HAW HAMBURG

Bachelorarbeit

Philip Aguilar Bremer

Erstellung eines Konzepts mit prototypischer Umsetzung einer Lehrendenkomponente als Microservice im Rahmen einer Online-Programmierplattform

Hochschule für Angewandte Wissenschaften Hamburg
Fakultät Technik und Informatik
Department Informatik

Hamburg University of Applied Sciences
Faculty of Computer Science and Engineering
Department Computer Science

Philip Aguilar Bremer

Erstellung eines Konzepts mit prototypischer Umsetzung einer Lehrendenkomponente als Microservice im Rahmen einer Online-Programmierplattform

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im gemeinsamen Bachelorstudiengang Wirtschaftsinformatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Jens von Pilgrim
Zweitgutachter: Prof. Dr. Axel Schmolitzky

Eingereicht am: 19.10.2021

Philip Aguilar Bremer

Thema der Arbeit

Erstellung eines Konzepts mit prototypischer Umsetzung einer Lehrendenkomponente als Microservice im Rahmen einer Online-Programmierplattform

Stichworte

HAW-OPPSEE, Learning Management Systeme, Programmierplattformen, Webpack, Visual Studio Code Extension API, React, Node.js, Express.js

Kurzzusammenfassung

Diese Bachelorarbeit befasst sich damit, wie im Rahmen der HAW-OPPSEE Plattform eine Frontendkomponente für den Lecturer konzipiert werden kann. Dabei werden unter Verwendung der VSCode Extension API Webviews innerhalb des Visual Studio Code Editors integriert, welche die Anforderungen für den Lecturer an eine Online Programmierplattform erfüllen. Zunächst werden dafür Plattformen untersucht, die Lecturer-spezifische Funktionen bereits umsetzen. Auf dieser Basis werden die Anforderungen ermittelt, die der Lecturer an eine Online-Programmierplattform stellt. Danach werden die Anforderungen in ein UML-Use-Case-Diagramm überführt. Ausgehend von diesem Modell wird die Architektur und der Entwurf für die prototypische Implementierung erstellt. Anschließend werden die wesentlichen Implementierungsschritte betrachtet. Zu guter Letzt werden die Anforderungen mit den tatsächlich implementierten Funktionen abgeglichen, bevor als letztes mögliche Probleme aufgelistet werden und die Lösung als Ganzes evaluiert wird.

Philip Aguilar Bremer

Title of Thesis

Developing a concept with a prototype implementation of a lecturer component as a microservice within the scope of an online programming platform

Keywords

HAW-OPPSEE, learning management systems, programming platforms, Webpack, Visual Studio Code Extension API, React, Node.js, Express.js

Abstract

This thesis addresses how a frontend component for a lecturer can be implemented within the framework of the HAW-OPPSEE platform. Using the VSCode Extension API, web views are integrated within the Visual Studio Code Editor that meet the lecturer's requirements for an online programming platform. First, platforms that already implement lecturer-specific features will be examined. Based on this, the requirements that the lecturer demands from an online programming platform are identified. Then, the requirements are transformed into a UML use case diagram. Starting from this model, the architecture and the design for the prototype implementation are created. Subsequently, the main implementation steps will be discussed. Finally, the requirements are compared with the actually implemented features, before potential problems are listed and the solution as a whole is evaluated.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungen

AI Angewandte Informatik.

CRUD Create, Read, Update, Delete.

CSS Cascading Style Sheets.

ER Modell Entity-Relationship Modell.

HAW Hochschule für Angewandte Wissenschaften.

ID Identifikationsnummer.

IDE Integrierte Entwicklungsumgebung.

LMS Learning Management System.

MVP Minimal Viable Product.

OPPSEE Online Programming Platform for Software Engineering Education.

RegEx Regular expression.

RWD Responsive Webdesign.

SQL Structured Query Language.

TI Technische Informatik.

UML Unified Modeling Language.

URI Uniform Resource Identifier.

VSCode Visual Studio Code.

WI Wirtschaftsinformatik.

1 Einleitung

Die Fragestellung, inwiefern die Erstellung eines Konzepts mit prototypischer Umsetzung einer Lehrendenkomponente als Microservice im Rahmen einer Online-Programmierplattform möglich ist, hat sich aus dem Online Programming Platform for Software Engineering Education (OPPSEE)-Projekt der Hochschule für Angewandte Wissenschaften (HAW) Hamburg ergeben. Das Projekt OPPSEE beschreibt dabei eine Online Programmierplattform, welche als zusätzliches freiwilliges Angebot für alle Informatikstudierenden zur Verfügung stehen soll. Hauptsächlich für jene, die ihre Programmierfähigkeiten alleine oder in einem Team durch das Bearbeiten von Übungsaufgaben, sogenannten Assignments verbessern wollen [HBPS21, S. 4].

Um jedoch die Benutzerfreundlichkeit für den Lecturer zu erhöhen, ist es nötig, nicht nur eine bloße Integrierte Entwicklungsumgebung (IDE) bereitzustellen, in der Programmieraufgaben bearbeitet und ausgewertet werden können, sondern auch eine Weboberfläche innerhalb der IDE anzubieten, welche die Anforderungen eines Lecturers an eine solche Plattform befriedigt. Wie dieses Vorhaben umgesetzt werden kann, wird im Laufe dieser Arbeit genauer betrachtet.

1.0.1 Ziel der Arbeit

Ziel dieser Bachelorarbeit ist es diese Anforderungen, die der Lecturer an eine Online Programmierplattform stellt herauszufinden sowie zu dokumentieren. Darauf aufbauend soll die Architektur und der Entwurf für die Lehrendenkomponente innerhalb der HAW-OPPSEE Plattform konzipiert werden.

Anschließend soll vor allem mit dem Fokus auf die Frontendkomponente untersucht werden, wie gut die Lecturer-spezifischen Funktionen unter Einhaltung der Visual Studio Code (VSCode) Extension Richtlinien implementiert werden können. Dazu wird sowohl für das Frontend als auch für das Backend ein Prototyp entwickelt und anschließend auf seine Funktionalität getestet. Schließlich soll evaluiert und validiert werden auf welche Art und Weise Anforderungen umgesetzt, oder auch nicht umgesetzt werden konnten.

1.0.2 Konkretisierung der Aufgabenstellung

Der Hauptfokus bezüglich Prototypen in dieser Arbeit liegt auf dem Entwickeln der Frontendkomponente. Sie soll mindestens die Funktionen eines sogenannten Minimal Viable Products (MVPs) umfassen. Die Frontendkomponente schafft eine Anwendungsoberfläche für den Lecturer und wird als VSCode Extension entwickelt. Daher wird sie im Folgenden Lecturer Extension genannt.

Die Begriffsdefinition des MVPs ist in der Literatur wie von Lenarduzzi und Taibi beschrieben nicht eindeutig, sondern hat sich über die Jahre hinweg immer weiterentwickelt [LT16, S. 112]. In dieser Arbeit wird davon ausgegangen, dass für die VSCode Extension als MVP das minimale Set an Features umgesetzt werden muss, um die Anforderungen des Lecturers zu befriedigen. Dies ist nach Lenarduzzi und Taibi ebenfalls die am weitesten verbreitete Definition für das MVP [LT16, S. 119]. Sie wird von knapp einem Drittel der 22 untersuchten Literaturergebnisse unterstützt [LT16, S. 118]. Ausgehend davon hat das zur Folge, dass für Features, welche die Implementierung über unterschiedliche Wege ermöglichen, sich für eine Option entschieden werden sollte.

Das für die Anwendung notwendige Backend dagegen, soll nur beispielhaft implementiert werden, weil es für die Lauffähigkeit der Lecturer Extension essenziell ist. Dies liegt daran, dass Services benötigt werden, um Authentifizierungsvorgänge und das Persistieren von Daten zu bearbeiten.

Die Art der Backend Implementierung soll daher nicht die spätere Umsetzung innerhalb der HAW-OPPSEE Plattform veranschaulichen. Deswegen wird sich der Einfachheit halber für die Verwendung von Express entschieden. Express beschreibt sich als populäres, minimales und flexibles Node.js Webanwendungs-Framework, das für die schnelle und robuste Erstellung von APIs verwendet werden kann.

1.0.3 Aufbau der Arbeit

Um die bereits genannten Ziele in 1.0.1 zu erreichen, wird sich in Kapitel 2 zunächst ein Überblick darüber verschafft, welche für den Lecturer interessanten Funktionen die populären Learning Management Systeme (LMS) bereits anbieten. Anschließend werden der Aufbau sowie die Funktionen dreier ausgewählter Programmierplattformen aus dem deutschsprachigen Raum begutachtet.

Darauf aufbauend werden in Kapitel 3 erst die Stakeholder sowie der Systemkontext bestimmt. Danach werden die funktionalen bzw. nicht-funktionalen Anforderungen des Lecturers zur Übersicht tabellarisch aufgelistet. Schließlich wird durch Anwendungsfälle genauer darauf eingegangen, wie sich das System im Normalfall bzw. in Sonderfällen verhalten muss.

Nachdem die Erfassung und genaue Beschreibung der Anforderungen abgeschlossen ist, wird in Kapitel 4 die geplante Architektur der Lecturer Extension sowie des Lecturer Backends beschrieben.

Im nächsten Abschnitt 4.1.1 wird zunächst ein Domänenmodell erstellt, um eine Übersicht über die Klassen, ihre Beziehungen und Attribute im System zum schaffen. Davon ausgehend können wir unser Datenbankmodell erstellen, welches die Grundlage zur Erstellung unser Datenbanktabellen im Lecturer Backend schafft.

In den nächsten zwei Abschnitten 4.2 und 4.3 werden die beiden zu implementierenden Komponenten nun innerhalb eines Systems betrachtet. Daraus resultierend werden zunächst die Subsysteme, sowie Schnittstellen untereinander spezifiziert. Im Feinentwurf soll dann weiter auf den Fachkern der Klassen eingegangen werden. Jegliche Attribute mitsamt ihrer Datenstrukturen, sowie Operationen sind hier aufzuzeigen. An dieser Stelle wird vor allem auch auf das erweiterte Klassendiagramm der Frontendkomponente eingegangen. Es beschreibt maßgeblich die Umsetzung der Anwendungsfälle, weil diese hauptsächlich Interaktionen zwischen Nutzer*in und der Lecturer Extension beschreiben.

Kapitel 5 beschreibt ein Konzept, wie das Front- bzw. Backend implementiert und aufgesetzt werden kann. Dabei wird u. a. behandelt, welche Technologien im Entwicklungsprozess verwendet werden, warum diese benutzt werden, welche Schwierigkeiten in der Entwicklung des Prototypen auftreten können und wie man diese lösen kann. Bezogen auf die Lecturer Extension im Frontend wird zudem noch auf das Design eingegangen. Die Korrektheit der implementierten Funktionen wird außerdem durch eine Reihe von Tests überprüft.

Im vorletzten Kapitel 6 wird mit Hilfe von Screenshots der Anwendung dokumentiert, welche Anforderungen umgesetzt werden konnten.

Zum Abschluss werden im letzten Kapitel 7 die Ergebnisse dieser Arbeit zusammengetragen und ein Ausblick gegeben.

1.0.4 Konventionen

Aus der Informatik haben sich heutzutage viele Anglizismen durchgesetzt, für die keine geläufige Übersetzung ins Deutsche existiert. Dazu kommt, dass der Großteil aller Programmiersprachen (beispielsweise JavaScript, Java, Python) auf Englisch verfasst wurden. Des Weiteren benutzen auch alle in dieser Arbeit verwendeten Frameworks und Bibliotheken ausschließlich englische Begriffe zur Bezeichnung von Schlüsselwörtern, Funktionen und Objekten. Daher ist es auch im deutschsprachigen Raum üblich, dass der Programmierquellcode auf Englisch verfasst wird. Auch während der Literaturrecherche für diese Arbeit wurden bestimmte Begriffe, u. a. für Systeme und Stakeholder, teilweise aus dem Englischen abgeleitet.

Auf dieser Grundlage werden für die Informatik typische Anglizismen, bereits in der hier verwendeten Literatur eingeführte Anglizismen und Quellcode nicht ins Deutsche übersetzt. Stattdessen wird ihre Bedeutung im Glossar beschrieben. Englische Substantive und Abkürzungen werden in dieser Arbeit durchweg groß geschrieben. Auch sollen sie nach den Regeln der deutschen Sprache dekliniert werden, um den Lesefluss zu verbessern. Anzumerken ist noch, dass das Einführen von neuen Anglizismen sowie das Konjugieren von englischen Begriffen, wie z. B. „gecancelled“, unerwünscht ist.

2 Related Work

In diesem Kapitel wird untersucht, inwiefern bereits bestehende Online-Lernplattformen den Workflow seitens des Dozierenden (im Folgenden Lecturer genannt) gestalten. Der Fokus liegt dabei auf Funktionen, die in der integrierten Entwicklungsumgebung (IDE) Visual Studio Code [VSCHome] mit Hilfe einer Erweiterung (im Folgenden Extension genannt) implementiert werden können. Dafür sollen sowohl allgemeine als auch informationstechnische Elemente betrachtet werden.

Um dies zu erreichen, wird für die Erfassung der Umsetzung von Lecturer-gebundenen Funktionen ein Mix zwischen populären Learning Management Systemen und bereits existierenden Online-Programmierplattformen genauer betrachtet. Ziel dieses Kapitels ist es, Anforderungen aus bereits bestehenden Konzepten herauszufiltern, die der Lecturer an eine solche Plattform stellt.

2.1 Learning Management Systeme (LMS)

Unter einem LMS versteht man ein System, das den Zugang zu Online-Lerndiensten für Studierende, Lehrerende und Administrierende organisiert und bereitstellt. Diese Lerndienste umfassen in der Regel die Zugriffskontrolle und Bereitstellung von Lehrinhalten, Kommunikationswerkzeuge und die Organisation von Benutzergruppen [Pau02, S. 5-6]. Da vor allem diese Funktionsbereiche interessant für den Lecturer sind, lohnt es sich einen genaueren Blick auf ausgewählte LMS zu werfen.

Die Welt der LMS ist allerdings groß. Laut Schätzungen der Talented Learning, LLC (ein nach eigenen Angaben unabhängiges Forschungs- und Beratungsunternehmen [TLAbout]) existierten im Jahr 2019 bereits über 800 LMS Anbieter [Leh21]. Aufgrund der vielfältigen Lösungsmöglichkeiten wird sich daher der Übersicht halber auf die drei aktiv und meistgenutzten Systeme von Hochschulen konzentriert.

Diese lauten nach LISTedTECH [LIS21], einem Marktforschungsunternehmen, das laut seiner Webseite über 90% des Nordamerikanischen Marktes und 60-75% des europäischen Marktes abdeckt:

- Moodle ($> 40\%$[1]) [MooHome]
- Blackboard ($> 10\%$[1]) [BlaHome]
- Canvas ($> 10\%$[1]) [InsHome]

Zu Beginn soll vorweggenommen werden, dass das Ziel dieser Arbeit nicht ist, alle grundlegenden Funktionen der LMS zu ersetzen. Vielmehr soll die Online-Programmierlernplattform nebenher betrieben werden. Dabei sind die Lecturer-spezifischen Funktionen auf diese Plattform zu übertragen und somit später als Microservice prototypisch zu implementieren. Zunächst werden im Folgenden die grundlegenden Funktionen hauptsächlich am Marktführenden Moodle analysiert. Anschließend werden diese mit Features der anderen LMS in Reihenfolge der obigen Auflistung ergänzt.

Anzumerken ist, dass die drei marktführenden LMS eine große Ähnlichkeit zueinander aufweisen [Pat13]. Eine doppelte Aufzählung der gleichen Funktion soll hierbei jedoch vermieden werden, da dies keinen Mehrwert für die folgende Anforderungsanalyse mit sich bringt. Um Wiederholungen zu vermeiden, liegt der Fokus in den nach Moodle folgenden Abschnitten daher hauptsächlich darauf, neue oder signifikant unterschiedlich implementierte Funktionen zu ermitteln.

2.1.1 Moodle

Moodle ist ein freies Open-Source (FOSS) Learning Management System (LMS), das in PHP und Javascript geschrieben ist. Eine SQL-Datenbank wird verwendet, um Dateien zu persistieren [MooGitH]. Moodle wurde unter der General Public License (GNU) veröffentlicht [MooLice]. Laut Moodles Webseite [MooFeat] werden folgende Funktionen angeboten, die für den Lecturer von Interesse sein können:

[1] Marktanteil

1. Ein (personalisiertes) Dashboard [MooDash]
   Das Moodle Dashboard dient dem Nutzer unter anderem zur Kursübersicht und erscheint in der Regel nach dem Login. Alternativ kann der Nutzer auf das Dashboard über den Link im Profil, oder in der Navigationsleiste zugreifen. Kurse können mit einem Stern markiert oder verborgen werden. Demnach kann man Kurse nach folgenden Parametern filtern: In Bearbeitung, zukünftig, vergangen, markiert/hervorgehoben, verborgen.

   Die Kursübersicht kann darüber hinaus entweder nach kürzlich besuchten Kursen, oder alphabetisch nach den Namen der Kurse sortiert werden. Zudem kann man die Kurse auf 3 verschiedenen Wegen anzeigen lassen. In Form von Kurskarten, als Liste, oder in einer zusammengefassten Kursübersicht.

   Zusätzlich zur Kursübersicht, kann das Dashboard vom Nutzer verwaltet und überarbeitet werden. Die Personalisierung basiert dabei auf sogenannten Blöcken, die je nach Berechtigung hinzugefügt, verschoben oder gelöscht werden können. Dazu gehören zum Beispiel „kürzlich besuchte Kurse“ oder „mit Sternchen markierte Kurse“, die so gesondert angezeigt werden. Blöcke folgen dabei dem Responsive Webdesign (RWD). Das heißt, dass die Blöcke sich je nach Displaygröße dem Display anpassen und verschieben. RWD bedeutet also, dass eine Webseite so konzipiert wird, dass sie auf allen Zielgeräten ordnungsgemäß angezeigt und verwendet werden kann [JSZ+18, S. 2]. Weitere Blöcke die standardmäßig neben der Kursübersicht auf dem Dashboard angezeigt werden, sind folgende:

   - Navigation
   - Einstellungen
   - Meine Dateien
   - Neue Auszeichnungen
   - Kalender
   - Aktuelle Termine
   - Online Aktivitäten

2. Authentifizierung
   Bevor ein genauerer Blick auf die Gestaltung der Kurse geworfen wird, soll genauer betrachtet werden, inwiefern man sich für die Moodle Plattform authentifizieren kann, und welche Rollen einem anschließend innerhalb und außerhalb von Kursen zugewiesen werden.

   Moodle listet auf seiner Webseite 13 verschiedene Authentifizierungs-Plugins auf [MooAuth]. Interessant ist hier vor allem die OAuth2 Authentifizierung, da die HAW Hamburg durch ihre GitLab-Plattform [Bla20] bereits einen OAuth2 Authentifizierungsservice besitzt. Diese Authentifizierungsmethode hat den Vorteil, dass keine Nutzerdaten mehr in der Applikation gespeichert oder verwaltet werden müssen [Hen20, S. 98]. Zudem erhält der Nutzer ein sogenanntes Access Token, das je nach Scope den Zugriff auf bestimmte Daten des Authentifizierungsproviders ermöglicht [GitOAut]. Dies ist vor allem in IDEs wie auch Visual Studio Code von Vorteil, da so direkt auf Git Ressourcen, wie zum Beispiel Repositorys zugegriffen werden kann.

   Eine weitere interessante Spezifikation, die im Zusammenhang mit Authentifizierung genannt wird, ist die vom IMS Global Learning Consortium entwickelte Learning Tools Interoperability® (LTI®). Sie ist ein Standard, der es erlaubt Kursinhalte und Lernwerkzeuge von verschiedenen Anbietern in eine Lernplattform zu integrieren [IMSFAQs]. Um diese Spezifikation zu unterstützen, gibt es je nach Programmiersprache Bibliotheken, die diesen Standard umsetzen. Um in unserer Anwendung den LTI Standard zu implementieren, und unsere Applikation somit integrierbar in Moodle zu machen, könnten wir beispielsweise das ltijs npm Modul [Cos21] verwenden.

   Aufgrund der Tatsache, dass unsere Lecturer-Komponente jedoch nur als Visual Studio Code Extension, und nicht als eigenständige Webanwendung entwickelt wird, macht die Anwendung dieses Standards hier keinen Sinn. Denn um unsere Applikation lauffähig zu gestalten, muss diese in einer IDE ausgeführt werden, welche die VS Code Extension API unterstützt. Daher wäre sie selbstständig in einem LMS nicht ausführbar.

3. Rollen
   Rollen sind dafür zuständig die Zugriffsrechte auf Einstellungen und Ressourcen in dem LMS zu definieren. Moodle listet auf seiner Webseite acht verschiedene Standardrollen auf, die einem Nutzer zugewiesen werden können [MooSRol]:

   - Administrator/in (vollständige Admin-Rechte auf der Moodle Webseite)
   - Manager/in (Sub-Administrator/in, darf weniger als Administrator/in)
   - Kursersteller/in (darf neue Kurse erstellen)
   - Trainer/in (darf Kurse und Kursinhalte verwalten)
   - Trainer/in ohne Bearbeitungsrecht (darf Bewertungen aber keine Änderungen an Kurs bzw. Kursinhalten vornehmen)
   - Teilnehmer/in (kann auf einen Kurs zugreifen und an Aktivitäten teilnehmen)
   - Gast (Kann Kurse ansehen, aber nicht dran teilnehmen)
   - Authentifizierte/r Nutzer/in (Rolle für alle angemeldeten Nutzer)

   Diese Rollen sind jedoch auch abhängig vom Kontext, in dem ein Nutzer steht. Als typisches Beispiel wird der Kurskontext mit Teilnehmer/in bzw. Trainer/in Rolle aufgeführt [MooRole].

   Davon ausgehend sind für den Lecturer der HAW-OPPSEE vor allem folgende Rollen von Interesse, um Kursinhalte sicher zu verwalten: Kursersteller/in, Trainer/in und Teilnehmer/in. Der/die Trainierende ohne Bearbeitungsrecht wird dabei nicht berücksichtigt, da u. a. nach Mitarbeitern des OPPSEE Teams [OPPAbou] der Fokus darauf liegt, Studierenden die Möglichkeit zu geben Inhalte eines bestimmten Themas zu üben, wiederholen oder gar vertiefen [GHS20, S. 30]. Resultierend aus diesen genannten Anforderungen kann man schließen, dass auf der Plattform Übungsaufgaben im Mittelpunkt stehen sollen. Folglich werden Benotungen innerhalb eines solchen Kurses und damit auch die zugehörige Rolle überflüssig.

4. Kurse
   Nach Moodle sind Kurse Bereiche "[...] in denen Trainer/innen Lerninhalte für ihre Kursteilnehmer/innen bereitstellen können"[MooKurs]. Das Erstellen eines Kurses in Moodle erfordert standardmäßig mindestens die Kursersteller/in Rolle. Die Trainer/in Rolle reicht hierfür nicht aus [MooKAnl].

   Um sich in einen Kurs einzuschreiben bietet Moodle wieder zahlreiche (mindestens zwölf) verschiedene Methoden an [MooEins]. Um diese übersichtlicher zu gestalten, werden diese von uns in fünf verschiedene Gruppen eingeteilt. Dabei analysieren wir, inwiefern die Einschreibemöglichkeiten sinnvoll in unsere Lehrendenkomponente integriert werden können:

   - Einschreibung über externe Quellen
     - Einschreibung über andere (Moodle) Installation
     - Einschreibung über LDAP-Server
     - Einschreibung über externe Datenbank

     Kurseinschreibungen über externe Quellen sind für unsere Lehrendenkomponente in Bezug auf den Lecturer nicht relevant, weil dieser keinen Einfluss auf diese Einschreibemethoden hat. Nutzer über ein gesamtes (Moodle-)Netzwerk einzuschreiben kann dabei z.B. dazu führen, dass jene unerwünscht zu einem Kurs hinzugefügt werden. Dies sollte weder im Interesse des Lecturers noch der weiteren Kursteilnehmer sein.

     LDAP steht für Lightweight Directory Access Protocol [Ser06] (leichtgewichtiges Verzeichniszugriffsprotokoll). LDAP Verzeichnisse sind dabei kurzgefasst eine Standardtechnologie um Nutzerinformationen, -gruppen und Zugriffsrechte zu speichern und über einen Server bereitzustellen. An diesem sogenannten LDAP Server besteht die Möglichkeit für einen Nutzer sich zu authentifizieren [RSKP20, S. 1]. Die Authentifizierung der User via LDAP ist wiederum Voraussetzung für die Kurseinschreibung durch LDAP [MooLDAP]. Da wir jedoch eine Authentifizierung über das OAUTH2 Protokoll mit GitLab anstreben, um gegebenenfalls Zugriff auf für Programmieraufgaben nicht unwichtige Git-Ressourcen zu erhalten, kommt das LDAP Protokoll hier nicht in Frage.

Mit einer externen Datenbank dagegen soll unsere Lehrendenkomponente kommunizieren können. Theoretisch wird es auch möglich sein, in dieser als datenbankadministrierende Person neue Kursteilnehmende einzutragen. Es wäre jedoch wünschenswert, wenn der Lecturer zumindest die Möglichkeit hätte diesen Vorgang von sich aus zu verwalten oder alternativ Studierenden eine Option zur Selbsteinschreibung geben könnte. Genauer darauf eingegangen wird jedoch in den folgenden Punkten.

- Von einem Anwendungsnutzenden initiierte Einschreibemethoden
  - Manuelle Einschreibung
  - Selbsteinschreibung

  In den von Anwendungsnutzenden initiierten Einschreibemethoden kann standardmäßig ein Administrierender bzw. Lecturer User manuell zum Kurs hinzufügen [MooMEnr]. Dies könnte Letztere/n insoweit helfen, weitere Kursleitende bereits gleich nach Erstellung des Kurses einzutragen. So kann die Last der organisatorischen oder verwaltungstechnischen Aufgaben umgehend abgegeben oder verteilt werden.

  Für die Methode „Selbsteinschreibung“ kann der Lecturer (bei Moodle z.B. ein/e Nutzer/in mit der Rolle Trainer/in) einen Kursschlüssel definieren, der benötigt wird um Zugriff auf den entsprechenden Kurs zu bekommen [MooSelf]. Dadurch kann die Verantwortung der Anmeldung für einen Kurs auf den Studierenden übertragen und der organisatorische Aufwand des Lecturers gleichzeitig verringert werden. Vor allem weil die Übungsplattform bereits eine gewisse Eigeninitiative seitens der Studierenden erfordert, macht diese Methode durchaus Sinn.

- Einschreibung von Nutzergruppen auf Basis von (Meta-)Daten
  - Globale Gruppen
  - Meta-Einschreibung
  - Kursbereichseinschreibung

  Die Methoden „Globale Gruppen“, „Meta-Einschreibung“ und „Kursbereichseinschreibung“ bieten dem Lecturer die Möglichkeit User nach einem bestimmten Bereich [MooKBer] (z.B. Wirtschaftsinformatik), einer globalen Gruppenzugehörigkeit [MooGlob] (z.B. Lecturer) oder einer bereits bestehenden Kurszugehörigkeit [MooMeta] hinzuzufügen. Auf Grundlage dieser Metadaten bietet sich der Vorteil, dass Kursteilnehmende mit einem gleichen Berechtigungsprofil entgegengesetzt zu den von Anwendungsnutzenden initiierten Einschreibemethoden auch zusammengefasst als Nutzergruppen verwaltet werden können.

- Einschreibung über das Hochladen...
  - ...einer CSV-Datei
  - ...einer XML-Datei

  Das Eintragen der Kursteilnehmenden über das Hochladen einer Datei bietet dem Lecturer eine attraktive Alternative, falls diese bereits in XML-/CSV-Format vorhanden sind. So könnte die Kursregistrierung für alle mit dem Upload einer einzigen Datei abgeschlossen werden.

- Sonstige
  - Gastzugang
  - Zugang durch Zahlung (u. a. via PayPal)

  Ein Gastzugang könnte für einen Lecturer von Bedeutung sein, falls er seinen Kurs allen Authentifizierten der Plattform bereitstellen möchte. Technisch ist diese Methode dagegen aber uninteressant, da hierfür lediglich eine Option im Kurs eingebaut werden müsste, die den Kurseinschreibungsschritt ignoriert.

  Zahlungen für Kurse im Rahmen eines begleitenden Angebots für Lehrveranstaltungen einer Hochschule sind vorerst nicht vorgesehen.

Auch für die Kurseinstellungen listet Moodle wieder 12 verschiedene Kategorien auf[MooSett]. Nachfolgend werden die Einstellungsmöglichkeiten betrachtet, die man auch auf einer Online-Programmierplattform sinnvoll einsetzen kann.

Dazu gehören u. a. folgende allgemeine Kurseinstellungen, Kursbeschreibungs- und Kursformateinstellungen, die auf der Moodle-Plattform vorgenommen werden können:

- Angabe eines vollständigen Kursnamen
- Kursabkürzung
- Kursbereich (z. B. Wirtschaftsinformatik, Technische Informatik, ...)
- Kurssichtbarkeit (z. B. nur sichtbar für Lecturer)
- Kurslaufzeit (mit Beginn und Ende)
- Kurs-ID
- Textuelle Kursbeschreibung
- Kursfoto
- Verborgene Abschnitte

Die Übertragung der genannten Moodle-Einstellungen auf unsere Plattform macht Sinn, um eine angemessene Beschreibung und Klassifizierung der Kurse zu gewährleisten. Anzumerken ist hierbei, dass die Kursichtbarkeit auf einer Lernplattform nicht von Bedeutung ist, falls die Kurse ausschließlich für Studierende konzipiert sind.

Zudem kann die Rolle der alphanumerischen Kurs-ID hier missinterpretiert werden. In Moodle kann man diese für unterschiedliche Zwecke, wie z. B. der Kommunikation mit externen System verwenden. Darüber hinaus, ist die Kurs-ID je nach Einstellung veränderbar. Dies soll und darf in unserem System nicht der Fall sein. Um Kurse eindeutig zu identifizieren, benötigen wir eine unveränderbare und einzigartige Identifikationsnummer.

5. Lernfortschritt
   Innerhalb von Kursen in Moodle kann der Lernfortschritt durch den Lecturer verfolgt werden [MooLern]. Dazu gehört u. a. der Aktivitätsabschluss [MooAkti]. Hier wird es dem Lecturer ermöglicht, auf Grundlage von ihm gestellten Kriterien, Aktivitäten mit einem Häkchen und dementsprechend abgeschlossen zu markieren. Dies kann z. B. durch Anklicken oder Erreichen einer bestimmten Punktzahl im Rahmen der Aufgabe geschehen.

6. Arbeiten mit Dateien [MooData]
   Dateien können auf Moodle u.a. mit der sogenannten „Drag and Drop"-Funktion hochgeladen werden. Dafür muss man eine Datei auf dem lokalen Desktop auswählen und anschließend in den vorgesehenen Bereich in Moodle verschieben. Im LMS selbst können Dateien wiederum mit der „Drag and Drop"-Funktion verschoben werden. Alternativ kann man in Moodle auf eine Datei auch die Verschieben-Funktion anwenden. Dabei wird eine (andere) Rubrik ausgewählt, in welche die Datei verschoben wird. Ebenfalls kann man mehrere Dateien auf einmal oder ZIP-komprimierte auf dem lokalen Desktop auswählen und direkt auf Moodle verschieben.

   Anstatt die „Drag and Drop"-Funktion zu verwenden, kann man in Moodle auch auf den Activity Chooser und File Picker zurückgreifen. Hier kann man über einen Button in der jeweiligen Rubrik den Activity Chooser öffnen. Über den Punkt „Ressourcen" in der Navigationsleiste kann eine neue Datei angelegt werden. Hier muss der Lecturer ein Name für die Datei angeben. Zudem kann er seiner Datei eine Beschreibung hinzufügen, welche je nach Einstellung den Studenten angezeigt werden kann, oder nicht.

   Der File Picker wird nun verwendet, um die hochzuladende Datei auszuwählen. Dabei stehen dem Lecturer mehrere Möglichkeiten zur Auswahl. Er kann die Dateien sowohl von seinem Desktop hochladen als auch aus dem „eigene Dateien"-Bereich in Moodle, einer URL, oder einem Cloud-Speicher (zum Beispiel Google Drive).

   Anschließend können weitere Informationen zur Datei angezeigt oder ausgeblendet werden. Dazu gehören das Upload-/Bearbeitungsdatum, die Dateigröße, und der Dateityp. Außerdem stehen dem Lecturer weitere Funktionen in Bezug auf die Datei zur Verfügung, wie zum Beispiel eine zeitliche Zugriffsbeschränkung.

Um die beschriebene Architektur von Moodle besser zu verstehen, wurde die von Al-Ajlan publizierte Moodle-Grafik aus dem Jahr 2012 [AA12, S. 203] überarbeitet und an die auf Version 3.10 basierte Dokumentation angepasst. Im Fokus des Architekturdiagramms stehen hier die Kurse. Der besseren Übersicht halber wurden daher einige Architekturkomponenten weggelassen oder zusammengefasst dargestellt.

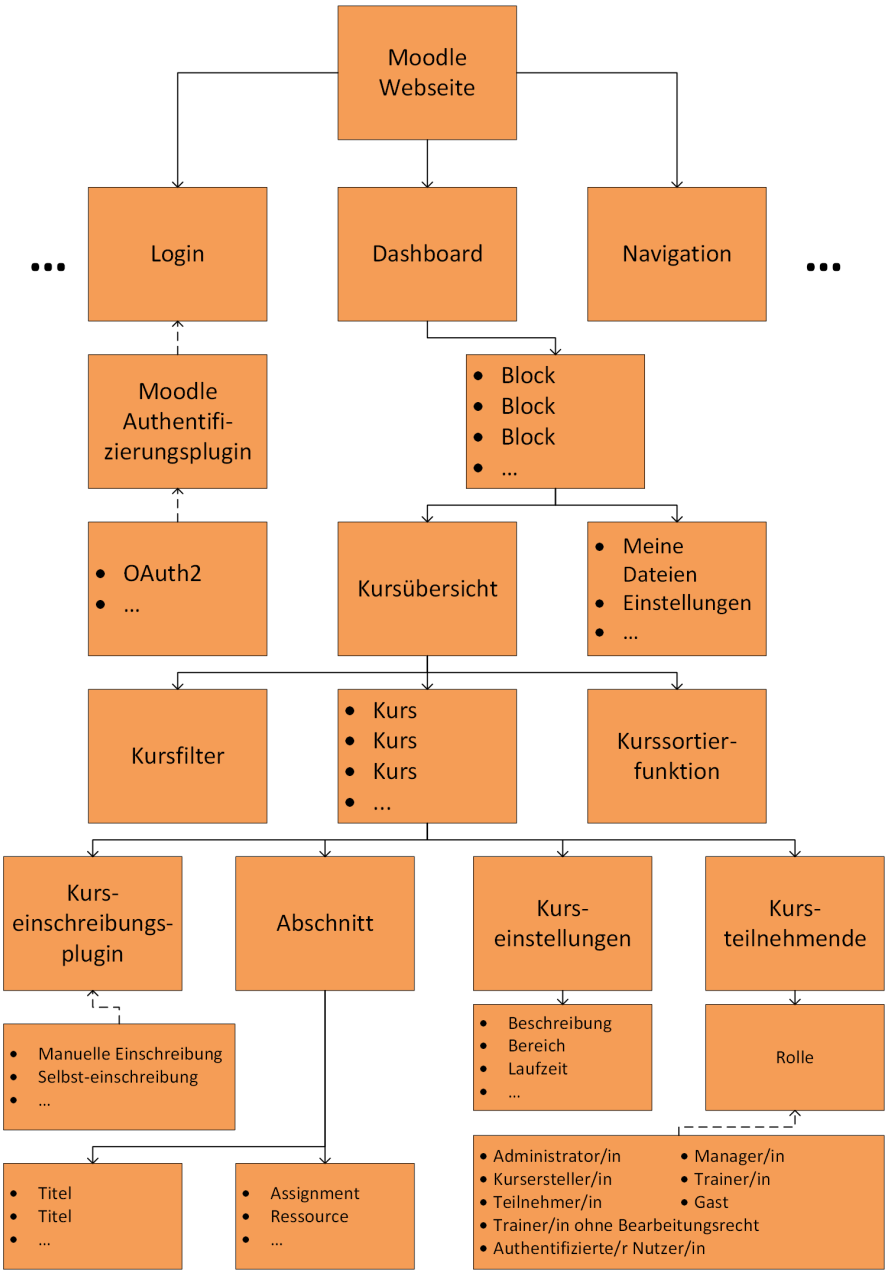

Abbildung 2.1: Moodle Webseitenarchitektur

Dabei wurden zur Erstellung des Diagramms u. a. Prinzipien des Architekturentwurfs nach Ludewig und Lichter befolgt. Danach soll eine Architektur modularisiert und damit in sinnvolle überschaubare Bestandteile, den Komponenten, aufgeteilt werden [LL13, S. 411]. „Die hierarchische Gliederung ist [dabei] ein bewährtes Vorgehen, um [die] Komplexität [von Komponenten] zu reduzieren“ [LL13, S. 419]. Daher wurden zur Beschreibung der Moodle-Webseiten-Architektur die nach Ludewig und Lichter beschriebene Aggregations-, sowie Generalisierungshierarchie verwendet.

Erstere steht für eine Ganzes-Teile-Hierarchie. Hier steht eine Komponente einer anderen stets so gegenüber, dass sie entweder aus dieser besteht oder andersherum ein Teil von ihr ist [LL13, S. 420]. Der Beziehungspfeil sieht dabei wie folgt aus: $\rightarrow$.

Zweitere folgt dagegen dem Ansatz Komponenten nach Merkmalen (Funktionen und Attributen) einzuordnen. Sogenannte spezialisierte Komponenten übernehmen dabei die Merkmale ihrer übergeordneten Komponenten und erweitern diese um weitere Funktionen oder Attribute [LL13, S. 420]. Der Beziehungspfeil hierfür sieht wie folgt aus: $\dashrightarrow$.

2.1.2 Blackboard

Blackboard Learn (im folgenden Blackboard genannt) ist ein LMS, welches Web-Tools nutzt, die das Erledigen von Aufgaben und Kursen in einer sicheren Online-Umgebung ermöglichen. Das LMS wird dabei u. a. im Kontext der Hochschulbildung verwendet [Pat13].

Die Grundlegende Systemarchitektur sieht dabei wie folgt aus. Je nach Betriebssystem kann Blackboard entweder auf einem ISS Web Server (Windows) oder einem Apache Web Server (Linux) betrieben werden [Pat13]. Blackboard selbst basiert dabei auf einer Open-Source-Implementierung der Java Servlets (einfache Java-Klassen die auf eine Request, in der Regel eine HTTP-Request, antworten [OraServ]) und des JavaServer Pages Containers namens Apache Tomcat [BlaTomc]. Der Web Container verwendet hierbei Blackboards Servlets, um dynamisch die Kursseiten inklusive der Kursinhalte und Assignments zu generieren. Letztere Daten werden dabei je nach Implementierung aus einer Microsoft SQL oder Oracle Datenbank geladen. Nebenher wird noch ein Kollaborationsserver betrieben, der die Kommunikation zwischen den Usern ermöglicht [Pat13].

Standardrollen existieren bei diesem LMS deutlich mehr als bei Moodle. Nach eigenen Angaben listet Blackboard über 20 Rollen auf. Letzten Endes ist das Prinzip dabei aber ähnlich wie bei Moodle. Auch hier werden Rollen abhängig vom Kontext vergeben. Unterteilt wird hier in Systemrollen, Kurs- und Organisationsrollen sowie in institutionelle Rollen. Erstere kontrollieren dabei die administrativen Rechte und Funktionen. Zweitere verwalten den Zugriff auf Kursinhalte und -tools. Letztere wiederum legen fest, welche Registerkarten und Module einem User abhängig von ihrer Rolle überhaupt angezeigt werden [BlaRole].

Was jedoch neu ist und im vorherigen Abschnitt noch nicht erwähnt wurde sind statistische Berichte und Auswertungen. Diese sind in Moodle zwar auch vorhanden, allerdings werden dort nur die Nutzeraktivitäten zwischen Nutzer*innen verschiedener Rollen angezeigt, nicht jedoch wie viele User*innen tatsächlich die Moodle-Webseite besucht haben [MooStat].

Blackboard dagegen schlüsselt die Daten genauer auf. Hier können in der Kursaktivitätsübersicht die gesamte und durchschnittlich verbrachte Zeit sowie die Anzahl und Art der Aktivitäten nach Kursteilnehmer*innen und Datum sortiert bzw. angezeigt werden. Dabei kann man die Daten auch nach bestimmten Nutzergruppen filtern [BlaStat].

Darauf aufbauend verdeutlichen die zwei folgenden Studien den Anwendungsbereich dieser Analysetools. Zum einen wird in einer von Blackboard selbst durchgeführten Studie mit über drei Millionen Teilnehmenden ermittelt, wie intensiv verschiedene Kursarten verwendet werden, aufgeschlüsselt nach den genutzten Tools. Mit dem Ergebnis, dass u. a. Kurse mit vielen Assessments von den Studierenden besonders in Anspruch genommen werden [WNHF16, S. 7]. Die Assessments beschreiben dabei vor allem Quiz und Tests [WNHF16, S. 4]. In einer weiteren Studie wird durch Feedback Studierender sowie den Blackboard Analysetools ermittelt inwiefern man das Lernen auf der Plattform verbessern kann. Dabei werden u. a. die Lernzeiten der Studierenden untersucht, mit der Erkenntnis, dass diese vor allem zwischen Mittwoch und Freitag aktiv sind. Zudem wird aus den Studiendaten geschlossen, dass das Einbinden von interaktiven Aktivitäten durch den Lecturer in die Plattform berücksichtigt werden sollte [DJ20, S. 54].

Die mit den Analysetools verbundenen Studienergebnisse zeigen, dass es hilfreich für den Lecturer ist, Nutzerstatistiken zu verwenden, um seine Kursinhalte optimal auf die Studierenden abzustimmen. Der Ansatz interaktive Übungsaufgaben zur Verfügung zu stellen, so wie es auch in der OPPSEE-Plattform geplant ist, hat sich in der Vergangenheit bewährt.

2.1.3 Canvas

Canvas ist ein von Instructure entwickeltes webbasiertes quelloffenes LMS, geschrieben in Ruby, unter der Verwendung des Ruby on Rails Webframeworks [InsGitH]. Es wird von Bildungseinrichtungen und dementsprechend auch Lehrenden sowie Studierenden verwendet, um auf Kursinhalte zuzugreifen sowie diese zu managen [InsCanv].

Dafür umfasst Canvas eine Vielzahl von anpassbaren Werkzeugen zu/zur:

- Kurserstellung, -verwaltung und -analyse
- Benutzeranalyse
- Statistiken
- Kommunikation

Die Inhalte werden dabei nach offiziellen Angaben in der Regel über einen durch Apache und Passenger konfigurierten Linux-betriebenen Webserver bereitgestellt. Zur Persistierung der Daten wird PostgreSQL vorgschlagen. Die Datenbank ist dabei entweder auf dem gleichen Server oder einem separaten Datenbankserver aufzusetzen und bereitzustellen. Zum Cachen der Daten wird idealerweise Redis genutzt, um den vollen Funktionsumfang des LMS auszunutzen [InsProd].

Wie anfangs bereits erwähnt wurde, sind sich die marktführenden LMS bezüglich ihrer Funktionalität sehr ähnlich. Ein gutes Beispiel dafür sind die sogenannten MasteryPaths, ein Feature vom Canvas LMS, was auf den ersten Blick einzigartig erscheint, da Moodle bzw. Blackboard kein Feature unter diesem Namen vertreiben. Hierbei handelt es sich um die Funktion, Kursinhalte abhängig vom Lernfortschritt des Studierenden freizuschalten [InsMast]. Nach genauerer Recherche fällt jedoch auf, dass Moodle und Blackboard diese lediglich unter anderem Namen auflisten [BlaCont][MooVora]. Dessen ungeachtet wurde dieses Feature hauptsächlich deswegen nachgetragen, da es für den Lecturer wie nachfolgend begründet nützlich sein kann.

Durch die Zunahme des Zugangs zur Hochschuldbildung ist es heutzutage schwierig, ein Mittelmaß zu finden um alle Studierende gleichermaßen effektiv zu fördern. Daher kann der Lecturer mit differenziertem Lernen sicherstellen, dass sowohl schwächelnde Studierende Zugriff auf Übungsaufgaben entsprechend ihren Fähigkeiten bekommen, als auch, dass sich fortgeschrittene Studierende mit weiterführenden Aufgaben fortbilden können [CBCO+11, S. 4].

2.2 Existierende Programmierplattformen im deutschsprachigen Raum

In diesem Abschnitt werden bereits existierende Lösungen für die Programmierlehre aus dem deutschsprachigen Raum untersucht. Dabei liegt der Fokus darauf, ob und inwiefern sich der Aufbau der Lehrendenkomponente auf diesen Plattformen im Vergleich zu den traditionellen LMS unterscheidet. Idealerweise sollen hier demnach noch nicht erfasste aber hilfreiche Funktionen für den Lecturer erfasst werden.

Untersucht werden dabei drei Systeme aus dem deutschsprachigen Raum. ArTEMiS der Technischen Universität München (TUM), Jack der Universität Duisburg-Essen (UDE) sowie Codeboard der Eidgenössischen Technischen Hochschule (ETH) Zürich.

2.2.1 ArTEMiS

ArTEMiS steht für AuTomated assEssment Management System for interactive learning und stellt ein LMS, das auf eine sofortige und automatische Codeevaluierung ausgerichtet ist, um Studierenden das iterative Lösen von Programmieraufgaben zu ermöglichen. ArTEMiS bietet dafür einen programmiersprachenunabhängigen Online-Code-Editor mit interaktiven Übungsanweisungen [KS18, S. 284]. Das LMS ist quelloffen und unter der MIT-Lizenz auf GitHub veröffentlicht [TUMLice].

Nachfolgend wird das System Design genauer betrachtet. ArTEMiS ist zwar nicht ausschließlich wie eine Übungsplattform aufgebaut, da dieses LMS ebenfalls das Durchführen von Klausuren ermöglicht [TUMInst]. Dennoch kann das System Design als Anhaltspunkt dienen, um die Architektur für die Lecturer-Komponente der OPPSEE-Plattform zu erstellen.

Das Top-Level-Design folgt dabei der Client-Server Architektur. Ein Application Client (eine Angular Webanwendung) greift hier über die ArTEMiS API auf die bereitgestellten Schnittstellen des Application Servers (ein Backend auf Basis von Spring Boot) zu [TUMSyst]. Letzterer dient dabei als sogenannte Fassade für den Client. Dies ist ein Entwurfsmuster, welches nützlich ist, um eine vereinfachte Schnittstelle zu einem komplexen Teilsystem anzubieten [GKRS17, S. 90].

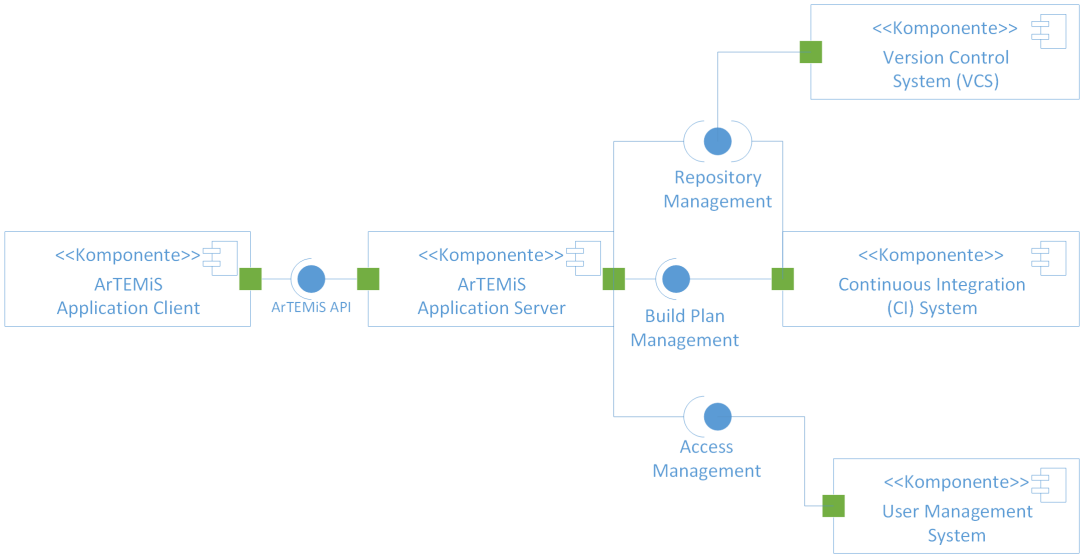

Abbildung 2.2: ArTEMiS Top Level Design [TUMSyst]

In diesem Fall verwaltet der Application Server alle Anfragen des Clients und greift je nach Anwendungsfall über Schnittstellen auf die benötigten Subsysteme zu. Im Falle der Authentifizierung verbindet er sich mit dem User Management System, für Programmieraufgaben dagegen verbindet er sich mit dem Version Control System (VCS) und dem Continuous Integration (CI) System [TUMSyst]. In Abbildung 2.2 wird dies grafisch mittels eines Unified Modeling Language (UML)-Komponentendiagramms veranschaulicht.

Da ArTEMiS sich in erster Linie durch die Programmieraufgabenbereitstellung von den anderen LMS abhebt und der Workflow auch für die Konzeption der Lecturer-Komponente in einer Übungsplattform von Interesse ist, wird dieser im Folgenden betrachtet [KS18, S. 286]:

1. Zunächst bereitet der Lecturer die Aufgabe vor:
   Ein Repository wird eingerichtet, welches den Übungscode und die Testfälle enthält. Darüber hinaus muss der Build Plan auf dem Continious Integration (CI) Server eingerichtet, und eine Übung in ArTEMiS erstellt werden

2. Der Studierende startet die Übung:
   Der/die Student*in muss die Übung in ArTEMiS starten. Dies triggert das Kopieren (forken) des Repositorys und Build Plans. ArTEMiS setzt die Berechtigungen dabei so, dass die Studierenden nur ihr eigenes Repository sehen können.

3. Optional kann das Repository auf den lokalen Rechner geklont werden.
4. Der Studierende kann die Aufgabe entweder auf dem lokalen Rechner oder in der Online-Entwicklungsumgebung lösen
5. Der Studierende committet und pusht seine Lösung über das VCS zum Remote-Server oder klickt auf „Commit & Run Tests“ im Online-Editor
6. Der CI-Server überprüft die Lösung
7. Der Lecturer sowie Studierende für ihre jeweils eigene Lösung können die Ergebnisse einsehen

Ein entscheidender Vorteil von ArTEMiS gegenüber den vorherigen Lösungen ist somit, dass der Lecturer durch das Versionskontrollsystem (VCS) einen genauen Einblick über den Entwicklungsprozess des Programms eines jeden einzelnen Studenten hat. Im Idealfall lernt der/die Student*in seine/ihre Lösungen möglichst häufig über das VCS zu pushen, sodass der Fortschritt innerhalb einer Aufgabe iterativ verfolgt werden kann.

Zu guter Letzt wird die Abbildung 2.3 betrachtet, welche einen Ausschnitt des Datenbankmodells beschreibt. Da ArTEMiS mit MySQL eine quelloffene relationale Datenbank verwendet [TUMSyst][OraGitH] kann dieses Diagramm als Grundlage dafür dienen unser eigenes Datenbankmodell zu erstellen. Weggelassen wurden hier u. a. Datenbanktabellen im Zusammenhang mit der Klausurdurchführung. Diese sind für den weiteren Verlauf dieser Arbeit nicht relevant.

Im Folgenden wird aus der Sicht des Lecturers evaluiert, inwiefern von ihm verwaltete Kurse und Aufgaben auf Basis eines solchen Datenbankmodells abgebildet werden. Nach Notation des Modells kann der Lecturer einem Kurs mehrere Aufgaben zuordnen. Aufgaben wiederum werden durch eine Aufgabenart repräsentiert. Hier, im Rahmen dieser Arbeit, soll der Fokus auf den Programmieraufgaben liegen. Die Aufgaben des Lecturers können nun von Teams aus Studierenden bearbeitet werden. Die Teilnahme an einer Aufgabe wird dabei gesondert in der Participation-Tabelle registriert. Diese enthält u. a. eine URL, die auf das zugehörige Repository der Aufgabe verweist. Studierende können nun bis zu einem gesetzten Ablaufdatum der Aufgabe mehrmals an dieser teilnehmen oder eine Lösung einreichen. Je nach Aufgabentyp können die Aufgaben automatisch, halbautomatisch oder manuell bewertet werden. Zusätzlich zu jeder Auswertung einer Aufgabe, kann der Lecturer ein Feedback verfassen bzw. eine Beschwerde über die Auswertung erhalten. Auf die Beschwerde kann er wiederum antworten.

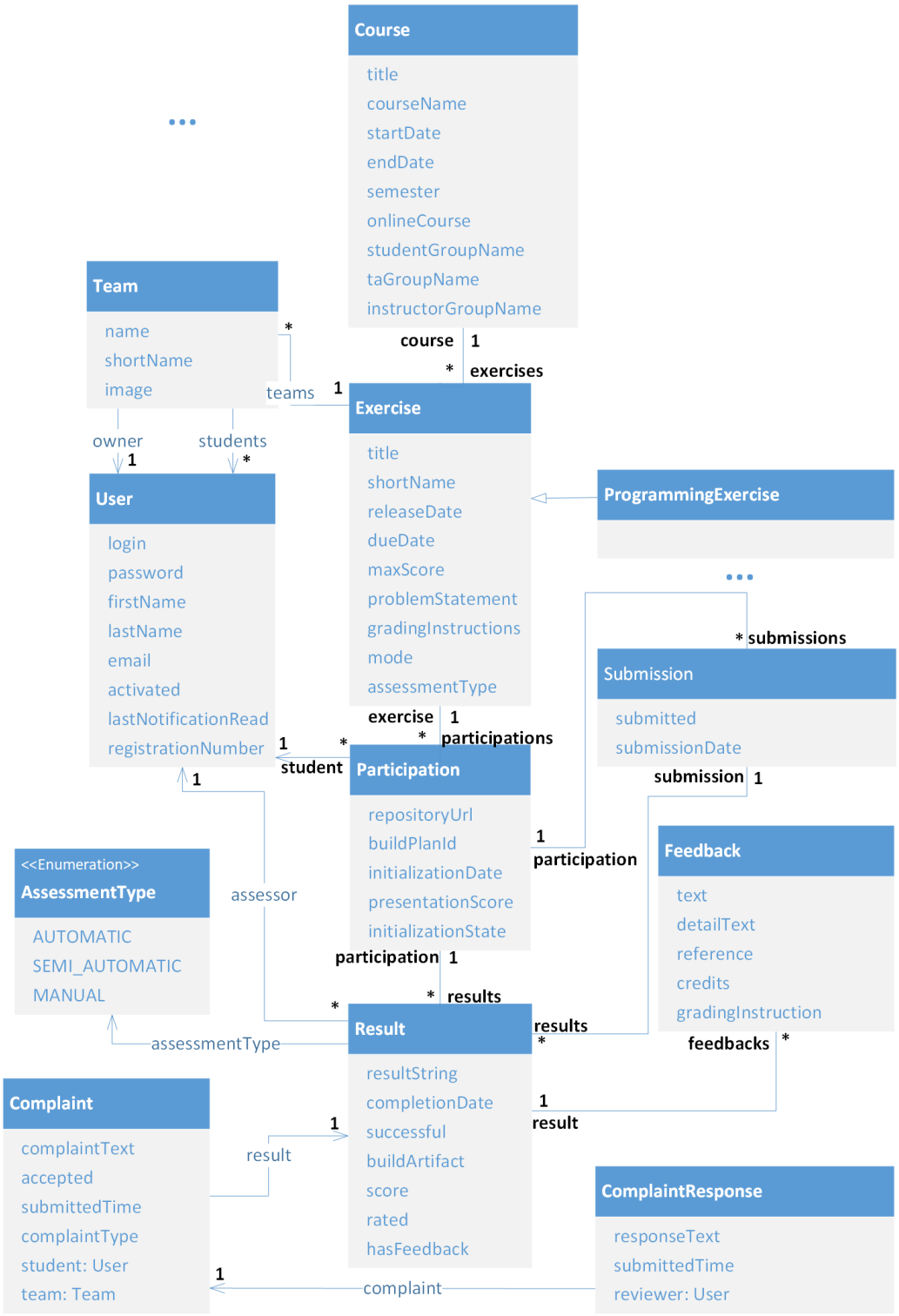

Abbildung 2.3: ArTEMiS Datenbankmodell [TUMSyst]

Ein Vorteil, der hieraus gegenüber den populären LMS hervorgeht ist, dass jede Aufgabe einer bestimmten Repository-URL zugeordnet werden kann. Damit besteht für den Lecturer die Möglichkeit Aufgabenprogrammiercode über ein VCS zu bearbeiten [TUMExer]. Nach diesem Modell und damit in ArTEMiS leider nicht möglich ist jedoch das Erstellen von Aufgaben auf Basis von bereits bestehenden Repositories. Zudem kann jede Aufgabe nur maximal in einem Kurs vorhanden sein, was die 1-zu-n-Beziehung in Abbildung 2.3 zwischen der Course- und Exercise-Tabelle verdeutlicht. Eine solche Umsetzung verhindert das Anlegen einer Aufgabensammlung innerhalb der Plattform, auf die alle Lecturer zugreifen können.

2.2.2 Jack

Jack ist ein server-basiertes System, welches auch die Durchführung computergestützter Übungen und Prüfungen mit automatischer Bewertung und Feedback-Generierung ermöglicht. Es kann ebenfalls in Moodle (und weiteren LMS) als externes Tool über den LTI-Standard integriert werden [JacELMS].

Das System ist so aufgebaut, dass es in Paketen auf einem Java Enterprise Edition (EE) Server, der mit einem separaten Datenbanksystem verbunden ist, bereitgestellt werden kann [GSB08, S. 3]. Ein Java EE Server ist dabei eine Serveranwendung, welche die APIs der Java EE Plattform implementiert und die Standard-Java-EE-Dienste zur Verfügung stellt. Java EE Server ermöglichen es Applikationsdaten dem Client zu übermitteln. Ähnlich, wie Webserver, die Webseiten dem Webbrowser bereitstellen. Daher werden Java EE Server manchmal auch als Applikationsserver bezeichnet [Ian12, S. 16].

Jack unterstützt ausschließlich die Programmiersprache Java [GSB08, S. 1] und verwendet wie die bisher betrachteten LMS ebenfalls eine relationale Datenbank, um u. a. auf bewährte Mechanismen für Datenbanktransaktionen zu vertrauen und Datenverlust bei der Durchführung von Klausuren zu verhindern [GSB08, S. 4].

Über das Frontend des Systems können Lecturer oder Studierende ein Webinterface nutzen, um mit dem System zu interagieren. Im Backend laufen sogenannte Checker-Komponenten, die eingereichte Lösungen aus der Datenbank auslesen und diese Bewerten [GSB08, S. 3].

Für den Studierenden gibt es in Jack neben einem Klausuren-Modus auch ein Selbsttrainingsmodus. Bei Letzterem kann der Studierende unbegrenzt viele Lösungen einreichen, ohne zeitliche Beschränkung. Kompliziertere Aufgaben muss der Studierende dabei jedoch herunterladen und nach Bearbeitung zum Einreichen der Lösung wieder hochladen.

Über die bisher genannten Funktionen hinaus lassen sich in Jack u. a. Kurse anlegen, Lösungsstatistiken anzeigen und Aufgaben über Dropbox speichern/teilen [JacHome].

Zusammenfassend hebt sich Jack von den bisher aufgeführten LMS vor allem dadurch ab, dass es einen Selbsttrainingsmodus explizit für Programmieraufgaben anbietet. Im Rahmen der Nutzung des Systems wurde nämlich erkannt, dass Studierende sich bei Verwendung des Selbsttrainingsmodus zum Erarbeiten der Lösungen hauptsächlich untereinander ausgetauscht oder das Internet zur Hilfe genommen haben. Durch die vorwiegend selbständige Arbeit mussten Lecturer maximal einige Minuten pro Tag aufwenden um Aufgaben manuell auszuwerten oder Lösungswege zu erklären [GSB08, S. 20]. Die Möglichkeit, dass die für Studierende bereitgestellten Aufgaben auch noch geteilt und somit unter den Lecturern wiederverwendet werden können erleichtert den zeitlichen Aufwand für den Lecturer zusätzlich.

2.2.3 Codeboard

Codeboard ist eine webbasierte IDE die es Lecturern erlaubt unvollständige Template-Projekte u. a. in C, C++, Eiffel, Haskell, Java und Python zu erstellen. Die sogenannten Projekte können je nach Bedarf öffentlich über einen Link verfügbar oder auf bestimmte Nutzer*innen begrenzt werden. Ist die Einreich-Funktion für Aufgaben vom Lecturer eingeschaltet, haben Studierende die Möglichkeit nach erfolgreichem Compilieren ihres Programmiercodes ihre Lösung über einen Submit-Button einzureichen. An dieser Stelle ermöglicht Codeboard das automatische Bewerten der Programme mit Hilfe von Tests, die nach Bedarf vom Lecturer vor dem/der Nutzer/in versteckt werden können [CodProj].

Im Gegensatz zu den vorherigen Programmierplattformen bietet Codeboard detailliertere Statistiken über die Nutzung der Aufgaben an. Neben der Anzahl von Zugriffen können auch Kompilierungen, Ausführungen und Abgaben eines Projekts eingesehen werden [CodProj].

3 Anforderungsermittlung

Nachdem die Funktionen bestehender Lernplattformen ausführlich analysiert worden sind, widmet sich dieses Kapitel den daraus resultierenden Anforderungen für den Lecturer. Zum Erfassen dieser wird sich an den Grundlagen des Requirements Engineerings (RE) nach Pohl und Rupp [PR15] orientiert.

Dafür wird zunächst das System sowie der Systemkontext genau abgegrenzt. Nun können die Stakeholder inklusive ihrer Bedeutung im Systemkontext ermittelt werden. Anschließend sind die Anforderungen der Übersichtlichkeit halber nach funktionalen Anforderungen, Qualitätsanforderungen und Randbedingungen zu unterteilen, bevor diese grafisch mit Hilfe von Anwendungsfällen, sogenannter UML-Use-Case-Diagramme, visuell aufbereitet werden.

Eine Anforderung an sich ist nach IEEE Standards [IEE90, S. 62] (übersetzt aus dem Englischen) dabei wie folgt definiert:

1. „Eine Bedingung oder Fähigkeit die ein Benutzer benötigt, um ein Problem zu lösen oder ein Ziel zu erreichen“.
2. „Eine erfüllte Bedingung oder Fähigkeit, die ein System oder eine Systemkomponente einhalten oder besitzen muss, um einen Vertrag, eine Norm, eine Spezifikation oder ein anderes formal vorgeschriebenes Dokument zu erfüllen“.
3. „Eine dokumentierte Darstellung einer Bedingung oder Fähigkeit wie in (1) oder (2)“.

3.1 Rolle des Requirements Engineering (RE)

Studien belegen, dass 60% der Fehler schon im RE auftreten [PR15, S. 1][Boe81]. Aus quantitativen Studien der Vergangenheit geht außerdem hervor, dass bereits 40% der Zeit in die Entwicklung eines Softwareprodukts für Programm- und Programmierungsspezifikationen entrichtet werden [Boe84, S. 9].

Eine erfolgreiche Systementwicklung setzt deswegen ein leistungsfähiges Requirements Engineering (RE) mit einer vollständigen, fehlerfreien, unmissverständlichen Anforderungsermittlung voraus. Dadurch können im Nachhinein langwierige Änderungsprozesse vermieden werden [PR15, S. 2-3].

3.2 System und Systemkontext

Im Rahmen des Systems und Systemkontextes müssen wir zwischen zwei Abgrenzungsprozessen unterscheiden [PR15, S. 15]:

1. Systemabgrenzung
2. Kontextabgrenzung

Erstere bestimmt, welche Aspekte durch das geplante System abgedeckt werden bzw. Teil der Umgebung dieses Systems sind. Zweitere bestimmt die Grenze zur irrelevanten Umgebung. Dies wird durch das Analysieren der Beziehungen zwischen den Aspekten und der Umgebung des geplanten Systems erreicht.

Um diese beiden Grenzen zu veranschaulichen, wird/werden im vereinfachten UML-Komponentendiagramm der OPPSEE-Plattform [HBPS21, S. 4] die fehlende/n Komponente/n für die Lehrendenkomponente in Abbildung 3.1 ergänzt und eine gestrichelte Linie zur Abgrenzung des Kontextes eingezeichnet.

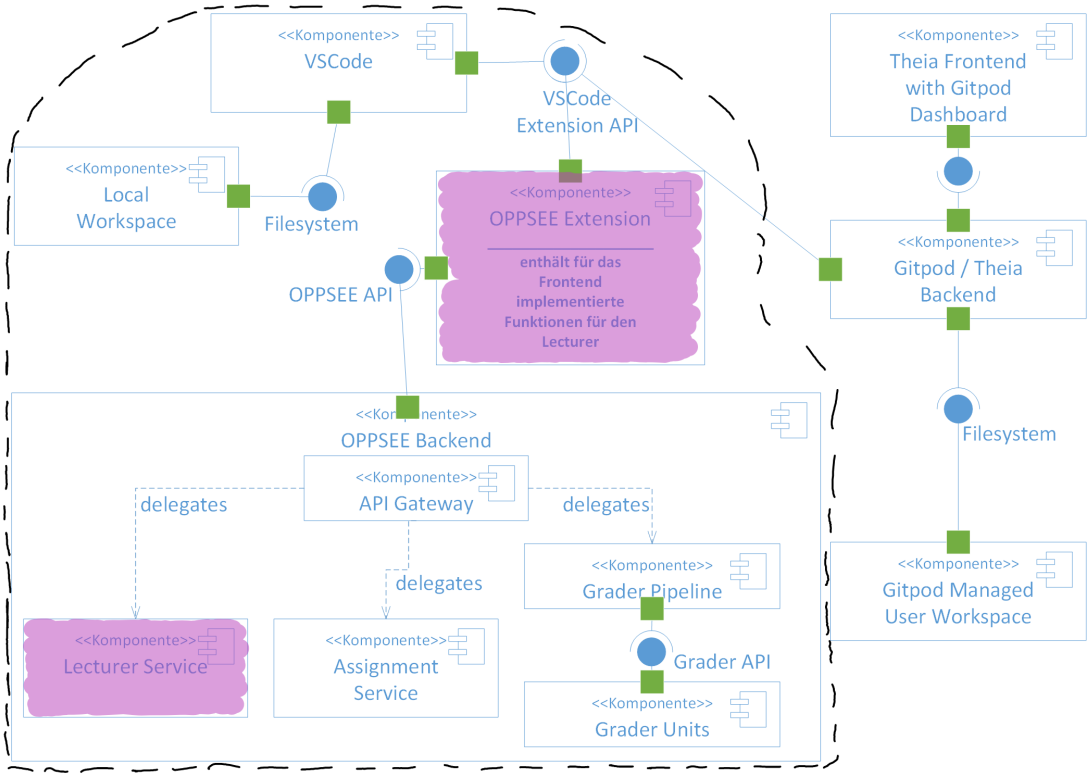

Abbildung 3.1: Vereinfachtes UML-Komponentendiagramm für die OPPSEE-Plattform

Die Systemabgrenzung wird in Abbildung 3.1 grafisch durch die (in Pink) markierten Komponenten veranschaulicht. Sie beschreibt jenes (Teil-)System, welches im Rahmen dieser Arbeit konzipiert wird.

Anzumerken ist, dass in Abbildung 3.1 die Lehrenden-Komponente als Teil der OPPSEE Extension integriert wird. Alternativ könnte auch eine zweite unabhängige Lecturer-Extension entwickelt werden, welche ausschließlich die Lecturer-Funktionen umsetzt. Parallel zur OPPSEE Extension, könnte die Lecturer-Extension dann über die VSCode Extension API in das System integriert werden, und über die OPPSEE API auf das OPPSEE Backend zugreifen. Diese Möglichkeit erleichtert das Weiterentwickeln des OPPSEE-Frontends ungemein, da so im Rahmen der Extension neue Funktionen entwickelt und getestet werden können, ohne dass eine Cloud-Infrastruktur auf dem lokalen Rechnern bereitgestellt werden muss [HBPS21, S. 5].

Darüber hinaus ermöglicht ein solcher Ansatz das Befolgen der Microservices-Architektur. Diese besagt, dass die Grundlage darin besteht, eine einzelne Anwendung, in diesem Fall unser Frontend, als eine Reihe kleiner und unabhängiger Dienste zu entwickeln, die in ihren eigenen Prozessen laufen und unabhängig voneinander entwickelt sowie bereitgestellt werden [IS18, S. 7].

Die zweite in Abbildung 3.1 markierte Komponente beschreibt einen Lecturer Service, der in dieser Arbeit beispielhaft als Mockup (englisch für Vorführmodell) konzipiert und implementiert wird. Um die unterschiedlichen Berechtigungen und Workflows zwischen Lecturern und anderen Stakeholdern zu verdeutlichen, wird in diesem Service ein Authentifizierungsservice integriert. Um die soeben beschriebene Microservices-Architektur zu befolgen, sollte in einer produktiven Umgebung der Authentifizierungsservice vom Lecturer Service getrennt werden.

Die durch die gestrichelte Linie gewählte Kontextabgrenzung in Abbildung 3.1 ist hier so gewählt, weil...

1. die VSCode-Umgebung essenziell ist, um unsere Lecturer Extension auszuführen.
2. der Local Workspace benötigt wird, um auf bzw. mit Assignments zu arbeiten (z. B. um diese zu klonen).
3. neben dem Lecturer Service auch der Assignment Service und die Grader API im Frontend aufgerufen werden muss, um dem Lecturer ebenfalls Daten bezüglich der Auswertung von/oder Assignments anzuzeigen.
4. Gitpod- und Theia-spezifische Komponenten für diese prototypischen Konzipierung irrelevant sind, da sie für das reine Betreiben und Testen der Lecturer-Funktionen nicht benötigt werden.

3.3 Stakeholderanalyse

Stakeholder sind eine essenzielle Quelle zur Identifikation möglicher Anforderungen an das System. Unberücksichtigte Stakeholder können durch die daraus folgende beeinträchtigte Anforderungsermittlung zu signifikant negativen Auswirkungen auf den gesamten Projektverlauf führen [PR15, S. 22].

Die verschiedenen Stakeholder*innen der Lehrendenkomponente können u. a. durch die Analyse der benötigten Berechtigungen in diesem System abgeleitet werden. Aus den bereits aufgelisteten Rollen des Moodle-Abschnitts im Related Work Kapitel geht hervor, dass dazu Administratoren/Administratorinnen gehören, die das System verwalten, der Lecturer selbst natürlich, aber auch wissenschaftliche Mitarbeiter*innen, Tutoren/Tutorinnen oder gar Studierende, die eine verwaltende Rolle im Kurskontext einnehmen.

Die Gruppe aus wissenschaftlichen Mitarbeiter*innen, Tutoren/Tutorinnen und Kursverwaltenden Studierenden kann dabei zum Stakeholder „Teacher“ zusammengefasst werden. Dies bezeichnet eine Gruppe die im Kurskontext erweiterte und dem regulären Studierenden übergeordnete Rechte besitzt, aber im Zweifel (nicht zwingend) weniger Rechte als der Lecturer selbst.

Auch reguläre Studenten haben ein indirektes Interesse an der Lehrendenkomponente. Sie erwarten, dass sie unkompliziert auf die bereitgestellten Inhalte des Lecturers zugreifen können. Darüber hinaus sollte eine Übungsplattform den Anspruch erheben, dass der Lecturer über die Plattform Studierende mit ausreichend Hilfestellung für die erfolgreiche Bearbeitung der Assignments versorgen kann.

Neben den bisher identifizierten Stakeholdern, wurden nach Gandraß, Hinrichs und Schmolitzky der Platform sowie Assignment Developer identifiziert [GHS20, S. 30]. Der Platform Developer mag Interesse daran haben, inwiefern die Lehrendenkomponente implementiert wurde, damit dieser auftretende Fehler im Zweifel leichter identifizieren und beheben kann. Der Assignment Developer wird dagegen vor allem daran interessiert sein, inwiefern er neue Assignments über die Lehrendenkomponente erstellen kann.

Wichtig ist anzumerken, dass die Rollen der Stakeholder scharf getrennt werden, und sich nicht überschneiden sollen. Ein Lecturer kann daher z. B. nicht gleichzeitig auch ein Administrator sein. Allerdings ist es durchaus möglich, dass eine Person, zu verschiedenen Zeiten, verschiedene Stakeholder verkörpert. Unter diesen Voraussetzungen werden die Stakeholder folgend nach ihrer Bedeutung bezüglich für die Lehrendenkomponente sortiert.

1. Lecturer
   Der Lehrende (Lecturer) steht bei der Entwicklung einer Lehrendenkomponente im Mittelpunkt. Ziel ist es ihm ein umfangreiches Werkzeug zur Verwaltung von Kursen und Assignments bereitzustellen. Der Lecturer wird daher maßgeblich das Aufstellen der funktionalen Anforderungen beeinflussen.

2. Teacher
   Teacher sind nahezu genauso wichtig, wie der Lecturer. Sie führen ähnliche Aufgaben wie der Lecturer im Bereich der Verwaltung von Kursen und Assignments aus bzw. übernehmen diese für den Lecturer sogar.

3. Assignment Developer
   Das Erstellen von Assignments ist essenziell dafür, dass der Lecturer seinen Studierenden diese überhaupt bereitstellen kann. Damit nimmt der Assignment Developer ebenfalls eine wichtige Rolle ein und fordert dabei gleichzeitig vom System nicht nur unkomplizierte Wege zur Contenterstellung, sondern auch zur Bearbeitung und Integration in die Lehrendenkomponente.

4. Studierender
   Der/die Studierende ist insofern wichtig, da der bereitgestellte Lehrinhalt letzten Endes genau auf diese/n ausgerichtet werden muss.

5. Administrator/in
   Administratoren/Administratorinnen verwalten hauptsächlich die Einstellung und Einhaltung der für User zugeordneten Rollen. Im Kurs- und Assignmentkontext spielen sie dagegen eine untergeordnete Figur.

6. Platform Developer
   Plattform Developer haben im Kurs- und Assignmentkontext gar keine Rolle. Sie sind daran interessiert in welche Qualität und auf welche Art geforderte Funktionen umgesetzt werden. Daher beeinflussen sie vor allem nicht-funktionale Anforderungen.

3.4 Anforderungsübersicht

Nach Pohl und Rupp wird zwischen 3 verschiedenen Anforderungsarten unterschieden. Diese sind dabei wie folgt definiert:

1. Funktionale Anforderungen
   „Eine funktionale Anforderung ist eine Anforderung bezüglich des Ergebnisses eines Verhaltens, das von einer Funktion des Systems bereitgestellt werden soll.“ [PR15, S. 8]

2. Qualitätsanforderungen (auch nicht-funktionale Anforderungen genannt)
   „Eine Qualitätsanforderung ist eine Anforderung, die sich auf ein Qualitätsmerkmal bezieht, das nicht durch funktionale Anforderungen abgedeckt wird.“ [PR15, S. 9]

3. Randbedingungen
   „Eine Randbedingung ist eine Anforderung, die den Lösungsraum jenseits dessen einschränkt, was notwendig ist, um die funktionalen Anforderungen und die Qualitätsanforderungen zu erfüllen.“ [PR15, S. 9]

Funktionale Anforderungen beschreiben also, welche Funktionalität ein System bereitstellen soll [PR15, S. 8]. Ein Beispiel für eine funktionale Anforderung ist es, sich mit Email und Passwort anmelden zu können.

Nicht-funktionale Anforderungen dagegen geben an, welche qualitativen Ansprüche an das System erhoben werden. Damit ist gemeint, wie (gut) Funktionen und Teile des Systems entwickelt und implementiert werden. Beispiele hierfür sind Performanz, Verfügbarkeit, Zuverlässigkeit, Skalierbarkeit oder Portabilität [PR15, S. 9].

Randbedingungen beschreiben unter welchen Voraussetzungen bzw. Bedingungen funktionale und nicht-funktionale Anforderungen umgesetzt werden können. Sie schränken somit den Lösungsraum des Entwicklungsprozesses ein [PR15, S. 9]. Das kann zum Beispiel ein System, wie beispielsweise VSCode sein, in das eine Lösung integriert werden muss.

Nachfolgend werden die für diese Arbeit identifizierten Anforderungen nach den soeben aufgelisteten Anforderungsarten tabellarisch (Tabelle 3.1 und 3.2) dargestellt. Grundlage der Erfassung dieser Anforderungen bilden dabei die bisherigen Erkenntnisse, maßgeblich aus dem Related Work Kapitel.

Funktionale Anforderungen
Als User*in möchte ich mich über einen HAW-Dienst auf der Plattform anmelden können.
Als User*in möchte ich Kurse nach verschiedenen Parametern filtern können (z. B. nach „zuletzt besucht“).
Als User*in möchte ich Kursen persönliche Präferenzen zuordnen können (markieren, verstecken etc.).
Als User*in möchte ich zwischen verschiedenen Sprachen (z.B. Deutsch / Englisch) wechseln können.
Als Lecturer möchte ich Zugriffsrechte für meine Kurse bearbeiten können.
Als Lecturer möchte ich Studierenden eine unkomplizierte Möglichkeit vorstellen, sich für Kurse einzuschreiben (z. B. Selbsteinschreibung durch Passwort).
Als Lecturer möchte ich User unkompliziert (z. B. über eine CSV-Datei) selbst in Kurse einschreiben können.
Als Lecturer möchte ich Assignments einen Bearbeitungszeitraum zuweisen können.
Als Lecturer möchte ich auf eine Sammlung von Assignments zugreifen können und diese ggf. meinem Kurs hinzufügen können.
Als Lecturer möchte ich Kurse anlegen können.
Als Lecturer / Teacher möchte ich Kurstitel und -beschreibung anlegen bzw. bearbeiten können.
Als Lecturer / Teacher möchte ich Kurse Bereichen zuordnen können.
Als Lecturer / Teacher möchte ich Tipps zum Lösen von Assignments einstellen können.
Als Lecturer / Teacher möchte ich Assignments abhängig vom Fortschritt des jeweiligen Kursteilnehmers freischalten können.
Als Lecturer / Teacher möchte ich mir Kursstatistiken (z. B. Kursaktivität) anzeigen lassen können.
Als Lecturer / Teacher möchte ich eine Auswertungsübersicht der Assignments in einem Kurs angezeigt bekommen können.
Als Lecturer / Assignment Developer möchte ich Assignments auf Basis von Git Repos anlegen.
Als Studierender / Assignment Developer möchte ich Zugriff auf Git-Ressourcen haben.
Als Studierender kann ich Hinweise auf fehlerhafte Auswertungen geben.
Als Administrator möchte ich Kurse löschen können.

Tabelle 3.1: Funktionale Anforderungen

Nicht-funktionale Anforderungen
Eine Token-basierte Authentifizierungsmethode soll verwendet werden.
Die verwendete Programmiersprache sollte typisiert sein.
Studierenden sollten Lecturer-Funktionen, die sie nicht ausführen können auch nicht angezeigt werden.
Das Icon-Set von VSCode namens „codicons“ [VSCicon] soll im Frontend verwendet werden.
Das Session-Interface sollte an das HAW-Interface angelehnt werden (30 Minuten Sessiondauer, mit Option auf Verlängerung durch einen Refresh-Button).
Neben einem Access-Token muss intern auch ein Refresh-Token verwendet werden.
Die VSCode Extension Guidelines sollten weitestgehend eingehalten werden.
Die Webviews im Frontend sollten responsive sein.
Eine quelloffene relationale Datenbank sollte zur Persistenz der Daten verwendet werden.
Kurs- und Assignmentbeschreibung sollten nach Eingabe korrekt formatiert werden.
Daten sollten serverseitig nach Coordinated Universal Time (UTC) abgespeichert werden. Der Client dagegen soll die lokale Zeit angegeben bekommen.
Der Zeitaufwand des Lecturers für das Anbieten seiner Kurse als freiwilliges Zusatzangebot sollte so gering wie möglich gehalten werden.
Front- und Backend sollten einer klaren Ordnerstruktur folgen.
Eine umfangreiche Dokumentation zum Aufsetzen / Arbeiten / Verwenden des Projekts sollte vorhanden sein.
Eine hohe Testabdeckung in Form von API-Tests sollte die Funktionalität, Zuverlässigkeit, Leistung und Sicherheit des Backends überprüfen.
Eine hohe Testabdeckung in Form von Unit- und Intergrationstests sollte die Funktionalität, Zuverlässigkeit, Leistung und Sicherheit des Frontends überprüfen.
Daten sollte das System validieren und bereinigen, bevor diese verarbeitet werden.
Eine Dokumentation für die API-Endpunkte sollten vorhanden sein.
Ein Framework sollte zur Entwicklung des Frontends verwendet werden.
Wichtige Informationen oder Fehler sollten über die VSCode Extension API durch die bereits in VSCode integrierten Informations- bzw. Fehlermeldungen dargestellt werden.

Tabelle 3.2: Nicht-funktionale Anforderungen

Zusätzlich zu den funktionalen und nicht-funktionalen Anforderungen gibt es noch einige Randbedingungen, die uns in der Entwicklung unserer Lehrendenkomponente beschränken. Dazu gehört zum einen, dass das Frontend in VSCode zu integrieren ist. Daher muss die VSCode Extension API zum Implementieren der Lecturer-spezifischen Funktionen verwendet werden. Weil die Extension API auf der Programmiersprache JavaScript (JS) basiert [VSCAPI], sind auch die Webviews in JS oder TypeScript (TS), einer stark typisieren auf JS basierenden Programmiersprache [MTScrip], zu entwickeln. Ebenso können Frameworks eingesetzt werden, die eine dieser beiden Programmiersprachen unterstützen. Als Authentifizierungsmöglichkeit steht zudem bereits der in Abschnitt 2.1.1 erwähnte GitLab-Server der HAW Hamburg als OAuth2 Authentifizierungsprovider bereit.

3.5 Anwendungsfälle

Anwendungsfälle (auch Use Cases genannt) können entweder durch ein UML-Diagramm modelliert [PR15, S. 22] oder natürlichsprachlich formuliert werden [LL13, S. 387].

Unabhängig davon lauten die wesentlichen Merkmale nach Ludewig und Lichter [LL13, S. 387] wie folgt:

- Neben dem zum entwickelnden System ist immer mindestens ein Akteur (Actor) am Anwendungsfall beteiligt. Ein Akteur wird in Rolle eines Benutzers oder externen Systems dargestellt, wie z. B. durch einen der identifizierten Stakeholder*innen. Anzumerken ist, dass Akteure nie selbst zum System gehören, sondern als Außenstehende mit dem System interagieren.
- Ein Anwendungsfall wird durch ein spezielles Ereignis, einem sogenannten Trigger, vom Hauptakteur ausgelöst.
- Ein Anwendungsfall verfolgt vom Hauptakteur ausgehend immer ein gewisses Ziel.
- Alle Interaktionen zwischen dem System und seinen beteiligten Akteuren, die für das Erreichen oder Verfehlen des Ziels notwendig sind, müssen erfasst werden.
- Ein Anwendungsfall endet, wenn sein Ziel erreicht oder endgültig verfehlt wurde.

In unserem Fall stellen die Anwendungsfälle typische Abläufe dar, die ein/e User*in innerhalb der VSCode Extension tätigt. Mit den bisherigen Anforderungen als Grundlage bieten uns die Anwendungsfälle eine erweiterte Sicht auf die externen Ansprüche an das System, indem neben einem detaillierten Normalablauf auch Vor- und Nachbedingungen sowie Sonderfälle berücksichtigt werden.

3.5.1 Authentifizierungsprozess

Zunächst wird nun der Authentifizierungsprozess für die Lehrendenkomponente betrachtet. Dieser muss stets durchlaufen werden, um auf weitere Funktionen der Lehrendenkomponente zugreifen zu können. Dabei wird wie bereits beschrieben auf einen Authentifizierungsprovider zurückgegriffen.

Name: Authentifizieren
Ziel: Der/die Userin möchte Zugang zur Lehrendenkomponente der OPPSEE-Plattform erhalten.
Akteure: User/in (Lecturer / Teacher / Studierender / Assignment Developer / Administrator/in), die VSCode Extension als Frontend, das Backend-Mockup

Vorbedingungen:

- Der/die User/in hat eine funktionierende Internetverbindung.
- Der/die User/in verfügt über gültige Zugangsdaten für den Authentifizierungsprovider.
- Sowohl die VSCode Extension im Frontend als auch das lokale Backend-Mockup müssen erfolgreich gestartet worden sein.

Nachbedingungen:

- Der/die User/in hat sich mit gültigen Einwahldaten beim Authentifizierungsprovider authentifiziert oder war vorab bereits bei diesem angemeldet.
- Nach erfolgreicher Authentifizierung wird dies dem/der Nutzer/in im durch die Extension geöffneten Fenster angezeigt.
- Der/die User/in kann nun auf Lecturer-Funktionen abhängig von seinen/ihren Nutzungsrechten zugreifen.

- Der/die Nutzer/in kann die 30-minütige Session durch einen Button aktualisieren oder vorzeitig beenden.
- Nach 30 Minuten läuft die Session des Nutzers oder der Nutzerin ab, sofern er diese nicht verlängert. Dabei wird der/die Nutzer/in auf diesen Umstand durch eine Infonachricht hingewiesen und automatisch ausgeloggt.

Nachbedingungen im Sonderfall:

- Eine nicht erfolgreiche Authentifizierung sollte der Authentifizierungsprovider dem/der User/in signalisieren. Da eine Weiterleitung zum Frontend, der VSCode Extension, nur im Erfolgsfall geschieht, passiert hier nichts. Es wird weiterhin der Login-Button angezeigt.
- Besteht keine Internetverbindung, wird die entsprechende Meldung des Browsers angezeigt.

Normalablauf:

1. Der/die Nutzer/in greift auf die Nutzeroberfläche der Lehrendenkomponente zu und betätigt den Login-Button.
2. Die OPPSEE-Extension leitet den/die Nutzer/in zum Authentifizierungsprovider in einem neuen Fenster weiter.
3. Der/die Nutzer/in wird aufgefordert sich beim Authentifizierungsprovider zu authentifizieren.
4. Der/die Nutzer/in meldet sich mit seinen korrekten Anmeldedaten beim Authentifizierungsprovider an.
5. Dem/der Nutzer/in wird angezeigt, dass diese/r sich erfolgreich authentifiziert hat.
6. Die OPPSEE-Extension fängt das Weiterleiten des Authentifizierungsproviders inklusive Access und Refresh Token ab und registriert dadurch, dass der/die Nutzer/in nun Zugriff auf die Funktionen der Lehrendenkomponente haben darf.
7. Dem/der Nutzer/in wird die Profilseite angezeigt, welche den/die Nutzer/in begrüßt und ihm/ihr die aktuelle Rolle im System anzeigt. Er/sie hat nun Zugriff auf das Hauptmenü.

Sonderfall 3a: Der/die Nutzer/in hat keine bestehende Internetverbindung.

3a.1 Dem/der Nutzer/in wird die Standardmeldung des Browsers für eine fehlende Internetverbindung angezeigt.

Sonderfall 3b: Das Mockup-Backend ist nicht erreichbar.

3b.1 Dem/der Nutzer/in wird die Standardmeldung des Browsers für eine abgelehnte Verbindung des Backend-Hosts angezeigt.

Sonderfall 3c: Der/die Nutzer/in bricht den Anmeldevorgang ab.

3c.1 Der Anmeldevorgang wird abgebrochen und dem/der Nutzer/in wird weiterhin der Login-Button angezeigt.

Sonderfall 4a: Der/die Nutzer/in versucht sich mit fehlerhaften Daten beim Authentifizierungsprovider anzumelden.

4a.1 Dem/der Nutzer/in wird die Standardmeldung des Authentifizierungsproviders für falsch übermittelte Anmeldeinformationen angezeigt.

3.5.2 Kurserstellung

In diesem Abschnitt werden die nötigen Schritte zur Erstellung eines Kurses festgehalten.

Name: Kurse anlegen
Ziel: Dem Lecturer einen neuen Kurs bereitstellen. In diesem kann er künftig seine Lehrinhalte und Assignments anbieten.
Akteure: Lecturer, die VSCode Extension als Frontend, das Backend-Mockup

Vorbedingungen:

- Der Lecturer hat sich erfolgreich authentifiziert.
- Dem Lecturer ist im System die Rolle Lecturer zugewiesen.
- Ein Kursname muss angegeben werden, der nicht bereits existiert.
- Ein Passwort mit mindestens 8 Zeichen, einem Groß- und Kleinbuchstaben sowie einer Zahl muss für den Kurs festgelegt werden.

- Der Lecturer muss das Passwort nach der Eingabe in einem zweiten Textfeld bestätigen.

Nachbedingungen:

- Der Kurs, sowie die Kursinhalte sind in einer Datenbank persistiert.
- Der Lecturer ist im Kurs als Ersteller hinterlegt und erhält somit automatisch Administrator/innen Rechte im Kurskontext.
- Der Lecturer kann seinen Kurs über den Reiter „Meine Kurse“ aufrufen.

Nachbedingungen im Sonderfall:

- Der Lecturer wird dazu aufgefordert nicht bzw. falsch ausgefüllte Pflichttextfelder im Formular auszufüllen bzw. zu korrigieren.
- Bei bereits vorhandenem Kursnamen: Dem Lecturer wird nach betätigen des „Kurs erstellen“-Buttons ein Infotext angezeigt, dass ein Kurs mit dem besagten Namen bereits existiert.

Normalablauf:

1. Der Lecturer wählt im Hauptmenü den Eintrag „Kurs erstellen“ aus.
2. Der Lecturer trägt einen Kursnamen ins Formular ein.
3. Der Lecturer gibt optional eine Beschreibung für den Kurs an.
4. Der Lecturer wählt ein Passwort für die Kursselbsteinschreibung.
5. Der Lecturer Bestätigt das Passwort in einem weiteren Textfeld.
6. Der Lecturer betätigt den „Kurs erstellen“-Button.
7. Die Daten vom Lecturer werden an das Backend übermittelt.
8. Das Passwort wird verschlüsselt, bevor es auf der Datenbank abgelegt wird.
9. Ein neuer Kurs mit den übermittelten Daten wird in der Datenbank angelegt.

Sonderfall 2a: Der Lecturer betätigt den „Kurs erstellen“-Button bevor jegliche Pflichtfelder ausgefüllt sind.

2a.1 Ein Hinweis wird angezeigt, dass das entsprechende Pflichtfeld vor dem Übermitteln der Daten auszufüllen ist.

Sonderfall 4a: Der Lecturer gibt ein Passwort ohne Klein- / Großbuchstaben / Ziffer / mit weniger als acht Zeichen an.

4a.1 Ein Hinweis wird angezeigt, dass das Passwort aus mindestens einem Klein- / Großbuchstaben, einer Ziffer bzw. acht Zeichen bestehen muss.

Sonderfall 5a: Das Passwort zur Bestätigung stimmt nicht mit dem angegebenen Passwort überein.

5a.1 Ein Hinweis wird angezeigt, dass die beiden Passwörter nicht übereinstimmen.

Sonderfall 7a: Ein Kurs mit dem angegeben Namen ist bereits im System hinterlegt.

7a.1 Dem Lecturer wird eine Infomeldung angezeigt, dass ein Kurs mit dem gewählten Namen bereits existiert.

Sonderfall 7b: Das Backend ist nicht erreichbar.

7b.1 Dem Lecturer wird eine Infomeldung angezeigt, dass die Erstellung des Kurses fehlgeschlagen ist.

3.5.3 Anlegung von Kursinhalten (Assignments)

Hier wird beschrieben, wie ein/e Nutzer*in Kursinhalte für einen Kurs erstellen und diesem zuweisen kann. Zudem wird u. a. der Sonderfall beschrieben, in dem der/die Nutzer*in Daten zu einer bereits bestehenden Aufgabe aktualisieren bzw. anpassen möchte.

Name: Assignments in Kursen zuweisen
Ziel: Der/die Nutzer/in möchte Assignments von bestehenden Git Repositorys auf der Plattform anlegen und seinen Kursen hinzufügen können. Dies geschieht mit dem Ziel, dass Studierende innerhalb des Kurses an den Assignments arbeiten können.
Akteure: Lecturer / Teacher / Assignment Developer, die VSCode Extension als Frontend, das Backend-Mockup

Vorbedingungen:

- Der/die Nutzer/in hat sich erfolgreich authentifiziert.
- Dem/der Nutzer/in ist im System die Rolle Lecturer zugewiesen.
- Ein Assignmentname muss angegeben werden, der nicht bereits existiert.
- Eine valide HTTPS-URL muss angegeben werden. Diese sollte auf ein Git Repository verweisen.

Nachbedingungen:

- Das Assignment, ist in der Datenbank persistiert.
- Der/die Nutzer/in kann das Assignment über den Reiter „Assignments“ aufrufen.

Nachbedingungen im Sonderfall:

- Der/die Nutzer/in wird dazu aufgefordert nicht bzw. falsch ausgefüllte Pflichttextfelder im Formular auszufüllen bzw. zu korrigieren.
- Bei bereits vorhandenem Assignmentnamen: Dem/der Nutzer/in wird nach betätigen des „Assignment erstellen“-Buttons ein Infotext angezeigt, dass ein Assignment mit dem besagten Namen bereits existiert.

Normalablauf:

1. Der/die Nutzer/in wählt im Hauptmenü den Eintrag „Assignment erstellen“ aus.
2. Der/die Nutzer/in trägt einen Assignmentnamen ins Formular ein.
3. Der/die Nutzer/in gibt den HTTPS-Link zum Git Repository des Assignments an.
4. Optional wird eine Beschreibung angegeben.
5. Der/die Nutzer/in klickt den Button „Assignment erstellen“ an.
6. Die Daten von dem/der Nutzer/in werden an das Backend übermittelt.
7. Ein neues Assignment mit den übermittelten Daten wird in der Datenbank angelegt.

8. Der/die Nutzer/in wählt im Hauptmenü den Eintrag „Assignment“ aus.
9. Der/die Nutzer/in wählt das Assignment aus, welches einem Kurs hinzugefügt werden soll.
10. Der/die Nutzer/in klickt auf den Button „Zu Kursen hinzufügen“.
11. Der/die Nutzer/in wählt einen Kurs aus dem Dropdown-Menü aus, zu welchem das Assignment hinzugefügt werden soll.
12. Der/die Nutzer/in gibt optional an, wie lange das Assignment im Kurs sichtbar ist.
13. Der/die Nutzer/in klickt auf „hinzufügen“.
14. Der Eintrag des Assignments im Kurs wird in der Datenbank persistiert.

Sonderfall 2a: Der/die Nutzer/in betätigt den „Assignment erstellen“-Button bevor jegliche Pflichtfelder ausgefüllt sind.

2a.1 Dem/der Nutzer/in wird ein Hinweis angezeigt, dass das entsprechende Pflichtfeld vor dem Übermitteln der Daten auszufüllen ist.

Sonderfall 3a: Der/die Nutzer/in gibt eine URL an, die nicht der Syntax einer HTTPS-URL entspricht.

3a.1 Dem/der Nutzer/in wird ein Hinweis angezeigt, dass eine valide HTTPS-URL eingegeben werden muss.

Sonderfall 6a: Ein Assignment mit dem angegebenen Namen ist bereits im System hinterlegt.

6a.1 Dem/der Nutzer/in wird eine Infomeldung angezeigt, dass ein Assignment mit dem gewählten Namen bereits existiert.

Sonderfall 6b: Das Backend ist nicht erreichbar.

6b.1 Dem/der Nutzer/in wird eine Infomeldung angezeigt, dass die Erstellung des Assignments fehlgeschlagen ist.

Sonderfall 10a: Der/die Nutzer/in betätigt den „Update“-Button.

10a.1 Der/die Nutzer/in wählt einen Kurs aus dem Dropdown-Menü aus, zu welchem das Assignment bereits zugewiesen ist.

10a.2 Der/die Nutzer/in kann ein neues Datum dafür angeben von bzw. bis wann das Assignment sichtbar ist.

10a.3 Der/die Nutzer/in betätigt erneut den „Update“-Button.

10a.4 Die Kurs-Assignment-Beziehung in der Datenbank wird aktualisiert.

Sonderfall 11a: Der/die Nutzer/in betätigt den „Abbrechen“-Button.

11a.1 Der Vorgang dem Assignment einen Kurs zuzuweisen wird abgebrochen.

Sonderfall 11b: Der/die Nutzer/in betätigt den „Hinzufügen“-Button bevor ein Kurs aus dem Dropdown-Menü ausgewählt wurde.

11b.1 Die Eingabe wird vom Frontend ignoriert.

Sonderfall 11c: Das Backend ist nicht erreichbar.

11c.1 Dem/der Nutzer/in wird im Dropdown-Menü angezeigt, dass keine Kurse zum hinzufügen geladen werden konnten.

Sonderfall 12a: Der/die Nutzer/in gibt Daten für die Sichtbarkeit an, die in der Vergangenheit liegen.

12a.1 Für den/die Nutzer/in wird der Teil des Datums markiert, der geändert werden muss, damit das Datum nicht mehr in der Vergangenheit liegt.

Sonderfall 14a: Das Backend ist nicht erreichbar.

14a.1 Dem/der Nutzer/in wird eine Infomeldung angezeigt, dass das Assignment dem Kurs nicht erfolgreich zugewiesen werden konnte.

3.5.4 Sortierung, Filtration, Markierung und Aufrufen von Kursen

Nachdem es dem/der Nutzer/in nun möglich ist sich sowohl Kurse als auch Kursinhalte zu erstellen, soll er/sie diese auf eine übersichtliche und damit benutzerfreundliche Weise verwalten und aufrufen können. Dafür sind die Methoden aus dem „personalisierten Dashboard"-Teil des Moodle Abschnitts 2.1.1, welche u. a. das Filtern, Sortieren und Markieren beschreiben, auf unsere Lehrendenkomponente zu übertragen.

Name: Kurse sortieren, filtern, markieren und aufrufen
Ziel: Der/die Nutzer/in möchte die Kurse alphabetisch, nach Erstellungszeitpunkt oder nach den zuletzt aufgerufenen Kursen sortieren. Zudem möchte er/sie einen Kurs als Favorit markieren oder verstecken können. Darüber hinaus ist das Filtern nach verschiedenen Kurseigenschaften von Interesse. Somit soll der/die Nutzer/in schnell den gewünschten Kurs finden und ihn durch einen Klick aufrufen können. Auf die Eigenschaften nach denen ein Kurs gefiltert werden kann wird im Normalablauf genauer eingegangen.
Akteure: Lecturer / Teacher / Studierender, die VSCode Extension als Frontend, das Backend-Mockup

Vorbedingungen:

- Der/die Nutzer/in hat eine funktionierende Internetverbindung.
- Der/die Nutzer/in hat sich erfolgreich authentifiziert.
- Es wurden bereits Kurse und Assignments im System angelegt.
- Der/die Nutzer/in hat sich entweder bereits in einem Kurs eingeschrieben oder hat selbst einen Kurs erstellt.

Nachbedingungen:

- Die Anzahl der angezeigten Kurse in der Sidebar konnte durch das Verwenden von Filtern und Markierungen gezielt eingegrenzt werden.
- Der gesuchte Kurs wurde erfolgreich im Kursfenster geöffnet.

Nachbedingungen im Sonderfall:

- Besteht keine Verbindung zum Backend, wird anstatt der Kurse bzw. des Kursinhalts lediglich das Sync-Icon [VSCSync] angezeigt. Nach einer bestimmten Zeit, z. B. 15 Sekunden, erscheint eine Informationsnachricht, dass die Daten nicht geladen werden konnten.
- Bei Fehlern bezüglich des Klonen eines Repositorys wird eine VSCode Fehlermeldung angezeigt.

Normalablauf:
Um diesen Anwendungsfall besser beschreiben zu können wird vorab davon ausgegangen, dass der/die Nutzer/in einen Kurs mit dem Namen „Abc“ in der Liste seine Kurse hervorheben möchte. Ein anderer Kurs mit dem Namen „Xyz“ dagegen möchte der/die Nutzer/in standardmäßig verbergen. Anschließend hat er/sie vor, die Kursinhalte des Kurses „Abc“ abzurufen. Dem Kurs „Abc“ ist hier außerdem bereits mindestens ein Assignment zugewiesen worden.

1. Der/die Nutzer/in wählt im Hauptmenü den Eintrag „Kursübersicht“ aus.
2. Der/die Nutzer/in kann im Sortiermenü wählen, ob er seine Kurse alphabetisch, nach Erstellungsdatum oder nach dem letzten Zugriff sortiert.
3. Der/die Nutzer/in kann optional über die Suchleiste nach dem Kurs „Abc“ bzw. „Xyz“ suchen.
4. Der/die Nutzer/in favorisiert den Kurs „Abc“ mittels „Stern“-Button bzw. verbirgt den Kurs „Xyz“ mittels des „Verstecken“-Buttons auf der entsprechenden Kurslistenkarte.
5. Der angeklickte Button auf der Kurskarte wird optisch hervorgehoben.
6. Wurden die Standardfiltereinstellungen nicht geändert, verschwindet der Kurs „Xyz“ automatisch aus der Kursliste.

7. Über das Filtermenü kann der/die Nutzer/in seine/ihre Filtereinstellungen anpassen. Dabei kann er/sie festlegen ob Kurse mit folgenden Eigenschaften angezeigt werden oder nicht:
   - Verborgene Kurse
   - Favorisierte Kurse
   - Nicht markierte Kurse (weder verborgen, noch favorisiert)
   - Kurse aus den Bereichen Wirtschaftsinformatik (WI), Technische Informatik (TI) und/oder Angewandte Informatik (AI)
   - Gemanagte Kurse (Kurse in dessen Kurskontext der/die Nutzer/in die Rolle Teacher oder Lecturer zugewiesen hat)
8. Wurden die Filtereinstellungen angepasst, wird die Kursliste anschließend sofort automatisch aktualisiert.
9. Der/die Nutzer/in klickt nun auf die Kurskarte des Kurses „Abc“.
10. Der Kurs inklusive seiner Kursinhalte wird in einem neuen Fenster geladen.
11. Auf der neu geladenen Kursseite kann der/die Nutzer/in ein ausgewähltes Assignment ausklappen und den „Klonen“-Button betätigen.
12. Der/die Nutzer/in muss nun einen Ordner im Dateisystem angeben, in den das Repository geklont werden soll.
13. Das Repository wird in den gewählten Ordner kopiert.
14. Der/die Nutzer/in wird in einer VSCode Informationsnachricht gefragt, ob er das Repository öffnen (optional im neuen Fenster) möchte.
15. Der/die Nutzer/in betätigt den „Öffnen“-Button.
16. Das Repository des Assignments wird im aktuellen Fenster geöffnet.

Sonderfall 2a bzw. 10a: Das Backend ist nicht erreichbar.

2a.1 Die Animation des Sync-Icons, die dem/der Nutzer/in das Laden der Kurse / Kursinhalte (10a.1) veranschaulicht, befindet sich in einer Dauerschleife.

2a.2 Nach 15 Sekunden erhält der/die Nutzer/in die Informationsnachricht, dass die Kurse / Kursinhalte (10a.2) bisher noch nicht geladen werden konnten und er/sie es bitte nochmal versuchen solle.

Sonderfall 5a: Das Backend ist nicht erreichbar.

5a.1 Der angeklickte Button wird nicht optisch hervorgehoben.

5a.2 Der Eigenschaften der bereits geladenen Kurse bleiben unverändert.

Sonderfall 12a: Der/die Nutzer/in hat keine bestehende Internetverbindung.

12a.1 Es erscheint ein kleines VSCode Fenster mit dem Fehler, dass der Hostname nicht aufgelöst werden konnte.

Sonderfall 12b: Die HTTPS-URL des Assignments ist keine valide Git Repository URL.

12b.1 Es erscheint ein kleines VSCode Fenster mit dem Fehler, dass das Repository zur zugehörigen URL nicht gefunden werden konnte.

Sonderfall 15a: Der/die Nutzer/in möchte das Repository in einem neuen VSCode Fenster öffnen.

15a.1 Der/die Nutzer/in betätigt den „Öffne im neuen Fenster“-Button.

15a.2 Das Repository öffnet sich in einem neuen VSCode Fenster.

3.5.5 Kurseinschreibung

Das Anbieten von einer bzw. mehreren Methoden zur Einschreibung in einen Kurs ist essenziell, um einen kontrollierten Kurszugriff innerhalb des Systems zu gewährleisten. Mögliche Methoden wurden hierfür bereits im Moodle Abschnitt 2.1.1 beschrieben. In 1.0.2 wird verdeutlicht, dass für ein Feature zunächst nur eine Art der Umsetzung innerhalb des Prototypen realisiert werden soll. Daher Beschreibt dieser Anwendungsfall ausschließlich die Kurseinschreibungsart „Selbsteinschreibung“.

Name: Selbsteinschreibung
Ziel: Der/die Nutzer/in möchte sich für einen Kurs anmelden und Zugriff auf die Kursinhalte erhalten.
Akteure: Lecturer / Teacher / Studierender, die VSCode Extension als Frontend, das Backend-Mockup

Vorbedingungen:

- Der/die Nutzer/in hat eine funktionierende Internetverbindung.
- Der/die Nutzer/in hat sich erfolgreich authentifiziert.
- Es wurden bereits Kurse im System angelegt.
- Dem/der Nutzer/in liegt das Passwort für die Anmeldung im Kurs vor.
- Der/die Nutzer/in hat sich für den ausgewählten Kurs noch nicht eingeschrieben.

Nachbedingungen:

- Dem/der Nutzer/in werden die Kursinhalte nun angezeigt.
- Die Kurseinschreibung ist in der Datenbank persistiert.
- Der/die Nutzer/in muss sich beim erneuten Aufrufen des Kurses nicht noch einmal in den Kurs einschreiben.
- Der/die Nutzer/in hat die Rolle Student/in im Kurskontext.

Nachbedingungen im Sonderfall:

- Das Fehlschlagen der Einschreibung wird den/der Nutzer/in als Informations- oder Fehlermeldung angezeigt.
- Der/die Nutzer/in wird nicht in den Kurs eingeschrieben.
- Der/die Nutzer/in erhält keinen Zugriff auf die Kursinhalte.

Normalablauf:

1. Der/die Nutzer/in wählt im Hauptmenü den Eintrag „Kursübersicht“ aus.
2. Der/die Nutzer/in klickt auf den Radiobutton „Alle Kurse“.
3. Der/die Nutzer/in verwendet optional die Suchleiste, um nach einem bestimmten Kurs gezielt zu suchen.
4. Der/die Nutzer/in wählt einen Kurs aus, in den er sich neu einschreiben möchte.
5. Es öffnet sich ein neues Kursfenster.
6. Der/die Nutzer/in trägt im Formular unter dem Eintrag Kurspasswort das korrekte Passwort ein.
7. Der/die Nutzer/in betätigt den Button „Für Kurs anmelden“.
8. Das Lecturer Backend überprüft, ob das richtige Passwort übermittelt wurde.
9. Der/die Nutzer/in wird in der Datenbank im Kurs mit der Rolle Student/in eingetragen.
10. Eine Informationsnachricht erscheint, dass sich der/die Nutzer/in erfolgreich für den Kurs angemeldet hat.
11. Die Kursinhalte werden geladen und dem/der Nutzer/in angezeigt.

Sonderfall 4a: Das Backend ist nicht erreichbar.

4a.1 Dem/der Nutzer/in wird eine Fehlermeldung angezeigt, dass die Kurse nicht geladen werden konnten.

Sonderfall 6a: Der/die Nutzer/in versucht sich ohne Passwort anzumelden.

6a.1 Der/die Nutzer/in gibt kein Passwort an.

6a.2 Der/die Nutzer/in betätigt den Button „Für Kurs anmelden“.

6a.3 Dem/der Nutzer/in wird ein Hinweis angezeigt, dass ein Passwort einzutragen ist.

Sonderfall 6b: Der/die Nutzer gibt ein fehlerhaftes Passwort an.

6b.1 Der/die Nutzer/in gibt im Formular unter dem Eintrag Kurspasswort ein fehlerhaftes Passwort an.

6b.2 Der/die Nutzer/in betätigt den Button „Für Kurs anmelden“.

6b.3 Das Lecturer Backend überprüft, ob das richtige Passwort übermittelt wurde.

6b.4 Dem/der Nutzer/in wird in einer Informationsnachricht angezeigt, dass die Anmeldung für den Kurs fehlgeschlagen ist.

Sonderfall 7a: Das Backend ist nicht erreichbar.

7a.1 Dem/der Nutzer/in wird in einer Fehlermeldung angezeigt, dass das Backend nicht erreichbar ist.

3.5.6 Vergeben von erweiterten Nutzerrechten im Kurskontext

Ein Lecturer möchte seinen Kurs nicht immer zwingend alleine verwalten. Daher muss es für ihn eine Möglichkeit geben, Kursteilnehmer*innen Rechte im Kurskontext zuweisen zu können.

Name: Erweiterte Nutzerrechte im Kurskontext vergeben
Ziel: Ein Lecturer möchte einem/einer anderen Nutzer/in erweiterte Kursrechte in seinem Kurs zuteilen.
Akteure: Lecturer, die VSCode Extension als Frontend, das Backend-Mockup

Vorbedingungen:

- Der Lecturer hat eine funktionierende Internetverbindung.
- Der Lecturer hat sich erfolgreich authentifiziert.
- Es wurden bereits Kurse im System angelegt.
- Es haben sich bereits Nutzer/innen für den Kurs eingeschrieben.
- Der Lecturer hat die Rolle „Course Admin“ im Kurskontext.

Nachbedingungen:

- Die Rolle im Kurskontext wurde für einen oder mehrere Nutzer aktualisiert.
- Die neuen Rollen der Kursteilnehmer wurden in der Datenbank persistiert.

Nachbedingungen im Sonderfall:

- Bei Fehlschlagen des Bearbeiten der Rolle eines oder mehreren Kursteilnehmenden im Kurskontext seitens des Lecturer wird letzterem dies per Informationsnachricht mitgeteilt.
- Die neue Rolle für den Kursteilnehmenden wird nicht in der Datenbank persistiert.

Normalablauf:

1. Der Lecturer wählt im Hauptmenü den Eintrag „Kursübersicht“ aus.
2. Der Lecturer verwendet optional die Suchleiste oder den Filter, um nach einem bestimmten Kurs gezielt zu suchen.
3. Der Lecturer wählt einen Kurs aus, in dem er Nutzerrechte bearbeiten möchte.
4. Es öffnet sich ein neues Kursfenster.
5. Der Lecturer betätigt den Button „Zeige Kursteilnehmer*innen“.
6. Der Lecturer wählt im Dropdown-Menü die neue Rolle für einen oder mehrere Kursteilnehmer*innen aus.
7. Der Lecturer markiert die Checkboxen der Kursteilnehmer*innen, dessen Nutzerrechte er bearbeiten möchte.

8. Der Lecturer betätigt den Button „Ausgewählte Nutzerrollen aktualisieren“.
9. Eine Anfrage die Nutzerrollen zu aktualisieren, wird an das Lecturer Backend übermittelt.
10. Das Lecturer Backend überprüft, ob der Nutzer autorisiert ist Nutzerrollen zu bearbeiten.
11. Die neuen Nutzerrollen werden in der Datenbank eingetragen.
12. Eine Informationsnachricht für jede aktualisierte Rolle im Kurskontext erscheint.

Sonderfall 2a: Das Backend ist nicht erreichbar.

2a.1 Dem Lecturer wird eine Fehlermeldung angezeigt, dass die Kurse nicht geladen werden konnten.

Sonderfall 5a: Der/die Nutzer/in hat lediglich die Rolle „Student/in“ im Kurskontext.

5a.1 Der Button um auf die Kursteilnehmer*innen zugreifen zu können wird nicht angezeigt.

Sonderfall 7a: Der Lecturer versucht die Checkbox mit seinem eigenen Nutzereintrag zu markieren.

7a.1 Die Checkbox ist deaktiviert und reagiert nicht auf die Eingabe.

Sonderfall 8a: Der Lecturer betätigt den Button „Rollenänderungen zurücksetzen“.

8a.1 Die Rollenänderungen im Kurskontext werden zurückgesetzt ohne an das Lecturer Backend übermittelt zu werden.

Sonderfall 11a: Der/die Nutzer/in hat lediglich die Rolle „Teacher“ im Kurskontext.

11a.1 Dem/der Nutzer/in wird angezeigt, dass das Aktualisieren der Nutzerrollen fehlgeschlagen ist (da dieser nicht autorisiert ist).

3.5.7 Kursergebnisse / Kursstatistiken

In diesem Abschnitt wird beschrieben, wie und wo ein/e Kursverwalter/in mit der Rolle „Course Admin“ oder „Teacher“ im Kurskontext Kursergebnisse / Kursstatistiken abrufen kann. Dabei werden diese beiden Anwendungsfälle in einem zusammengefasst, da der Ablauf im Prinzip der Selbe ist. Lediglich die Schaltfläche, welche der/die Kursverwalter/in betätigen muss unterscheidet sich hierbei.

Name: Kursergebnisse / Kursstatistiken abrufen
Ziel: Der Lecturer bzw. Teacher möchte Statistiken, wie z. B. Aufrufe seines Kurses einsehen können. Zudem möchte er die Ergebnisse von Studierenden bezüglich der Bearbeitung seiner Assignment im Kurs einsehen können.
Akteure: Lecturer, Teacher, die VSCode Extension als Frontend, das Backend-Mockup

Vorbedingungen:

- Der/die Nutzer/in hat eine funktionierende Internetverbindung.
- Der/die Nutzer/in hat sich erfolgreich authentifiziert.
- Es wurden bereits Kurse im System angelegt.
- Es haben sich bereits Nutzer*innen für den Kurs eingeschrieben.
- Es haben Nutzer*innen bereits angebotene Assignment aus dem Kurs bearbeitet.
- Der/die Nutzer/in hat die Rolle „Course Admin“ im Kurskontext.

Nachbedingungen:

- Dem/der Nutzer/in werden Kursergebnisse / Kursstatistiken angezeigt.

Nachbedingungen im Sonderfall:

- Ohne Berechtigung bleibt die Schaltfläche um Kursergebnisse / Kursstatistiken einzusehen verborgen.
- Fehler beim Abrufen der Kursergebnisse / Kursstatistiken werden dem/der Nutzer/in angezeigt.

Normalablauf:

1. Der Lecturer wählt im Hauptmenü den Eintrag „Kursübersicht“ aus.
2. Der Lecturer verwendet optional die Suchleiste oder den Filter, um nach einem bestimmten Kurs gezielt zu suchen.
3. Der Lecturer wählt einen Kurs aus, für den er Statistiken bzw. Ergebnisse einsehen möchte.
4. Es öffnet sich ein neues Kursfenster.
5. Der Lecturer betätigt den Button „Kursstatistiken“ bzw. „Kursergebnisse“.

Sonderfall 2a: Das Backend ist nicht erreichbar.

2a.1 Dem Lecturer wird eine Fehlermeldung angezeigt, dass die Kurse nicht geladen werden konnten.

Sonderfall 5a: Der/die Nutzer/in hat lediglich die Rolle „Student/in“ im Kurskontext.

5a.1 Der Button um auf die Kursstatistiken bzw. Kursergebnisse zugreifen zu können wird nicht angezeigt.

3.5.8 Weitere (Sub-)Anwendungsfälle

Die Anwendungsfälle in diesem Kapitel sind teilweise so weit gefasst, dass sie weitere Subanwendungsfälle des Lecturer enthalten, die hier nicht noch einmal einzeln aufgelistet sind. Dies wurde aus dem Grund so umgesetzt, damit die gleichen Schritte im Normalablauf nach Möglichkeit nicht zu oft unter verschiedenen Anwendungsfällen aufgelistet werden.

Diese hier sogenannten Subanwendungsfälle beschreiben kurze und häufig benötigte Anwendungsfälle innerhalb der Applikation. Beispiele dafür wären u. a. Vorgänge wie eine Kursseite abzurufen, ein Assignment erstellen oder Assignmentinhalte abrufen. Da diese Subanwendungsfälle durch ihren Titel einen Vorgang bereits sehr präzise beschreiben, werden sie auch im folgenden Abschnitt 3.5.9 referenziert.

Zudem fast alle Anwendungsfälle in diesem Kapitel mit Fokus auf dem Lecturer erstellt worden. Das UML-Use-Case-Diagramm im folgenden Abschnitt beinhaltet jedoch auch Anwendungsfälle aus Sicht der Studierenden oder anderer Akteure.

3.5.9 Anwendungsfalldiagramm

Zum Abschluss dieses Abschnitts wird ein UML-Anwendungsfalldiagramm 3.2 erstellt, um die soeben beschriebenen Anwendungsfälle und ihre Beziehungen untereinander darzustellen. Ziel des Anwendungsfalldiagramms ist es, alle Akteure sowie Anwendungsfälle aufzulisten. Dabei wird auch visualisiert, welche Akteure an den entsprechenden Anwendungsfällen teilnehmen [RJB04, S. 34]. Als Notation wird die von Rumbaugh, Jacobson und Booch beschriebene UML-Notation für Use-Case-Diagramme verwendet [RJB04, S.79 - 80].

Konventionen

Anwendungsfälle, an denen der Lecturer aktiv teilnimmt und welche in diesem Kapitel hier bereits beschrieben wurden, sind hier in blauer Farbe dargestellt. Anwendungsfälle, die den Lecturer zwar betreffen können, in denen dieser aber nicht unmittelbar und aktiv involviert ist, werden dagegen in grün dargestellt. Die Assoziation zwischen Akteuren und Anwendungsfällen sind nur aus Gründen der Lesbarkeit teilweise in unterschiedlichen Farben dargestellt. Zudem beinhaltet jeder Anwendungsfall auch den Vorgang sich zu authentifizieren. Auch dies wurde aus Gründen der Lesbarkeit nicht im Diagramm berücksichtigt.

Wie im Abschnitt 3.3 bereits beschrieben darf ein Akteur zur gleichen Zeit nicht zwei Rollen einnehmen. Das bedeutet beispielsweise, dass ein Lecturer durchaus die Rolle des Assignment Developers annehmen und neue Assignments erstellen darf. Im folgenden Diagramm führt er diesen Vorgang jedoch als Assignment Developer und nicht als Lecturer aus.

Akteure sind im folgenden Diagramm unterhalb und oberhalb des Teilsystems „Lecturer Extension“ eingezeichnet. Ist ein Anwendungsfall sowohl mit einem Akteur oberhalb, als auch unterhalb des Teilsystems verknüpft, so ist der Akteur oberhalb Initiator des Vorgangs. Der Akteur unterhalb spielt dann lediglich den Empfänger, z. B. für den Erhalt einer neuen Rolle. Anzumerken ist an dieser Stelle, dass das Lecturer Backend als Empfänger nicht berücksichtigt wurde, da es in jedem Anwendungsfall, mit Ausnahme der Authentifizierung, Daten übermittelt bekommt. Dies hätte die Übersichtlichkeit des Diagramms negativ beeinflusst.

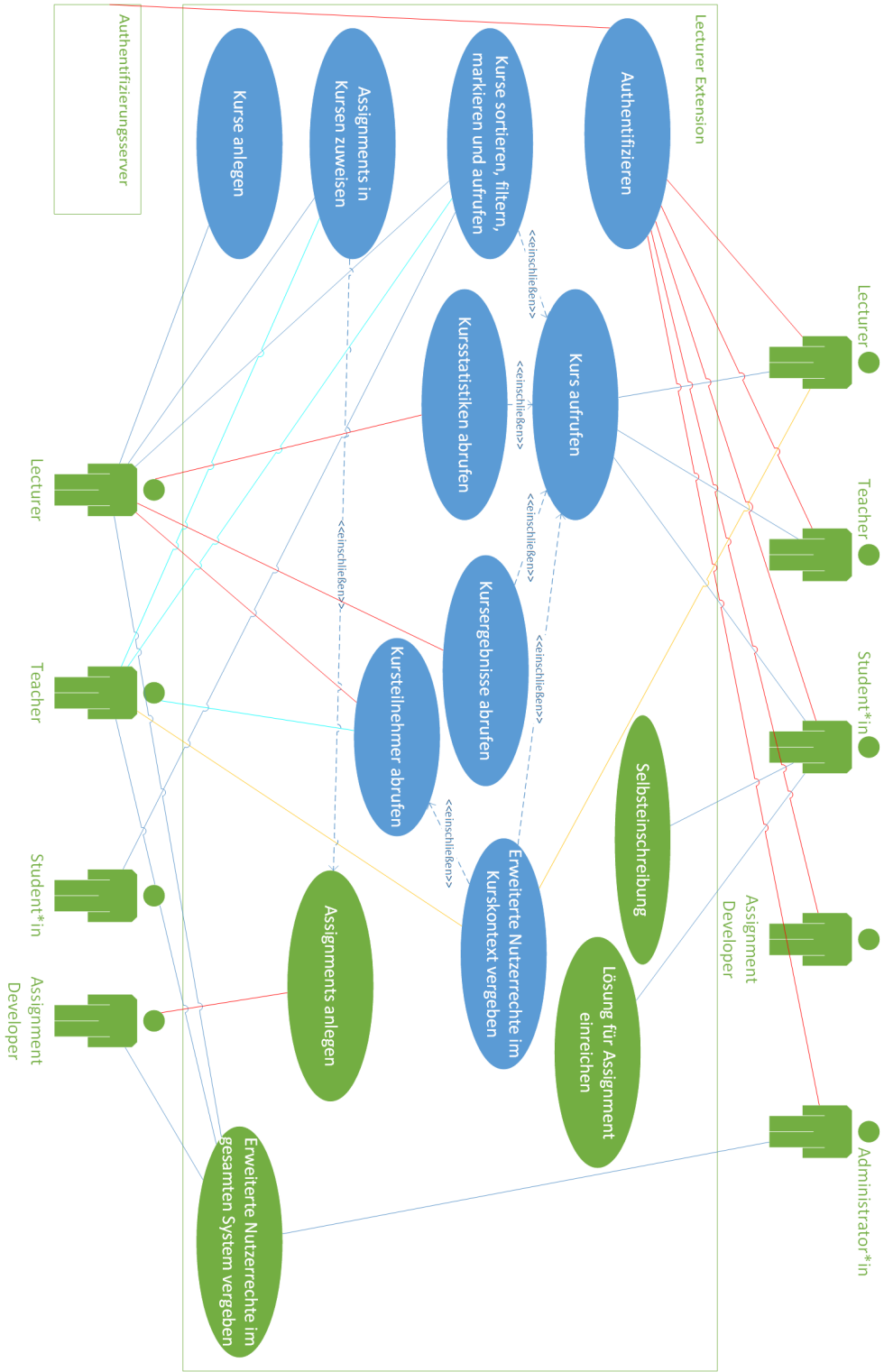

Abbildung 3.2: Anwendungsfalldiagramm

4 Modellierung, Architektur und Entwurf

Dieses Kapitel ist in die drei Abschnitte Modellierung, Architektur und Grobentwurf, sowie Feinentwurf unterteilt. Im ersten Abschnitt 4.1 werden Modelle aufgestellt, die Elemente (hier Klassen bzw. Entitäten) darstellen, welche im Rahmen unser zuvor betrachteten Anwendungsfälle abgebildet werden können. Dazu gehören zum Beispiel Kurse, Assignments, aber auch User*innen.

Da sich diese Modelle (teilweise) von den Anforderungen ableiten und somit noch zur Anforderungsermittlung gehören, werden diese in einem von der Architektur und dem Entwurf separaten Abschnitt modelliert.

Im zweiten Abschnitt 4.2 wird dann das Gesamtsystem betrachtet und in seine Subsysteme unterteilt. Zudem werden die Schnittstellen zwischen den Subsystemen spezifiziert und dokumentiert.

Im dritten und letzten Abschnitt 4.3 werden erweiterte Klassendiagramme der Front- und Backendkomponente konzipiert, die neben den Attributen nun auch zu implementierende Methoden spezifizieren.

4.1 Modellierung

Dieser Abschnitt umfasst das Domänenmodell, welches sich von den Anforderungen ableiten lässt. Darüber hinaus wird hier auch das Datenbankmodell erstellt. Letzteres gehört nicht zur Anforderungsanalyse, wird aber an dieser Stelle konzipiert, da Entitäten und ihre Beziehungen untereinander ähnlich den Klassen und Kardinalitäten des Domänenmodells sind.

4.1.1 Domänenmodell

Das Domänenmodell, ist ein Modell, welches rein auf fachlicher Basis erstellt werden sollte. Es erlaubt eine genaue Formulierung von Anforderungen [GKRS17, S. 63] und kann sowohl von Fachexperten als auch Entwicklern interpretiert werden.

Für die Notation des folgenden Modells wird hier die UML-Klassendiagrammnotation nach Rumbaugh, Jacobson und Booch verwendet [RJB04, S. 28]. Daher wird es im Folgenden auch Domänenklassenmodell genannt.

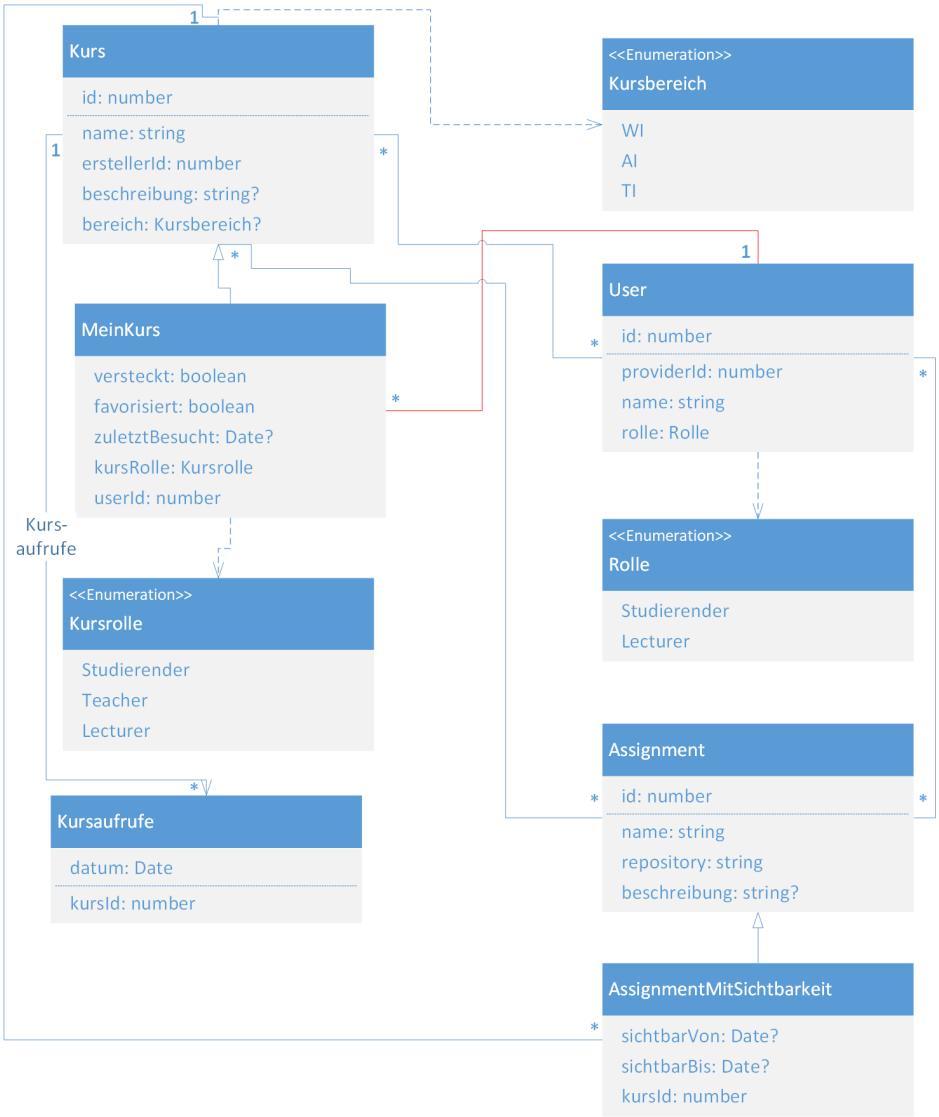

Abbildung 4.1: Dömanenklassenmodell

Das Modell kann dabei wie folgt interpretiert werden. In unserem System existieren Kurse, denen neben einer Identifikationsnummer (ID) zur eindeutigen Identifikation auch ein Name und die Ersteller ID des Lecturer zugewiesen werden müssen. Eine Beschreibung und ein Kursbereich ist dagegen optional. Falls letzterer angegeben wird, muss dieser entweder WI, TI oder AI lauten. Kurse haben n:m-Assoziation zu Usern und Assignments. Das bedeutet, dass ein Kurs mehrere User/Assignment enthalten kann. Andersherum heißt dies aber auch, dass ein/e Nutzer/in in mehrere Kurse gleichzeitig eingeschrieben sein kann, bzw. ein Assignment in mehreren Kursen enthalten. Assignment mit deklarierter Sichtbarkeit können dagegen nur einem Kurs zugewiesen sein. Dies ist notwendig, damit die Sichtbarkeit des gleichen Assignment in verschiedenen Kursen unterschiedlich eingestellt werden kann.

Die „MeinKurs“-Klasse enthält dagegen noch zusätzliche Informationen, die benutzerdefiniert sind. Daher kann eine Instanz der „MeinKurs“-Klasse ausschließlich einem User zugewiesen werden. Die „MeinKurs“-Klasse definiert auch die Rolle im Kurskontext. Hier stehen die Möglichkeiten „Studierender“, „Teacher“ und „Lecturer“ zur Auswahl.

Über die bisherigen Assoziationen hinaus besteht noch eine n:m-Assoziation zwischen Usern und Assignments. Dies liegt daran, dass Studierende die Möglichkeit haben sollen, beliebig viele Assignments zu bearbeiten. Andersherum soll ein Assignment ebenfalls von beliebig vielen Studierenden bearbeitet werden können.

4.1.2 Datenbankmodell

Um die Architektur unserer Datenbank zu beschreiben wird die durch Chen beschriebene Notation des Entity-Relationship Modells (ER Modells) verwendet, welche aus der Motivation heraus entstanden ist als Werkzeug für Datenbankmodellierung zu dienen [Che76, S. 9]. Dieses wird sich dem Domänenklassenmodell ähneln, die Beziehungen durch das verknüpfen mit Attributen jedoch genauer beschreiben. Auf Grundlage der Entitäten bzw. Beziehungen dieses Modells können in Abschnitt 5.1.2 die Tabellen der Datenbank aufgesetzt werden.

Da auf die Entitäten bzw. Kardinalitäten bereits im vorherigen Abschnitt 4.1.1 eingegangen worden ist, wird der Vorgang an dieser Stelle nicht wiederholt. Neu dagegen ist die Beschreibung durch Attribute der Beziehungen zwischen den Entitäten.

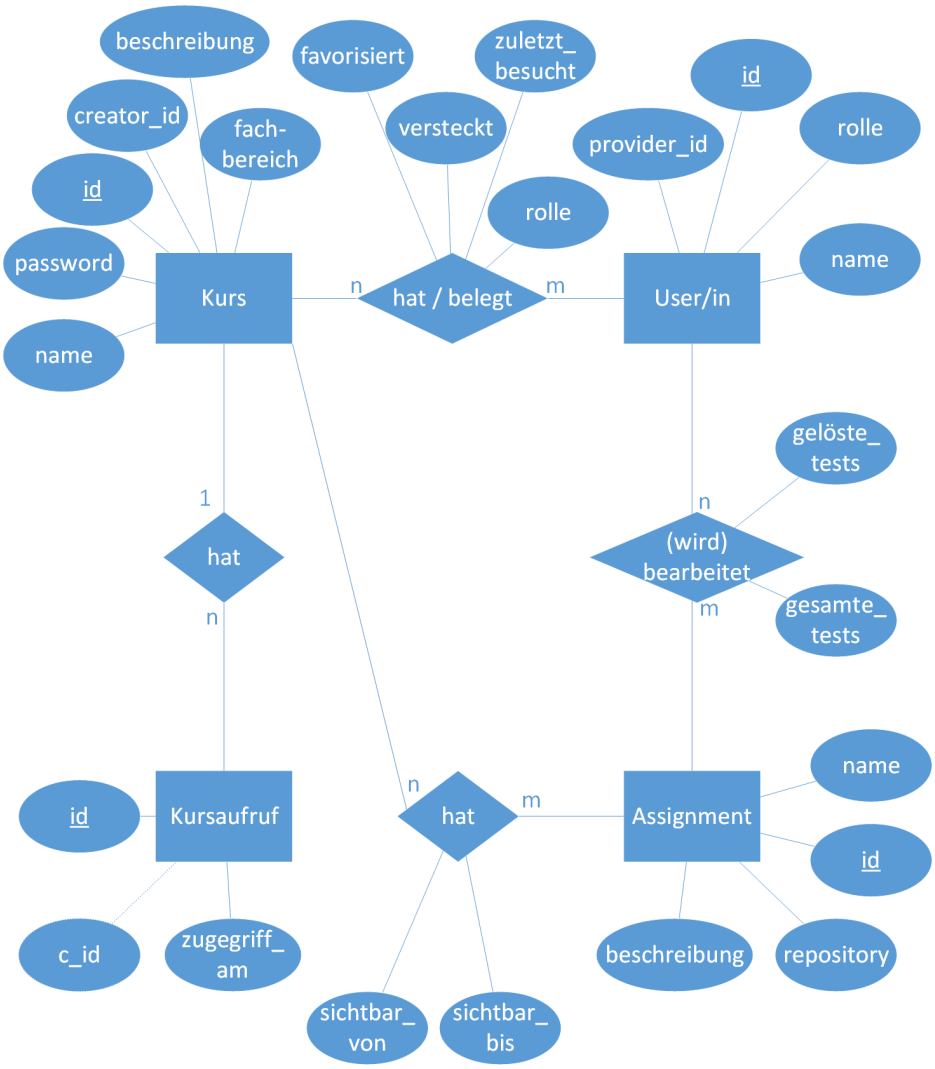

Abbildung 4.2: Entity-Relationship-Modell

Zwischen Kurs und User/in, Kurs und Assignment, sowie User/in und Assignment liegt eine n:m-Beziehung vor. Das bedeutet, dass für diese drei Beziehungen eine Beziehungstabelle inklusive der jeweiligen Attribute erstellt werden muss. Die Kurs-User/in-Beziehung muss daher die Felder „favorisiert“, „versteckt“, „zuletzt_besucht“ und „rolle“ in der Datenbank beinhalten. Die Kurs-Assignment-Beziehung die Felder „sichtbar_von“ und „sichtbar_bis“. Die User*in-Assignment-Beziehung schließlich muss die Anzahl der gelösten und gesamten Tests zu einem Assignment hinterlegen.

4.2 Architektur und Grobentwurf

Um einen Überblick über die Systemarchitektur der im Rahmen dieser Arbeit implementierten Prototypen zu gewähren, wird zunächst ein Komponentendiagramm erstellt, welches das ganze System in die kritischen Systeme für unsere Anwendung unterteilt. Anschließend werden jegliche Schnittstellen spezifiziert und dokumentiert, die unser Lecturer Backend der Lecturer Extension bereitstellen muss.

4.2.1 Subsystemspezifikation (Komponentendiagramm)

Für die Konzipierung unseres Komponentendiagramms wird das OPPSEE-Diagramm aus Abbildung 3.1 als Ausgang verwendet. Es wird dabei heruntergebrochen auf die Komponenten, welche im Rahmen des Lecturer Backend und der Lecturer Extension entwickelt bzw. benötigt wurden. Daraus ergibt sich das folgende Diagramm 4.3:

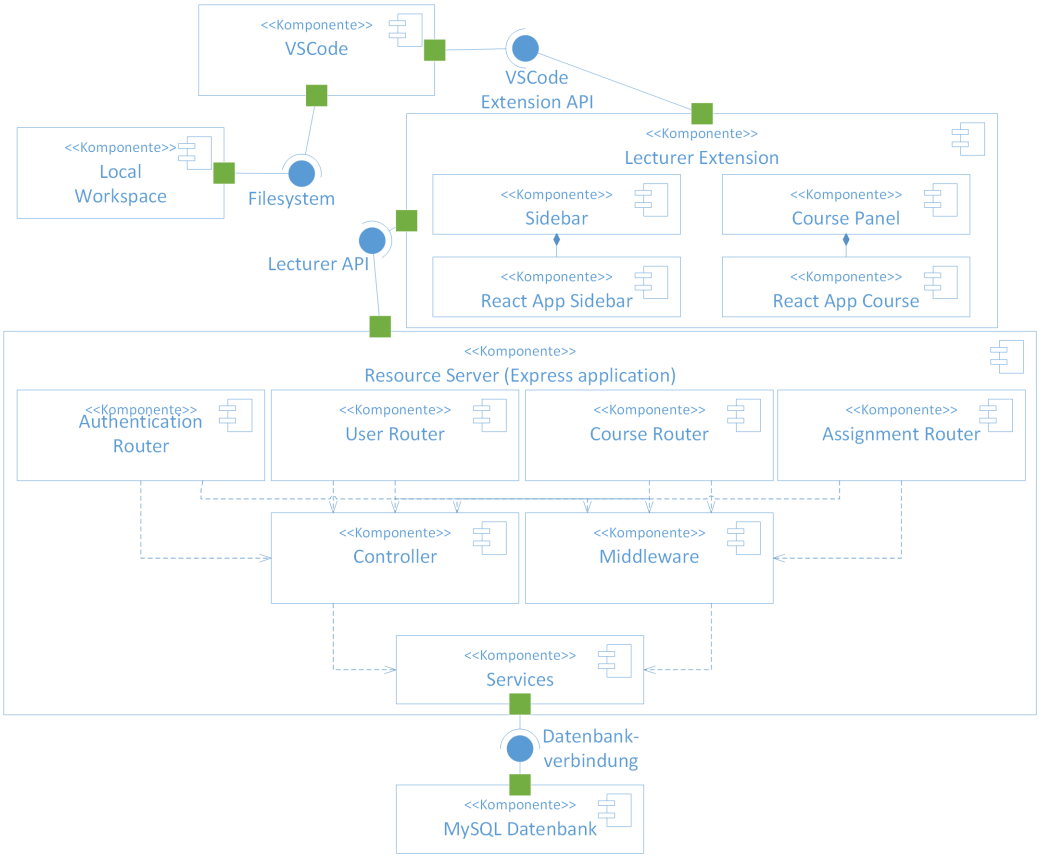

Abbildung 4.3: Lecturer Komponentendiagramm

Lecturer Extension
Die Lecturer Extension enthält einen sogenannten Sidebar-Provider und ein Course-Panel, welche den Lebenszyklus der Webviews verwalten. Sie enthalten das HTML-„root"-Element und wenden etwaige Stile oder Skripts auf das Webview, welches sie verwalten an. So auch den kompilierten Programmiercode der React-Anwendungen. Aus diesem Grund sind eben jene existenzabhängig von dem Sidebar-Provider bzw. Course-Panel und daher als Komposition dieser modelliert.

Lecturer Backend
Um sich zu authentifizieren, oder sogenannte Create, Read, Update, Delete (CRUD)-Operationen auf die persistierten Daten auszuführen, muss die Lecturer Extension mit dem Lecturer Backend über die Lecturer API kommunizieren. Nach einer Anfrage wird diese vom entsprechenden Router innerhalb der Express-Anwendung an die Controller und Middleware weitergeleitet. Controller und Middleware implementieren dabei wiederverwendbare Funktionen. Sie unterscheiden sich lediglich in dem Punkt, dass Controller fachliche Operationen ausführen. Middleware dagegen prüft auf etwaige Fehler (Eingabefehler, fehlerhafte bzw. unbekannte Routen, Authentifizierung, Autorisierung). Durch diesen Architekturaufbau muss in den Routern beispielsweise die Authentifizierung oder Autorisierung nicht ständig neu implementiert werden. Es reicht aus, die bereits implementierte Funktion aus der Middleware bzw. dem Controller immer wieder aufzurufen.

Datenbankzugriff
Im Falle von Zugriffen auf die Datenbank, müssen Controller und Middleware auf Services zurückgreifen. Diese stellen eine Verbindung zur Datenbank her und abstrahieren bzw. übersetzen die Structured Query Language (SQL)-Befehle in TypeScript-Funktionen.

Lokales Dateisystem
Wie aus dem Diagramm 4.3 hervorgeht, ist der Zugriff auf das lokale Dateisystem über VSCode hier auch abgebildet. Das hat den Grund, dass beim Klonen von Assignments auf dieses zugegriffen werden muss.

4.2.2 Schnittstellenspezifikation

In diesem Abschnitt werden alle API Endpunkte und damit die Schnittstellen des Lecturer Backends für die Lecturer Extension spezifiziert und dokumentiert. Da das Modellieren aller API Endpunkte zu einem unübersichtlichen Diagramm führen würde, werden diese stattdessen in 4 Bereiche unterteilt und beschrieben.

Authentifizierung

API Endpunkt	Beschreibung
GET /authenticate/provider	Weiterleitung zur Authentifizierung beim Provider (GitHub oder GitLab).
GET /authenticate/provider/callback	Callback nachdem die Authentifizierung erfolgreich war. Der Nutzer wird auf eine URL weitergeleitet und kann hier das Access und Refresh Token abfangen.
POST /authenticate/refresh	Vergeben eines neuen Access Token durch das übergeben eines Refresh Tokens.

Tabelle 4.1: API Endpunkte Authentifizierung

User

API Endpunkt	Beschreibung
POST /users	Erstellen eines/einer neuen Nutzer/in.
GET /users/profile	Gibt die Profilinformationen eines/einer Nutzer/in zurück (ID, Name und Rolle).
GET /users/assignment/:id	Gibt alle Nutzer/innen zurück, die ein bestimmtes Assignment bearbeitet haben. Autorisiert nur im Kurskontext als Teacher oder Lecturer.

Tabelle 4.2: API Endpunkte User

Kurse

API Endpunkt	Beschreibung
POST /courses	Erstellen eines Kurses. Autorisiert nur mit Lecturer Rolle im gesamten Systemkontext.
POST /courses/signUp	Einschreiben in einen Kurs.
GET /courses	Alle Kurse anzeigen.

GET /courses/course/:id	Kursdaten inklusive benutzerdefinierte Einstellungen/Informationen (zuletzt besucht, favorisiert etc.) für einen bestimmten Kurs abrufen.
GET /courses/course/:id/users	Alle Nutzer eines bestimmten Kurses anzeigen. Funktioniert nur mit der Rolle „Teacher“ oder höher („Kursadministrator/in“) im Kurskontext.
GET /courses/course/:id/assignments	Alle Assignments innerhalb eines bestimmten Kurses anzeigen. Funktioniert nur mit der Rolle „Teacher“ oder höher im Kurskontext.
GET /courses/course/:id/assignments/visible	Alle sichtbaren Assignments innerhalb eines bestimmten Kurses anzeigen.
GET /courses/course/:id/assignments/norelation	Alle Assignments anzeigen, die noch nicht in einem bestimmten Kurs hinterlegt sind. Funktioniert nur mit der Rolle „Teacher“ oder höher im Kurskontext.
GET /courses/myCourses	Alle Kurse anzeigen, in denen ich eingeschrieben, Teacher oder Lecturer bin.
PUT /courses/name/:id	Name eines Kurses bearbeiten. Nur mir der Rolle „Kursadministrator/in“ möglich.
PUT /courses/description/:id	Beschreibung eines Kurses bearbeiten. Nur mir der Rolle „Teacher“ oder höher im Kurskontext möglich.
PUT /courses/hidden/:id	Verborgen-Eigenschaft für bestimmten Kurs in der eigenen Kursansicht bearbeiten.
PUT /courses/starred/:id	Favorisieren-Eigenschaft für bestimmten Kurs in der eigenen Kursansicht bearbeiten.
PUT /courses/visited/:id	Zuletzt-Besucht-Eigenschaft für bestimmten Kurs aktualisieren.

PUT /courses/role/:cId/:uId	Rolle eines/einer Nutzers/Nutzerin bearbeiten. Nur möglich mit der Rolle „Kursadministrator*in“.
DELETE /courses/course/:cId/assignment/:aId	Assignment aus einem Kurs löschen. Funktioniert nur mit der Rolle „Teacher“ oder höher im Kurskontext.
GET /courses/accessed/:id	Erfasse Anzahl der Nutzer/innen, die den Kurs innerhalb der letzten Woche aufgerufen haben.
GET /courses/accessed/total/:id	Erfasse Anzahl der gesamten Aufrufe, die den Kurs innerhalb der letzten Woche aufgerufen haben.
POST /courses/accessed	Fügt einem Kurs einen Aufruf hinzu.

Tabelle 4.3: API Endpunkte Kurse

Assignments

API Endpunkt	**Beschreibung**
POST /assignments	Erstellen eines Assignments. Autorisiert mit Lecturer Rolle im gesamten Systemkontext oder mindestens Teacher Rolle im Kurskontext.
POST /assignments/assignment/:aId/course/:cId	Fügt ein bereits existierendes Assignment einem Kurs hinzu. Erfordert mindestens die Teacher Rolle im Kurskontext.
GET /assignments/assignment/:aId/course/:cId	Aktualisiert die Sichtbarkeit der Assignments innerhalb des Kurses.
GET /assignments	Gibt alle existierenden Assignments zurück.
GET /assignments/assignment/:id/courses	Gibt alle Kurse zurück, in denen das Assignment vorkommt.
GET /assignments/assignment/:id/ courses/no-relation	Gibt alle Kurse zurück, in denen das Assignment noch nicht vorkommt.

POST /assignments/assignment/:id/user	Erstellt eine Nutzer*in-Assignment-Beziehung, um abzubilden, wie viele Tests des Assignments erfolgreich durchlaufen wurden.
POST /assignments/assignment/:id/user/update	Aktualisiert die Nutzer*in-Assignment-Beziehung.

Tabelle 4.4: API Endpunkte Assignments

4.3 Feinentwurf

Im letzten Abschnitt dieses Kapitels wird ein erweitertes Klassendiagramm des Frontends modelliert. Das Diagramm soll nun jegliche Klassen, Attribute und auch Funktionen enthalten, die für die Implementierung relevant sind. Hierbei wird nur das erweiterte Klassendiagramm des Frontends modelliert. Dies liegt daran, dass das Lecturer Backend lediglich den Lecturer Service des OPPSEE-Backends simulieren soll. Es gibt daher nicht die tatsächliche Architektur im OPPSEE-Backend wieder. Daher wird an dieser Stelle auf das Betrachten detaillierterer Komponenten als im Grobentwurf verzichtet.

4.3.1 Erweitertes Klassendiagramm des Frontends

Für das erweiterte Klassendiagramm des Frontends wird die gleiche Notation wie für das Domänenklassenmodell im Abschnitt 4.1.1 verwendet. Zunächst wird jedoch nur der Extension-Kontext betrachtet. Dabei wird u. a. ein Blick auf die Sidebar und das Course Panel geworfen. Auf die detaillierte Architektur der React-Anwendungen innerhalb der Extension wird dagegen im Abschnitt 5.2.2 genauer eingegangen.

Extension-Kontext
Im Laufe der Entwicklung unseres Prototypen werden viele benutzerdefinierte Type Guards verwendet. Diese stellen sicher, dass die Funktionen welche wir an einem instantiierten Objekt aufrufen, auch wirklich definiert sind. Da diese fachlich jedoch eine untergeordnete Rolle spielen, wurden dazugehörige Klassen bzw. Funktionen an dieser Stelle weggelassen.

Des Weiteren werden private oder nicht verwendete Funktionen ebenfalls außen vor gelassen. Das daher resultierende erweiterte Klassenmodell für die Extension sieht wie folgt aus:

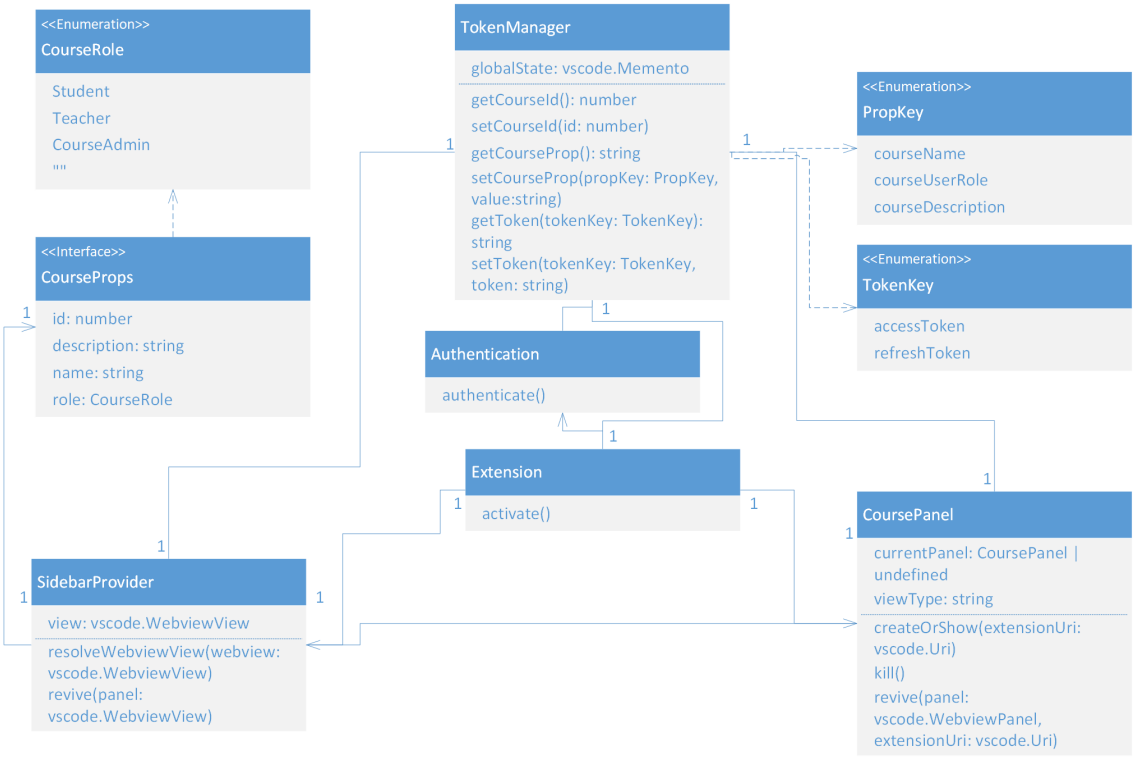

Abbildung 4.4: Erweitertes Klassendiagramm der Extension

Der Einstiegspunkt hier ist die activate()-Funktion der Extension. Diese wird aufgerufen, sobald die Lecturer Extension gestartet ist. Anschließend wird ein Icon in der Sidebar angezeigt, über welches auf das Menü der Lecturer Extension zugegriffen werden kann. Zudem sind nun auch alle weiteren Befehle der Extension ausführbar, wie z. B. das Aufrufen der authenticate()-Funktion oder des Course Panels.

Einen weiteren zentraler Punkt für unsere Anwendung stellt der TokenManager dar. Dieser bietet durch das Attribut „globalState“ die Möglichkeit an, Daten lokal im VS-Code Editor zu speichern oder abzurufen. Dies kann nützlich sein, um Anmeldedaten zu speichern oder zwischengespeicherte Daten der Sidebar im Course Panel aufzurufen.

Anzumerken ist hierbei jedoch, dass die OPPSEE-Plattform im Browser läuft. Daher könnte in dem Fall anstelle von „vscode.Memento“, dem lokalen Speicher in VSCode vermutlich etwas wie der Local Storage im Browser verwendet werden. Dies ist ein über HTML5 verfügbarer Key-Value-Speicher [WP12, S. 84].

5 Implementierung

Das Lecturer Backend und die Lecturer Extension sollen die bisher beschriebenen Anforderungen auf Grundlage der konzipierten Architektur implementieren. Nachfolgend wird beschrieben, inwiefern dies auf technologischer Ebene gelingen kann.

5.1 Backend

Nachfolgend wird beschrieben, welche Technologien zum Erstellen des Backends verwendet wurden. Anschließend wird der Aufbau und die Nutzung erklärt. Zu guter Letzt werden die in Abschnitt 4.2.2 beschriebenen Schnittstellen mit Hilfe von Postman [Postman] getestet. Das hier beschriebene Lecturer Backend ist auch auf GitHub abrufbar [VscMock].

5.1.1 Verwendete Technologien

Wie in der Einleitung (1.0.2) bereits erwähnt, wird das Lecturer Backend auf Basis von Express erstellt. Dies hat neben den erläuterten Vorzügen den Vorteil, dass das komplette Projekt im Backend sowie auch im Frontend in TypeScript entwickelt werden kann. Als Datenbank wird wie bei ArTEMiS eine MySQL-Datenbank verwendet.

Neben dem Express-Framework wurden u. a. folgende npm-Module [npmjs] verwendet und via „npm i“ installiert:

npm-Modul	**Zweck**
express-validator	Zum Validieren und Bereinigen von übergebenen Parametern über einen API Endpunkt.

nodemon	Zum Starten und automatischem Neustarten der Express-Anwendung bei Änderungen.
mysql2	„mysql2“ fungiert als Datenbanktreiber zum Zugriff auf die MySQL-Datenbank.
dotenv	Zum Laden etwaiger Variablen aus der process.env-Datei. In Letzterer können Tokens oder Passwörter hinterlegt werden, um das Offenbaren der sensitiven Daten im Quellcode zu vermeiden.
bcrypt	Zum Verschlüsseln von Passwörtern, zum Beispiel dem Kurseinschreibeschlüssel.
jsonwebtoken	Um eigens erstellte Tokens zu vergeben. Diese können für den Authentifizierungsprozess genutzt werden.
passport und passport-oauth2	Um sich mittels eines OAuth 2.0 Authentifizierungsprovider authentifizieren zu können und auf Basis letzteren Moduls unsere eigenen Passport-Strategien zu erstellen.

Tabelle 5.1: Verwendete npm-Module im Lecturer Backend

5.1.2 Aufbau

Um das Lecturer Backend Projekt aufzusetzen, muss zunächst die MySQL Community Server Installation [MySqlc] durchgeführt werden. Anschließend kann man sich mittels des Terminals bei MySQL anmelden:

```
mysql -u root -p
Enter password: <root-Passwort>
```

Nun muss eine neue Datenbank für dieses Projekt angelegt werden, z. B. mit:

```
CREATE DATABASE vsc_mock_backend_db;
```

Da es aus Sicherheitsgründen nicht üblich ist auf dem root-User zu arbeiten, sollte ein/e neue/r Nutzer/in für die Datenbank angelegt werden, z. B. mit:
CREATE USER 'vsc_user'@'localhost IDENTIFIED WITH mysql_native_password' BY <neues Passwort>

Anschließend sollten diesem/dieser alle Rechte auf der erstellten Datenbank übertragen werden:
GRANT ALL PRIVILIGES ON vsc_mock_backend_db.* TO vsc_user@localhost

Nun kann man sich in drei bis vier Schritten mit dem neuen Account anmelden und auf die Datenbank zugreifen:

1. exit;
2. mysql -u vsc_user -p
3. Enter password: <neues Passwort>
4. USE vsc_mock_backend_db

Schließlich können die Datenbanktabellen aus dem ER Modells in Abschnitt 4.1.2 abgeleitet werden:

Listing 5.1: Tabelle User

```
1        CREATE TABLE User (
2                id INT AUTO_INCREMENT,
3                provider_id INT NOT NULL UNIQUE,
4                name VARCHAR(50) NOT NULL,
5                role ENUM('Lecturer', 'Student') NOT NULL
                     DEFAULT 'Student',
6                PRIMARY KEY (id)
7        );
```

Listing 5.2: Tabelle Course

```
CREATE TABLE Course (
        id INT AUTO_INCREMENT,
        name VARCHAR(50) NOT NULL UNIQUE,
        password VARCHAR(255) NOT NULL,
        creator_id INT NOT NULL,
        description TEXT,
        subject ENUM('WI', 'TI', 'AI'),
        PRIMARY KEY (id),
        FOREIGN KEY (creator_id) REFERENCES User(id)
);
```

Listing 5.3: Tabelle Assignment

```
CREATE TABLE Assignment (
        id INT AUTO_INCREMENT,
        name VARCHAR(50) NOT NULL UNIQUE,
        repository VARCHAR(255) NOT NULL UNIQUE,
        description TEXT,
        PRIMARY KEY (id)
);
```

Listing 5.4: Tabelle CourseUserRelation

```
CREATE TABLE CourseUserRelation (
        u_id INT,
        c_id INT,
        hidden BOOLEAN NOT NULL DEFAULT 0,
        starred BOOLEAN NOT NULL DEFAULT 0,
        role ENUM('CourseAdmin', 'Teacher', 'Student')
            NOT NULL DEFAULT 'Student',
        visited DATETIME,
        PRIMARY KEY (u_id, c_id),
        FOREIGN KEY (u_id) REFERENCES User(id),
        FOREIGN KEY (c_id) REFERENCES Course(id)
);
```

Listing 5.5: Tabelle CourseAssignmentRelation

```
CREATE TABLE CourseAssignmentRelation (
        a_id INT,
        c_id INT,
        visible_from DATETIME,
        visible_to DATETIME,
        PRIMARY KEY (a_id, c_id),
        FOREIGN KEY (a_id) REFERENCES Assignment(id),
        FOREIGN KEY (c_id) REFERENCES Course(id)
);
```

Listing 5.6: Tabelle AssignmentUserRelation

```
CREATE TABLE AssignmentUserRelation (
        u_id INT,
        a_id INT,
        solved_tests INT,
        total_tests INT,
        PRIMARY KEY (u_id, a_id),
        FOREIGN KEY (u_id) REFERENCES User(id),
        FOREIGN KEY (a_id) REFERENCES Assignment(id)
);
```

Listing 5.7: Tabelle CourseAccess

```
CREATE TABLE CourseAccess(
        ca_id INT AUTO_INCREMENT,
        c_id INT,
        accessed DATETIME,
        PRIMARY KEY (ca_id),
        FOREIGN KEY (c_id) REFERENCES Course(id)
);
```

Nach Erstellung der Tabellen, müssen in der process.env-Datei noch die folgenden Umgebungsvariablen für die Verbindung mit der Datenbank eingetragen werden:
PORT=3000 (oder einen anderen Port, falls bevorzugt)
DB_HOST="localhost"
DB_USER=<Nutzername> (z. B. "vsc_user")
DB_PWD=<Passwort>
DB_NAME=<Datenbankname> (z. B. "vsc_mock_backend_db")

Um zu testen, ob alles funktioniert hat, kann nun folgender API Call gemacht werden:
GET http://localhost:3000/users
Für diese Route wird aus Testzwecken keine Authentifizierung vorausgesetzt.

Nachdem die Datenbank läuft, müssen des Weiteren Umgebungsvariablen für die Passport-Strategien eingerichtet werden. Dies läuft wie folgt ab. Zunächst muss eine OAuth-Anwendung, bei GitHub [GitHOAu] oder GitLab [GitLOAu] eingerichtet werden. Nach Abschluss dieses Prozesses erhält man eine CLIENT_ID sowie ein CLIENT_SECRET. Diese müssen ebenfalls in der process.env eingetragen werden:
CLIENT_ID=<GitHub Client ID/GitLab App ID>
CLIENT_SECRET=<GitHub/GitLab Client Secret>

Darüber hinaus ist die Callback-URL in der process.env wie folgt zu setzen (vorausgesetzt der Port ist 3000):
CALLBACK_URL="http://localhost:3000/authenticate/provider/callback"

Zu guter Letzt müssen nur noch das JWT_ACCESS_SECRET sowie das JWT_REFRESH_SECRET gesetzt werden, um eigene Access-/Refresh-Token innerhalb unserer Anwendung zu generieren. Die Schlüssel dafür können beispielsweise mit dem Build-In-Modul „crypto“ in Node.js realisiert werden. Nachdem der Befehl „node“ im Terminal ausgeführt wurde, kann man mit „crypto“ wie folgt Schlüssel generieren:
require('crypto').randomBytes(32).toString('hex')

Momentan ist im Lecturer Backend GitHub voreingestellt. Um auf GitLab (der HAW) zu wechseln, muss aufgrund der selbst implementierten Passport-Strategien nur ein minimaler Schritt getätigt werden. In Zeile 20 des authenticationRouter.ts muss lediglich die „GitHubOAuth2Strategy“ durch die „GitLabOAuth2Strategy“ ersetzt werden.

Ein weiterer Vorteil der benutzerdefinierten Passport-Strategien war, dass das Nutzerprofil des Authentifizierungsproviders ausgelesen werden konnte. Die „GitLabOAuth2Strategy“ beschreibt beispielsweise wie jede andere Passport-Strategie das User-Profil:

Listing 5.8: GitLabOAuth2Strategy.ts Zeile 37 - 62

```
1         userProfile(accessToken: string, done: (err?: Error |
              null, profile?: any) => void): void {
2             var json: any = {};
3             this._oauth2.get(this.profileUrl, accessToken, (err,
                  body) => {
4                 if (err) {
5                     return done(new InternalOAuthError('Failed␣
                          to␣fetch␣user␣profile', err));
6                 } else if (typeof body === 'string') {
7                     try {
8                         json = JSON.parse(body);
9                         console.log(json);
10                    } catch (error) {
11                        return done(new Error('Failed␣to␣parse␣
                              user␣profile'));
12                    }
13                } else {
14                    return done(new Error('Failed␣to␣parse␣body'
                          ));
15                }
16                if (isGitLabJSON(json)) {
17                    done(null, this.parseProfile(json));
18                } else {
19                    return done(new Error('id,␣username,␣email␣
                          or␣name␣is␣not␣defined'));
20                }
21            });
22        }
```

In Zeile 9 der obigen Abbildung haben wir uns das gesamte Nutzerprofil ausgeben lassen, in der Hoffnung, dass ein Attribut herausgefiltert werden kann, welches zwischen Studierenden und Lecturern unterscheidet. Leider war dies nicht der Fall. Daher muss der/die Datenbankadministrator/in die Rollen innerhalb des Anwendungskontexts festlegen.

5.2 Frontend

Dieser Abschnitt befasst sich damit, welche Technologien für die Lecturer Extension zum Einsatz kamen, und auch, aus welchen Gründen sich für diese entschieden wurde. Anschließend wird auf den Aufbau aus dem Kapitel 4 eingegangen und dieser weiter verfeinert. Zum Abschluss wird über das Design diskutiert. Vor allem darüber, wie die Lecturer Extension optimal in VSCode eingebunden werden kann. Dabei wird auch die VSCode Extension Guideline [MicrExt] sowie die Color API [MicrThe] berücksichtigt. Der Code für die Frontendkomponente ist ebenfalls auf GitHub [VscProt] verfügbar.

5.2.1 Verwendete Technologien

Neben der VSCode Extension API wurde Webpack [Webpack] als Modul-Bundler und React als JavaScript-Framework zur Erstellung der Webviews verwendet.

Für Webpack wurde sich entschieden, da zum einen die Konfigurationsdatei beim Erstellen einer VSCode Extension bereits automatisch mit generiert werden kann. Zum anderen verfügt es über einige nützliche Funktionen, die uns das Aufsetzen zusammen mit React erleichtert haben.

Dazu gehört zum Beispiel das Code Splitting [WebCdSp]. Es erlaubt uns mehrere Eingangspunkte zu definieren von denen Webpack aus startet die Bundles zu generieren. Dies ist hilfreich, da der Extension Programmiercode in einem anderen Kontext (Node.js) als die React Anwendungen (Webkontext) gebündelt werden muss. Für die beiden React Anwendungen, die untereinander keine Abhängigkeiten zueinander pflegen, können dazu auch zwei verschiedene Eingangspunkte definiert werden. So entstehen am Ende mindestens drei Bundles. Eines für die VSCode Extension, eines für die Sidebar, und ein weiteres für das Course Panel. Ein weiterer Vorteil ist nun, dass Webpack so konfiguriert werden kann, dass doppelt vorkommende Abhängigkeit nicht mehrfach gebündelt, sondern in einem gemeinsam genutzten Bundle geteilt werden. Zudem unterstützt Webpack Tree Shaking [WebTrSh], eine Funktion um nicht erreichbaren JavaScript bzw. TypeScript Code zu eliminieren.

Gründe, warum sich für React entschieden wurde, werden hier nach Hartmann und Zeigermann aufgeführt [HZ20, S. 7 - 8]:

- Einfachheit:
  Die Schwierigkeit React zu erlernen ist überschaubar. Daten fließen über das virtuelle DOM nur in eine Richtung und auch die jsx-Syntax bzw. tsx-Syntax ist ähnlich zu JavaScript bzw. TypeScript und HTML gehalten.

- Kontinuität und Stabilität:
  Das Einführen von Änderungen in die React-API geschieht vorsichtig, sodass zwar neue Features eingeführt werden, diese aber abwärtskompatibel sind. Dies erspart dem Nutzer langwierige Migrationsprozesse. Ein gutes Beispiel dafür ist die React-Hooks-API [FBHook], welche u. a. auch für die Lecturer Extension verwendet wird.

- Integrierbarkeit:
  React konzentriert sich hauptsächlich über die Verwaltung des Zustands der Applikation. Es werden nur wenige Vorgaben bezüglich der Umgebung getätigt. Daher lässt sich React u. a. optimal via Webpack in die Extension einbinden.

- Ökosystem:
  React bietet eine Reihe an Tools an, die bei der Entwicklung unterstützen. Innerhalb der Lecturer Extension wurden beispielsweise u. a. die npm-Module „react-table“, „react-select“, „react-datetime-picker“ und „react-input-autosize“ verwendet. Wie die Namen dieser Module suggerieren, wurden diese ausschließlich für React entwickelt und haben bei der Gestaltung von UI-Elementen geholfen.

- Praxiserprobte Technologie:
  React wird neben Facebook auf anderen sehr bekannten Firmen wie Netflix, Microsoft, Jira oder Outlook zum Einsatz. Neue Features werden darüber hinaus normalerweise auf der riesigen facebook.com-Codebasis getestet.

5.2.2 Aufbau

In diesem Abschnitt wird das erweiterte Klassendiagramm der React Anwendungen genauer betrachtet und erklärt. Da dieses sehr umfassend ist, wurde es in 2 Teile mit den wichtigsten Klassen aufgeteilt. „onClick“- und „onChange“-Funktionen sowie geerbte React-Funktionen (render(), onComponentDidMount() etc.) werden nicht berücksichtigt, da diese wenig Aussagekraft haben. Gestartet werden kann das Programm über die Taste F5 (in VSCode der Shortcut für run -> start debugging).

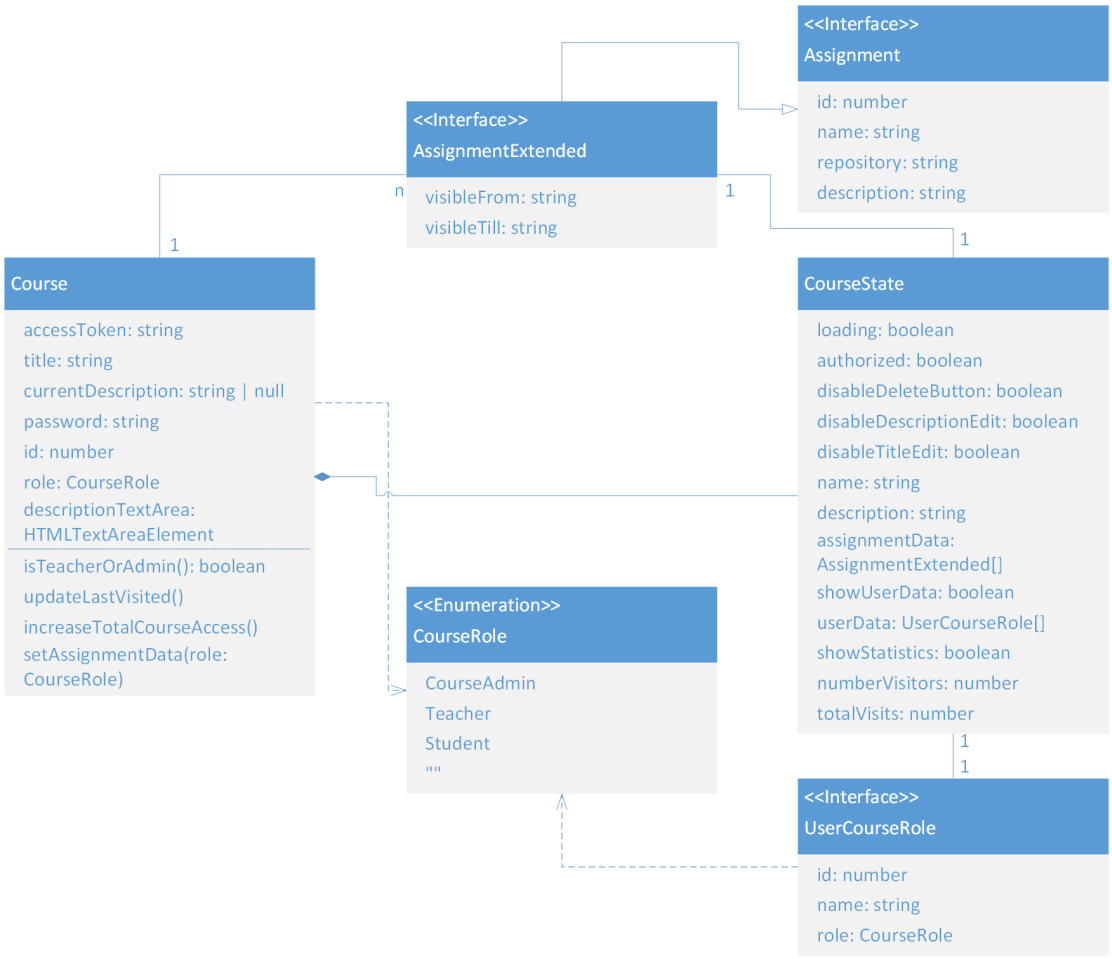

Abbildung 5.1: Erweitertes Klassendiagramm Frontend Teil 1

Im der ersten Abbildung 5.1 betrachten wir die Architektur des Course Panels. Dieses kann nur durch die Sidebar aufgerufen werden. Das liegt daran, dass der Befehl das Course Panel zu öffnen intern ist.

Interne Befehle werden in VSCode Extensions genauso wie extern zugängliche Befehle implementiert. Der Unterschied liegt darin, dass sie in der package.json unter „commands“ nicht aufgeführt werden.

Die Klasse „Course“ ist im Course Panel der zentrale Part. Sie enthält im State u. a. die Variable „authorized“ um zu überprüfen, ob dem/der Nutzer/in die Kurseinschreibeoberfläche oder Kursinhalte angezeigt werden. Ist letzteres der Fall, dann wird zusätzlich abgefragt, ob der/die authentifizierte Nutzer/in erweiterte Rechte im Kurskontext hat. Ist dies der Fall, kann er zusätzliche Aktionen ausführen, wie z. B. das Bearbeiten von Benutzerrechten, Kurstitel und Kursbeschreibung.

Neben den Kurs- und Nutzerdaten werden an dieser Stelle ebenfalls die Assignments innerhalb des Kurses angezeigt. Ein Lecturer kann ihre Sichtbarkeit hier überprüfen und Assignments bei Bedarf löschen.

Anzumerken ist noch, dass der State (hier als CourseState) in React etwas besonderes ist. Ändert sich eines der Attribute dieser Klasse, wie z. B. das Assignment-Array, so wird die Kursseite neu gerendert.

In der nächsten Grafik 5.2 kann ein Ausschnitt des generierten Klassendiagramms der Sidebar betrachtet werden. Die zentrale Klasse an dieser Stelle ist die Navbar. Sie hält im State das „accessToken“ sowie die Profildaten und „weiß“ dadurch immer, ob ein/e Nutzer/in gerade authentifiziert ist oder nicht. In letzterem Fall wird der/die Nutzer/in automatisch abgemeldet.

Das Abmelden kann ebenfalls nach 30 Minuten automatisch durch die AutoLogoutCountdown-Komponente getriggert werden. Diese prüft einmal die Sekunde, wie lange das Access Token noch gültig ist. Alternativ kann die Session aber auch durch einen Knopfdruck verlängert werden. Dabei wird das Refresh Token verwendet, um ein neues Access Token anzufragen.

Der Grund, warum sich für ein Access Token entschieden wurde liegt darin, dass es im Rahmen einer Online-Programmierplattform wichtig ist, auf eigene Git Ressourcen zurückgreifen zu können. Momentan wird in der Anwendung ein selbst erstelltes JSON Web Token verwendet. Das Token, welches über die OAuth 2.0 Authentifizierung in dieser Anwendung vergeben wird könnte beim Authentifizierungsvorgang jedoch mit bzw. stattdessen übergeben werden. Und eben mit diesem Token, wäre der Zugriff auf die eigenen Git Ressourcen gewährleistet.

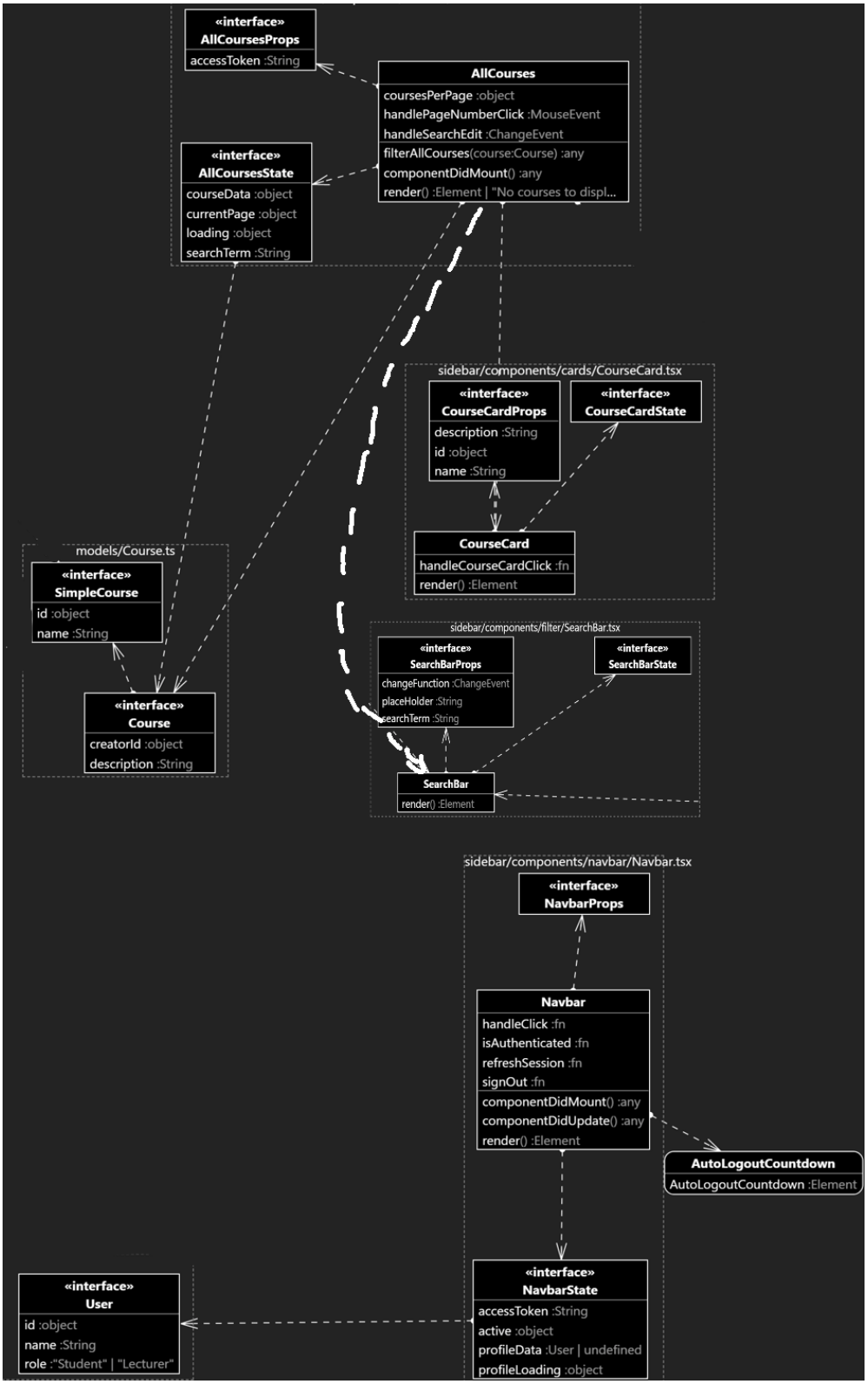

Abbildung 5.2: Erweitertes Klassendiagramm Frontend Sidebar Ausschnitt

Als Beispielkomponente neben der Navabar wurde hier die Kursübersicht mit eingeblendet. Was auffallen mag ist, dass zwischen diesen Klassen, wie auch anderen Klassen und der Navbar keine Abhängigkeiten bestehen, obwohl zwischen diesen Klassen navigiert werden kann. Dies liegt daran, dass der React Router die Navigation innerhalb der React Anwendung implementiert.

Im weiteren Verlauf könnte nun auf jede Komponente und ihre Umsetzung eingegangen werden. Darauf wird jedoch zu Gunsten des folgenden Kapitels 6 verzichtet. Dort wird nochmal ausführlich behandelt, welche Funktionen von der Extension letzten Endes umgesetzt wurden und warum.

5.2.3 Design

In diesem Kapitel wird genauer auf das Design unserer Lecturer Extension eingegangen. Um den generellen Stil aus dem VSCode Editor auf die Webviews anzuwenden, kann auf die Cascading Style Sheets (CSS)-Dateien „reset.css“ und „vscode.css“ aus dem offiziellen Beispiel für Webviews von Microsoft zurückgegriffen werden. Dadurch erhalten HTML-Elemente wie Buttons oder Input-Felder automatisch den VSCode-Stil. Eingebunden in die Webviews wird es wie folgt. Zunächst muss eine VSCode Uniform Resource Identifier (URI) für beide CSS-Dateien angelegt werden:

Listing 5.9: Einbinden von Layout-Dateien, SidebarProvider.ts Zeile 142 - 147

```
1         const styleResetUri = webview.asWebviewUri(
2         vscode.Uri.joinPath(this._extensionUri, "media", "reset.
              css")
3         );
4         const styleVSCodeUri = webview.asWebviewUri(
5         vscode.Uri.joinPath(this._extensionUri, "media", "vscode
              .css")
6         );
```

Anschließend können die Layout-Dateien über Stylesheets im HTML-Code der Webviews (hier SidebarProvider.ts) eingebunden werden:

...
<link href="$styleResetUri" rel="stylesheet">
<link href="$styleVSCodeUri" rel="stylesheet">
...

Das Problem hierbei ist jedoch, dass dadurch nicht alle HTML-Elemente abgedeckt werden. Daher musste ein Weg gefunden werden, auch die React-Komponenten im VSCode Stil zu designen.

Da VSCode für solche Fälle CSS-Variablen anbietet, auf die zurückgegriffen werden kann [MicrThe], wurde die Lecturer komplett in CSS gestaltet. Der Prozess ist jedoch sehr langwierig, da VSCode über 500 solcher Variablen definiert. Teilweise wurde auch die Erfahrung gemacht, dass die gleiche Variable in einigen Color Themes anwendbar ist, in anderen jedoch nicht. Daher kann man, bzw. muss man einige Farben in der package.json für bestimmte Themes überschreiben. Dies funktioniert z. B. wie folgt:

Listing 5.10: Farbvariablen überschreiben, package.json Zeile 21 - 41

```
1         "contributes": {
2             "colors": [
3             {
4                 "id": "override.background",
5                 "description": "In case we want to change the
                      table's default cell color",
6                 "defaults": {
7                     "dark": "#1e1e1e",
8                     "light": "#FFFFFF",
9                     "highContrast": "editor.background"
10                }
11            },
12            {
13                "id": "override.selection.background",
14                "description": "In case we want to override the
                      table's default cell color on selection",
15                "defaults": {
16                    "dark": "#264f78",
17                    "light": "#add6ff",
18                    "highContrast": "editor.selectionBackground"
19                }
20            }
21            ],
```

Anwenden kann man die neu definierten Farbvariablen folgendermaßen:

Listing 5.11: Farbvariablen anwenden, index.css Zeile 54

```
1     background:var(--override-background, var(--vscode-editor-
          background));
```

Die neu definierte Farbvariable ist zuerst aufzurufen. Anschließend wird die VSCode-Variable als Fallback verwendet, für Themes in denen die benutzerdefinierte Variable nicht definiert ist.

Noch aufwendiger als das Überschreiben der Farbvariablen für bestimmte Themes war jedoch das Anpassen von importierten React-Modulen, da diese teilweise Inline-Styles, also CSS in JavaScript bzw. TypeScript verwendet haben. Ein Beispiel, nur für eine einfache Select-Komponente musste folgender Aufwand betrieben werden:

Listing 5.12: react-select Styling, Assignment.tsx Z. 107 - 160

```
1     const customStyles = {
2         control: (provided: CSSObject, state: ControlPropsType)
              => ({
3             ...provided,
4             width: "100%",
5             border: state.isFocused
6             ? "1px solid var(--vscode-focusBorder)"
7             : "none"
8         }),
9         placeholder: (provided: CSSObject) => ({
10            ...provided,
11            color: "var(--vscode-input-placeholderForeground)"
12        }),
13        singleValue: (provided: CSSObject) => ({
14            ...provided,
15            color: "color: var(--vscode-input-foreground)"
16        }),
17        dropdownIndicator: (provided: CSSObject) => ({
18            ...provided,
19            color: "var(--vscode-list-
                  inactiveSelectionForeground)",
```

```
20              backgroundColor: "var(--vscode-input-background)",
21              borderRight: "1px solid white",
22              ":hover": {
23                  color: "var(--vscode-list-
                        activeSelectionForeground)",
24              },
25              ":focus": {
26                  color: "var(--vscode-list-
                        activeSelectionForeground)",
27              }
28          }),
29          valueContainer: (provided: CSSObject) => ({
30              ...provided,
31              color: "color: var(--vscode-input-foreground)",
32              backgroundColor: "var(--vscode-input-background)"
33          }),
34          menu: (provided: CSSObject) => ({
35              ...provided,
36              backgroundColor: "var(--vscode-list-
                    inactiveSelectionBackground)",
37              color: "var(--vscode-list-
                    inactiveSelectionForeground)",
38          }),
39          option: (provided: CSSObject, state: OptionPropsType) =>
                ({
40              ...provided,
41              backgroundColor: state.isFocused
42              ? "var(--vscode-list-hoverBackground)"
43              : state.isSelected
44              ? "var(--vscode-list-activeSelectionBackground)"
45              : "var(--vscode-list-inactiveSelectionBackground)",
46              color: state.isFocused || state.isSelected
47              ? "var(--vscode-list-activeSelectionForeground)"
48              : "var(--vscode-list-inactiveSelectionForeground)"
49          }),
50          input: (provided: CSSObject) => ({
```

```
51              ...provided,
52              color: "color: var(--vscode-input-foreground)"
53          })
54      };
```

Anschließend musste die „customStyles“-Variable nur noch mit den sogenannten Props übergeben werden. Props sind in React Argumente, die einer Komponente übergeben werden können [FBProp].

Doch auch mit „react-table“ gab es Probleme, obwohl dort der Mehraufwand weggefallen ist, da es mit CSS gestylt werden kann. Hier wurden Checkboxen fälschlicherweise markiert, obwohl diese deaktiviert waren. Daher haben wir als erster folgenden Lösungsansatz entwickelt und ebenfalls auf GitHub dokumentiert [TL2021]:

Listing 5.13: React Table Issue

```
1   Header: ({ toggleRowSelected, rows, selectedFlatRows }) => {
2       const style = {
3           cursor: 'pointer',
4       };
5       const checked: boolean = rows.length > 1 && rows.length
            - selectedFlatRows.length === 1; // In my case: If
            all but one rows are selected -> all selectable rows
            are selected and header checkbox should be checked
6       const title: string = 'Toggle All Rows Selected';
7       const indeterminate: boolean = !checked &&
            selectedFlatRows.length > 0; // If my header
            checkbox is not checked (therefore not all rows are
            selected) but some rows are selected -> header
            checkbox is indeterminate
8       const disabled: boolean = rows.length === 1; // In my
            case: If only one row exists, it is a row with a
            disabled checkbox. Hence, we can disable the header
            checkbox because it can and should not modify the
            row's selection.
9       const overiddenOnChange = (event: React.ChangeEvent<
            HTMLInputElement>) => {
10          rows.forEach(row => {
```

```
11                  // The condition whether a row is selectable or
                        not
12                  if (isJwtPayloadWithExp(payload) && row.original
                        .id !== payload.userId) {
13                      toggleRowSelected(row.id, event.
                            currentTarget.checked);
14                  }
15              });
16          };
17          const modifiedToggleAllRowsProps = {
18              onChange: overiddenOnChange,
19              style: style,
20              checked: checked,
21              title: title,
22              indeterminate: indeterminate,
23              disabled: disabled
24          };
25          return (
26          <div>
27          <CheckBox {...modifiedToggleAllRowsProps} />
28          </div>
29          );
30      },
```

Diese Tabelle wurde eingesetzt, um die Nutzerrechte zu bearbeiten. Da ein/e Nutzer/in seine eigenen Nutzungsrechte nicht bearbeiten darf, mussten die Props für die Checkbox im Tabellen-Header angepasst werden. So musste der Header-Checkbox signalisiert werden, dass bereits alle anzuklickenden Checkboxen markiert sind, wenn alle bis auf eine Tabellenzeile ausgewählt wurden. Außerdem musste die onChange-Funktion der Checkbox überschrieben werden, indem dort nicht alle Checkboxen ohne Überprüfung markiert werden, sondern nur jene mit Ausnahme der aus der Spalte des/der aktuellen Nutzers/-Nutzerin.

Zusätzlich zum Design wurde darauf geachtet, dass möglichst viel mit CSS-Flexbox gearbeitet wird. CSS-Flexbox-Container passen sich der Bildschirmgröße an, und sind daher ein ideales Werkzeug, um das im Abschnitt 2.1.1 beschriebene RWD umzusetzen. In den Abbildungen im Anhang A wird vielleicht auch deutlich, dass mit dem automatischen Logout nach 30 Minuten und dem HAW-Logo versucht wurde, dem Nutzer zu vermitteln, dass er sich auf einer Seite der HAW Hamburg befindet. Für die Icons innerhalb der Extension wurden außerdem ausschließlich der offizielle Icon-Font für VSCode von Microsoft verwendet.

5.3 Tests

Abschließend, wurde das System ausführlich getestet. Dabei lag der Hauptfokus auf den Backendtests, weil das Frontend komplett von den Daten aus dem Backend abhängig ist. Das Frontend dagegen soll nach den VSCode Extension Guidelines möglichst einfach gehalten werden [MicrExt]. Es ist daher lediglich dazu da, die vom Backend bereitgestellten Daten anzuzeigen.

Beim Testen des Backends wurden ausschließlich Blackbox Tests durchgeführt. Dies bedeutet, dass auf die Implementierung nicht genauer eingegangen wurde, sondern nur auf die Schnittstellen bzw. die hier angebotenen API Endpunkte des Systems. Das Testen der Datenweitergabe und Kommunikation zwischen Systemen wird dabei auch Interface Testing genannt [IEE90].

Für diese Art von Testen eignet sich das bereits erwähnte Tool Postman. Mit diesem Tool können sogenannte Collections erstellt werden. Hier kann man eine Reihe von API-Anfragen definieren, die in einer Collection automatisch hintereinander abgerufen werden. Zusätzlich ist es möglich vor und nach jedem API-Aufruf JavaScript-Code auszuführen. Dabei kann auch auf die Anfrage des Clienten bzw. die Antwort des Servers zugegriffen werden. So ist es möglich Werte zwischen Anfrage und Antwort abzugleichen. Zudem können ebenfalls Werte über Variablen der Collection zwischen den API-Anfragen weitergegeben werden.

Mit diesem Funktionsumfang wurde jeder API Endpunkt unserer Anwendung mindestens einmal getestet. Die Skripts zu den Tests wurden durch .json-Dateien exportiert und sind ebenfalls auf GitHub abrufbar [VscMock]. Mit dem NoAuth-Skript und dem 404Test-Skript werden lediglich Testfälle aufgerufen in denen der Nutzer nicht authentifiziert ist oder versucht API Endpunkte aufzurufen, die nicht existieren.

Die Antwort von dem System darauf verlief wie erwartet. Alle falschen Anfragen wurden kontrolliert abgelehnt. Was diese Tests jedoch aufgedeckt haben, ist, dass die Reihenfolge der Middleware im Backend wichtig ist. Werden beispielsweise die Parameter vor der Authentifizierung überprüft, so wird eine sogenannte „Bad Request“ statt einer Meldung, dass kein Zugriff erlaubt ist herausgegeben.

Das interessantere Skript für den Lecturer ist das Lecturer_Auth-Skript. Hier testet der Lecturer alle für ihn relevanten Endpunkte. Vor Nutzung muss das Skript jedoch in Zeile 1594 angepasst werden. An dieser Stelle muss ein Access Token mit Lecturer-Rechten übergeben werden.

Aufgrund des großen Umfangs aller Tests wird hier nicht auf jeden Test eingegangen, sondern nur das grobe Konzept dahinter beschrieben. Außerdem wird erklärt, welche Erkenntnisse bzw. Fehler während des Testens gesammelt werden konnten.

Unter einem der ersten 5 Tests wurde überprüft, ob der/die Nutzer/in tatsächlich ein Lecturer ist. Ein paar Testfälle später wurden ein Assignment, als auch ein Kurs erstellt. Gleichzeitig wurde abgefragt, ob die übergebenen Parameter tatsächlich mit denen des erstellten Elements übereinstimmen. Wieder ein paar Anfragen später, wurde das Assignment einem Kurs zugewiesen. Dies wurde u. a. aus dem Grund getan, um das Verhalten bei Änderungen der Sichtbarkeit des Assignments im Kurskontext zu überprüfen. Hierbei ist durch die Test aufgefallen, dass im Quellcode ein „Date“-Objekt fälschlicherweise als String interpretiert wird. Der Fehler konnte anschließend behoben werden. Zu guter Letzt wurde getestet, ob ein gelöschtes Assignment aus einem Kurs immer noch mit diesem verbunden wird. Dies war nicht der Fall. Abgesehen von dem einen beschriebenen Fehler liefen alle anderen Tests reibungslos durch.

6 Validierung

Dieses Kapitel dokumentiert zunächst die Umsetzung der dokumentierten Anwendungsfälle in Abschnitt 3.5. Anschließend wird auf die aufgelisteten (nicht-)funktionalen Anforderungen eingegangen und evaluiert inwiefern diese realisiert werden konnten. Dabei wird auf grafische Ausschnitte aus der Anwendung verwiesen. Diese sind für eine bessere Übersichtlichkeit erst im Anhang einsehbar.

6.1 Authentifizierung (3.5.1)

Der Anmeldevorgang in der Lecturer läuft wie folgt ab. Zunächst wird die Schaltfläche „Sign in“ im Startbildschirm A.1 betätigt. Darauf öffnen sich ein neues Fenster im Browser. Hier muss sich der User bei GitHub oder GitLab, je nach Einstellung authentifizieren. Nach erfolgreichem Anmelden wird im Webbrowser angezeigt, dass die Authentifizierung erfolgreich abgeschlossen ist. Wie vom Anwendungsfall 3.5.1 verlangt, wird er nun auf eine Seite mit der Profilübersicht weitergeleitet A.2.

Für die Implementierung wurde eine Token-basierter Ansatz gewählt. Die Gründe dafür sind bereits in 5.2.2 beschrieben.

Ein Problem was sich bei der Implementierung ergeben hat, ist nur, dass der/die Nutzer/in keine Fehlermeldung erhält, wenn keine Verbindung zum Backend besteht. Da der Anmeldevorgang extern ist, und die Anwendung lediglich darauf wartet, bis die Callback-URL aufgerufen wird, wäre dies nur durch ein Timeout möglich. Da aus Studien hervorgeht, dass Nutzer/innen sowieso nur bereit sind 2 Sekunden lang auf das Laden von Inhalten zu warten [Nah04, S. 153], wird davon abgesehen.

6.2 Kurserstellung (3.5.2)

Für die Kurserstellung muss im Menü „Create Course“ ausgewählt werden. Anschließend öffnet sich eine neue Seite mit einem Formular A.3. Bei korrekter Eingabe und Übermittlung der Daten wird ein neuer Kurs erstellt. Andernfalls erscheint wie geplant, ein Hinweis welches Feld (anders) auszufüllen ist A.4. Existiert ein Kurs mit dem gleichen Namen dagegen schon, wird die Anfrage abgefangen und der/die Nutzer/in mit einer Informationsnachricht über diesen Konflikt informiert A.5. Wichtig war an dieser Stelle darauf zu achten, dass die eingegebenen Daten erhalten bleiben. So muss der/die Nutzer/in die Felder beim nicht erfolgreichen Versuch des Erstellens eines Kurses nicht nochmal ausfüllen.

Beim Entwickeln dieses Lösungsansatzes sind keine Probleme aufgetreten. Es sei allerdings anzumerken, dass der Kursbereich im Frontend nicht berücksichtigt wurde, obwohl er im Backend bereits implementiert ist. Dies liegt daran, dass die einzige Funktion für den Kursbereich das Filtern nach diesem gewesen wäre. Dies kann und wird im folgenden Kapitel jedoch schon mit anderen Eigenschaften beschrieben. Daher hatte die Berücksichtigung des Kursbereichs eine untergeordnete Priorität.

6.3 Anlegung von Kursinhalten (Assignments) (3.5.3)

Im nächsten Schritt wird betrachtet, wie dem eben beschriebenen Kurs Kursinhalte, darunter Assignments, zugewiesen werden können. Für diesen Vorgang muss der/die Nutzer/in zunächst die Schaltfläche „Create Assignments“ im Hauptmenü tätigen, um auf die Seite zur Erstellung eines Assignments A.6 zu gelangen. Anschließend muss er ähnlich wie bei der Kurserstellung wieder ein paar Textfelder ausfüllen. Hierbei wird mittels einer Regular expression (RegEx) überprüft, ob eine URL übergeben wurde, die der „https://“-Syntax entspricht. Ist dies nicht der Fall, so wird ein entsprechender Hinweis dazu angezeigt A.7.

Anschließend kann der/die Nutzer/in über den Menüpunkt „Assignments“ sich alle Assignments anzeigen lassen. Hier kann er sein soeben erstelltes Assignment auswählen. Über dieses wiederum kann er durch ein Dropdown-Menü einen Kurs zuweisen A.8. Das Dropdown-Menü ist für die beste Performance so gestaltet, dass erst nach Eingabe von mindestens einem Buchstaben die Optionen laden. Der API Endpunkt zum Backend ist nämlich so konzipiert, dass dann nur die Kurse aus dem Backend abgerufen werden, die den entsprechenden String enthalten. Zudem werden die Vorschläge im Frontend gecacht. So wird die Anzahl der Aufrufe und damit der Datenverkehr minimiert. Dies ist vor allem wichtig im Hinblick darauf, dass das Backend mit der Zeit eine immer größere Anzahl an Kursen durchsuchen muss.

Neben dem Kurs kann auch ein Datum für die Sichtbarkeit angegeben werden, ab wann bzw. bis wann das Assignment für die Kursteilnehmer angezeigt werden soll. Das Datum selbst wird überprüft, und dem Nutzer wird durch einen roten Hintergrund markiert, ob er ein gültiges Datum eingegeben hat oder nicht.

Ein Problem, was sich bei der Implementierung dieses Schrittes ergeben hat, ist, dass eine „https“-URL für ein Assignment angegeben werden kann, die nicht auf ein Git Repository verweist. Wie damit umgegangen wurde, wird im folgenden Schritt betrachtet.

6.4 Sortierung, Filtration, Markierung und Aufrufen von Kursen (3.5.4)

Das Sortieren, Filtern, Markieren und Aufrufen von Kursen ist gerade für den Lecturer essenziell, da mit zunehmender Zeit die Anzahl seiner Kurse wachsen kann und die Übersicht darüber gleichzeitig abnehmen. Um dem entgegen zu wirken, soll er alte Kurse ausblenden, wichtige Kurse hervorheben und bei Bedarf gezielt nach einem Kurs suchen können. Die ihm zur Verfügung stehenden Möglichkeiten werden durch die folgende Abbildung A.9 veranschaulicht. Neben einer Suchleiste, einem Filter nach bestimmten Kurseigenschaften und den Buttons zum Favorisieren bzw. Verstecken, existiert links neben der Suchleiste darüber hinaus ein Menü zur Sortierung. Hier kann ausgewählt werden, ob die Kurse alphabetisch, nach Erstellungszeitpunkt oder nach den zuletzt besuchten Kursen sortiert werden sollen.

Mit dem Klicken auf die Kurskarte kann der Nutzer nun auf die Kursseite gelangen. Dies ist eine der wenigen Stellen, an denen gegen die VSCode Extension Guidelines verstoßen wurde. Diese besagen nämlich, dass ein Webview keine Schaltfläche für das Öffnen eines neuen Webviews enthalten sollte [MicrExt]. Da die Sidebar Kursinhalte jedoch nicht nutzerfreundlich anzeigen konnte, musste dagegen leider verstoßen werden.

Ideen dieses Problem zu umgehen verliefen im Entwicklungsprozess leider ins Leere. Eine Idee die aufkam war beispielsweise Dateien direkt von einem Server zu laden und lediglich im Code Editor anzuzeigen. Weil es in VSCode möglich ist andere Extensions als Abhängigkeit zu verwenden [MicExM] wäre es interessant gewesen dafür auf die Remote Development Extensions von Microsoft bzw. GitHub zuzugreifen. Hierüber hätte man sich per SSH mit einem Server verbinden oder gar direkt auf GitHub-Dateien zugreifen können. Da die Projekte dieser Extensions jedoch nicht Open Source sind, und die Lizenzbedingungen eine Nutzung als Abhängigkeit nicht erlauben [MicReD], musste diese Idee leider verworfen werden.

Nun angekommen im Kursfenster, kann der/die Nutzer/in das Assignment ausklappen und über den „Clone this Assignment“-Button auf die Inhalte zugreifen A.10. Hier ist anzumerken, dass es zu dem in Abbildung A.11 beschriebenen Fehler kommt, wenn die Repository URL auf kein Git Repository verweist.

6.5 Kurseinschreibung (3.5.5)

Für die Kurseinschreibung wurde wie im Anwendungsfall beschrieben die Methode für Selbsteinschreibung implementiert. Diese wurde mit der Begründung bevorzugt, dass sich der Lecturer in diesem Fall nicht aktiv um die Teilnahme an der freiwilligen Plattform beschäftigen muss, sondern nur einmalig beim Erstellen. Das Passwort für die Anmeldung ist dabei verschlüsselt auf der Datenbank abgelegt.

Die Sicht des Studierenden bei der Anmeldung für einen Kurs ist in Abbildung A.12 beschrieben. Bei falscher Eingabe des Passworts wird dies dem/der Nutzer/in an dieser Stelle durch eine Informationsnachricht signalisiert. Ansonsten wird er/sie zu den Kursinhalten weitergeleitet. Was an Abbildung A.12 auch noch auffallen mag, ist, dass dem Studierenden nur die Funktionen angezeigt werden, welche er auch wirklich ausführen kann. Menüpunkte für die Kurserstellung o. Ä. sind daher in dieser Sicht nicht vorhanden.

6.6 Vergeben von erweiterten Nutzerrechten im Kurskontext (3.5.6)

Nutzer*innen eines Kurses kann sich der Lecturer durch die Schaltfläche „Show Participants“ anzeigen lassen. Zum Bearbeiten der Rollen wurde wie in Abschnitt 5.2.1 bereits erwähnt react-table verwendet. In der UI schaut dies wie in Abbildung A.13 aus. Der Vorteil an der Implementierung mit react-table ist, dass der Lecturer die Rechte mehrerer Nutzer gleichzeitig bearbeiten kann, sowie Filter und das auf- bzw. absteigende Sortieren auf die Tabelle anwenden kann. Zudem ist es für den Lecturer möglich, geänderte Rollen zurückzusetzen, bevor er diese bestätigt. Wie in 5.2.1 bereits beschrieben, wurde ebenfalls berücksichtigt, dass er seine eigene Rolle nicht bearbeiten kann.

6.7 Kursergebnisse / Kursstatistiken 3.5.7

Etwaige Statistiken in dieser Anwendung können unter Abbildung A.14 betrachtet werden. Dabei fällt auf, dass lediglich angezeigt wird, wie viele Nutzer*innen den Kurs innerhalb der letzten 7 Tagen besucht haben. Wünschenswerter wäre hier ein Dashboard mit erweiterten Statistiken, wie in Kapitel 2.1 erwähnt. Da der Mehraufwand UI-Komponenten im Stile von VSCode zu designen jedoch sehr hoch ist (siehe Abschnitt 5.2.3), wurde hierauf zu Gunsten fachlicher Funktionen verzichtet.

Auch die Nutzerergebnisse können in der Anwendung noch nicht angezeigt werden. Dies liegt daran, dass darauf bewusst verzichtet wurde, weil innerhalb der Anwendung noch keine Funktion zur Abgabe von Assignments implementiert wurde. Letzteres hätte auch nicht mehr unmittelbar mit dem Lecturer zu tun gehabt, der hier in der gesamten Arbeit im Mittelpunkt steht.

6.8 Umgesetztes Design

Zu guter Letzt soll noch kurz auf das Design eingegangen werden. Wie Abbildung A.15 zeigt, ist die Sidebar responsive und passt die Menüansicht der Bildschirmgröße an. Wird der Bildschirm kleiner, dann wird dieses nicht mehr durchgehend angezeigt, wie Abbildung A.17 veranschaulicht. Es erscheint stattdessen erst beim Klicken auf das Menü-Icon. Auch die Kursseite passt sich völlig responsive jeglicher Bildschirmgröße an. Zudem wurden weitestgehend die Themes in VSCode unterstützt, wie Abbildung A.16. Vor allem für Personen mit eingeschränkter Sicht kann beispielsweise die Unterstützung des „High Contrast“-Modus sehr wichtig sein.

7 Fazit

Zum Abschluss dieser Arbeit werden die Erkenntnisse über den Verlauf des Projekts zusammengefasst.

Bereits nach relativ kurzer Zeit ist aufgefallen, dass die Einarbeitungszeit in ein solches Projekt relativ langwierig ist. Übliche Vorgehensweisen, wie das de facto Standard-Setup der React-App können nicht verwendet werden, weil diese mit der VSCode Extension API interagieren und vor allem in diese integriert werden muss. Darüber hinaus ist es durch das Nicht-Benutzen eines css-Frameworks sehr langwierig gewesen alle Komponenten an VSCode UI anzupassen. Durch die bisherig wenige Erfahrung im VSCode Extension Umfeld, kamen zudem immer mal wieder neue Hürden auf, wie die Anpassung der Farbpalette oder das Versuchen des Einbindens von anderen Extensions als Abhängigkeit. So hohe Hürden, das letzten Endes auch an einer Stelle gegen die VSCode Extension Guidelines verstoßen werden musste, um die Kursdaten nutzerfreundlich anzuzeigen.

Was jedoch auch klar geworden sein sollte, ist, dass das Einbinden von Webviews, wenn es denn richtig gemacht wird, sehr gut funktioniert. Selbst das Authentifizieren sowie Aufrufen von API Endpunkten aus der App heraus war gar kein Problem. Zudem konnte über die globale „vscode“-Variable unkompliziert mit der VSCode Extension API kommuniziert werden. So war das Aufrufen von Informationsmeldungen oder Fehlerbenachrichtigungen über den Code Editor kein Problem.

Abschließend kann man zusammenfassen, dass das Endprodukt ein durch getestetes erfolgreiches Programm war, welches den Bezug zum Fachlichen aufgrund der Hürden im Design etwas verloren hat.

Literaturverzeichnis

[AA12] Al-Ajlan, Ajlan S.: A Comparative Study Between E-Learning Features. Version: 2012. DOI 10.5772/29854. In: Pontes, Elvis (Hrsg.); Silva, Anderson (Hrsg.); Guelfi, Adilson (Hrsg.) ; Kofuji, Sergio T. (Hrsg.): *Methodologies, Tools and New Developments for E-Learning*. Rijeka : InTechOpen, 2012, Kapitel 10. DOI 10.5772/29854

[Bla20] Blank, Ilona: *GitLab*. Version: 2020. https://userdoc.informatik.haw-hamburg.de/doku.php?id=docu:gitlab. – [online], Abruf: 04-07-2021

[BlaCont] Blackboard Inc.: *Freigeben von Inhalten*. https://help.blackboard.com/de-de/Learn/Instructor/Ultra/Course_Content/Release_Content. – [online], Abruf: 27-07-2021

[BlaHome] Blackboard Inc.: *Blackboard homepage*. https://www.blackboard.com. – [online], Abruf: 27-02-2021

[BlaRole] Blackboard Inc.: *Roles and Priviliges*. https://help.blackboard.com/de-de/Learn/Administrator/Hosting/User_Management/Roles_and_Privileges. – [online], Abruf: 20-07-2021

[BlaStat] Blackboard Inc.: *Run Statistics Reports*. https://help.blackboard.com/de-de/Learn/Administrator/Hosting/System_Management/Reports/Running_Statistics_Reports. – [online], Abruf: 23-07-2021

[BlaTomc] Blackboard Inc.: *About Apache Tomcat*. https://help.blackboard.com/Learn/Administrator/Hosting/Architecture/About_Apache_Tomcat. – [online], Abruf: 20-07-2021

[Boe81] Boehm, Barry W.: *Software Engineering Economics*. 1st. USA : Prentice Hall PTR, 1981. ISBN 0138221227

[Boe84] Boehm, Barry W.: Software Engineering Economics. DOI 10.1109/TSE.1984.5010193. In: *IEEE Transactions on Software Engineering* SE-10 (1984), Nr. 1, S. 4–21

[VscMock] Bremer, Philip A.: *vsc-mock-backend*. https://github.com/PhilipAB/vsc-mock-backend. – [online], Abruf: 17-10-2021

[VscProt] Bremer, Philip A.: *vsc-proto*. https://github.com/PhilipAB/vsc-prototype. – [online], Abruf: 18-10-2021

[CBCO+11] Carter, Janet; Bouvier, Dennis; Cardell-Oliver, Rachel; Hamilton, Margaret; Kurkovsky, Stanislav; Markham, Stefanie; McClung, O. W.; McDermott, Roger; Riedesel, Charles; Shi, Jian ; White, Su: Motivating All Our Students? DOI 10.1145/2078856.2078858. In: *Proceedings of the 16th Annual Conference Reports on Innovation and Technology in Computer Science Education - Working Group Reports*. Association for Computing Machinery (ITiCSE-WGR '11). S. 1–18. – ISBN 9781450311229

[Che76] Chen, Peter Pin-Shan: The Entity-Relationship Model—toward a Unified View of Data. DOI 10.1145/320434.320440. In: *ACM Trans. Database Syst.* 1 (1976), März, Nr. 1, S. 9–36. ISSN 0362–5915

[CodProj] Codeboard: *Project*. https://codeboard.io/docs/project. – [online], Abruf: 12-08-2021

[Cos21] Costa, Carlos: *ltijs*. Version: 2021. https://cvmcosta.me/ltijs/#/. – [online], Abruf: 04-07-2021

[DJ20] Darko, Charles; Jagger, Daniel: How Can We Manage the Blackboard Learning Management System to Enhance Student's Learning? DOI 10.1145/3417188.3417205. In: *Proceedings of the 2020 4th International Conference on Deep Learning Technologies (ICDLT)*. Association for Computing Machinery (ICDLT 2020). S. 49–54. – ISBN 9781450375481

[FBHook] Facebook: *Hooks Reference*. https://reactjs.org/docs/hooks-reference.html. – [online], Abruf: 18-10-2021

[FBProp] Facebook: *Component and Props*. https://reactjs.org/docs/components-and-props.html. – [online], Abruf: 18-10-2021

[GHS20] Gandrass, Niels; Hinrichs, Torge ; Schmolitzky, Axel: Towards an Online Programming Platform Complementing Software Engineering Education. http://ceur-ws.org/Vol-2531/paper05.pdf. In: *Software Engineering im Unterricht der Hochschulen* 2531 (2020), Februar, S. 27–35. ISSN 1613–0073

[GitHOAu] GitHub: *Creating an OAuth App*. https://docs.github.com/en/developers/apps/building-oauth-apps/creating-an-oauth-app. – [online], Abruf: 17-10-2021

[GitLOAu] GitLab: *User owned applications*. https://code.visualstudio.com/api/references/theme-color. – [online], Abruf: 17-10-2021

[GitOAut] GitLab: *GitLab as OAuth2 authentication service provider*. https://docs.gitlab.com/ee/integration/oauth_provider.html. – [online], Abruf: 04-07-2021

[GKRS17] Gharbi, Mahbouba; Koschel, Arne; Rausch, Andreas ; Starke, Gernot: *Basiswissen für Softwarearchitekten : Aus- und Weiterbildung nach iSAQB-Standard zum Certified Professional for Software Architecture – Foundation Level*. dpunkt.verlag http://ebookcentral.proquest.com/lib/hawhamburg-ebooks/detail.action?docID=5150375. ISBN 9783960883326

[GSB08] Goedicke, Michael; Striewe, Michael ; Balz, Moritz: Computer aided assessments and programming exercises with JACK. Universität Duisburg-Essen, Institut für Informatik und Wirtschaftsinformatik (ICB) (28). – ICB-Research Report. – Online–Ressource. http://hdl.handle.net/10419/58160

[HBPS21] Hinrichs, Torge; Burau, Henri; Pilgrim, Jens von ; Schmolitzky, Axel: A Scaleable Online Programming Platform for Software Engineering Education. http://ceur-ws.org/Vol-2814/paper-A6-2.pdf. In: Götz, Sebastian (Hrsg.); Linsbauer, Lukas (Hrsg.); Schaefer, Ina (Hrsg.) ; Wortmann, Andreas (Hrsg.): *Proceedings of the Software Engineering 2021 Satellite Events, Braunschweig/Virtual, Germany, February 22 - 26, 2021* Bd. 2814, CEUR-WS.org. – ISSN 1613–0073

[Hen20] Henry, Gavin: Justin Richer on OAuth. DOI 10.1109/MS.2019.2949648. In: *IEEE Software* 37 (2020), Nr. 1, S. 98–100

[HZ20] Hartmann, Nils; Zeigermann, Oliver: *React: Grundlagen, fortgeschrittene Techniken und Praxistipps – mit TypeScript und Redux*. dpunkt.verlag, 2020. ISBN 9783864905520

[Ian12] Ian, Evans: *Your First Cup: An Introduction to the Java EE Platform*. Version: 2012. https://docs.oracle.com/javaee/6/firstcup/doc/. – [online], Abruf: 11-08-2021

[IEE90] IEEE: IEEE Standard Glossary of Software Engineering Terminology. DOI 10.1109/IEEESTD.1990.101064. In: *IEEE Std 610.12-1990* (1990), S. 1–84

[IMSFAQs] IMS Global Learning Consortium: *LTI Fundamentals FAQ*. https://www.imsglobal.org/lti-fundamentals-faq. – [online], Abruf: 04-07-2021

[InsCanv] Instructure: *What is Canvas?* https://community.canvaslms.com/t5/Canvas-Basics-Guide/What-is-Canvas/ta-p/45. – [online], Abruf: 24-07-2021

[InsGitH] Instructure: *Canvas LMS*. https://github.com/instructure/canvas-lms. – [online], Abruf: 26-07-2021

[InsHome] Instructure Inc.: *Canvas homepage*. https://www.instructure.com/canvas. – [online], Abruf: 27-02-2021

[InsMast] Instructure: *What is MasteryPaths?* https://community.canvaslms.com/t5/Canvas-Basics-Guide/What-is-MasteryPaths/ta-p/404483. – [online], Abruf: 27-07-2021

[InsProd] Instructure: *Production Start*. https://github.com/instructure/canvas-lms/wiki/Production-Start. – [online], Abruf: 26-07-2021

[IS18] Indrasiri, Kasun; Siriwardena, Prabath: *Microservices for the Enterprise : Designing, Developing, and Deploying*. Apress L. P. http://ebookcentral.proquest.com/lib/hawhamburg-ebooks/detail.action?docID=5598950. ISBN 9781484238585

[JacELMS] Jack: *Jack Einbinden in Lernplattformen*. https://jack-community.org/wiki/index.php/Einbinden_in_Lernplattformen. – [online], Abruf: 02-03-2021

[JacHome] Jack: *Hauptseite*. https://jack-community.org/wiki/index.php/Hauptseite. – [online], Abruf: 12-08-2021

[JSZ+18] Jameel, Asad; Shahzad, Khurram; Zafar, Afia; Ahmed, Usman; Hussain, Syed J. ; Sajid, Ahthasham: The Users Experience Quality of Responsive Web Design on Multiple Devices. DOI 10.1145/3231053.3234632. In: *Proceedings of the 2nd International Conference on Future Networks and Distributed Systems*. Association for Computing Machinery (ICFNDS '18). – ISBN 9781450364287

[TUMLice] Krusche, Stephan: *License*. https://github.com/ls1intum/Artemis/blob/develop/LICENSE. – [online], Abruf: 27-07-2021

[KS18] Krusche, Stephan; Seitz, Andreas: ArTEMiS: An Automatic Assessment Management System for Interactive Learning. DOI 10.1145/3159450.3159602. In: *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*. Association for Computing Machinery (SIGCSE '18). S. 284–289. – ISBN 9781450351034

[Leh21] Leh, John: *Why So Many LMS Vendors?* Version: 2021. https://talentedlearning.com/why-so-many-lms-vendors/. – [online], Abruf: 03-07-2021

[TL2021] Linsley, Tanner: *react-table Issue 2988*. https://github.com/tannerlinsley/react-table/issues/2988. – [online], Abruf: 18-10-2021

[LIS21] ListedTECH: *Our data on education institutions - Preview of our data*. https://www.listedtech.com/our-data-on-education-institutions. – [online], Abruf: 27-02-2021

[LL13] Ludewig, Jochen; Lichter, Horst: *Software Engineering : Grundlagen, Menschen, Prozesse, Techniken*. dpunkt.verlag http://ebookcentral.proquest.com/lib/hawhamburg-ebooks/detail.action?docID=1203992. ISBN 9783864912986

[LT16] Lenarduzzi, Valentina; Taibi, Davide: MVP Explained: A Systematic Mapping Study on the Definitions of Minimal Viable Product. DOI 10.1109/SEAA.2016.56. In: *2016 42th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, 2016 S. 112–119

[MicExM] Microsoft: *Extension Manifest*. https://code.visualstudio.com/api/references/extension-manifest. – [online], Abruf: 19-10-2021

[MicReD] Microsoft: *Remote Development FAQ*. https://code.visualstudio.com/docs/remote/faq. – [online], Abruf: 19-10-2021

[MicrExt] Microsoft: *Extension Guidelines*. https://code.visualstudio.com/api/references/extension-guidelines. – [online], Abruf: 17-10-2021

[MicrThe] Microsoft: *Theme Color*. https://code.visualstudio.com/api/references/theme-color. – [online], Abruf: 17-10-2021

[MTScrip] Microsoft: *TypeScript*. https://www.typescriptlang.org/. – [online], Abruf: 30-08-2021

[VSCAPI] Microsoft: *VS Code API*. https://code.visualstudio.com/api/references/vscode-api. – [online], Abruf: 04-07-2021

[VSCHome] Microsoft: *Visual Studio Code Homepage*. https://code.visualstudio.com/. – [online], Abruf: 03-07-2021

[VSCicon] Microsoft: *VSCode Codicons*. https://github.com/microsoft/vscode-codicons. – [online], Abruf: 29-08-2021

[VSCSync] Microsoft: *VSCode Codicons Sync icon*. https://github.com/microsoft/vscode-codicons/blob/main/src/icons/sync.svg. – [online], Abruf: 12-10-2021

[MooAkti] Moodle: *Aktivitätsabschluss*. https://docs.moodle.org/310/de/Aktivit%C3%A4tsabschluss. – [online], Abruf: 16-07-2021

[MooAuth] Moodle: *Authentifizierung*. https://docs.moodle.org/310/de/Authentifizierung. – [online], Abruf: 03-07-2021

[MooDash] Moodle: *Dashboard*. https://docs.moodle.org/310/de/Dashboard. – [online], Abruf: 03-07-2021

[MooData] Moodle: *Arbeiten mit Dateien und Verzeichnissen*. https://docs.moodle.org/310/de/Arbeiten_mit_Dateien_und_Verzeichnissen. – [online], Abruf: 03-07-2021

[MooEins] MOODLE: *Einschreibung*. https://docs.moodle.org/310/de/Einschreibung. – [online], Abruf: 06-07-2021

[MooFeat] MOODLE: *Moodle features*. https://docs.moodle.org/310/en/Features. – [online], Abruf: 02-03-2021

[MooGitH] MOODLE: *Moodle GitHub*. https://github.com/moodle/moodle. – [online], Abruf: 02-03-2021

[MooGlob] MOODLE: *Einschreibung über globale Gruppen*. https://docs.moodle.org/310/de/Einschreibung_%C3%BCber_globale_Gruppen. – [online], Abruf: 09-07-2021

[MooHome] MOODLE: *Moodle homepage*. https://moodle.org. – [online], Abruf: 27-02-2021

[MooKAnl] MOODLE: *Kurs anlegen*. https://docs.moodle.org/310/de/Kurs_anlegen. – [online], Abruf: 09-07-2021

[MooKBer] MOODLE: *Kursbereichseinschreibung*. https://docs.moodle.org/310/de/Kursbereichseinschreibung. – [online], Abruf: 09-07-2021

[MooKurs] MOODLE: *Kurse*. https://docs.moodle.org/310/de/Kurse. – [online], Abruf: 11-07-2021

[MooLDAP] MOODLE: *LDAP-Einschreibung*. https://docs.moodle.org/310/de/LDAP-Einschreibung. – [online], Abruf: 07-07-2021

[MooLern] MOODLE: *Lernfortschritt*. https://docs.moodle.org/310/de/Lernfortschritt. – [online], Abruf: 16-07-2021

[MooLice] MOODLE: *Moodle license*. https://docs.moodle.org/dev/License. – [online], Abruf: 02-03-2021

[MooMEnr] MOODLE: *Capabilities/enrol/manual:enrol*. https://docs.moodle.org/310/de/Capabilities/enrol/manual:enrol. – [online], Abruf: 09-07-2021

[MooMeta] MOODLE: *Meta-Einschreibung*. https://docs.moodle.org/310/de/Meta-Einschreibung. – [online], Abruf: 09-07-2021

[MooRole] MOODLE: *Rollen und Rechte*. https://docs.moodle.org/310/de/Rollen_und_Rechte. – [online], Abruf: 05-07-2021

[MooSelf] MOODLE: *Selbsteinschreibung*. https://docs.moodle.org/310/de/Selbsteinschreibung. – [online], Abruf: 09-07-2021

[MooSett] MOODLE: *Kurseinstellungen*. https://docs.moodle.org/310/de/Kurseinstellungen. – [online], Abruf: 11-07-2021

[MooSRol] MOODLE: *StandardRollen*. https://docs.moodle.org/310/de/Standardrollen. – [online], Abruf: 05-07-2021

[MooStat] MOODLE: *Statistiken*. https://docs.moodle.org/310/de/Statistiken. – [online], Abruf: 23-07-2021

[MooVora] MOODLE: *Voraussetzungen*. https://docs.moodle.org/310/de/Voraussetzungen. – [online], Abruf: 27-07-2021

[Nah04] NAH, Fiona Fui-Hoon: A study on tolerable waiting time: how long are Web users willing to wait? DOI 10.1080/01449290410001669914. In: *Behaviour & Information Technology* 23 (2004), Nr. 3, S. 153–163

[npmjs] NPM, INC.: *npmjs*. https://www.npmjs.com/. – [online], Abruf: 17-10-2021

[MySqlc] ORACLE: *MySQL Community Download*. https://dev.mysql.com/downloads/. – [online], Abruf: 17-10-2021

[OraGitH] ORACLE: *MySQL*. https://github.com/mysql. – [online], Abruf: 04-08-2021

[OraServ] ORACLE: *What Is a Servlet?* https://javaee.github.io/tutorial/servlets001.html#BNAFE. – [online], Abruf: 20-07-2021

[Pat13] PATTERSON, Terry: *Blackboard learn administration*. Packt Publishing Ltd, 2013

[Pau02] PAULSEN, Morten F.: Online education systems: Discussion and definition of terms. https://www.porto.ucp.pt/open/curso/modulos/doc/Definition%20of%20Terms.pdf. In: *NKI distance education* 202 (2002), S. 1–8

[Postman] POSTMAN, INC.: *Postman*. https://www.postman.com/. – [online], Abruf: 17-10-2021

[PR15] Pohl, Klaus; Rupp, Chris: *Basiswissen Requirements Engineering : Aus- und Weiterbildung nach IREB-Standard zum Certified Professional for Requirements Engineering Foundation Level*. dpunkt.verlag http://ebookcentral.proquest.com/lib/hawhamburg-ebooks/detail.action?docID=2029882. ISBN 9783864916731

[RJB04] Rumbaugh, James; Jacobson, Ivar ; Booch, Grady: *Unified Modeling Language Reference Manual, The (2nd Edition)*. Pearson Higher Education, 2004. ISBN 0321245628

[RSKP20] R, Nikhil; S, Anisha B. ; Kumar P, Ramakanth: Users Sync Authentication using External Ldap in Organizations. DOI 10.1109/INOCON50539.2020.9298432. In: *2020 IEEE International Conference for Innovation in Technology (INOCON)*, 2020 S. 1–4

[Ser06] Sermersheim, Jim: *Lightweight Directory Access Protocol (LDAP): The Protocol*. RFC 4511. DOI 10.17487/RFC4511. Version: Juni 2006 (Request for Comments)

[OPPAbou] Schmolitzky, Axel; Pilgrim, Jens von; Hinrichs, Torge; Burau, Henri; Stäps, Florian ; Bergmann, Stefan: *About Us ...* https://oppsee.informatik.haw-hamburg.de/about/. – [online], Abruf: 05-07-2021

[TLAbout] Talented Learning LLC: *About Talented Learning*. https://talentedlearning.com/about-talented-learning-consulting/. – [online], Abruf: 03-07-2021

[TUMExer] Technical University of Munich, Chair for Applied Software Engineering: *Exercise Creation*. https://docs.artemis.ase.in.tum.de/user/exercises/programming/#exercise-creation. – [online], Abruf: 11-08-2021

[TUMInst] Technical University of Munich, Chair for Applied Software Engineering: *Instructors Guide*. https://docs.artemis.ase.in.tum.de/user/exams/instructors_guide/. – [online], Abruf: 27-07-2021

[TUMSyst] Technical University of Munich, Chair for Applied Software Engineering: *System Design*. https://docs.artemis.ase.in.tum.de/dev/system-design/. – [online], Abruf: 28-07-2021

[WebCdSp] Webpack: *Code Splitting*. https://webpack.js.org/guides/code-splitting/. – [online], Abruf: 17-10-2021

[Webpack] Webpack: *Webpack*. https://webpack.js.org/. – [online], Abruf: 17-10-2021

[WebTrSh] Webpack: *Tree Shaking*. https://webpack.js.org/guides/tree-shaking/. – [online], Abruf: 17-10-2021

[WNHF16] Whitmer, John; Nuñez, Nicolas; Harfield, Timothy ; Forteza, Diego: *Patterns in Blackboard Learn tool use: Five course design archetypes*. https://www.blackboard.com/sites/default/files/resource/pdf/Bb_Patterns_LMS_Course_Design_r5_tcm136-42998.pdf. Version: 2016

[WP12] West, William; Pulimood, S M.: Analysis of privacy and security in HTML5 web storage. In: *Journal of Computing Sciences in Colleges* 27 (2012), Nr. 3, S. 80–87

A Anhang

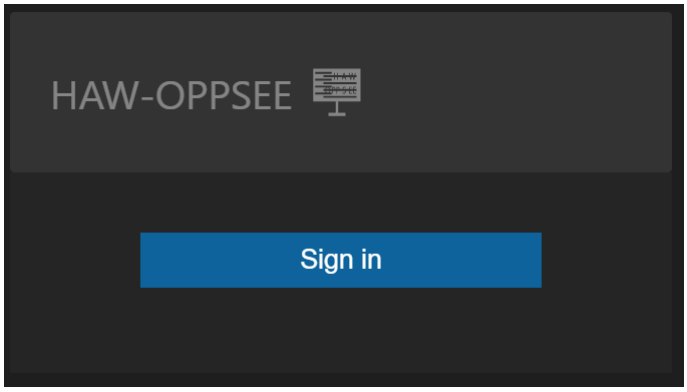

Abbildung A.1: Startbildschirm

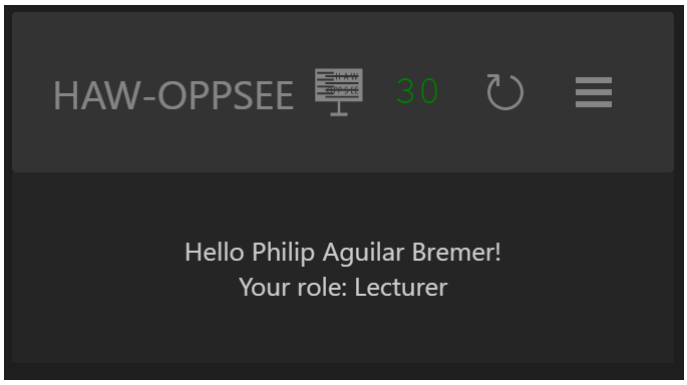

Abbildung A.2: Bildschirm nach erfolgreichem Login

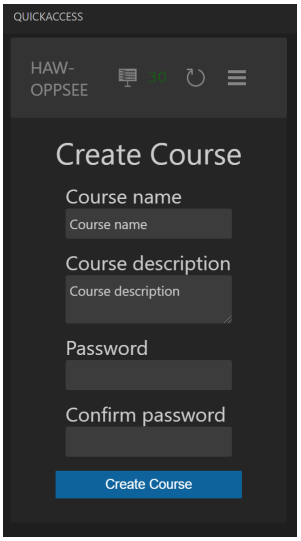

Abbildung A.3: Das Formular zur Kurserstellung

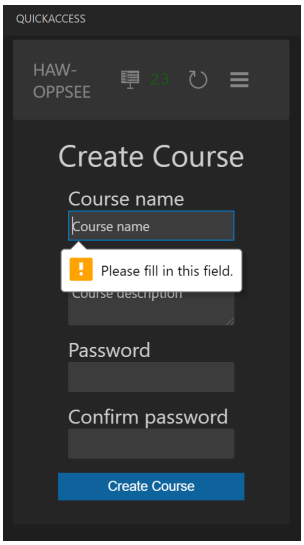

Abbildung A.4: Hinweis für falsche Eingabe

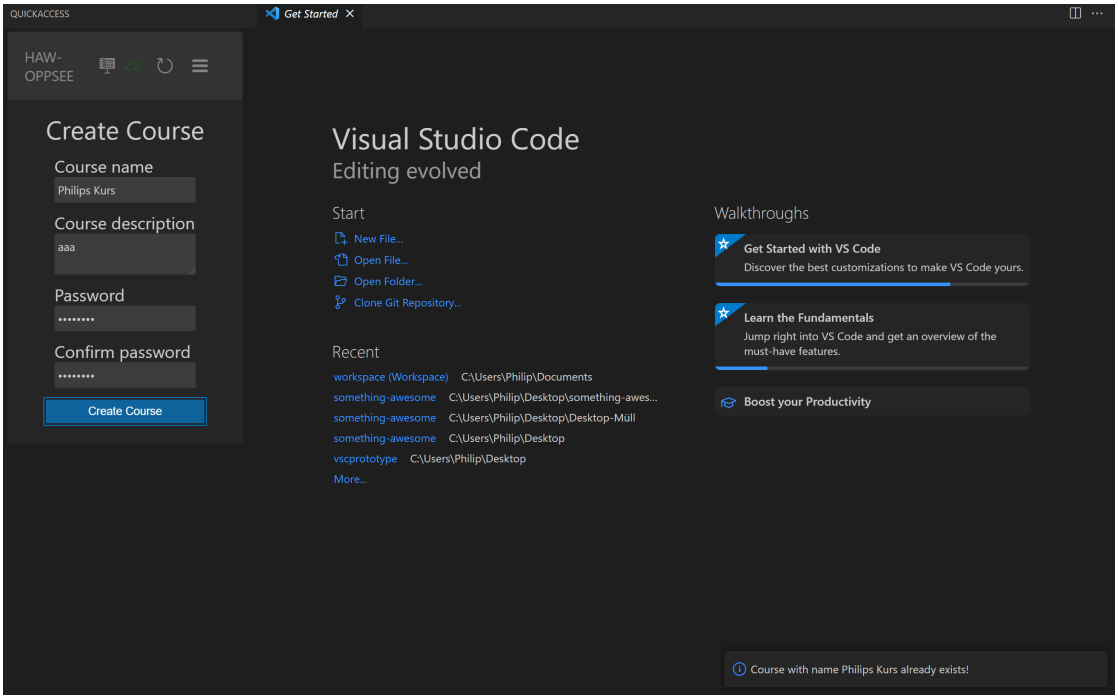

Abbildung A.5: Information über Namenskonflikt (rechts unten)

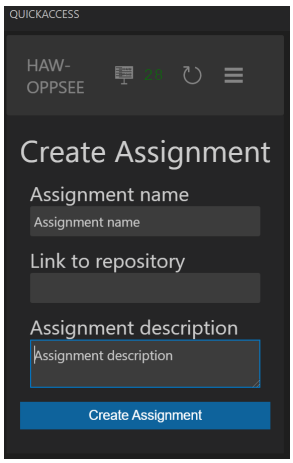

Abbildung A.6: Erstellen eines Assignments

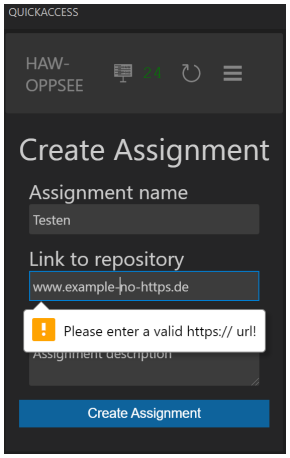

Abbildung A.7: Prüfen der URL

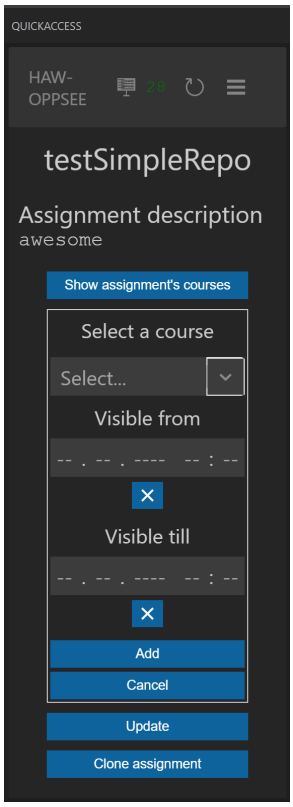

Abbildung A.8: Assignment zu Kurs hinzufügen

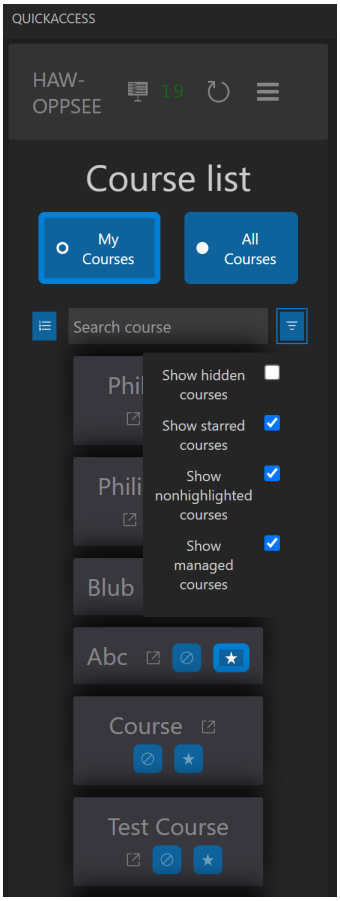

Abbildung A.9: Sortieren, filtern, markieren, aufrufen

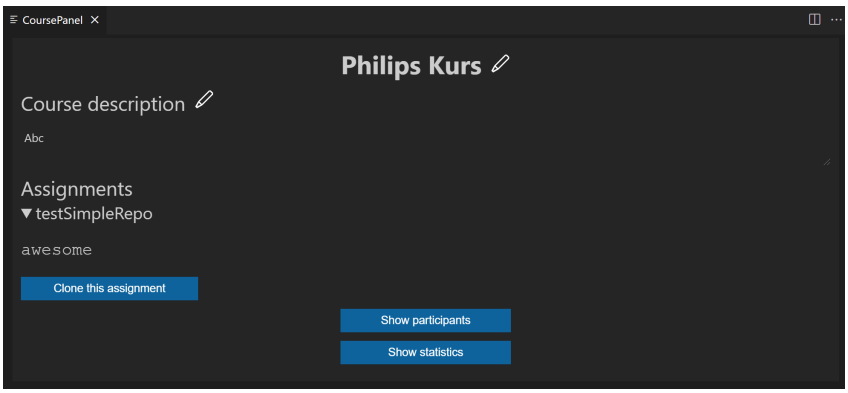

Abbildung A.10: Assignment im Kurs abrufen

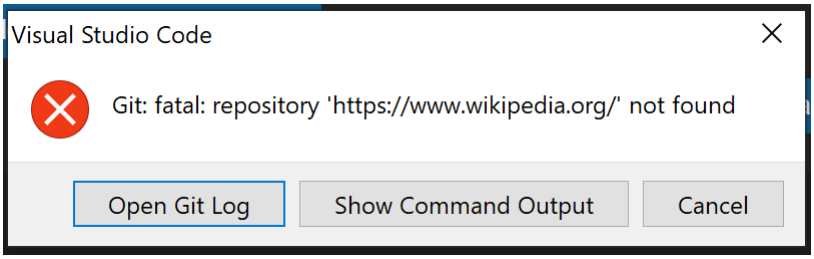

Abbildung A.11: Fehler bei „falscher“ Repository URL

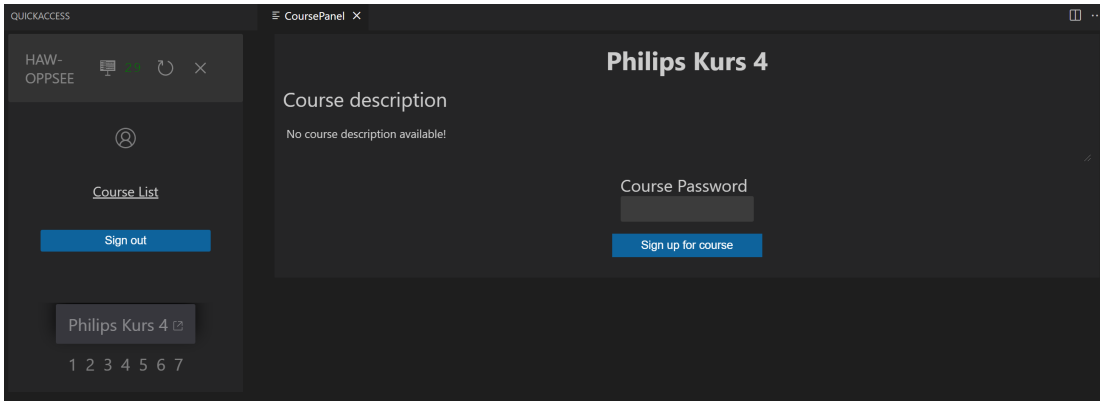

Abbildung A.12: Studierendensicht Kurseinschreibung

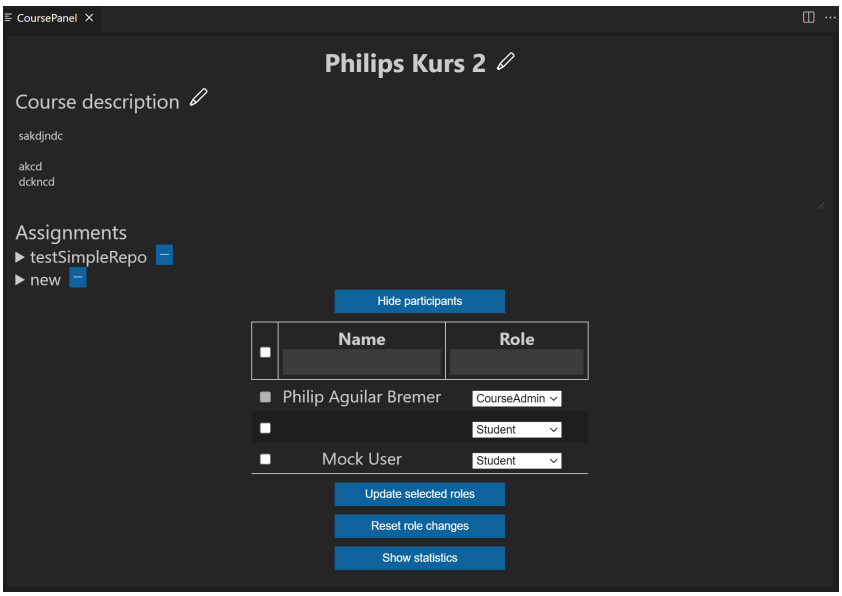

Abbildung A.13: Kursrechte bearbeiten

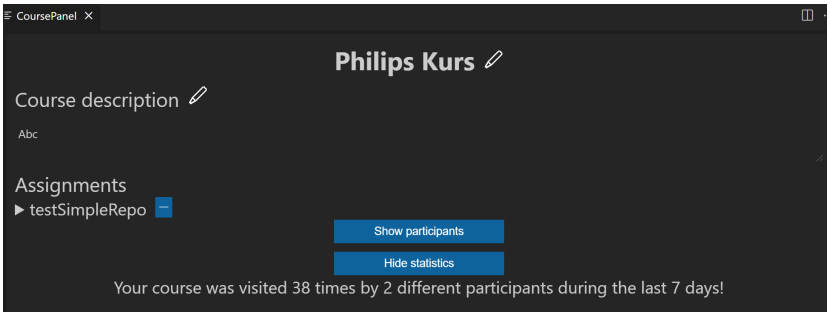

Abbildung A.14: Statistiken

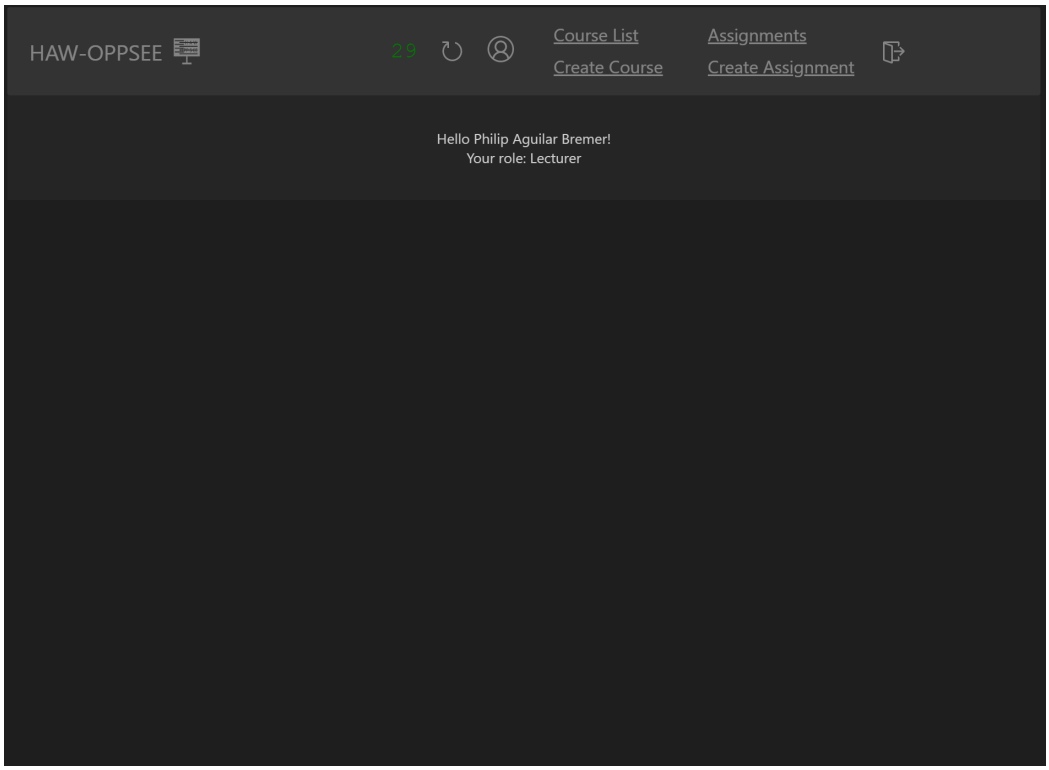

Abbildung A.15: Responsive Web Design

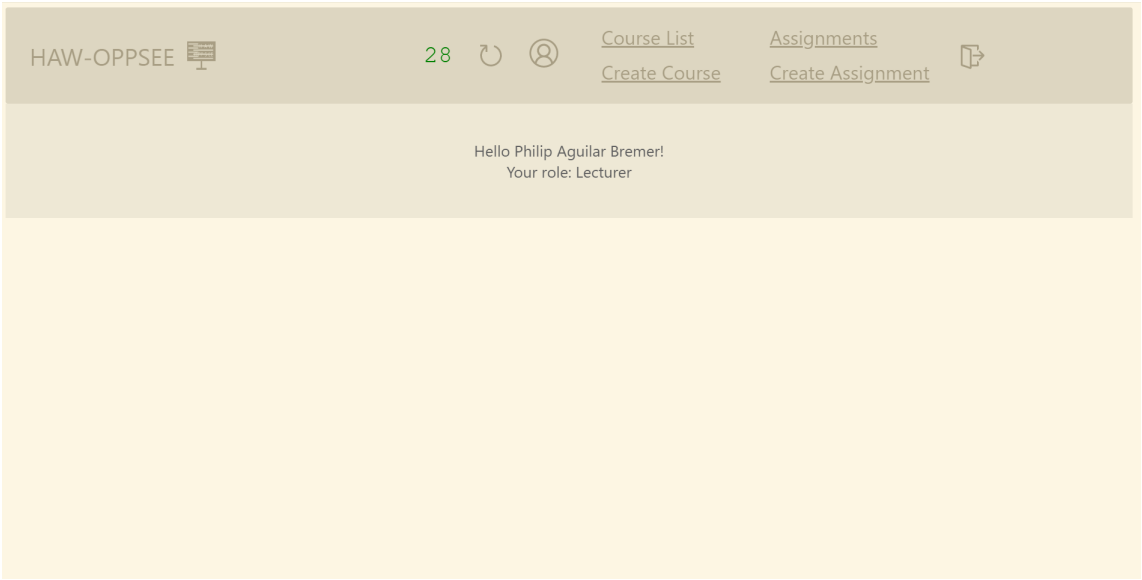

Abbildung A.16: Unterstützung verschiedener Themes

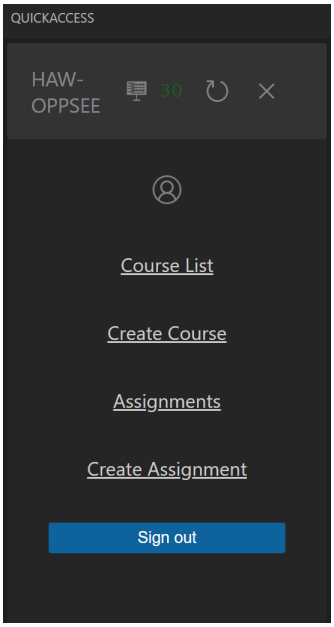

Abbildung A.17: Ansicht des Sidebar Menüs

Listings

Glossar

Assignment Eine Aufgabe (im Kontext dieser Arbeit werden damit hauptsächlich Programmierübungsaufgaben beschrieben).

Lecturer Ein/e Dozent*in.

Lecturer Backend Das Backend für die Lecturer Extension, welches u. a. die Authentifizierung und das Persistieren von Daten behandelt.

Lecturer Extension Die Frontendkomponente für die Umsetzung der Lecturer Funktionen innerhalb des Visual Studio Code Kontexts.

Microservice Die Microservice Architektur beschreibt eine Service-orientierte Architektur, in der eine Anwendung aus einer Sammlung von Lose gekoppelten Diensten besteht. Lose gekoppelte Dienste innerhalb unseres Systems könnten dabei zum Beispiel der Lecturer Service oder ein Authentifizierungsservice sein.

Teacher Eine Person mit einschränkten Kursverwaltungsrechten innerhalb eines Kurses.

UML Die UML ist eine allgemeine, entwicklungsorientierte Modellierungssprache die im Bereich der Softwareentwicklung verwendet wird..

VSCode Extension Eine Erweiterung für den Code Editor Visual Studio Code.

Erklärung zur selbstständigen Bearbeitung einer Abschlussarbeit

Gemäß der Allgemeinen Prüfungs- und Studienordnung ist zusammen mit der Abschlussarbeit eine schriftliche Erklärung abzugeben, in der der Studierende bestätigt, dass die Abschlussarbeit „— bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit [(§ 18 Abs. 1 APSO-TI-BM bzw. § 21 Abs. 1 APSO-INGI)] — ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt wurden. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich zu machen."

Quelle: § 16 Abs. 5 APSO-TI-BM bzw. § 15 Abs. 6 APSO-INGI

Erklärung zur selbstständigen Bearbeitung der Arbeit

Hiermit versichere ich,

Name: _____

Vorname: _____

dass ich die vorliegende Bachelorarbeit – bzw. bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit – mit dem Thema:

Erstellung eines Konzepts mit prototypischer Umsetzung einer Lehrendenkomponente als Microservice im Rahmen einer Online-Programmierplattform

ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

_____	_____	_____
Ort	Datum	Unterschrift im Original