittelt werden, was im Kontext eines generischen Service von besonderer Wichtigkeit ist. Eine weitere Anforderung umfasst die umschriebene Möglichkeit zur Umsetzung und Rekombination verschiedener Plattformgenres. Teilziele, wie die Modularisierung von Komponenten, die zu einer Gesamtplatform zusammengesetzt werden, sollen zur Erfüllung der Gesamtanforderung genutzt werden. Standards digitaler Lehr-Lern-System, wie sie unter Kapitel 2.2.3 zusammengefasst wurden, sollen innerhalb des

Service integriert sein. Zuletzt soll auch das plattformunabhängige Deployment für den Client unkompliziert durchführbar und möglichst einfach in bereits vorhandene Systeme integrierbar sein.

Von hohem Stellenwert ist die nicht-funktionale Anforderung der Clientseite, innerhalb einer Request angeforderte Daten möglichst performant vom Server zu erhalten.

	ID	**Beschreibung**
Funktionale Anforderungen	**FA-70**	**Erweiterbarkeit der persistenten Strukturen**
	FA-80	**Automatisiertes Deployment**
Nicht-funktionale Anforderungen	**NA-20**	**Ressourcensparender Betrieb**
	NA-30	**Dokumentation des Quellcodes**
	NA-40	**Sicherstellung der Skalierbarkeit durch Verteilung**
	NA-41	Wartbarkeit des Systems
	NA-50	**Automatisierte Tests**

Tabelle 3.2.: Anforderungen für die Wartung und Entwicklung des Servers

Im Vordergrund der funktionalen Anforderungen zur Weiterentwicklung steht die Möglichkeit, neue Plattformarten in die Servicestruktur einzupflegen. Damit zusammenhängend muss sichergestellt sein, dass vorhandene Datenbankstrukturen derartige Änderungen möglich machen, ohne eine umfangreiche Migration vorhandener Inhalte nach sich zu ziehen. Änderungen sollen zudem automatisiert in der Deployment-Pipeline genutzt werden können, um neue Versionen des Service bereitzustellen.

Auf der Seite nicht-funktionaler Anforderungen steht neben einer ressourcensparenden Nutzung der Anwendung die Realisierbarkeit und Integrierbarkeit von Änderungen im Fokus. Entwickler sollen sich in den vorhandenen Quellcode einarbeiten können, weshalb eine Dokumentation des Systems unverzichtbar ist. Damit geht eine Wartbarkeit des Systems einher, die das Einpflegen größerer Änderungen möglich macht. Auch sollen Neuerungen auf die Kompatibilität mit dem Gesamtsystem testbar sein, weshalb automatisierte Tests entwickelt werden sollen.

3.3. Konzept der Anwendung

Anhand der im Kapitel 3.2 abgeleiteten Anforderungen werden im Folgenden die Grundlagen zur konkreten Implementierung der Anwendung konzeptioniert. Zuerst erfolgt eine grundlegende Systemarchitektur, welche die verschiedenen Systemkomponenten in ihrer gegenseitigen Interaktion zusammenfasst. Darauf aufbauend erfolgt die Wahl der Technologien, welche für die Implementierung verwendet werden. Außerdem werden Aspekte, die die konkrete Implementierung vorplanen, näher beleuchtet.

3.3.1. Systemarchitektur und Technologien

Im Wesentlichen besteht die Servicestruktur aus zwei Komponenten. Für die Speicherung von Datensätzen, wie etwa Nutzern, Kursinhalten oder Hierarchien, wird ein Datenbanksystem notwendig. Um Daten aus der Datenbank zu laden, neue Datensätze anzulegen oder vorhandene Daten zu aktualisieren, stellt der Service eine zentrale Schnittstelle bereit. Um die beschriebenen Aufgaben ausführen zu können, muss eine Kommunikation zwischen Schnittstelle und Datenbank gewährleistet werden. Der Service kann von potenziellen Clients, wie beispielsweise Frontends, genutzt werden, um Anfragen an den Server zu senden und angefragte Daten zu erhalten. Die grundlegende Architektur ist in Abbildung 3.1 dargestellt. Darüber hinaus soll das Deployment der gesamten Struktur über eine CI/CD-Pipeline gewährleistet werden. Neue Versionen des Service sollen somit unkompliziert erstell- und veröffentlichbar sein.

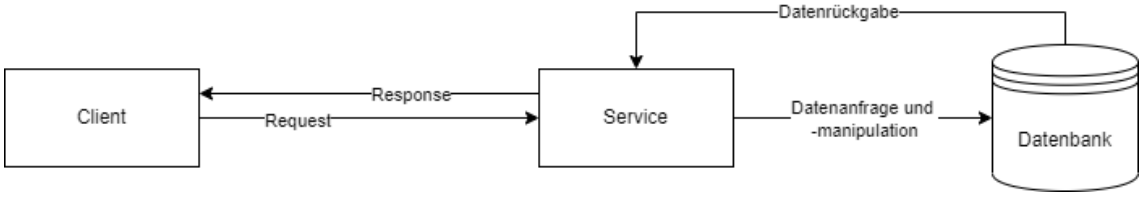

Abbildung 3.1.: Grundstruktur des Service unter Nutzung eines Clients

Ausgehend von der geplanten Systemarchitektur sollen Entscheidungen für Technologien, die zur Implementierung und Distribution der Anwendung verwendet werden, getroffen werden. Die Wahl ist dabei an die grundlegenden Anforderungen angelehnt und soll eine bestmögliche Umsetzung dieser ermöglichen.

Für die Implementierung des Backends wird die Sprache JavaScript mit der Laufzeitumgebung *NodeJS* verwendet. Für die Einbindung von zusätzlichen Paketen, die die Entwicklung nach Möglichkeit unterstützen, soll der zu NodeJS zugehörige Paketmanager Node Package Manager (NPM) verwendet werden. NPM bietet eine große Zahl von frei verfügbaren Paketen, die in die von NodeJS bereitgestellte Umgebung eingebunden werden können. Derartige NPM-Pakete erleichtern die Entwicklung somit erheblich, indem bestimmte Tools oder Schnittstellen vereinfachend bereitgestellt werden.

Als zugrundeliegendes Datenbankmanagementsystem soll MongoDB verwendet werden. Die unter Kapitel 2.3.1 aufgezeichneten Vorteile des nicht-relationalen Datenbanksystems tragen zur Flexibilität und Erweiterbarkeit der Anwendung bei. Zum Modellieren, Validieren und Anfragen von Daten wird die NPM-Bibliothek *Mongoose* genutzt. Diese erleichtert die Modellierung von Datenstrukturen unter MongoDB erheblich. Darüber hinaus ist es möglich, Einschränkungen und Metadaten in die Modelle einzubinden.

Die Anfrage des Webservice wird über GraphQL realisiert. Besonders die Möglichkeit der selektiven Datenanfrage über GraphQL soll die Flexibilität des Service bedeutend unterstützen. Durch eine übersichtliche Datenmodellierung wird in Kombination mit MongoDB zudem die bestmögliche Erweiterbarkeit der Anwendung garantiert. Die verwendete Bibliothek, die über NPM eingebunden wird, ist *Apollo*. Neben der reinen GraphQL-Implementierung verwirklicht Apollo ebenfalls die grundlegende Serverstruktur. Für die Authentifizierung von Anfragen diverser Clients sollen JSON Web Tokens (JWTs) zum Einsatz kommen. JWTs realisieren dabei für den Server sicher validierbare Session-Tokens, über die Anfragen authentifiziert werden können. Zudem kann innerhalb des Payloads des Tokens ein JSON-Inhalt hinterlegt werden, was das flexible Übergeben von Daten ermöglicht.

Um eine reibungslose Bereitstellung der Anwendung zu garantieren, soll ein CI/CD-Workflow eingerichtet werden. Um eine einfache Integration in Verbindung mit der Versionskontrolle zu ermöglichen, wird hierfür GitHub Actions verwendet. Durch die

von GitHub bereitgestellten Runner sowie Actions, die die notwendigen Komponenten des Deployments realisieren, wird der Entwicklungsprozess weiter vereinfacht. Es wird damit gleichzeitig vorausgesetzt, dass das Quellcode-Repository auf der freien Hosting-Lösung *GitHub* hochgeladen wird. Die Einrichtung des Workflows mit den entsprechenden Runnern von GitHub soll damit unmittelbar innerhalb des Repositories geschehen.

Für das Deployment des Backends wird Docker genutzt. Die Bereitstellung des entsprechenden Docker-Images soll dabei auf der offiziellen Image-Bibliothek *Docker Hub* geschehen, was die öffentliche Verfügbarkeit des Images garantiert. Durch die verwendete CI/CD-Pipeline wird der gesamte Prozess, vom Bauen des Images bis hin zu dessen Veröffentlichung, vollautomatisiert geschehen. Für das Testen des Images kann ein entsprechender Container auf der lokalen Maschine erstellt werden. Hierfür soll *Docker Desktop* unter dem Betriebssystem Windows 11 Pro verwendet werden. Die verwendete Datenbank soll im Prozess des Deployments ebenfalls in Docker ausgelagert und über das entsprechende Image integriert werden. Um ein gezieltes Deployment der Anwendung mitsamt der Datenbank schnell auch auf anderen Systemen möglich zu machen, soll eine vorgefertigte Docker-Compose-Datei bereitgestellt werden. Das Starten des Service ist somit zu Testzwecken auf einer lokalen Hostmaschine ebenso möglich wie auf virtuellen Maschinen oder externen Servern.

3.3.2. Konzept der Datenstrukturen

Für die Anwendungen werden komplexe Datenstrukturen, die in gegenseitigem Zusammenspiel stehen, notwendig. Es soll deshalb bereits vor der Entwicklung geplant werden, welche Arten von Daten anfallen, wie diese kategorisierbar sind und in welcher Beziehung sie untereinander stehen.

Durch die in Kapitel 3.3.1 dargelegte Wahl von MongoDB als verwendetes Datenbanksystem wird deutlich, dass die Modellierung der Datenstrukturen auf einem nicht-relationalen Ansatz beruht. Um eine gezieltere Implementierung der Datenstrukturen zu ermöglichen, wird zunächst eine Modellierung angefertigt. Die Form der Modellierung orientiert sich grob an der Modellierungssprache UML, ist jedoch für die Nutzung der zugrundeliegenden Implementierung Mongoose stark vereinfacht. Für jede Eigenschaft einer Dokumentstruktur ist ein Datentyp vermerkt.

Dieser stellt entweder eine Mongoose Objekt-ID, ein Datum, eine Zeichenkette, einen Zahlenwert oder eine Referenz auf ein anderes Datenmodell dar. Bei Referenzen handelt es sich um ganze Datensätze, die innerhalb der gegebenen Struktur einbezogen werden, oder lediglich um Zeiger auf Objekt-IDs von bereits bestehenden Datensätzen. Daten, welche mit einem Ausrufezeichen versehen sind, erlauben dabei keine Nullwerte. Zuletzt kann mit einem Doppelpunkt im Dokumentenkopf eine Vererbungsbeziehung angedeutet werden. Eine solche Vererbung bedeutet, dass der neue Dokumententyp alle Eigenschaften des zugrundeliegenden Basisdokuments besitzt und diese lediglich um weitere Eigenschaften erweitert.

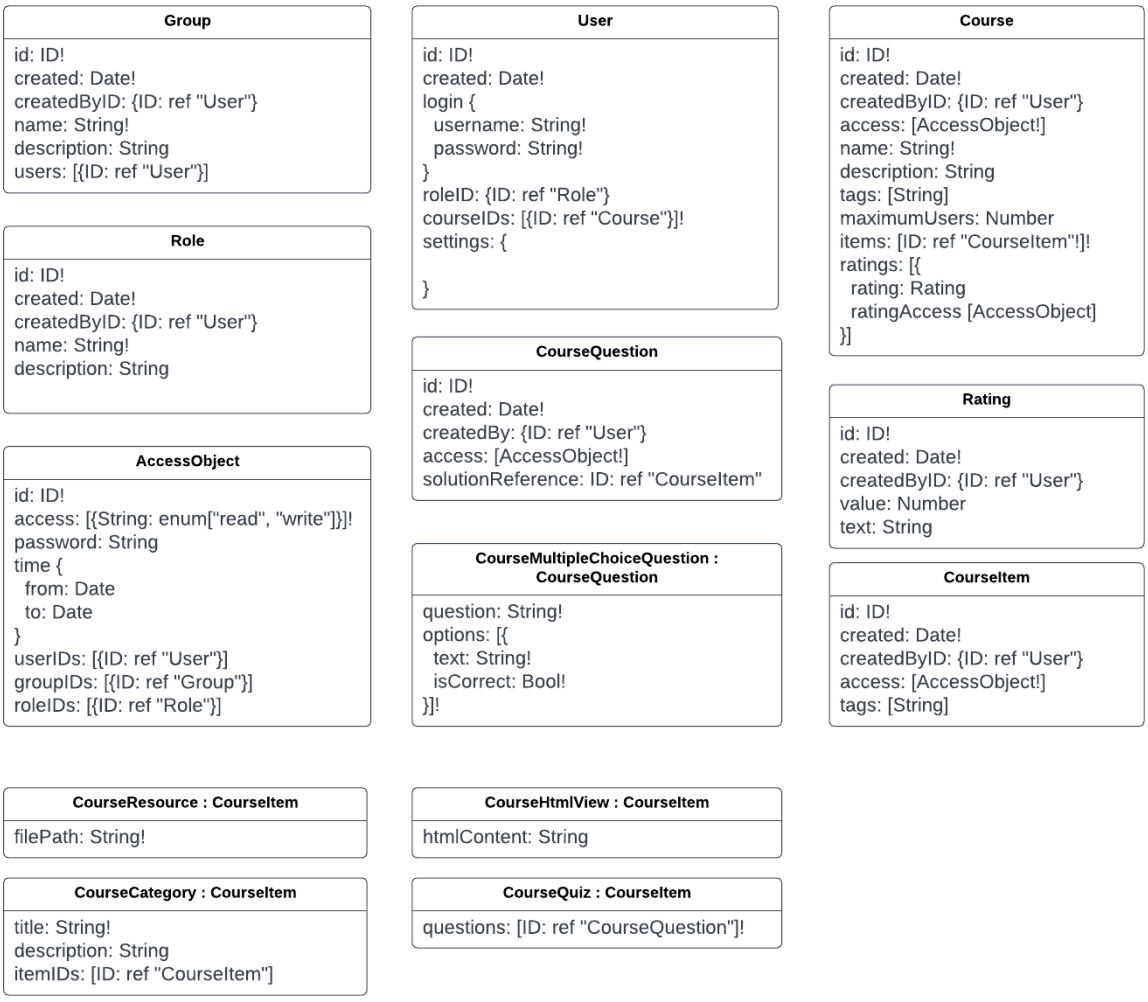

Abbildung 3.2.: Modellierung der Mongoose-Datenstrukturen

Die Modellierung der Datenstruktur für die prototypische Anwendung ist unter der Abbildung 3.2 dargestellt. Es wird dabei die abgewandelte Modellierungsform zur Darstellung der MongoDB-Dokumentenstruktur verwendet.

3.4. Evaluationskonzept

Nach der Fertigstellung der Anwendung soll eine Evaluierung des Systems vorgenommen werden, wofür mehrere Ansätze zur Verfügung stehen. Dafür spielen die unter Kapitel 3.2 gestellten Anforderungen eine besondere Rolle. Um eine Überprüfung der Anforderungen zu ermöglichen, werden im folgenden Kapitel Testfälle aufgeführt. Zusätzlich wird spezifiziert, wie die Durchführung der Testfälle auf der Clientseite vorgenommen wird.

3.4.1. Testfälle

Die Validierung des entwickelten Systems soll anhand von spezifischen Testfällen geschehen. Die Testfälle dienen dabei der Überprüfung der Anforderungen, welche unter Kapitel 2.1.1 dargelegt wurden. Für jede Anforderung soll dabei ein Testfall, der die Erfüllung durch die gewählte Implementierung vornimmt, entworfen werden. Entsprechend der Unterscheidung in client- und serverspezifische Anforderungen erfolgt auch die Einteilung der zugehörigen Testfälle in die zwei Kategorien.

ID	Anforderung	Beschreibung
FA-10-TF	FA-10	Prüfung der Authentifizierung
FA-11-TF	FA-11	Prüfung der Zugänglichkeit nach Nutzergruppen
FA-12-TF	FA-12	Prüfung der Verwaltung von Nutzergruppen und -berechtigungen
FA-20-TF	FA-20	Prüfung der selektiven Anfrage von Daten
FA-30-TF	FA-30	Prüfung der Wiederverwendbarkeit von Kurs-Komponenten
FA-31-TF	FA-31	Prüfung der Verfügbarkeit von Grundkonzepten verschiedener E-Learning-Systeme
FA-32-TF	FA-32	Prüfung der Modularisierung von Teilen des Service
FA-40-TF	FA-40	Prüfung der Verwaltbarkeit verschiedener Kursstrukturen
FA-41-TF	FA-41	Prüfung der Kategorisierung von Kursen und Ressourcen nach ihrem Kontext und Einsatz
FA-42-TF	FA-42	Prüfung von Suchanfragen
FA-50-TF	FA-50	Prüfung der Implementierung des xAPI-Standards
FA-60-TF	FA-60	Prüfung der Möglichkeit zum einfachen Deployment der Gesamtstruktur
FA-61-TF	FA-61	Prüfung der Verfügbarkeit unter Windows, Linux und Mac
NA-10-TF	NA-10	Prüfung der Performanz der Datenanfrage

Tabelle 3.3.: Testfälle für die Nutzung auf der Clientseite

ID	Anforderung	Beschreibung
FA-70-TF	FA-70	Prüfung der Erweiterbarkeit der persistenten Strukturen
FA-80-TF	FA-80	Prüfung des automatisierten Deployments
NA-20-TF	NA-20	Prüfung des ressourcensparenden Betriebs
NA-30-TF	NA-30	Prüfung der Dokumentation des Quellcodes
NA-40-TF	NA-40	Prüfung der Skalierbarkeit durch Verteilung
NA-41-TF	NA-41	Prüfung der Wartbarkeit des Systems
NA-50-TF	NA-50	Prüfung der automatisierten Tests

Tabelle 3.4.: Testfälle für die Wartung und Entwicklung des Servers

Für alle Testfälle werden detailliertere Spezifikationen festgelegt. Darin werden mögliche Abhängigkeiten von anderen Anforderungen, die vor der Durchführung des Testfalls erfüllt sein müssen, genannt. Außerdem wird die Ausgangssituation, die zu Beginn des Tests hergestellt sein muss, beschrieben. Auch wird vermerkt, mit welchem Ereignis der Testfall ausgelöst wird und welches Ergebnis im Falle der Durchführung erwartet wird. Im Folgenden wird je ein exemplarischer Testfall für die Nutzung des Service auf der Clientseite sowie für die Wartung und Entwicklung des Servers dargestellt. Alle weiteren Testfälle sind unter dem Anhang A aufgeführt.

FA-20-TF
Anforderung: FA-20 Selektive Anfrage von Daten
Abhängigkeiten: Keine.
Ausgangssituation: Server ist gestartet. Nutzer besitzt Authentifizierung zur Datenabfrage.
Ereignis: Nutzer fragt selektiv nur bestimmte Daten vom Server an.
Erwartetes Ergebnis: Server antwortet mit Datensatz, der dem geforderten Umfang der Anfrage entspricht.

FA-80-TF
Anforderung: FA-80 Automatisiertes Deployment
Abhängigkeiten: FA-60.
Ausgangssituation: Entwickler hat Server durch eine Änderung erweitert.
Ereignis: Entwickler lädt Entwicklungsstand in das GitHub Remote-Repository hoch.
Erwartetes Ergebnis: Automatische Deployment-Pipeline wird ausgelöst. Version mit entwickelten Änderungen steht nach erfolgreichem Durchlaufen der Pipeline im Rahmen des Deployments bereit.

3.4.2. Überprüfung clientspezifischer Anforderungen

Der wesentliche Teil der vorliegenden wissenschaftlichen Arbeit beschäftigt sich mit der für die E-Learning Domäne notwendigen Services. Derartige Strukturen werden jedoch entworfen, um schlussendlich für Anfragen eines Clients einsetzbar zu sein.

Ein Teil der Überprüfung der unter 3.4.1 aufgeführten Testfälle ist somit nur mit einem besonderen Fokus auf die Clientseite möglich. Um eine Validierung der Anforderungen durch die Durchführung entsprechender Testfälle zu ermöglichen, soll zu Testzwecken ein primitiver Client genutzt werden. Dieser soll nicht den spezifischen Zweck einer Lehr-Lernplattform erfüllen, sondern lediglich durch zielgerichtet entwickelte Softwarekomponenten Anfragen ausführen und entsprechende Ergebnisse darstellen können. Die Flexibilität und Ergebnisdarstellung soll aufgrund der untergeordneten Relevanz für die Evaluation möglichst simpel erfolgen.

Für die Testzwecke wird hierfür der Apollo-Testclient benutzt, welcher automatisch mit der Servicetechnologie Apollo-Server bereitsteht. Der Client erlaubt dabei das Ausführen von GraphQL Queries und Mutations inklusive von Datenheadern.

4. Implementierung

Das folgende Kapitel stellt die Umsetzung der Servicestruktur anhand der gestellten Anforderungen dar. Es soll dabei auf das methodisch strukturierte Vorgehen innerhalb der einzelnen Entwicklungsschritte eingegangen werden. Neben dem generellen Verfahren werden die bereitgestellten Schnittstellen und ihre Möglichkeiten zur Erweiterbarkeit vorgestellt. Abschließend wird die Einrichtung und Durchführung des Service-Deployments beschrieben.

4.1. Entwicklung der Servicestruktur

Die Entwicklung der Serverstruktur mit ihren zentralen Komponenten stellt den Hauptteil des Implementierungsprozesses dar. Aus diesem Grund werden in den folgenden Unterkapiteln die wichtigsten Zwischenschritte, die zur Umsetzung des Service notwendig sind, genauer erläutert. Wo notwendig, werden zudem die verwendeten Technologien genauer thematisiert, um einen detaillierten Einblick in die Softwarearchitektur geben zu können.

4.1.1. Vorbereitung

Vor dem eigentlichen Beginn der Implementierung sind einige vorbereitende Schritte notwendig. Diese betreffen primär die Installation von Bibliotheken und Tools, welche im Anschluss für die Entwicklung genutzt werden sollen.

Zu Beginn wird jegliche Software installiert, die für die Entwicklung notwendig ist. Im Falle der vorliegenden Arbeit ist darauf zu achten, dass die Software mit dem zur Entwicklung genutzten Betriebssystem Windows 11 Pro kompatibel ist. Voraussetzung für die Entwicklung der auf NodeJS basierenden Anwendung ist die Installation von NodeJS. Für eine einfache Installation wird dafür, ebenso wie für

die Versionskontrollsoftware *Git*, der offizielle Download der Windows-Installation benutzt. Zuletzt erfolgt die Installation des freien Code-Editors *Visual Studio Code*. Dieser erleichtert das Anlegen und Bearbeiten von Quellcode-Dateien und unterstützt durch integrierte Funktionen und einfach installierbare Erweiterungen den Entwicklungsprozess.

Das Projekt selbst wird in einem Ordner angelegt. Über die Kommandozeilenwerkzeuge von NPM können zahlreiche Operationen ausgeführt werden, die für die Initialisierung und Einrichtung des Projektes benötigt werden. Die Ausführung der Kommandos erfolgt durch eine Instanz der *Windows PowerShell*, die im zuvor erstellten Projektordner geöffnet wurde. Mithilfe des Befehls `npm init` wird innerhalb des Ordners eine *package.json*-Datei angelegt, die neben Metadaten zum Projekt alle Paket-Abhängigkeiten abspeichert. Diese Datei ist für NPM charakteristisch.

```
$ npm init
```

Im Anschluss an die Ausführung des Kommandos werden diverse Metadaten, wie beispielsweise der Projektname oder -autor, angegeben. Diese können ebenfalls zu einem späteren Zeitpunkt manuell in der *package.json*-Datei angepasst werden.

Innerhalb des durch NPM initialisierten Projektes sollen die unter Kapitel 3.3.1 aufgeführten Pakete installiert werden. Über das Kommando `npm install` werden die drei Pakete *Apollo*, *GraphQL* und *Mongoose* installiert. Die vorangestellte Option *save* stellt dabei sicher, dass die Pakete als Abhängigkeiten für das Projekt vermerkt werden.

```
$ npm install --save apollo-server graphql mongoose
```

Zusätzlich wird das Paket *nodemon* installiert. Dieses ist nicht für die Produktion der Anwendung notwendig, sondern erleichtert die Entwicklung des Servers, indem ein automatischer Neustart der Anwendung bei Änderungen im Quellcode eingeleitet wird. Da das Paket nicht für die Produktion notwendig ist, wird durch die Option *save-dev* sichergestellt, dass die Abhängigkeit lediglich für Zwecke der Entwicklung benötigt wird.

```
$ npm install --save-dev nodemon
```

Abschließend soll das Projekt durch die Versionskontrollsoftware *Git* versioniert werden. Ebenso wie bei NPM sind alle Initialisierungsbefehle über die in *Git* integrierte Kommandozeile ausführbar. Zunächst wird innerhalb des Projektverzeichnisses ein neues git-Repository initialisiert.

```
$ git init
```

Darüber hinaus wird für das Repository die kostenfreie Remote-Lösung GitHub verknüpft. Dafür wird zunächst ein Repository auf GitHub angelegt, welches anschließend als Remote-Repository dienen soll. Das Repository liegt dabei unter dem persönlichen Nutzeraccount des Autors, weshalb die öffentliche Sichtbarkeit des Projektes nicht garantiert ist. Unter dem lokal initialisierten Git-Repository kann nun das Remote-Repository hinzugefügt werden. Anschließend wird der initiale Commit, der bereits lokal versioniert wurde, über den `git push` Befehl in das Remote Repository hochgeladen. Die Autorisierung für das Hochladen der Dateien erfolgt über einen SSH-Schlüssel. Dieser wurde auf der Maschine des Autors generiert und anschließend im GitHub-Profil hinterlegt.

```
$ git remote add origin git@github.com:schlosser/ba.git
$ git branch -M main
$ git push -u origin main
```

Mit der grundlegenden Einrichtung des Projektes ist es möglich, einen primitiven Apollo-Server einzurichten, der auf GraphQL-Anfragen reagieren kann. Für die initiale Konfiguration des Servers ist es notwendig, alle Typedefinitions und Resolvers anzugeben, mit denen der Service operieren kann. Für ein primitives Anfangsszenario wird dabei eine Typedefinition einer Query hinterlegt, welche einen Textinhalt abrufbar macht. Der zugehörige Resolver gibt lediglich einen einfachen Testinhalt “Hello World!“ zurück.

```
const { ApolloServer } = require('apollo-server');
const { gql } = require('apollo-server');

const typeDefs = gql`
    type Query {
        hello: String
    }
`;

const resolvers = {
    Query: {
        hello: () => {
            return 'Hello World!';
        }
    }
};

const server = new ApolloServer({typeDefs, resolvers});
return server.listen({port: 5000});
```

Der Server wird im Anschluss über das *nodemon*-Package gestartet. Durch die Angabe des spezifischen Ports ist der Service im Anschluss unter der Adresse *localhost:5000* erreichbar.

```
$ nodemon run
```

Apollo stellt eine grundlegende Weboberfläche bereit, mit der GraphQL-Anfragen an den zugrundeliegenden Server gestellt werden können. Dabei ist ein Überblick über alle vorhandenen Queries und Mutations des Servers gegeben, sodass Anfragen sowohl durch simples Selektieren der Bestandteile als auch manuell durch die GraphQL-Syntax konzeptioniert werden können. Wie in Abbildung 4.1 zu sehen ist, kann die zuvor erstellte Query abgerufen werden, um die zugehörige “Hello World!“-Nachricht abzurufen.

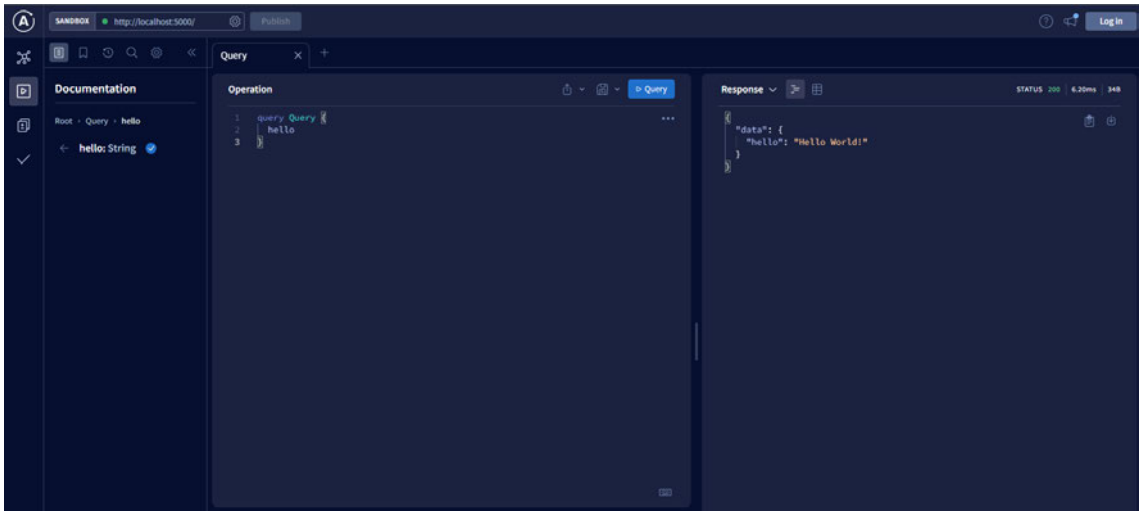

Abbildung 4.1.: Nutzung der Apollo GraphQL-Oberfläche zum Testen der erstellten Quellcodebasis

Im Folgenden soll das Datenbankmanagementsystem, welches für die Persistierung von Daten des Backends genutzt wird, eingerichtet werden. Die bereits eingebundene Bibliothek *Mongoose* muss dafür zunächst initialisiert werden. Die Herstellung zu einer auf MongoDB basierenden Datenbank geschieht dabei durch einen so genannten *Connection String*. Dieser Textinhalt, welcher vom Aufbau an einen im Internet gebräuchlichen URL-Link angelehnt ist, beinhaltet alle Informationen, die zum Verbindungsaufbau notwendig sind. Dazu zählen neben dem Host, auf welchem die Datenbank bereitsteht, auch der Nutzer, welcher an der Datenbank angemeldet wird, um entsprechende Operationen auszuführen. Zusätzlich können verschiedene optionale Schlüssel-Wert-Paare angegeben werden, welche die Verbindung genauer konfigurieren.

Die ursprüngliche Startdatei wird so erweitert, dass vor dem Starten des Servers die Verbindung zur Datenbank hergestellt wird, welche somit eine Voraussetzung für das System darstellt. Hierfür wird ein beispielhafter Connection-String angelegt, welcher sich mit einer auf dem lokalen Host laufenden MongoDB-Datenbank unter dem Port 27017 verbindet. Zur Authentifizierung wird dabei der Nutzername `admin` mit demselben Passwort verwendet. Im folgenden Listing sind die neuen und geänderten Komponenten des Quellcodes dargestellt.

```
1  const mongoose = require('mongoose');
2
3  const mongoConnectionString = "mongodb://admin:admin@localhost
       :27017";
4
5  mongoose.connect(mongoConnectionString, {useNewUrlParser: true})
6      .then(() => {
7          console.log('MongoDB Database connected.');
8          return server.listen({port: 5000});
9      })
10     .then((res) => {
11         console.log(`Server running at ${res.url}`)
12     });
```

Die zugehörige Testumgebung für die Datenbank wird auf dem lokalen System unter Docker bereitgestellt. Hierfür wird das offizielle MongoDB Image genutzt, welches frei auf Docker Hub zur Verfügung steht. Beim Starten des Images wird der Port 27017 verwendet. Es wird zudem ein Root-Nutzer angelegt, welcher unbegrenzte Zugriffsrechte auf die Datenbank besitzt. Dieser Nutzer erhält die unter dem folgenden Listing spezifizierten Anmeldedaten, sodass der Server uneingeschränkt auf die Datenbank zugreifen kann.

```
$ docker run --name test-database -p 27017:27017 -e
  MONGO_INITDB_ROOT_USERNAME=admin -e MONGO_INITDB_ROOT_PASSWORD=
  admin mongo
```

Nach einem Neustart des Servers, welcher durch die Änderung an Dateien durch *nodemon* automatisch geschieht, kann überprüft werden, ob die Verbindung der Anwendung zur Datenbank erfolgreich hergestellt wurde. Als Kriterium gelten dabei die eingebauten Konsolenmeldungen, die die Datenbankverbindung und das anschließende Starten des Servers melden.

```
B:\Development\Webservices\bachelorarbeit\server>nodemon run
[nodemon] 2.0.16
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node run index.js`
MongoDB Database connected.
Server running at http://localhost:5000/
```

Abbildung 4.2.: Meldung des Serverstarts unter Verbindung der MongoDB-Datenbank

Wie unter Abbildung 4.2 deutlich wird, sind sowohl die Datenbankverbindung als auch das Starten des Servers auf der lokalen Maschine erfolgreich durchführbar. Die Einrichtungsphase des Projektes kann somit als abgeschlossen angesehen werden. In den folgenden Schritten wird die eigentliche Funktionalität inklusive aller notwendigen Datenstrukturen in die Basis des Projektes integriert.

4.1.2. Basis von Models und GraphQL Schemas

Innerhalb des Anwendungskerns wird zwischen Modellen, Type Definitions und zugehörigen Resolvern unterschieden. Auch wenn diese bereits teilweise unter Kapitel 2.3.2 erläutert wurden, soll zunächst eine genauere, auf den Kontext zugeschnittene Begriffsabgrenzung geschehen.

Die in Kapitel 3.3.2 dargelegten Datenstrukturen müssen für eine persistente Datenspeicherung unter MongoDB in der Projektumgebung modelliert werden. Dafür werden unter Mongoose so genannte *Models* nachgestellt. Diese dienen als Basis für die später unter MongoDB generierten Documents.

Für die Anfrage- und Antwortlogik unter GraphQL sind *Type Definitions* und *Resolvers* notwendig. Type Definitions geben dabei die genauen Formate vor, in denen die Requests entgegengenommen und die Responses zurückgegeben werden. Es erfolgt somit in den meisten Fällen eine Abstraktion oder Übertragung der zuvor erstellten Models auf ihre spätere Abruf- und Modifizierbarkeit. Die Anwendungslogik, welche die Funktion der Type Definitions realisiert, wird in so genannten Resolvers

implementiert. Diese entsprechen dem Requestformat der zugrundeliegenden Type Definition und machen somit ein variables Umgehen mit eventuellen Eingabedaten möglich. Die Aufgaben der Resolvers können von simplen Datenabrufen bis hin zu der Manipulation von gespeicherten Dokumenten reichen.

Aus der Charakteristik von Models und Type Definitions wird deutlich, dass es eine Überschneidung der Datenmodellierung gibt. Während das zugehörige Model auszeichnet, wie die Daten innerhalb der Datenbank gespeichert werden, stellen die zugehörigen Type Definitions oft deckungsgleiche Repräsentationen der Model-Daten dar. Es kommt somit zu einer redundanten Repräsentation der gleichen Datengrundlage innerhalb des Quellcodes. Dieser Umstand, der einen Mehraufwand und eine unübersichtlichere Quellcodebasis nach sich zieht, soll durch ein weiteres NPM-Paket aufgehoben werden. Dieses Paket, welches unter dem Namen *GraphQL Compose Mongoose* besteht, ermöglicht das Einlesen von zuvor erstellten Mongoose Models und erzeugt aus diesen ein so genanntes *GraphQL Schema*. Das Schema ist als eine Kombination der benötigten Type Definitions und Resolvers zu verstehen. *GraphQL Compose* bietet dabei vordefinierte Resolverfunktionen an, welche gängige Operationen wie das Abrufen, Erstellen und Modifizieren von Daten realisieren. Auch erweiterte Responsemodifikationen, wie das Filtern und Sortieren von Daten oder eine Limitierung der Ergebnismenge, können einfach in das zugrundeliegende Schema eingepflegt werden. Alle zugrundeliegenden Type Definitions können zusätzlich manuell angepasst und erweitert werden, sodass das System keine direkten Restriktionen aufweist.

Wie aus der Modellierung hervorgeht, stehen die verschiedenen Models in teils gegenseitiger Beziehung zueinander. Eine simple Anwendung um ein Model in ein weiteres Model zu integrieren ist, den entsprechenden Datensatz separat in dem bestehenden Model abzulegen. Dabei ist zu beachten, dass der Datensatz autonom innerhalb des Documents gespeichert ist und nicht ohne weiteres in anderen Datensätzen wiederverwendet werden kann. Wird eine Wiederverwendbarkeit verfolgt oder ist ein genereller Bezug zu einem bereits bestehenden Datensatz zu ziehen, so wird das entsprechende Model referenziert. Die Referenz speichert dabei nicht den gesamten Datensatz selbst, sondern lediglich die eineindeutige ID, welche dem Document zuzuordnen ist.

Eine Referenz auf ein Dokument ergibt allerdings Nachteile bei der standardmäßigen Repräsentation, da durch die ID kein direkter Rückschluss auf die Daten des Datensatzes möglich werden. Stattdessen müsste über eine zweite Anfrage die ID genutzt werden, um den entsprechenden Datensatz zu suchen. Um dieses umständliche Vorgehen zu vereinfachen, werden innerhalb des Schemas so genannte *Projections* festgelegt. Diese übernehmen die Zuordnung eines Datenattributs, wie etwa der beschriebenen ID, zum zugehörigen Model. Entsprechend ist es durch lediglich eine Anfrage möglich, alle aus Referenzen projizierten Daten abzurufen. GrpaphQL macht es dabei, wie bei jeglichen Queries, möglich, die in der Response zu übertragenden Daten der Projektion zu selektieren.

4.1.3. Authentifizierung

Das Registrieren eines neuen Nutzeraccounts wird über eine separate GraphQL-Mutation gelöst. Durch die Möglichkeit der Übergabe variabler Daten innerhalb der Anfrage können Daten, wie ein Nutzername, eine zugehörige E-Mail-Adresse oder das Nutzerpasswort, angegeben werden. Der entsprechende Resolver sorgt für eine Eintragung der Daten in der Datenbank. Um das Passwort aus Sicherheitsgründen nicht als menschenlesbaren Text abzulegen, wird eine Hashingfunktion verwendet. Dafür wird das NPM-Paket *Bcrypt* verwendet, welches diverse Hashing-Algorithmen bereitstellt und durch diese eine reverse Rückverfolgung des Passworts verhindert.

Analog zur Registrierung soll es durch eine GraphQL-Anfrage möglich sein, einen bestehenden Nutzer einzuloggen. Dafür wird die Übergabe des eindeutigen Nutzernamens sowie des zugehörigen Passworts notwendig. Über *Bcrypt* wird überprüft, ob das eingegebene Passwort dem zuvor gehashten Wert in der Datenbank entspricht. Nach einem erfolgreichen Login wird für den betreffenden Nutzer ein JWT generiert. Dieses beinhaltet neben Metadaten zum eingeloggten Nutzer ein Token, welches als eineindeutiges Mittel zur Authentifizierung genutzt wird. Bei der Durchführung einer sensiblen Anfrage ist das entsprechende Token somit zwingend zu übertragen, damit der Server das JWT validieren und den Nutzer im Anschluss als authentifiziert erkennen kann.

Neben der Authentifizierung spielt auch die Autorisierung eines Nutzers oder einer Nutzergruppe eine entscheidende Rolle für die Anwendung. Trotz der Angabe eines validen Tokens muss überprüft werden, ob ein Nutzer die konkrete Berechtigung

besitzt, um eine bestimmte Antwort vom Server zu erhalten. Die unter dem Kontext einer Lernplattform entwickelten Autorisierungswege sind vielfältig. Wie unter Kapitel 3.3.1 durch die Architektur angedeutet ist, können Ressourcen verschiedener Art durch so genannte *Access Objects* abgesichert werden. Diese geben an, unter welchen Bedingungen ein Zugriff in einer bestimmten Form möglich ist. Es ist dabei möglich, spezifische Einzelnutzer, erstellte Nutzerrollen oder auch Gruppen der Plattform für einen Zugriff freizugeben. Auch der Zugang über ein Passwort oder einen zeitlich eingeschränkten Rahmen ist einstellbar. Es kann dabei differenziert werden, ob nur eine lesende oder zusätzlich ebenfalls eine schreibende Berechtigung erteilt wird. Die zentrale Aufgabe des Autorisierungssystems ist es somit, innerhalb einer sensiblen Anfrage zu überprüfen, ob der authentifizierte Nutzer mindestens eine Anforderung aller definierten *Access Objects* erfüllt, um eine lesende oder schreibende Aufgabe auszuführen. Ist dies nicht gegeben, wird die Anfrage als unautorisiert an den Client zurückgewiesen.

4.1.4. Inhalte der Anwendung

Für die prototypische Anwendung werden neben den erläuterten Aspekten verschiedene Wege unterstützt, Inhalte für Lernende einzubinden. Die im Folgenden beschriebenen Implementierungen beziehen sich auf den Prototyp der Anwendung. Vorhandene Strukturen können für andere Zwecke erweitert oder verändert werden.

Für die Bereitstellung von Inhalten, welche zu einem Kurs gehören, enthält der Service vielgestaltige Instrumente. Für eine simple Darstellung von Inhalten wird die Auszeichnungssprache *Hypertext Markup Language* verwendet. Kursinhalte werden dabei in klassischen Tags angelegt. Auf der Seite des Clients ist somit eine Stilisierung der Elemente nach Belieben möglich, da der Service lediglich die zugrundeliegende Struktur der Inhalte abspeichert. Auch können Kurse Ressourcen anbieten. Dabei verweist ein Kurseintrag auf eine Datei, welche auf dem Server abgelegt ist. Kurseinträge können untereinander rekombiniert und inhaltlich verknüpft werden. Zur Unterstützung der Kursorganisation ist es dabei möglich, mehrere zusammengehörige Kurselemente in einer Kategorie zu organisieren und mit einem Titel sowie einer Beschreibung zu versehen.

Eine besondere Inhaltsstruktur sind Kursaufgaben. Für einen Aufgabentypen können mehrere Fragen hinterlegt werden. Die Anwendung unterstützt dabei erweiterbare Aufgabentypen und beinhaltet für den Prototyp eine beispielhafte Implementierung für Multiple Choice Aufgaben. Es ist zudem möglich, eine Interaktion eines Nutzers mit einer Aufgabe zu simulieren. Antworten, welche von Nutzern abgegeben werden, werden separat im System gespeichert und können somit zu Zwecken der Analyse abgerufen werden. Fragen selbst unterstützen das Verweisen auf Kursinhalte, welche die Hinleitung zur richtigen Antwort der Frage enthalten.

4.2. Schnittstellen zur Nutzung und Weiterentwicklung

Neben der reinen Logik, welche für die Realisierung von Lehr-Lern-Szenarien genutzt wird, spielen auch Schnittstellen zur externen Nutzung sowie Ansätze der Weiterentwicklung eine besondere Rolle. Im Folgenden werden diverse Aspekte, welche für die Erweiterbarkeit der Implementierung integriert wurden, genauer herausgestellt.

4.2.1. Grundlegende Konfiguration des Servers

Für die Nutzung des Servers müssen bestimmte Daten konfiguriert werden. Es wird dabei zwischen notwendigen und optionalen Daten unterschieden. Zu den notwendigen Daten zählt insbesondere der *Connection String*, mit welchem sich der Server mit der Datenbank verbinden kann. Optionale Daten hingegen umfassen Optionen, die die Ausführung des Servers auf eine bestimmte Art und Weise verändern, jedoch nicht zwingend für die Inbetriebnahme des Systems konfiguriert werden müssen. Alle für die Einrichtung benutzten Daten sollen nicht statisch im Quelltext kodiert werden, sondern müssen dynamisch anpassbar sein.

Für die Umgebungsvariablen werden im Projekt Standards hinterlegt, welche beim Starten manuell überschrieben werden können. Dafür wird das NPM-Paket *Dotenv* verwendet. Es wird somit sichergestellt, dass keine leeren Werte verwendet werden, falls bestimmte Parameter nicht gesetzt werden. Dies verhindert insbesondere im Rahmen der optionalen Daten eine Fehlfunktion des Servers, falls beim Starten des Systems keine Überschreibung stattfindet.

Von besonderem Vorteil ist die Verwendung von Umgebungsvariablen auch für das spätere Deployment über Docker. Von Docker gestartete Container erlaubt eine einfache Übergabe von Systemvariablen, sodass die Einrichtung des Servers fließend in das Deployment integriert werden kann.

4.2.2. Schnittstelle für xAPI

Die Realisierung des E-Learning-Standards xAPI spielt für die Nutzung der Servicedaten durch andere Plattformen eine zentrale Rolle. Für die vorliegende Servicestruktur soll der xAPI-Standard exemplarisch am Beispiel der bereits verfügbaren Übungsaufgaben implementiert werden. Als Basis für die Implementierung des Standards wird die offizielle xAPI-Spezifikation genutzt. [xAP22] Es ist dabei genau festgelegt, durch welche technischen Formate eine Repräsentation der Plattformdaten erfolgen muss. Aus diesen Anforderungen ergibt sich ein genaues Bild der für die Servicestruktur notwendigen Implementierungen.

Das Abrufen der xAPI-Repräsentation einer Antworthistorie wird über das Setzen eines Schalters innerhalb der Query angegeben. Ist der Schalter nicht aktiv, so erfolgt die Übertragung der Historie als normale JSON-Repräsentation. Bei einer Aktivierung erfolgt die Umleitung der Request auf einen separaten Controller. Nachdem dieser die Antwortdaten manuell aus der Datenbank abgerufen hat, erfolgt eine Übertragung der Daten in eine xAPI-Repräsentation. Dafür werden die drei zwingend notwendigen Attribute *actor*, *verb* und *object* jeweils manuell mit den Zieldaten gesetzt. Für die Definition des Objects wird zudem die Richtigkeit der Antwort hinterlegt.

Die für das Verb hinterlegte Aktion des Nutzers erfordert xAPI eine separate Beschreibung, welche über eine URL abzulegen ist. Um die Vollständigkeit innerhalb des Service zu garantieren, wird diese URL als neuer Endpoint im Service abgelegt. Für die beispielhafte Implementierung der Antworthistorie eines Quiz wird dabei das Verb *QuizAnswer* unter dem Endpoint */xapi/verbs/quizanswer* genauer beschrieben. Die Beschreibung ist dabei rein textuell und ermöglicht für xAPI die Einordnung des Verbs in den Kontext der Anwendung.

Die umgesetzte xAPI-Schnittstelle im prototypischen Service ist exemplarisch für die Möglichkeit, auch andere Teilaspekte des Service für die Spezifikation zu nutzen. So könnten auch andere Datengrundlagen und Aktivitäten von Nutzern gesammelt und durch den xAPI-Standard repräsentiert werden.

4.3. Deployment

Aus der Implementierung entwickelt sich das Bedürfnis nach der Einbettung der Anwendung in einem Rahmen, der auf lange Sicht für die Produktion genutzt werden kann. Trotz des prototypischen Charakters des Servers ist das Deployment inklusive einer Verteilbarkeit eine wichtige Anforderung. Teilschritte, die zum gezielten Deployment unternommen wurden, sind im folgenden Kapitel näher beschrieben.

4.3.1. Dockerizing der Anwendung

Als gewählte Form zur einfachen Verteilbarkeit der Anwendung wurde die Technologie Docker gewählt. Um die Anwendung schlussendlich als Container innerhalb von Docker starten zu können, ist zunächst eine Einrichtung der Anwendung für Docker - das so genannte *Dockerizing* - notwendig. Dafür wird zunächst ein *Dockerfile* angelegt. Dieser beinhaltet alle Informationen, damit Docker aus der bestehenden Anwendung ein Docker Image erzeugen kann. Nachfolgend ist die Dockerfile-Datei gezeigt, welche für den bestehenden Prototyp eingesetzt wird.

```
1  FROM node:16
2
3  WORKDIR /usr/src/app
4
5  COPY ./server/package*.json ./
6
7  RUN npm install
8
9  COPY ./server ./
10
11 EXPOSE 8000
12 CMD [ "node", "index.js" ]
```

Innerhalb des Dockerfiles wird angegeben, dass das offizielle Node-Image als Basis für das Image des Servers benötigt wird. Um eine Installation der für die Anwendung notwendigen Pakete zu ermöglichen, wird die für NPM benötigte Datei *package.json* ebenfalls mit in das Image kopiert. Nach einer Installation der notwendigen Pakete wird die Anwendungslogik ebenfalls in das Image kopiert. Über den Befehl `node index.js`, welcher innerhalb des Images gestartet wird, erfolgt der normale Start der Anwendung, wie er auch in der Testumgebung stattfand.

Mit der Einrichtung des *Dockerfiles* kann das Image in der Testumgebung gebaut werden. Dafür wird der `docker build` Befehl verwendet. Zusätzlich wird für das Image ein *Tag* angegeben, welcher eine spätere Identifikation erleichtert.

```
$ docker build -t schlosser/bachelorarbeit:0.1 .
```

Das neu erzeugte Image kann im Anschluss über das `docker run` Kommando gestartet werden:

```
$ docker run schlosser/bachelorarbeit:0.1
```

Ein anschließendes Ausführen des Kommandos `docker ps`, das die Liste aller gestarteten Container zeigt, ist zu sehen, dass der Container erfolgreich gestartet wurde. Die Anwendung kann in dieser Form jedoch nicht benutzt werden, da die zugrundeliegende Datenbank MongoDB nicht verfügbar ist. Aus diesem Grund soll über Docker-Compose eine Konfiguration angelegt werden, welche ein vollumfängliches Deployment inklusive der Datenbank ermöglicht. Die Docker-Compose-Datei umfasst dafür die Definition des Bauprozesses für das lokale Service-Image und die Referenz auf das offizielle MongoDB-Image, welches über Docker Hub bereitsteht. Die Anwendung kann somit über Docker-Compose lokal über den `docker-compose up` gebaut und bereitgestellt werden.

```
$ docker-compose up --build -d
```

Es ist zu bemerken, dass das volle Deployment inklusive der Datenbank primär für Testzwecke zu benutzen ist. Für das automatisierte Deployment des Images ist lediglich die Generierung des Anwendungs-Images über den zuerst angelegten *Dockerfile* relevant.

4.3.2. Einrichtung der CI/CD-Pipeline

Für die Einrichtung der CI/CD-Lösung GitHub Actions wird zunächst das Webinterface des Remote-Repositories aufgerufen. Dort wird der Tab *Actions* gewählt. Der erste einzurichtende Workflow soll den CI-Prozess zum Bauen des Docker Images aus dem aktuellen Projektstand realisieren. Hierfür wird, ausgehend vom GitHub Actions Marketplace, ein Workflow-Template so modifiziert, dass der Workflow den Tag des Images automatisch anhand des aktuellen Datums generiert. Neben der Möglichkeit, den Workflow manuell zu starten, wird in der *on*-Sektion zudem angegeben, dass ein Push auf den Hauptbranch *main* den CI-Prozess automatisch startet.

```
1  name: Docker Image CI
2
3  on:
4    push:
5      branches: [ "main" ]
6
7  jobs:
8    build:
9      runs-on: ubuntu-latest
10     steps:
11     - uses: actions/checkout@v3
12     - name: Build the Docker Image
13       run: docker build . --file ./server/Dockerfile --tag
     bachelorarbeit:$(date +%s)
```

Abschließend wird das generierte Image innerhalb eines eigenen Prozesses veröffentlicht. Dafür erfolgt die Erweiterung des Workflows um zwei weitere Schritte. Im ersten Schritt erfolgt der Login des Docker Hub Nutzeraccounts, welcher Zugriff auf das private Image Repository besitzt. Hierfür werden aus den unter GitHub hinterlegten Secrets die Anmeldedaten geladen. Im zweiten Schritt wird das gebaute Image über *docker push* im Image Repository hochgeladen.

```
    - name: Docker Login
      env:
        DOCKER_USER: ${{secrets.DOCKER_USER}}
        DOCKER_PASSWORD: ${{secrets.DOCKER_PASSWORD}}
      run: |
        docker login -u $DOCKER_USER -p $DOCKER_PASSWORD

    - name: Push the Docker Image
      run: docker push schlosser/bachelorarbeit
```

4.3.3. Verteilung der Anwendung

Für die Verteilung der Anwendung wird die in Docker integrierte Orchestrierung Docker Swarm verwendet. Für ein exemplarisches Einrichten der verteilten Anwendung werden virtuelle Maschinen unter dem Betriebssystem Ubuntu 22.04 mit einer aktiven Docker-Installation genutzt. Die Einrichtung der VM erfolgt unter dem Programm *Oracle VirtualBox* und setzt einen Bridget Network Adapter ein, um für die Verteilung ein lokales Netz zu simulieren.

Die Orchestrierung der Anwendung erfolgt ausgehend von der ersten VM, welche als Manager Node und damit initialer Verwalter des Swarms gilt. Unter dem lokalen Netzwerk wird der Swarm initialisiert. Mit der Ausführung des folgenden Befehls wird gleichzeitig ein Token generiert, welches von anderen Maschinen genutzt werden kann, um dem erstellten Swarm beizutreten.

```
$ docker swarm init --advertise-addr 192.168.100.2
```

Für die Bereitstellung des Services wird innerhalb des Manager-Hosts ein Service angelegt. Dieser gilt als Registry für den Swarm. Anschließend wird die Anwendung innerhalb des Swarms über einen Stack deployt. Dafür wird die unter 4.3.1 angelegte Docker-Compose-Datei verwendet.

```
$ docker service create --name registry --publish published=5000,
  target=5000 registry:2
$ docker stack deploy --compose-file docker-compose.yml
  bachelorarbeit
```

Zuletzt erfolgt ein Beitritt der zweiten VM in den aktiven Swarm über den folgenden Befehl:

```
$ docker swarm join --token <TOKEN> 192.168.100.2
```

Für den Platzhalter *<TOKEN>* wird das Join-Token eingesetzt, welches mit der Initialisierung des Swarms automatisch generiert wurde. Unter der Ausführung des folgenden Befehls von der Maschine der Manager Node können alle Maschinen, die dem Swarm als Node beigetreten sind, aufgelistet werden:

```
$ docker node ls
```

Zusätzlich kann über den folgenden Befehl eingesehen werden, welche Replizierungen der genutzten Images im vorhandenen Service angelegt wurden:

```
$ docker service ls
```

5. Evaluation

In diesem Kapitel wird die Durchführung der Evaluation beschrieben, welche einer Validierung der gestellten Anforderungen dienen soll. Als Grundlage für die Durchführung der Evaluation gelten dabei die Testfälle, welche im Kapitel 3.4.1 erstellt wurden. Die Testfälle beziehen sich dabei jeweils auf eine der in Kapitel 3.2 dargestellten Anforderungen. Schlussendlich werden die für die wissenschaftliche Arbeit erzielten Ergebnisse zusammengefasst.

5.1. Evaluation der Nutzung auf der Clientseite

FA-10-TF: Der Testfall FA-10-TF gilt der Validierung der Anforderung FA-10. Ziel ist die Prüfung der Verfügbarkeit der Authentifizierung innerhalb der Servicestruktur. Für die Vorbereitung des Testfalls wird zunächst eine vorliegende Version des Servers genutzt, um eine neue Kursressource anzulegen. Der Zugriff auf den Kurs wird so eingeschränkt, dass nur authentifizierte Nutzer lesenden Zugriff auf die Inhalte erhalten.

Zur Durchführung des Testfalls wird zunächst über den Apollo Testclient die Anfrage auf den Kurs ohne Login simuliert. Dafür wird kein Authorization-Header für die Durchführung der Anfrage übergeben. Der Server antwortet mit einem Fehlerobjekt, welches entspreche Fehlermeldungen als Inhalt enthält. Im Anschluss wird die identische Request mit der Übergabe des Authorization-Headers eines validen Nutzeraccounts durchgeführt. Der Server antwortet in mit den innerhalb der Query angeforderten Ressourcen. Das Ergebnis des Tests ist erfolgreich und zeigt, dass eine generelle Authentifizierung für bestimmte Ressourcen aktivierbar ist und Inhalte nur dann abrufbar macht, wenn eine Authentifizierung durch einen Nutzer gegeben ist.

FA-11-TF: Der Testfall FA-11-TF gilt der Validierung der Anforderung FA-11. Ziel ist die Prüfung der Zugänglichkeit verschiedener Ressourcen durch eine Autorisierung durch Nutzergruppen. Voraussetzung ist die in FA-10-TF validierte generelle

Authentifizierung innerhalb des Service. Für die Vorbereitung des Testfalls wird zunächst eine vorliegende Version des Servers genutzt, um eine neue Kursressource anzulegen. Der Zugriff auf den Kurs wird so eingeschränkt, dass nur authentifizierte Nutzer, welche eine bestimmte Rolle besitzen, lesenden Zugriff auf die Inhalte erhalten.

Zur Durchführung des Testfalls wird zunächst über den Apollo Testclient die Anfrage auf den Kurs ohne Login simuliert. Dafür wird der Authorization-Header eines Nutzers, welcher die entsprechende Rolle nicht besitzt, an die Anfrage übergeben. Auch wenn die Authentifizierung des Nutzers geben ist, so schlägt die Autorisierung aufgrund der nicht verfügbaren Rolle fehl, indem der Server mit einem Fehlerobjekt antwortet, welches entsprechende Fehlermeldungen enthält. Anschließend wird die identische Anfrage mit der Übergabe des Authorization-Headers eines Nutzers, welcher die spezifische Rolle besitzt, ausgeführt. Der Server antwortet mit den angeforderten Ressourcen innerhalb der Response. Das Ergebnis des Tests ist erfolgreich und zeigt, dass innerhalb des Servers neben der generellen Authentifizierung auch eine Autorisierung, die auf spezifische Nutzermerkmale zugeschnitten ist, möglich ist.

FA-12-TF: Der Testfall FA-12-TF gilt der Validierung der Anforderung FA-12. Ziel ist die Prüfung der Verwaltbarkeit angelegter Nutzergruppen sowie deren zugeteilte Berechtigungen. Voraussetzung ist die in FA-10-TF validierte generelle Authentifizierung innerhalb des Service sowie die in FA-11-TF validierte Autorisierung durch Nutzergruppen. Für die Vorbereitung des Testfalls wird zunächst eine vorliegende Version des Servers genutzt, um eine neue Kursressource anzulegen. Der Zugriff auf den Kurs wird so eingeschränkt, dass nur autorisierte Nutzer einer bestimmten Nutzergruppe sowie spezifische Nutzer lesenden Zugriff auf die Ressource erhalten.

Für den Test wird zunächst über den Apollo Testclient die Anfrage auf den Kurs mit dem Login eines Nutzers, der weder die für die Autorisierung benötigte Nutzergruppe noch eine individuelle Berechtigung besitzt, simuliert. Der Server antwortet dabei nicht mit den geforderten Inhalten, sondern mit einem Fehlerobjekt, welches entsprechende Fehlermeldungen als Inhalt der Response enthält. Anschließend wird der Nutzergruppe des Nutzers eine lesende Berechtigung auf den Kurs erteilt. Bei der erneuten Durchführung der Request antwortet der Server mit den angeforderten Ressourcen innerhalb der Response. Für einen weiteren Test wird der Gruppe die lesende Berechtigung entzogen. Anschließend wird dem Nutzer selbst die lesende

Berechtigung erteilt. Bei einer erneuten Durchführung der Request antwortet der Server mit den angeforderten Ressourcen innerhalb der Response. Das Ergebnis des Tests ist erfolgreich und zeigt, dass innerhalb des Servers neben der generellen Autorisierung eine differenzierte Zuweisung für Nutzer und Nutzergruppen möglich ist. Die Berechtigung ist dabei für Ressourcen verwaltbar und ermöglicht eine dynamische Anpassung der Berechtigungen.

FA-20-TF: Der Testfall FA-20-TF gilt der Validierung der Anforderung FA-20. Ziel ist die Prüfung der selektiven Datenanfrage beim Server. Für die Vorbereitung des Testfalls wird zunächst eine vorliegende Version des Servers gestartet. Es wird ein Kurs sowie eine Ressource angelegt. Damit der Zugriff auf die Ressourcen gesichert ist, wird keine Authentifizierung vorausgesetzt.

Zur Durchführung wird eine Query definiert, welche zunächst alle vorhandenen Kurse mit ihrem Namen und ihrer Beschreibung abruft. Nach der Ausführung der Query wird der angelegte Kurs mit seinem Namen und einer eingetragenen Beschreibung in der Response übermittelt. Auch wenn für den Kurs eine Ressource hinterlegt wurde, ist diese nicht in der Response enthalten. Die Query wird anschließend so erweitert, dass für jeden ausgegebenen Kurs ebenfalls alle Ressourcen angefordert werden. Nach der Ausführung der Query wird neben dem angelegten Kurs auch seine hinzugefügte Ressource in der Response übermittelt. Das Ergebnis des Tests ist erfolgreich und zeigt, dass über GraphQL abgesetzte Queries so einschränkbar sind, dass eine selektive Datenanfrage möglich wird.

FA-30-TF: Der Testfall FA-30-TF gilt der Validierung der Anforderung FA-30. Ziel ist die Prüfung der Wiederverwendbarkeit von Kurs-Komponenten innerhalb des Service. Für die Vorbereitung des Testfalls wird zunächst eine vorliegende Version des Servers gestartet. Es werden zwei Kurse sowie eine Ressource angelegt. Für einen lesenden und schreibenden Zugriff wird keine Authentifizierung vorausgesetzt.

Zur Durchführung des Testfalls wird beiden angelegten Kursen die neue Ressource hinzugefügt. Anschließend wird eine Query ausgeführt, um die Ressourcen beider Kurse abzurufen. Dabei wird für beide Kurse die identische Ressource ausgegeben, die zuvor hinterlegt wurde. Im Folgenden wird die hinterlegte Ressource modifiziert, indem exemplarisch der Name verändert wird. Wird nun erneut die bereits zuvor genutzte Query ausgeführt, so wird erneut die gleiche Ressource für beide Kurse angezeigt. Im Gegensatz zur ersten Query ist jedoch zu beobachten, dass der Name der Ressource dem geänderten Namen entspricht. Das Ergebnis des Tests ist erfolgreich

und zeigt, dass Ressourcen und andere Komponenten für mehrere Kurse verwendbar sind. Durchgeführte Änderungen werden dabei auf alle zugrundeliegenden Kurse übertragen, indem die Referenz auf eine Komponente gespeichert wird, statt die Komponente selbst.

FA-31-TF: Der Testfall FA-31-TF gilt der Validierung der Anforderung FA-31. Ziel ist die Prüfung der Verfügbarkeit von Grundkonzepten verschiedener E-Learning-Systeme anhand der Möglichkeit, zu einem Kurs zugehörige Übungen anzulegen. Für die Vorbereitung des Testfalls wird zunächst eine vorliegende Version des Servers gestartet. Es wird zudem ein Kurs angelegt. Für einen lesenden und schreibenden Zugriff auf den Kurs wird lediglich eine Authentifizierung, aber keine spezifische Autorisierung vorausgesetzt. Zum Durchführen der Übung wird ein exemplarischer Nutzer angelegt.

Für den Kurs wird eine Übungsressource hinterlegt. Zu dieser wird anschließend eine neue Multiple-Choice-Frage mit einer richtigen Antwort erstellt. Das Abrufen der Übungen des Kurses über eine Query übermittelt die zuvor angefertige Übungsressource mit der Übungsfrage. Über eine Mutation wird ausgehend vom angelegten Nutzer zunächst eine falsche Antwort an die Frage übermittelt. Die Rückgabe des *Results* in der Response gibt an, dass die Antwort falsch war. Über eine weitere Mutation wird anschließend die richtige Antwort übermittelt. Die Rückgabe des *Results* in der Response gibt an, dass die Antwort richtig war. Ein Abrufen des Antwortregisters der Frage für den verwendeten Nutzer enthält die Eintragungen der beiden gegebenen Antworten. Als letzte gegebene Antwort ist die richtige Antwort vermerkt. Das Ergebnis des Tests ist erfolgreich und zeigt, dass für einen Kurs Übungsaufgaben verschiedener Genres erstellt werden können. Gegebene Antworten werden dabei im System verzeichnet und ermöglichen eine Überwachung des Fortschritts der Nutzer.

FA-32-TF: Der Testfall FA-32-TF gilt der Validierung der Anforderung FA-32. Ziel ist die Prüfung der Modularisierung von Teilen des Systems. Für die Vorbereitung des Testfalls wird zunächst eine vorliegende Version des Servers gestartet. Es wird zudem ein Kurs angelegt, zu welchem ein neu erstellter Kursinhalt in Form einer Ressource hinzugefügt wird. Unabhängig dazu wird ein zweiter Kursinhalt in Form einer HTML-Ansicht gespeichert, welcher dem Kurs noch nicht zugeordnet wird. Für einen lesenden und schreibenden Zugriff wird keine Authentifizierung vorausgesetzt.

Zur Durchführung des Tests wird zunächst der Kurs inklusive seiner Inhalte abgerufen. In der Response wird dabei der Kurs mitsamt der hinterlegten Ressource angezeigt. Anschließend soll ein Austausch der Ressource durch die HTML-Ansicht vorgenommen werden. Dazu wird eine Mutation verwendet, welche die Referenz der Lernressource durch die zweite neu angelegte HTML-Ansicht vornimmt. Anschließend wird die zuvor genutzte Query erneut ausgeführt. Die Response übermittelt neben dem Kurs statt der zuvor betrachteten Ressource nun die HTML-Ansicht. Das Ergebnis des Tests ist erfolgreich und zeigt, dass Komponenten eines Kurses modularisiert und nach Belieben mit anderen Kursinhalten ausgetauscht werden können. Es ist dabei nicht notwendig, weitere Änderungen am Kurs vorzunehmen, sodass die geänderten Komponenten sofort innerhalb des Kurses übernommen werden.

FA-40-TF: Der Testfall FA-40-TF gilt der Validierung der Anforderung FA-40. Ziel ist die Prüfung der Verwaltbarkeit angelegter Kursstrukturen. Für die Vorbereitung des Testfalls wird zunächst eine vorliegende Version des Servers gestartet. Es wird ein Kurs sowie drei zugehörigen Ressourcen erstellt, welche dem Kurs zugeordnet werden. Für einen lesenden und schreibenden Zugriff wird keine Authentifizierung vorausgesetzt.

Für die Durchführung des Tests wird zunächst eine Query zum Abrufen des Kurses und der zugehörigen Komponenten durchgeführt. In der Response werden neben den Metadaten des Kurses die hinzugefügten Ressourcen in ihrer chronologischen Reihenfolge des Hinzufügens angezeigt. Nachfolgend wird eine der Ressourcen innerhalb des Kurses entfernt. Ein erneutes Ausführen der identischen Query zeigt, dass der Kurs nur noch die zwei anderen Komponenten enthält. Die chronologische Reihenfolge wurde dabei beibehalten. Abschließend wird eine Mutation ausgeführt, welche die Reihenfolge der angelegten Ressourcen verändert, indem die zwei Beispiel-Komponenten vertauscht werden. Ein erneutes Ausführen der zuvor genutzten Query

zeigt, dass beide Komponenten innerhalb des Kurses die neue Reihenfolge angenommen haben. Das Ergebnis des Tests ist erfolgreich und zeigt, dass Kurse nach dem Anlegen im Service neu strukturiert und hinsichtlich ihrer Bestandteile modifizierbar sind.

FA-41-TF: Der Testfall FA-41-TF gilt der Validierung der Anforderung FA-41. Ziel ist die Prüfung der Kategorisierung von Kursen und zugehörigen Kursinhalten nach ihrem Kontext und Einsatz. Für die Vorbereitung des Testfalls wird zunächst eine vorliegende Version des Servers gestartet. Es wird ein Kurs sowie zugehörige Ressourcen, die dem Kurs untergeordnet sind, angelegt. Für einen lesenden und schreibenden Zugriff wird keine Authentifizierung vorausgesetzt.

Kurse und Ressourcen erlauben das Anlegen von Metadaten. Neben einem identifizierenden Namen sowie einer Beschreibung ist es möglich, für Kurse und Kursinhalte Tags zu hinterlegen, welche eine nähere Einordnung durch Stichworte oder Stichwortgruppen ermöglichen. Zur Testdurchführung wird zunächst eine Query zum Abrufen des Kurses und der zugeordneten Inhalte inklusive aller Tags abgesetzt. Die Response zeigt, dass weder dem Kurs noch den Inhalten Tags zugewiesen sind. Anschließend werden Mutations, welche dem Kurs und den Inhalten beispielhafte Tags zuordnen, im Testclient ausgeführt. Ein erneutes Ausführen der zuvor genutzten Query zeigt, dass die Tags sowohl beim Kurs als auch bei den zugehörigen Ressourcen hinterlegt wurden. Das Ergebnis des Tests ist erfolgreich und zeigt, dass der Service eine Kategorisierung von Kursen und Kursinhalten ermöglicht.

FA-42-TF: Der Testfall FA-42-TF gilt der Validierung der Anforderung FA-42. Ziel ist die Prüfung der Realisierung von Suchanfragen. Voraussetzung ist die in FA-41-TF validierte Kategorisierung von Kursen und Kursinhalten. Für die Vorbereitung des Testfalls wird zunächst eine vorliegende Version des Servers gestartet. Es wird ein Kurs sowie zugehörige Ressourcen, die dem Kurs untergeordnet sind, erstellt. Für einen lesenden und schreibenden Zugriff wird keine Authentifizierung vorausgesetzt. Dem Kurs und den Inhalten werden, neben dem Namen und einer Beschreibung, Tags mit beschreibenden Stichworten zugeordnet.

Zur Durchführung des Testfalls wird zunächst eine Suchquery für Kurse durchgeführt. Als suchendes Attribut wird zunächst der Name verwendet. Bei der Angabe keines Namens wird in der Response kein Kurs übermittelt. Bei der Angabe von Teilen des Kursnamens wird der Kurs übermittelt. Bei der Angabe von Teilen des

Namens einer unterliegenden Ressource wird ebenfalls der Kurs übermittelt. Nachfolgend wird die Beschreibung als suchendes Attribut verwendet. Bei der Angabe keiner Beschreibung wird in der Response kein Kurs übermittelt. Bei Angabe von Teilen der Kursbeschreibung wird der Kurs übermittelt. Bei Angabe von Teilen der Beschreibung einer untergeordneten Kursressource wird der Kurs übermittelt. Anschließend werden als suchende Attribute Tags verwendet. Bei der Angabe keiner Tags wird in der Response kein Kurs übermittelt. Bei der Angabe von Teilen eines Tags, welcher im Kurs enthalten ist, wird der Kurs übermittelt. Bei der Angabe von Teilen eines Tags, welcher in den Kursinhalten enthalten ist, wird der Kurs übermittelt. Zuletzt wird die kombinierte Suchquery verwendet. Bei der Übergabe keines Suchtextes wird in der Response kein Kurs übermittelt. Wird als Suchtext der Teil des Kursnamens, der Kursbeschreibung oder eines Kurstags übergeben, wird der Kurs übermittelt. Wird als Suchtext der Teil eines Ressourcennamens, einer Ressourcenbeschreibung oder eines Ressourcentags übergeben, wird der Kurs übermittelt. Das Ergebnis des Tests ist erfolgreich und zeigt, dass innerhalb von Kursstrukturen Suchanfragen realisierbar sind. Es kann dabei zwischen einer Suche nach Namen, Beschreibung, Tags sowie einem kombinierten Suchverfahren differenziert werden.

FA-50-TF: Der Testfall FA-50-TF gilt der Validierung der Anforderung FA-50. Ziel ist die Prüfung der Implementierung des xAPI-Standards innerhalb der Anwendung. Voraussetzung ist die in FA-31-TF validierte Möglichkeit zum Anlegen und Durchführen von Übungen innerhalb eines Kurses. Für die Vorbereitung des Testfalls wird zunächst eine vorliegende Version des Servers gestartet. Es wird ein Kurs erstellt, zu welchem eine neue Übung angelegt wird. Außerdem wird ein Testnutzer bereitgestellt. Für einen lesenden und schreibenden Zugriff auf den Kurs wird lediglich eine Authentifizierung, aber keine spezifische Autorisierung vorausgesetzt. Zum Durchführen der Übung wird zudem ein Nutzer angelegt.

Zur Durchführung wird vom Nutzer zunächst eine falsche sowie eine richtige Antwort innerhalb der angelegten Übung abgegeben. Ein Abrufen der Übungsergebnisse zeigt für den Nutzer beide Einträge innerhalb der Übung an. Anschließend wird die Query zum Abrufen der Übungsergebnisse im xAPI-Standard verwendet. Die in der Response übermittelten JSON-Objekte enthalten neben der eindeutigen ID des Lerneintrags die drei von xAPI als zwingend spezifizierten Attribute *actor*, *verb* und *object*. Der *AgentType* des *actors* ist als *Agent* definiert, was eine Zuordnung zu einem Individuum zeigt. Darüber hinaus ist in der Eigenschaft *openid* die in der Datenbank hinterlegte ID des Nutzers angegeben. Das definierte *verb* gibt die

jeweils ausgeführte Aktion des Actors an. Als *id* des Verbs ist ein separater Endpoint gegeben, der für eine genauere Beschreibung des durchgeführten Verbs genutzt wird (*http://localhost:5000/xapi/verbs/quizanswer*). Zudem ist als lesbare Repräsentation der Aktion unter dem englischen Sprachschlüssel die Richtigkeit der Antwort vermerkt (*Right* oder *Wrong*). Unter dem *object* ist ein der Übung zuzuordnendes JSON-Element aufgeführt. Das Ergebnis des Tests ist erfolgreich und zeigt, dass für im Service aufgeführte Übungsformate eine Übertragung in ein Datenformat, das die xAPI-Spezifikation erfüllt, möglich ist.

FA-60-TF: Der Testfall FA-60-TF gilt der Validierung der Anforderung FA-60. Ziel ist die Prüfung des Deployments der gesamten Serverstruktur. Für die Vorbereitung des Testfalls wird zunächst eine stabile Version des Servers durch einen Commit in der Versionsverwaltung erfasst. Anschließend wird der lokale Commit durch die Push-Operation in das auf GitHub bereitstehende Remote-Repository eingereicht. Der Commit wird dabei nicht unter dem Branch *main* vorgenommen, welcher der Hauptbranch des Repositories ist, sondern unter einem separat angelegten Testbranch.

Auf der GitHub-Webseite wird das Remote-Repository aufgerufen. Dort wird der *Actions* Tab gewählt, unter welchem die CI/CD-Logik für GitHub-Actions hinterlegt ist. Anschließend wird der für das Deployment verwendete Workflow angewählt. Der Workflow wird zunächst auf den angelegten Testbranch angepasst und anschließend über einen Klick gestartet. Es werden nun alle unterliegenden Actions ausgeführt. Wichtigster Punkt ist hierbei das Docker Image, welches auf Basis des versionierten Quellcodes gebaut wird. Das Image wird im Workflow als Artifact hinterlegt und in einer weiteren Action auf *Docker Hub* hochgeladen. Das Image ist danach unter der Plattform verfügbar und kann aus dem Internet bezogen werden. Das Ergebnis des Tests ist erfolgreich und zeigt, dass die Servicestruktur durch die CI/CD-Lösung GitHub Actions automatisiert gebaut und im Anschluss in der Image-Registry Docker Hub bereitgestellt werden kann.

FA-61-TF: Der Testfall FA-61-TF gilt der Validierung der Anforderung FA-61. Ziel ist die Prüfung der Verfügbarkeit der Servicestruktur unter den Betriebssystemen Windows, Linux und Mac. Für die Vorbereitung des Testfalls wird zunächst eine vorliegende Version des Servers genutzt, um ein Docker-Image zu erzeugen. Um eine einfache Testbarkeit zu ermöglichen, wird das generierte Image im Anschluss auf der Online-Plattform *Docker Hub* hochgeladen.

Für den Test auf Windows wird die Software *Docker Desktop* unter dem Betriebssystem Windows 11 Pro auf einer lokalen Testmaschine verwendet. Für den Test auf Mac wird die Software *Docker Desktop* unter dem Betriebssystem macOS Big Sur 11 auf einer lokalen Testmaschine verwendet. Für den Test auf Linux wird Docker über den Paketmanager *APT* unter dem Betriebssystem Ubuntu 22.04 auf einer virtuellen Testmaschine installiert. Der anschließende Test verläuft auf allen drei gegebenen Systemen gleich. Es wird eine *docker-compose.yaml*-Datei angelegt, welche für das Starten der zugrundeliegenden Datenbank sowie dem erstellten Server-Image zuständig ist. Durch einen Start über den Befehl `docker-compose up -d` wird für beide Images je ein Container gestartet. Anschließend ist der Server lokal unter dem Port 5000 verfügbar. Zum Testen des Verhaltens wird der Server mit einem vorinstallierten Webbrowser unter der Adresse *http://localhost:5000* aufgerufen. Mithilfe des Apollo-Interfaces wird eine Mutation zur Erstellung eines Users ausgeführt. Anschließend wird eine Query zum Abrufen aller verfügbaren User ausgeführt. Es ist zu beobachten, dass das Verhalten unter allen Betriebssystemen identisch ist und der angelegte Nutzer in der Liste aller Nutzer angezeigt wird. Das Ergebnis des Tests ist erfolgreich und zeigt, dass das Deployment über Docker keine Restriktionen unter einer Nutzung der Betriebssysteme Windows, Mac oder Linux aufweist.

NA-10-TF: Der Testfall NA-10-TF gilt der Validierung der Anforderung NA-10. Ziel ist die Prüfung der Performanz bei der Anfrage von Daten über den Service. Für die Vorbereitung des Testfalls wird zunächst eine vorliegende Version des Servers über das zugrundeliegende Docker Image gestartet.

Für die Durchführung des Testfalls werden mehrere Queries betrachtet. Um differenzierte Anfragen in den Testfall einfließen zu lassen, wird dabei die Anzahl zu verknüpfender Documents als Kriterium für komplexere Queries verwendet. Im Prototyp des Servers beträgt die Maximalanzahl zu verknüpfender Verbindungen, die in einer Query durchführbar sind, drei. Im Folgenden sind die Messwerte der durchgeführten Queries mit der jeweiligen Anzahl verbundener Documents verzeichnet.

Anzahl verknüpfter Documents	Responsedauer in ms
1	5,3
1	6,2
1	5,4
2	8,1
2	8,8
2	8,5
3	10,2
3	13,5
3	12,8

Tabelle 5.1.: Messwerte für die Perfomanz ausgewählter Queries

Aus den Messwerten für eine Verknüpfung ergibt sich ein Mittelwert von 5,6 ms. Aus den Messwerten für zwei Verknüpfungen ergibt sich ein Mittelwert von 8,5 ms. Aus den Messwerten für drei Verknüpfungen ergibt sich ein Mittelwert von 12,2 ms. Alle Messwerte überschreiten somit den gegebenen Grenzwert von 100 ms nicht. Das Ergebnis des Tests ist erfolgreich und zeigt, dass die Durchführung von Queries performant ist und Responses ebenso performant an den Client übermittelt werden.

5.2. Evaluation der Wartung und Entwicklung des Servers

FA-70-TF: Der Testfall FA-70-TF gilt der Validierung der Anforderung FA-70. Ziel ist die Prüfung der Erweiterbarkeit der persistenten Strukturen des Service. Für die Vorbereitung des Testfalls liegt eine Testversion des Servers vor, welche bereits Datenbank-Einträge für vorhandene Kurse beinhaltet. Dafür wird ein Kurs mit einem exemplarischen Inhalt angelegt.

Die Seite des Servers wird zunächst um eine neue Kursinhalts-Komponente erweitert, welche neue Merkmale aufweist. Der Kurs wird anschließend um die neu angelegte Komponente ergänzt. Durch den für die Anwendung verwendete MongoDB Connection String wird über einen separaten Datenbankclient eine Verbindung zur

Datenbank aufgebaut. Anschließend wird das Document für den Kurs betrachtet. Das Document speichert als Liste der Kursinhalte ein Array an IDs, welche eindeutig einem Kursinhalt zugeordnet werden können. Das Hinzufügen eines neuen Kursinhaltes berührt somit in keiner Weise das bestehende Datenbankschema und ermöglicht das reibungslose Hinzufügen neuer persistenter Inhaltsstrukturen. Für eine weitere Überprüfung wird einer der bestehenden Kursinhalte erweitert. Dafür wird ein neues Datenfeld in der persistenten Struktur eingefügt. Ein Aufrufen des für diesen Inhaltstypen generierten Documents zeigt, dass das neue Datenfeld nicht im Document verzeichnet ist. Wird im Kurs ein neuer Inhalt des gleichen Typs angelegt, so ist das im Service hinzugefügte Feld im Document sichtbar. Das Ergebnis des Tests ist erfolgreich und zeigt, dass eine Erweiterung der persistenten Strukturen ohne Beeinträchtigung der bestehenden Inhalte möglich ist. Es ist dabei zu beachten, dass Daten, welche nachträglich in den Service integriert wurden, nicht automatisch in bereits angelegten Komponenten verfügbar sind.

FA-80-TF: Der Testfall FA-80-TF gilt der Validierung der Anforderung FA-80. Ziel ist die Prüfung der Erweiterbarkeit der persistenten Strukturen des Service. Voraussetzung ist die in FA-60-TF validierte Möglichkeit zum Deployment der Gesamtstruktur. Für die Vorbereitung des Testfalls liegt eine stabile Version des Servers vor.

Statt der unter FA-60-TF genutzten Möglichkeit zum manuellen Auslösen des Deployments soll im Speziellen die Anpassung der Pipeline auf Änderungen in der Versionsverwaltung getestet werden. Dafür wird die stabile Version unter dem Branch *main*, der als Hauptbranch des Repositories agiert, versioniert. Die lokale Änderung wird anschließend in das GitHub Remote-Repository gepusht. Im Anschluss wird das Webinterface der GitHub Actions Funktion im GitHub Remote-Repository betrachtet. Es wird dabei sichtbar, dass der bereits existierende Workflow automatisch durch die Einreichung des neuen Commits auf dem main-Branch ausgelöst wurde. Das gebaute Docker-Image wird als Artifact für eine neue Version des auf Docker Hub bereitgestellten Images genutzt. Das Ergebnis des Tests ist erfolgreich und zeigt, dass Änderungen unter der Versionierung des Hauptbranches automatisch die CI/CD-Pipeline auslösen und zum Deployment der geänderten Serverstruktur führen.

NA-20-TF: Der Testfall NA-20-TF gilt der Validierung der Anforderung NA-20. Ziel ist die Prüfung eines ressourcensparenden Betriebs bei der Ausführung des Service. Für die Vorbereitung des Testfalls wird zunächst eine vorliegende Version des Servers über das zugrundeliegende Docker Image gestartet.

Für die Messung der Auslastung wird der Service in verschiedenen Situationen analysiert. Die im Folgenden aufgeführten Messungen wurden im Moment der jeweils durchgeführten Belastung gemacht. Bei der Durchführung mehrerer Operationen erfolgt die Ausführung parallel.

Nutzungssituation	**Arbeitsspeicherauslastung in MB RAM**
Leerlauf, kein Zugriff	11,8
Ausführung einer Query	11,9
Ausführung von 100 Queries	12,0
Ausführung einer Mutation	11,8
Ausführung von 100 Mutations	12,2

Tabelle 5.2.: Messwerte der Arbeitsspeicherauslasung für ausgewählte Szenarien

Die Arbeitsspeicherauslastung ist auch bei aufwendigeren Operationen nahezu gleichbleibend und besitzt einen Mittelwert von 11,9 MB RAM Auslastung. Der Grenzwert von 1 GB RAM wird somit nicht überschritten. Das Ergebnis des Tests ist erfolgreich und zeigt, dass der Betrieb des Service sparsam ist und auch Belastungsszenarien keine Überlastung nach sich ziehen.

NA-30-TF: Der Testfall NA-30-TF gilt der Validierung der Anforderung NA-30. Ziel ist die Prüfung der Dokumentation des Quellcodes. Für die Vorbereitung des Testfalls werden aus einer vorhandenen Version des Servers mehrere beliebige Quellcodeausschnitte selektiert. Es werden dabei Ausschnitte verschiedener Komponenten, wie der Modellierung von Daten, der Resolverlogik oder anderer Anwendungslogik berücksichtigt.

Zunächst wird ein Codeausschnitt aus dem Modeling betrachtet. Die Datenklasse besitzt eine generelle Beschreibung, in welcher die Bedeutung für das Gesamtsystem gegeben ist. Für die einzelnen Datenfelder ist zudem das *description*-Attribut hinterlegt. Dieses wird nicht nur im Apollo Testclient genutzt, um Informationen detailliert darzustellen, sondern ermöglicht ebenfalls eine genauere Einordnung im Quellcode.

Ein betrachteter Ausschnitt für die Domäne der Resolver zeigt, dass die einzelnen Zuordnungen neuer Resolverfelder ins Schema durch erläuternde Kommentare unterstützt werden. Es ist hiermit möglich, den Kontext der Resolver einzuordnen und mit dem zugrundeliegenden Model zu assoziieren. Für den Aspekt anderer Systemkomponenten wird die für die Authentifizierung entwickelte Middleware betrachtet. Die für die Middleware bereitgestellte Funktion besitzt vorangestellte Kommentare, welche den Ablauf und die Wirkung der Authentifizierungsfunktion erläutern. Das Ergebnis des Tests ist erfolgreich und zeigt, dass verschiedene Komponenten des Systems so dokumentiert sind, dass ein grundlegendes Verständnis über die Funktion erlangt werden kann. Außerdem wird eine Einordnung in den Gesamtkontext des Systems möglich.

NA-40-TF: Der Testfall NA-40-TF gilt der Validierung der Anforderung NA-40. Ziel ist die Prüfung der Skalierbarkeit der Servicestruktur durch Verteilbarkeit. Für die Vorbereitung des Testfalls wird eine stabile über Docker verteilte Version des Servers bereitgestellt. Zur Simulation der Verteilung über zwei Systeme wird das Programm *Oracle VirtualBox* verwendet.

Unter VirtualBox werden zwei VM mit dem Betriebssystem Ubuntu 22.04 eingerichtet. Um das Aufbauen eines lokalen Netzwerkes zwischen beiden Maschinen zu simulieren, wird der Bridget Network Adapter verwendet. Über die Kommandozeile wird über den Paketmanager *APT* unter beiden Systemen Docker bezogen. Für eine Verteilung des Systems wird die von Docker bereitgestellte Orchestrierungstechnologie *Docker Swarm* verwendet. Die erste VM gilt dabei als Manager Node, von welcher der Swarm erstellt wird. Mit der Erstellung des Swarms wird ein Token generiert, über welches die zweite VM dem Swarm im Anschluss als Worker Node beitritt. Der Swarm besteht zu diesem Zeitpunkt aus einer aktiven Manager Node sowie einer Worker Node. Ausgehend von der Manager Node wird ein Docker Stack gestartet, welcher die zugrundeliegende Docker Compose Datei verwendet. Über den Befehl *docker stack services* ist einsehbar, dass je ein Image der Datenbank und des Service als Container gestartet wurden und innerhalb des Stacks verteilt sind. Das Ergebnis des Tests ist erfolgreich und zeigt, dass Komponenten der Anwendung über mehrere Systeme verteilt werden können. Auch ist das Replizieren der Komponenten möglich, um eine Skalierbarkeit der Anwendung zu ermöglichen.

NA-41-TF: Der Testfall NA-41-TF gilt der Validierung der Anforderung NA-41. Ziel ist die Prüfung der zentralen Wartbarkeit des Systems. Voraussetzung ist die in NA-40-TF validierte Verteilbarkeit der Anwendung. Für die Vorbereitung des Testfalls liegt eine stabile Version des Servers vor, welche über mehrere Systeme verteilt wurde.

Um ein zentrales Stoppen des verteilten Service zu erzielen, wird das als Manager Node verwendete System angesteuert. Auf diesem wird zunächst über den *docker stack rm* Befehl der angelegte Anwendungsstack beendet. Anschließend wird das Verlassen der Manager Node über *docker swarm leave –force* erzwungen. Durch das Fehlen einer Manager Node ist die Orchestrierung nicht länger verfügbar. Ein Ansteuern des Service im lokalen Netzwerk schlägt fehl. Das Ergebnis des Tests ist erfolgreich und zeigt, dass das Stoppen des verteilten Systems ausgehend von einer zentralen Komponente eingeleitet werden kann.

NA-50-TF: Der Testfall NA-50-TF gilt der Validierung der Anforderung NA-50. Ziel ist die Prüfung der Verfügbarkeit von automatischen Tests des Systems. Für die Vorbereitung des Testfalls liegt eine stabile Version des Servers mitsamt den angelegten Testfällen vor.

Für die Durchführung des Testfalls wird innerhalb des Anwendungsverzeichnisses eine Kommandozeile geöffnet. Die angelegten Tests sind über ein separat angelegtes Script verfügbar, welches über den Befehl *npm run test* gestartet wird. Mit dem Starten der Tests ist zu beobachten, dass die angelegten Tests der Reihe nach ausgeführt werden. Am Ende der Testreihe ist zu sehen, dass alle angelegten Tests durchgeführt wurden und die überprüften Anforderungen erfüllen:

```
  29 passing (238ms)
```

Das Ergebnis des Tests ist erfolgreich und zeigt, dass für das System Unit Tests erstellt wurden, welche ausgewählte technische Anforderungen automatisch validieren.

5.3. Auswertung

Bei der Durchführung der Evaluation auf Client- und Serverseite wurden verschiedene Ergebnisse erzielt. Durch die Durchführung der zuvor erstellten Testfälle wird es möglich, die Anwendung hinsichtlich ihrer Anforderungen zu bewerten.

Auf der Seite der clientseitigen Anforderungen konnten alle Testfälle erfolgreich durchgeführt werden. Die Tests, welche Kriterien wie Authentifizierung, Datenanfrage, die Anwendbarkeit für E-Learning-Zwecke sowie Möglichkeiten zum Deployment und einen ressourcensparenden Betrieb überprüfen sollten, hatten dabei allesamt ein erfolgreiches Ergebnis. Die Wahl der Implementierung sowie der Produktionsumgebung sind somit für die E-Learning Domäne geeignet und gut für potenzielle Clients nutzbar.

Auch alle Testfälle, die die Wartbarkeit und Weiterentwickelbarkeit des Service überprüfen sollten, konnten erfolgreich durchgeführt werden. Zentrale Merkmale wie die Erweiterbarkeit von Datenstrukturen, verschiedene Teilkonzepte des Deployments und der Wartbarkeit sowie das Einbauen automatisierter Tests und einer ausführlichen Dokumentation wurden geprüft. Alle Testfälle waren dabei erfolgreich, sodass belegt ist, dass der Service zentrale Kriterien zur guten Weiterentwickelbarkeit und Wartbarkeit während der Produktion erfüllt.

Eine Übersicht aller durchgeführten Testfälle mitsamt der erzielten Ergebnisse ist im Anhangskapitel A.3 gegeben. Durch den Erfolg aller Testfälle konnte gezeigt werden, dass die unter 2.1.1 gestellten Anforderungen an die Anwendung erfüllt werden konnten.

6. Fazit

Im abschließenden Kapitel sollen die Ergebnisse der vorliegenden wissenschaftlichen Arbeit zusammengefasst werden. Außerdem werden mögliche zukünftige Erweiterungen und Optimierungen am System benannt. Dabei wird der Mehrwert der Arbeit für zukünftige Forschungen herausgestellt.

6.1. Zusammenfassung

Wie in Kapitel 1.1 beschrieben, war das Hauptziel der vorliegenden wissenschaftlichen Arbeit, eine Serviceumgebung zu schaffen, die sich zur Einführung von Lehr-Lern-Szenarien eignet. Durch eine Abstraktheit des Servers sollten sowohl etablierte Umgebungen als auch neu entwickelte Szenarien in ihrer Umsetzung unterstützt werden. Aus diesen Hauptzielen sowie zahlreichen weiteren Aspekten, die sich aus der Analyse bestehender Systeme und Standards ergaben, konnten Anforderungen für die Entwicklung der Servicestruktur gestellt werden. Die Validierung der Anforderungen auf Client- und Serverseite sollte anhand von abgeleiteten Testfällen geschehen.

Der Erfolg aller Testfälle hat gezeigt, dass die prototypische Servicestruktur als generischer Ansatz für die Nutzung innerhalb der E-Learning-Domäne geeignet ist. Verwendete Implementierungen ermöglichen das Aufbauen von Lehr-Lern-Szenarien und unterstützen eine primitive Repräsentation innerhalb verschiedener Strukturen, wie etwa Kursen. Nutzer können innerhalb der Anwendung angelegt und über verschiedene Verwaltungsmöglichkeiten, wie Gruppen oder spezifische Rollen, organisiert werden. Mithilfe derartiger Zuordnungen ist es zudem möglich, den Zugriff auf bestimmte Kurse und Ressourcen über Elemente der Authentifizierung und Autorisierung einzuschränken. Der Prototyp unterstützt zudem mehrere erweiterbare Inhaltstypen, welche in Kursen angelegt werden können. Dazu zählen neben einfachen Text- und Dateiinhalten ebenfalls Aufgaben, welche von Nutzern gelöst

werden können. Anhand von gespeicherten Lösungen der Aufgaben ist zudem exemplarisch der xAPI-Standard innerhalb der Anwendung realisiert worden. Auch wird die Weiterentwickelbarkeit, gezielte Produktion und einfache Verteilbarkeit der Struktur durch die Technologie Docker sowie eine CI/CD-Pipeline ermöglicht.

6.2. Ausblick

Trotz der erzielten positiven Ergebnisse innerhalb der spezifizierten Testfälle kann die prototypische Servicestruktur zukünftig verändert und erweitert werden. Neben der Implementierung neuer modularer Komponenten für erweiterte Lehr-Lern-Szenarien können auch komplexere Systeme zur Benutzerverwaltung integriert werden. Auch wurde die direkte Interaktion zwischen Nutzern des Systems im Prototyp nicht gesondert betrachtet und ist somit Gegenstand eventueller zukünftiger Erweiterungen. Für derartige Weiterentwicklungen könnten neue Analysen, die Umsetzungen bestehender Plattformen betrachten, durchgeführt und in der Implementierung berücksichtigt werden. Insgesamt ist feststellbar, dass eine Erprobung des Prototyps an praktischen, realitätsnahen Kriterien durchgeführt werden muss, um Anforderungen an Funktionalität und System gesondert zu testen.

Die vorliegende wissenschaftliche Arbeit kann zudem in zukünftigen Forschungen genutzt werden, um analysierte Ansätze im Raum von E-Learning-Lösungen einzubeziehen und die daraus abgeleiteten Anforderungen für generische Services für andere Zwecke zu nutzen. Kriterien, die eventuell zu einer Veränderung des Servers führen, könnten neue Anforderungen hinsichtlich der Nutzbarkeit des Servers nach sich ziehen und somit Gegenstand neuer Forschungen sein.

Literaturverzeichnis

[Arn18] Patricia Arnold: *Handbuch E-Learning Lehren und Lernen mit digitalen Medien*, UTB 4965, 5. Aufl., 2018.

[Bal09] Helmut Balzert: *Lehrbuch der Softwaretechnik: Basiskonzepte und Requirements Engineering*, SpringerLink Bücher, 3. Aufl., 2009.

[Bar05] Phil Barker: *What is IEEE learning object metadata/IMS learning resource metadata*, *CETIS Standards Briefing Series, JISC (Joint Information Systems Committee of the Universities' Funding Councils)*, 2005.

[Ben14] Günther Bengel: *Grundkurs Verteilte Systeme Grundlagen und Praxis des Client-Server und Distributed Computing*, SpringerLink Bücher, 4. Aufl., 2014.

[Cha21] Chaminda Chandrasekara: *Hands-on GitHub Actions Implement CI/CD with GitHub Action Workflows for Your Applications*, Springer eBook Collection, 2021.

[Edw15] Shakuntala Gupta Edward: *Practical MongoDB Architecting, Developing, and Administering MongoDB*, Expert's voice in open source, 2015.

[Erp15] John Erpenbeck: *E-Learning und Blended Learning Selbstgesteuerte Lernprozesse zum Wissensaufbau und zur Qualifizierung*, essentials, 2015.

[Fri18] Thomas Frisendal: *Visual Design of GraphQL Data A Practical Introduction with Legacy Data and Neo4j*, 2018.

[Her22] Andrea Herrmann: *Grundlagen der Anforderungsanalyse Standardkonformes Requirements Engineering*, Springer eBook Collection, 2022.

[Hü12] Michael Hüttermann: *DevOps for Developers*, The expert's voice in Web development DevOps for developers, 2012.

[ISO22] ISO: *ISO/IEC 25010 Qualitätsmodell*, 2022, URL: https://iso25000.com/index.php/en/iso-25000-standards/iso-25010, besucht am 16.05.2022.

[Jon22] Edward R Jones: *Implications of SCORM and emerging e-learning standards on engineering education*, in *2002 GSW*, 2022.

[Ker20] David Kergel: *E-Learning, E-Didaktik und digitales Lernen*, Diversität und Bildung im digitalen Zeitalter, 2020.

[Lus03] Markus Lusti: *Dateien und Datenbanken Eine anwendungsorientierte Einführung*, Springer-Lehrbuch, 4. Aufl., 2003.

[Mei10] Andreas Meier: *Relationale und postrelationale Datenbanken*, eXamen.press, 2010.

[Met20] Anja Metzner: *Software Engineering - kompakt*, Hanser eLibrary, 2020.

[Nie20] Helmut Niegemann: *Handbuch Bildungstechnologie Konzeption und Einsatz digitaler Lernumgebungen*, Springer eBook Collection, 2020.

[ORC21] Mohammed Ouadoud, Nouha Rida und Tarik Chafiq: *Overview of E-learning Platforms for Teaching and Learning.*, *Int. J. Recent Contributions Eng. Sci. IT*, Bd. 9(1):S. 50–70, 2021.

[PLP21] Harsh Vardhan Pant, Manoj Chandra Lohani und Jeetendra Pande: *MOOCs in Higher Education: Current Trends in India and Developed Countries*, in *Ubiquitous Technologies for Human Development and Knowledge Management*, S. 58–77, IGI Global, 2021.

[Pou20] Nigel Poulton: *Docker Deep Dive Harness the full potential of your applications with Docker*, 2020.

[SP21] Amalia Suzianti und Sabrina Ayu Paramadini: *Continuance intention of e-learning: The condition and its connection with open innovation*, *Journal of Open Innovation: Technology, Market, and Complexity*, Bd. 7(1):S. 97, 2021.

[SS12] Alexander Schill und Thomas Springer: *Verteilte Systeme Grundlagen und Basistechnologien*, eXamen.press, 2. Aufl., 2012.

[Ste21] René Steiner: *Grundkurs relationale Datenbanken Einführung in die Praxis der Datenbankentwicklung für Ausbildung, Studium und IT-Beruf*, Springer eBook Collection, 10. Aufl., 2021.

[Tan08] Andrew S Tanenbaum: *Verteilte Systeme Prinzipien und Paradigmen*, it-informatik, 2. Aufl., 2008.

[Vad18] Sricharan Vadapalli: *DevOps: Continuous Delivery, Integration, and Deployment with DevOps Dive into the core DevOps strategies*, 2018.

[VW16] Stephen Victor und Art Werkenthin: *Cracking the mobile learning code: xAPI and cmi5*, Techn. Ber., Technical report. Obsidian Learning, 2016.

[Wir19] Ralf Wirdemann: *RESTful Go APIs Design und Implementierung leichtgewichtiger Hypermedia Services*, Hanser eLibrary, 2019.

[xAP22] xAPI: *xAPI-Spezfikation*, 2022, URL: https://github.com/adlnet/xAPI-Spec/blob/master/xAPI-Data.md, besucht am 24.06.2022.

Anhang

A. Testfälle

A.1. Testfälle für die Nutzung auf der Clientseite

FA-10-TF
Anforderung: FA-10 Authentifizierung
Abhängigkeiten: Keine.
Ausgangssituation: Anlegen einer Kursressource, welche eine beliebige Authentifikation zum Zugriff benötigt.
Ereignis: Nacheinander Zugriff auf die Ressource mit und ohne Authentifizierung des Nutzers.
Erwartetes Ergebnis: Ohne Authentifizierung ist Zugriff auf Ressource nicht möglich. Mit Authentifizierung ist Zugriff auf Ressource möglich.

FA-11-TF
Anforderung: FA-11 Zugänglichkeit nach Nutzergruppen
Abhängigkeiten: FA-10
Ausgangssituation: Anlegen einer Kursressource, welche einen Zugriff nur für bestimmte Nutzergruppen ermöglicht. Anlegen je eines Nutzers mit und ohne der benötigten Gruppe.
Ereignis: Nacheinander Zugriff auf die Ressource der Nutzer mit und ohne der benötigten Nutzergruppe.
Erwartetes Ergebnis: Ohne die Nutzergruppe ist Zugriff auf Ressource nicht möglich. Mit der Nutzergruppe ist Zugriff auf Ressource möglich.

FA-12-TF
Anforderung: FA-12 Verwaltung von Nutzergruppen und -berechtigungen
Abhängigkeiten: FA-10, FA-11

Ausgangssituation: Anlegen von verschiedenen Zugriffskomponenten, welche vestehende Nutzer oder Nutzergruppen einbeziehen.
Ereignis: Modifikation des Zugriffs durch Hinzufügen neuer Berechtigungen oder Veränderung der bestehenden Gruppen.
Erwartetes Ergebnis: Zugriff ist je nach Art der Autorisierung mit aktualisierten Berechtigungen verändert.

FA-20-TF
Anforderung: FA-20 Selektive Anfrage von Daten
Abhängigkeiten: Keine.
Ausgangssituation: Server ist gestartet. Nutzer besitzt Authentifizierung zur Datenabfrage.
Ereignis: Nutzer fragt selektiv nur bestimmte Daten vom Server an.
Erwartetes Ergebnis: Server antwortet mit Datensatz, der dem geforderten Umfang der Anfrage entspricht.

FA-30-TF
Anforderung: FA-30 Wiederverwendbarkeit von Kurs-Komponenten
Abhängigkeiten: Keine.
Ausgangssituation: Server ist gestartet. Anlegen von zwei Kursen und einer Kurs-Komponente.
Ereignis: Erstellte Kurs-Komponente soll in beiden Kursen verwendet werden. Anschließend erfolgt eine Anpassung der Komponente.
Erwartetes Ergebnis: Die Kurs-Komponente ist in beiden Kursen verfügbar. Die getätigte Änderung wirkt sich auf beide Kurse aus.

FA-31-TF
Anforderung: FA-31 Möglichkeit zum Anlegen von Übungen verschiedener E-Learning-Systeme
Abhängigkeiten: Keine.
Ausgangssituation: Server ist gestartet.

Ereignis: Durch Operationen auf dem Service werden Kurse angelegt. Durch Operationen auf dem Service werden Übungsaufgaben angelegt.
Erwartetes Ergebnis: Die Kurse und Übungsaufgaben stehen bereit.

FA-32-TF
Anforderung: FA-32 Modularisierung von Teilen des Service
Abhängigkeiten: Keine.
Ausgangssituation: Server ist gestartet. Anlegen einer Kursstruktur mit einer Kurs-Komponente. Anlegen einer weiteren unabhängigen Kurs-Komponente.
Ereignis: Austauschen der im Kurs eingesetzten Komponente mit der zweiten, unabhängigen Komponente.
Erwartetes Ergebnis: Einsetzen des anderen Moduls erfordert keine weiteren Änderungen. Kurs wird automatisch auf Änderungen der Komponente angepasst.

FA-40-TF
Anforderung: FA-40 Verwaltbarkeit verschiedener Kursstrukturen
Abhängigkeiten: Keine.
Ausgangssituation: Server ist gestartet. Erstellen eines Kurses mit initialen Kurs-Komponenten.
Ereignis: Veränderung der Bestandteile durch Erweiterung von Kursinhalten und der Neustrukturierung der Kurs-Komponenten.
Erwartetes Ergebnis: Änderungen wirken sich auf den Kurs aus. Inhalte des Kurses sind immer noch verfügbar und eindeutig dem Kurs zuordenbar.

FA-41-TF
Anforderung: FA-41 Kategorisierung von Kursen und Ressourcen nach ihrem Kontext und Einsatz
Abhängigkeiten: Keine.
Ausgangssituation: Server ist gestartet. Mindestens ein Kurs mit zugehörigen Komponenten ist angelegt.
Ereignis: Der Kurs und die Inhalte werden durch bestimmte Merkmale kategorisiert.
Erwartetes Ergebnis: Die Kategorisierung ist hinterlegt und kann zwischen den Kursen und Inhalten genutzt werden, um Verknüpfungen herzustellen.

FA-42-TF
Anforderung: FA-42 Realisierung von Suchanfragen
Abhängigkeiten: FA-41
Ausgangssituation: Anlegen verschiedener Ressourcen mit zugehörigen Kategorisierungskriterien.
Ereignis: Ausführen mehrerer Suchanfragen mit Abstimmung auf erstellte Kriterien.
Erwartetes Ergebnis: Pro Suchanfrage korrektes Filtern der gewünschten Resultate.

FA-50-TF
Anforderung: FA-50 Implementierung des xAPI-Standards
Abhängigkeiten: FA-31
Ausgangssituation: Server ist gestartet. Beispielübung zur Erfassung in xAPI wird angelegt.
Ereignis: Nutzerinteraktion anhand der Übung wird mit richtiger und falscher Antwort ausgeführt.
Erwartetes Ergebnis: Nutzerintaktion wird aufgezeichnet und ist über eine Query im xAPI-Datenformat abrufbar. Daten, welche der Spezifikation entsprechen, stehen bereit.

FA-60-TF
Anforderung: FA-60 Möglichkeit zum einfachen Deployment der Gesamtstruktur
Abhängigkeiten: Keine.
Ausgangssituation: Neue Version der Servicestruktur steht bereit. Änderungen sind im Versionskontrollsystem erfasst.
Ereignis: Deployment-Pipeline wird durch einen manuellen Trigger unter GitHub Actions ausgelöst.
Erwartetes Ergebnis: Neue Version des Service wird korrekt gebaut. Resultierendes Docker-Image ist anschließend über die Plattform Docker Hub verfügbar.

FA-61-TF
Anforderung: FA-61 Verfügbarkeit unter Windows, Linux und Mac
Abhängigkeiten: Keine.
Ausgangssituation: Neue Version der Docker Images steht lokal oder über Docker Hub bereit.

Ereignis: Erstelltes Image wird unter den Betriebssystemen Windows, Mac und Linux installiert und anschließend ausgeführt.
Erwartetes Ergebnis: Die ausgeführte Servicestruktur ist ohne Einschränkungen unter allen Betriebssystemen identisch nutzbar.

NA-10-TF
Anforderung: NA-10 Performanz der Datenanfrage
Abhängigkeiten: Keine.
Ausgangssituation: Server ist gestartet.
Ereignis: Es werden neun verschiedene Anfragen ausgelöst. Die Zeit für die Übermittlung einer Response wird gemessen.
Erwartetes Ergebnis: Die Übermittlung der Response überschreitet nicht den Wert von 100 Millisekunden.

A.2. Testfälle für die Wartung und Entwicklung des Servers

FA-70-TF
Anforderung: FA-70 Erweiterbarkeit der persistenten Strukturen
Abhängigkeiten: Keine.
Ausgangssituation: Server besitzt stabile Version.
Ereignis: Aktueller Stand des Servers soll um neue potentielle Kurs-Komponente erweitert werden.
Erwartetes Ergebnis: Erweiterung legt neue benötigte Datenstrukturen in der Datenbank an. Es findet keine Kollision mit bestehenden Daten statt.

FA-80-TF
Anforderung: FA-80 Automatisiertes Deployment
Abhängigkeiten: FA-60.
Ausgangssituation: Entwickler hat Server durch eine Änderung erweitert.
Ereignis: Entwickler lädt Entwicklungsstand in das GitHub Remote-Repository hoch.

Erwartetes Ergebnis: Automatische Deployment-Pipeline wird ausgelöst. Version mit entwickelten Änderungen steht nach erfolgreichem Durchlaufen der Pipeline im Rahmen des Deployments bereit.

NA-20-TF
Anforderung: NA-20 Ressourcensparender Betrieb
Abhängigkeiten: Keine.
Ausgangssituation: Server wird auf einem zentralen System gestartet.
Ereignis: Der Server wird auf verschiedenen Wegen ausgelastet. Es wird der Arbeitsspeicherverbrauch der Anwendung erfasst.
Erwartetes Ergebnis: Der Arbeitsspeicherverbrauch überschreitet nicht den Wert von 1GB RAM.

NA-30-TF
Anforderung: NA-30 Dokumentation des Quellcodes
Abhängigkeiten: Keine.
Ausgangssituation: Quellcode des Servers steht bereit.
Ereignis: Es werden für verschiedene Komponenten des Systems beliebig Ausschnitte des Quellcodes betrachtet.
Erwartetes Ergebnis: Für alle zu betrachtenden Elemente ist eine im Quellcode vorgenommene Dokumentation vorhanden. Es ist eine Einordnung der jeweiligen Komponente in den Gesamtkontext des Systems ersichtlich.

NA-40-TF
Anforderung: NA-40 Sicherstellung der Skalierbarkeit durch Verteilung
Abhängigkeiten: Keine.
Ausgangssituation: Server besitzt stabile, verteilte Version. Verfügbarkeit mehrerer Beispielsysteme.
Ereignis: Komponenten des Servers werden auf die vorhandenen Systeme verteilt. Anschließend wird die verteilte Anwendung gestartet.
Erwartetes Ergebnis: Die einzelnen Komponenten sind ausführbar. Die Anwendung ist verfügbar und setzt sich aus den verschiedenen Komponenten zusammen.

NA-41-TF
Anforderung: NA-41 Wartbarkeit des Systems
Abhängigkeiten: NA-40
Ausgangssituation: Server ist gestartet. Anwendung ist über mehrere Komponenten verteilt.
Ereignis: Für Wartung des Servers soll verteilte Anwendung gestoppt werden. Das Beenden erfolgt über eine zentrale Komponente.
Erwartetes Ergebnis: Das Stoppen über die zentrale Komponente wirkt sich auf alle verteilten Komponenten aus.

NA-50-TF
Anforderung: NA-50 Automatisierte Tests
Abhängigkeiten: Keine.
Ausgangssituation: Stabile Version des Servers ist mitsamt entwickelter Tests verfügbar.
Ereignis: Automatisierte Tests werden ausgeführt.
Erwartetes Ergebnis: Tests laufen fehlerfrei und geben Auskunft über Testergebnisse.

A.3. Übersicht der Durchführung aller Testfälle

	ID	Erfüllung der Anforderung
Testfälle für funktionale Anforderungen	FA-10-TF	erfüllt
	FA-11-TF	erfüllt
	FA-12-TF	erfüllt
	FA-20-TF	erfüllt
	FA-30-TF	erfüllt
	FA-31-TF	erfüllt
	FA-32-TF	erfüllt
	FA-40-TF	erfüllt
	FA-41-TF	erfüllt
	FA-42-TF	erfüllt
	FA-50-TF	erfüllt
	FA-60-TF	erfüllt
	FA-61-TF	erfüllt
	FA-70-TF	erfüllt
	FA-80-TF	erfüllt
Testfälle für nicht-funktionale Anforderungen	NA-10-TF	erfüllt
	NA-20-TF	erfüllt
	NA-30-TF	erfüllt
	NA-40-TF	erfüllt
	NA-41-TF	erfüllt
	NA-50-TF	erfüllt

Tabelle A.1.: Zusammenfassung der durchgeführten Testfälle

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig angefertigt, nicht anderweitig zu Prüfungszwecken vorgelegt und keine anderen als die angegebenen Hilfsmittel verwendet habe. Sämtliche wissentlich verwendete Textausschnitte, Zitate oder Inhalte anderer Verfasser wurden ausdrücklich als solche gekennzeichnet.

Mittweida, den 29. Juli 2022