

Repräsentation von Kurseinheiten der FernUniversität als Hyperaudio-Dokumente in Moodle: Design und Implementierung

Bachelorarbeit

eingereicht von
Michael Lämmermann
(Matrikelnummer 9611711)

angefertigt am
Lehrgebiet Kooperative Systeme
Fakultät Mathematik und Informatik
FernUniversität in Hagen

Betreuer
Dr. Niels Seidel

August 2018

Michael Lämmermann

**Repräsentation von Kurseinheiten der FernUniversität als Hyperaudio-Dokumente
in Moodle: Design und Implementierung**

Zusammenfassung

...

Summary

...

Michael Lämmermann. *Repräsentation von Kurseinheiten der FernUniversität als Hyperaudio-Dokumente in Moodle: Design und Implementierung*. Bachelorarbeit. Fakultät Mathematik und Informatik, FernUniversität in Hagen, 2018.

Diese Publikation ist unter *Creative Commons – Namensnennung 3.0 Deutschland* lizenziert und darf als Ganzes oder ausschnittweise vervielfältigt, verbreitet und öffentlich zugänglich gemacht werden, sofern dies im Text nicht anders vermerkt ist.



Autor: Michael Lämmermann
Gestaltung und Satz: Michael Lämmermann/ \LaTeX
Datum: 21. August 2018

Inhaltsverzeichnis

1 Einführung	9
1.1 Motivation	9
1.2 Problemstellung	9
1.2.1 Ist-Situation	10
1.2.2 Probleme	10
1.3 Zielsetzung und Forschungsfragen	12
2 Grundlagen	13
2.1 Moodle	13
2.2 Kooperation im Lernumfeld	14
2.3 Hypermedia	15
2.3.1 Grundbegriffe	16
2.3.2 Lernen mit Hypermedia	16
2.3.3 Repräsentation von Kurseinheiten im Hyperradio-Format	17
2.4 Zusammenfassung	18
3 Analyse	19
3.1 Zielgruppe	19
3.1.1 Personas	19
3.1.2 User Stories	20
3.2 Anforderungsdefinition	22
3.2.1 Anforderungen der Administrierenden	23
3.2.2 Anforderungen der Nutzenden	23
3.3 Möglichkeiten der Moodle Plugin-Entwicklung	24
3.4 Aktueller Stand der Technik	26
3.4.1 Etablierte Audio- und Video-Plattformen	26
3.4.2 Technologien für den Einsatz in Moodle	27
3.5 Zusammenfassung	30
4 Konzept	32
4.1 Zusammenhänge der Komponenten der Hyperradio-Anwendung	32
4.1.1 Komponenten	32
4.1.2 Zusammenhänge	32
4.2 Definition des Schnittstellenformats für Hyperradio-Dokumente	33
4.3 Datenbankentwurf	34
4.4 Gestaltung der Benutzeroberfläche	36
4.5 Zusammenfassung	41
5 Implementierung	43
5.1 Architektur des Moodle-Plugins	43
5.2 Iterative Entwicklung	44
5.2.1 Speichern und Abspielen einer Audio-Datei	45
5.2.2 Speichern und Anzeige von Zusatzinhalten	47

Inhaltsverzeichnis

5.2.3	Einbindung der Konfigurationsdatei	49
5.2.4	Speichern und Anzeige von Kommentaren	50
5.2.5	Antworten auf Kommentare	54
5.2.6	Notizen	55
5.2.7	Audio Cues	56
5.2.8	Galerie der Zusatzinhalte	57
5.2.9	Zeitabhängige Visualisierung der Kommentare	57
5.2.10	Markierungen	57
5.3	Zusammenfassung	57
6	Evaluation	58
7	Zusammenfassung und Ausblick	59
A	Erster Teil des Anhangs	60
B	Zweiter Teil des Anhangs	62
	Literaturverzeichnis	64
	Abbildungsverzeichnis	67
	Tabellenverzeichnis	69
	Auflistungsverzeichnis	71
	Verzeichnis der Algorithmen	73
	Abkürzungsverzeichnis	75

1 Einführung

Diese Arbeit beschäftigt sich mit dem Design und der Implementierung eines Plugins für die Lernplattform Moodle, welches es ermöglichen soll, den Studierenden die Lerninhalte mittels Hyperaudio-Dokumenten bereitzustellen. Ziel dieses Plugins ist die Erweiterung der Lernmöglichkeiten an der FernUniversität in Hagen, um die Studierenden beim Erreichen ihrer Lernziele besser zu unterstützen.

1.1 Motivation

Die Motivation zur Behandlung dieses Themas besteht darin, dass 80% der Studierenden gleichfalls neben dem Studium arbeiten (FernUniversität in Hagen, 2018c). Unter diesen Umständen beschäftigen sich viele Studierende erst kurz vor der Prüfung - dafür aber entsprechend intensiv - mit den Lerninhalten. An der Fakultät Mathematik/Informatik und auch an der Fakultät für Wirtschaftswissenschaften bestehen diese Lerninhalte zu einem guten Teil aus textlastigen Kurseinheiten, die Abbildungen und Formeln enthalten.

Diese Aussage kann anhand der Pflichtmodule des Bachelorstudiengangs Wirtschaftsinformatik bestätigt werden. Die Pflichtmodule an der Fakultät Mathematik/Informatik weisen einen Textanteil von XXX auf. An der Fakultät für Wirtschaftswissenschaften liegt der Anteil in den Pflichtmodulen nochmals höher bei XXX. Die Analyse der Kurseinheiten wurde mittels der Textanalyse des Programms *PDF-Analyzer 5.0*¹ vorgenommen.

Abschnitt nach Abschluss der Analyse der Kurseinheiten aktualisieren

Eher selten werden auch Videos angeboten, in welchen bestimmte Lerninhalte aus den Kurseinheiten nochmals rekapituliert werden. Hier sind als Beispiel die Videos von Univ.-Prof. Dr. Ulrike Baumöl zum Kurs „Informationsmanagement“ zu nennen.

Die Idee besteht nun darin, den Lernenden erstens eine alternative Repräsentation (Modalität) der Lerninhalte anzubieten und ihnen zweitens das Lernen während ungenutzter Alltagssituationen zu ermöglichen (z.B. lange Autofahrten, Pendeln in Bus und Bahn, beim Joggen, etc.). Auf diese Weise könnten die Lernenden die Inhalte häufiger rezipieren und einüben. Ergänzt um gute E-Assessments (Selbsttests) hätten Sie in Summe eine Chance, sich frühzeitig und kontinuierlich auf die Prüfung vorzubereiten und vielleicht bessere Lernerfolge zu erzielen.

1.2 Problemstellung

Diese Arbeit wird sich in diesem Zusammenhang vor allem mit dem Problem des hohen Textanteils vieler Kurse und dem damit verbundenen Lernverhalten beschäftigen. Doch zunächst soll die Ist-Situation für die Studierenden an der FernUniversität in Hagen beschrieben werden.

¹<http://www.is-soft.de>

1.2.1 Ist-Situation

Jeder Studierende hat Zugriff auf den *Virtuellen Studienplatz*, häufig auch *Virtuelle Universität* (VU) oder *Lernraum Virtuelle Universität* (LVU) genannt. Hierbei handelt es sich um ein eigenentwickeltes Webportal der FernUniversität in Hagen. Der *Virtuelle Studienplatz* stellt unter anderem das Kurs- und Studiumsportal der FernUniversität dar. Hierüber können die Studierenden Kurse belegen, die Rückmeldung für das nächste Semester vornehmen oder ihre persönlichen Daten einsehen und bearbeiten. Auch eine Übersicht über das Veranstaltungsangebot wird dem Studierenden geboten. Zusätzlich bietet der *Virtuelle Studienplatz* eine Übersicht über alle belegten Kurse des Studierenden (FernUniversität in Hagen, 2018d). Mittels dieser Übersicht kann der Studierende direkt auf das jeweilige persönliche Kursportal seiner belegten Kurse gelangen. Neben allgemeinen Informationen zu dem Kurs bietet das Kursportal unter anderem Zugriff auf die Online-Studieninhalte (z.B. Kurseinheiten², Einsendeaufgaben, Musterlösungen) und Verweise zu anderen Diensten (Moodle, Online-Übungssystem, Kommunikationsangebote Adobe Connect Videokonferenzen), die in diesem Kurs zur Verfügung stehen (FernUniversität in Hagen, 2014).

Neben der Möglichkeit, die Kurseinheiten als PDF über den *Virtuellen Studienplatz* herunterzuladen, werden diese im Regelfall für die belegten Kurse automatisch in gedruckter Form an die Studierenden versendet. Die Kurseinheiten dienen als zentrales Lernmaterial für die Studierenden.

Jeder Kurs hat die Möglichkeit, zusätzliches Lernmaterial über Moodle, die zentrale Lernplattform der FernUniversität in Hagen, zur Verfügung zu stellen. Moodle bietet den Kursen „als sogenanntes Learningmanagementsystem (LMS) vielfältige Möglichkeiten zur Gestaltung der mediengestützten Lehre an“ (FernUniversität in Hagen, 2018b). Besonders hervorzuheben ist hierbei der mögliche Einsatz von Lehrvideos, Foren und Tests. Die Plattform Moodle und ihre Erweiterungsmöglichkeiten werden in Abschnitt 2.1 vorgestellt. Im Bezug auf die Pflichtmodule des Bachelorstudiengangs Wirtschaftsinformatik an der FernUniversität in Hagen wird Moodle in 21 von 30 Kursen beziehungsweise 10 von 15 Modulen eingesetzt (vgl. Anhang A.1). Auffällig ist hierbei, dass Moodle an der Fakultät für Wirtschaftswissenschaft für jeden Kurs angeboten wird, während an der Fakultät für Mathematik und Informatik kein einziger Kurs auf Moodle zurückgreift.

Durch den Einsatz von Adobe Connect besteht an der FernUniversität in Hagen auch die Möglichkeit sogenannter *Virtual Classrooms*. Dabei handelt es sich um eine Video- und Tonübertragung mit Textchat und Freigabemöglichkeiten für Präsentationen und Bildschirmminhalte. „Ein *Virtual Classroom* [Hervorhebung v. Verf.] eignet sich insbesondere für Veranstaltungen, in denen die synchrone Kommunikation ein hohes Gewicht erhält: Seminare, Tutorien, Sprechstunden, Arbeitsgruppen u.Ä.“ (FernUniversität in Hagen, 2018a).

Darüber hinaus werden den Studierenden mit den *Diskussionsforen* (Newsportal) und dem *Conference Center* als Chat-System zwei weitere Systeme zur Kommunikation geboten (FernUniversität in Hagen, 2018a).

1.2.2 Probleme

Die im vorangegangen Abschnitt beschriebenen Lernangebote weisen jedoch Defizite im Bezug auf die Vermittlung der Lerninhalte auf. Der *Virtuelle Studienplatz* dient aktuell ausschließlich als Portal, um die Studierenden zu den von ihnen benötigen Informationen zu leiten und unterstützt somit nur indirekt die Vermittlung von Lerninhalten.

Die Kurseinheiten als zentrales Lernmaterial bestehen, wie anhand der Zahlen aus Abschnitt 1.1 erkenntlich, zum Großteil aus Text. Dies hat zur Folge, dass sich die Studierenden während der Auseinandersetzung mit den Lerninhalten nicht mit anderen Dingen beschäftigen können, welche die Aufmerksamkeit ihrer Augen und Hände benötigen. Zusätzlich besteht oft das Problem, dass

²Kurseinheiten werden an der FernUniversität in Hagen auch als Studienbriefe bezeichnet.

Abbildungen, Formeln und Tabellen, auf die im Text verwiesen wird, nicht direkt auf der Seite ersichtlich sind, auf der diese im Text erwähnt werden. Hierdurch ist oftmals Blättern bzw. Scrollen nötig, je nachdem ob die Kurseinheit in Papierform oder digital bearbeitet wird. Dies erschwert zusätzlich das Verinnerlichen des in der Kurseinheit zu vermittelnden Inhalts. Im Vergleich zum Frontalunterricht bringt die Vermittlung der Lerninhalte in Form von Kurseinheiten den Nachteil mit sich, dass bei Verständnisproblemen keine direkten Fragen gestellt werden können.

Dieses Problem tritt bei Lehrvideos ebenso auf. Hier ist im besten Fall ein asynchroner Austausch mittels einer Kommentarfunktion möglich. Ähnlich wie Kurseinheiten verlangen auch Lehrvideos die durchgehende visuelle Aufmerksamkeit des Studierenden. Nur durch ununterbrochenes Betrachten eines Videos kann ein Studierender sicherstellen, dass er alle dargestellten Inhalte wahrnimmt.

Im Gegensatz zu Lehrvideos, sind die Foren und der Chat ohnehin nur als zusätzliche Kommunikationswege für die Studierenden implementiert. Es besteht das grundsätzliche Problem, dass diese Funktionalitäten nicht direkt an die Lerninhalte gekoppelt sind und deswegen separat aufgerufen werden müssen, falls beim Lernen Fragen auftreten sollten.

Die in Moodle verfügbaren Tests dienen hingegen in ihrer Form nur zur reinen Selbstkontrolle. Dadurch, dass diese Tests nicht direkt während der Erarbeitung der Lerninhalte durch Kurseinheit oder Lehrvideos erfolgen kann, wird nicht unmittelbar überprüft, ob die Inhalte korrekt verstanden wurden.

Virtual Classrooms zählen wiederum zu den Lerninhalt vermittelnden Angeboten. Bedingt durch die Tatsache, dass der Unterricht in *Virtual Classrooms* in Echtzeit abgehalten wird, entsteht der Nachteil, dass der Studierende nur zu einem festgelegten Zeitpunkt die Lehrveranstaltung wahrnehmen kann. Außerdem verlangen *Virtual Classrooms*, genau wie Lehrvideos, die ständige Aufmerksamkeit des Studierenden. Im Gegensatz zu Lehrvideos schaffen *Virtual Classrooms* jedoch die Möglichkeit der synchronen Kommunikation.

Zusammenfassend handelt es sich bei den Kurseinheiten, Lehrvideos und *Virtual Classrooms* um die Lerninhalt vermittelnden Angebote. Kurseinheiten und Lehrvideos stellen hierbei asynchrone Lehrmethoden dar, während es sich bei den *Virtual Classrooms* um eine synchrone Lehrmethode handelt. Die asynchronen Angebote bringen im Gegensatz zum synchronen Angebot den Vorteil mit sich, dass diese zu jeder beliebigen Zeit wahrgenommen werden können. Dennoch haben diese drei Lehrangebote gemeinsam, dass die Studierenden ihnen zumindest die volle visuelle Aufmerksamkeit beim Lernen schenken müssen.

Somit ist mit dem aktuellen Lehrangebot beispielsweise auch kein Lernen während der sportlichen Betätigung möglich. Dabei führt leichte körperliche Betätigung während des Lernens nach einer Studie von Schmidt-Kassow et al. (2013) sogar zu einem besseren Lernergebnis. Stattdessen ist der Studierende weiterhin daran gebunden im Sitzen oder gar vor dem Bildschirm zu lernen. Indes haben mehrere Studien gezeigt, dass langes Sitzen negative Auswirkungen auf die gesundheitliche Verfassung mit sich bringt (Tremblay et al., 2011).

Aufgrund der Tatsache, dass 80% der Studierenden an der FernUniversität in Hagen neben dem Studium ebenfalls einer Arbeit nachgehen (FernUniversität in Hagen, 2018c), muss auch die dem Studierenden zur Verfügung stehenden Zeit berücksichtigt werden. Zur bezahlten Arbeit kommt immer auch unbezahlte Arbeit hinzu. Diese betrug in den Jahren 2012/2013 im Durchschnitt ca. 24,5 Stunden in der Woche für Personen ab 18 Jahren. Als unbezahlte Arbeit gelten Haushaltstätigkeiten, wie Kochen, Putzen, Gartenpflege und Einkaufen, aber auch ehrenamtliche Tätigkeiten sowie Wegzeiten (Statistisches Bundesamt, 2015). Diese unbezahlte Arbeit kann aktuell zum Großteil nicht zum Lernen verwendet werden, da durch die heutigen Lernangebote stets die volle Aufmerksamkeit des Studierenden erforderlich ist. Beispielsweise ist es unmöglich, während des Fensterputzes eine Kurseinheit in ihrer aktuellen Repräsentation zu lesen, da die Putztätigkeit bereits die volle visuelle Aufmerksamkeit sowie den Einsatz der Hände erfordert. Diese Zeit könnte jedoch zum Lernen genutzt werden, sofern die Lerninhalte in einer anderen Form präsentiert würden, die sich auf andere Sinne beschränkt. Da der Hörsinn in vielen Alltagstätigkeiten, wie dem Putzen oder Einkaufen, nicht

vorrangig benötigt wird, bietet sich dazu eine Repräsentation in auditiver Form an.

1.3 Zielsetzung und Forschungsfragen

Ziel dieser Arbeit soll es sein, den Studierenden eine auditive Repräsentation der Lerninhalte anzubieten, welche es ihnen ermöglichen soll mehr ihrer Zeit zum Lernen nutzen zu können und dabei die Effizienz des Lernens zu erhöhen. Die alternative Repräsentation der Lerninhalte soll in die Moodle-Plattform integriert werden, da diese mit ihrem Plugin-System und dem hohen Verbreitungsgrad an der FernUniversität in Hagen gute Voraussetzungen für die Bereitstellung neuer Lehrmethoden bietet.

Es muss eine Lernumgebung gestaltet werden, welche es ermöglicht, die hauptsächlich auditiven Lerninhalte bereitzustellen. Daneben müssen auch Lerninhalte, welche nicht in auditiver Form abgebildet werden können, berücksichtigt werden. Akustische Signale können unterstützend eingesetzt werden, um Studierende auf Zusatzinhalte aufmerksam zu machen, die ihre tiefergehende Aufmerksamkeit erfordern. Typische Nutzerinteraktionen mit textuellen Lernmedien, wie beispielsweise persönliche Notizen und Markierungen oder das Durchsuchen des Inhaltsverzeichnisses, sollen hierbei erhalten bleiben.

Zusätzlich zur Bereitstellung der Lerninhalte in alternativer Form soll die Hyperaudio-Lernumgebung auch den Kommunikationsaustausch zwischen Lehrenden und Studierenden ermöglichen und fördern. Hiermit soll dem allgemeinen Problem des mangelnden sozialen Kontakts beim Fernstudium begegnet werden (Kerres und Jechle, 2002). Um den Studierenden beim Lernen möglichst große Flexibilität zu bieten, ist eine mobile Verfügbarkeit der neugestalteten Lerninhalte erstrebenswert.

Aus dieser Zielsetzung ergeben sich folgende Forschungsfragen:

- Wie kann den Studierenden ermöglicht werden mehr Zeit zum Lernen nutzen zu können?
- Wie ist eine Hyperaudio-Lernumgebung in Moodle zu gestalten?
- Wie lassen sich Inhalte in der Hyperaudio-Lernumgebung darstellen, die nicht in auditiver Form abgebildet werden können?
- Wie können nicht-auditive Inhalte mit den auditiven Inhalten verknüpft werden?
- Wie lassen sich alle Interaktionen, die eine textuelle Darstellung der Lerninhalte bietet, in der Hyperaudio-Lernumgebung umsetzen?
- Wie kann der Austausch zwischen Studierenden und Lehrenden umgesetzt werden?

2 Grundlagen

Als Fundament für die weitere Arbeit sollen zunächst einige Grundlagen thematisiert werden. Um das Vorhaben und das Vorgehen dieser Arbeit besser nachvollziehen zu können, wird zunächst das Modell der *Tetraden der Medieneffekte* vorgestellt. Hierbei handelt es sich um eine Idee von Marshall McLuhan, welcher sich mit den Effekten beschäftigt, die ein Medium mit sich bringt. Er hat festgestellt, dass im Wesentlichen vier Effekte von Interesse sind, welche er mit den folgenden Fragen bestimmen will (McLuhan, 1977):

1. What does the medium enhance?
2. What does the medium make obsolete?
3. What does the medium retrieve that had been obsoleted earlier?
4. What does the medium reverse or flip into when pushed to extremes?

So stellt McLuhan (1977) fest, dass das Radio eine unmittelbare und auditive Art der Kommunikation beförderte. Im gleichen Moment wurde dadurch die Bedeutsamkeit von Printmedien geschwächt. Hierdurch hat die vorangegangene auditive Kommunikation, welche durch die Einführung von Printmedien obsolet wurde, wieder an Bedeutung gewonnen. Wenn man nun das Medium an sein Limit bringt, dann befördert dies die Entwicklung hin zum Fernsehen (McLuhan, 1977).

Mit diesen Gedanken im Hinterkopf werden nun die Grundlagen für diese Arbeit betrachtet. Alle in den Grundlagen betrachteten Themen beschäftigen sich ebenfalls mit Medien und deren Effekten. Zuletzt soll auch in Kapitel 6 der Kreis geschlossen werden und das Medium Hyperaudio-Dokument nochmals anhand der *Tetraden der Medieneffekte* bewertet werden.

2.1 Moodle

Bei Moodle (Modular Object-Oriented Dynamic Learning Environment) handelt es sich um ein frei verfügbares Open-Source-Learningmanagementsystem (GNU Public License), mit dem Internetbasierte Kurse entwickelt und durchgeführt werden können (Moodle, 2015c). Ziel der Lernplattform ist es, den Lehrenden, Administratoren und Lernenden ein robustes, sicheres und integriertes System zu liefern, mit dessen Hilfe sie eine personalisierte Lernumgebung gestalten können (Moodle, 2018a). Unter dieser Zielsetzung ist Moodle als Lernplattform weltweit verbreitet und hat aktuell³ 101.447 registrierte Seiten in 232 Ländern mit insgesamt mehr als 130 Millionen Benutzern (Moodle, 2018e).

Zugriff auf Moodle erhält der Nutzer über die Startseite, welche auf die eigenen Bedürfnisse angepasst werden kann. Auch kann Moodle so konfiguriert werden, dass die Startseite erst nach Anmeldung an der Login-Seite erfolgen kann. Die Grundstruktur von Moodle ist, wie in Abbildung 2.1 zu sehen, anhand von Kursbereichen und Kursen organisiert. Kurse werden wiederum als Seiten repräsentiert, auf welchen die Lehrenden Arbeitsmaterialien und Aktivitäten für die Studierenden bereitstellen können. Kurse werden üblicherweise in einzelne Kursabschnitte unterteilt, in welchen die Arbeitsmaterialien und Aktivitäten eingebunden werden. Kursseiten können durch Blöcke noch um weitere zusätzliche Informationen angereichert werden.

Ein Beispiel für eine Kursseite ist in Abbildung 2.2 anhand des Kurses „Einführung in Mensch-Computer-Interaktion“ zu sehen. Auf der linken Seite sind innerhalb der Navigation die einzelnen

³Stand: 28.07.2018

2 Grundlagen

Kursabschnitte, hier als Kurseinheiten beziehungsweise Einführung benannt, sichtbar. Unterhalb der Navigation sowie auf der rechten Seite sind die verschiedenen Blöcke (z.B. „Aktivitäten“, „Suche in Foren“ oder „Neue Ankündigungen“) innerhalb des Kurses angeordnet. Im mittleren Bereich befinden sich untereinander die Beschreibungen der einzelnen Kursabschnitte mit Links zu den verwendeten Arbeitsmaterialien und Aktivitäten.

Kurse werden wiederum innerhalb von Kursbereichen organisiert. Es ist auch ein mehrstufiges Kursbereichssystem umsetzbar (Moodle, 2015a).

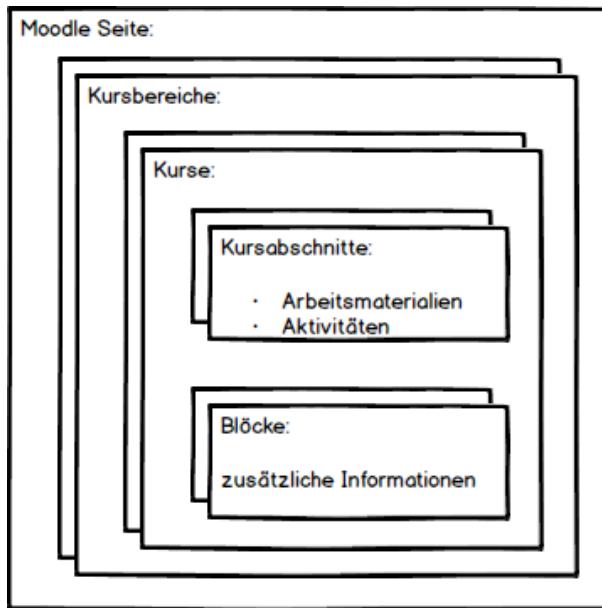


Abbildung 2.1: Schematischer Aufbau einer Moodle Seite

Technisch baut Moodle auf einem Aufbau aus Webserver und Datenbank unter Verwendung von PHP (PHP: Hypertext Preprocessor) auf. Um die zum Ziel gesetzte Personalisierbarkeit zu erreichen, setzt Moodle unter anderem auf ein Plugin-System. Plugins werden in über 50 verschiedene Plugin-Typen kategorisiert, wobei jeder dieser Typen dazu dient, einen speziellen Bereich von Moodle zu erweitern beziehungsweise anzupassen (Moodle, 2017b).

2.2 Kooperation im Lernumfeld

„Lernen ist in vieler Hinsicht ein sozialer Prozess, der kulturelle Einflüsse einschließt sowie soziale Aktivitäten und gemeinsames Problemlösen umfasst“ (Reinmann-Rothmeier und Mandl, 1995). Weit gefasst versteht man unter kooperativem Lernen eine Situation, in der zwei oder mehrere Personen zusammen lernen oder versuchen, zusammen zu lernen (Dillenbourg, 1999).

Während in dieser Arbeit, wie im deutschsprachigen Raum üblich, keine Unterscheidung zwischen dem *kooperativen* und dem *kollaborativen* Lernen vorgenommen wird, werden die Begriffe außerhalb des deutschsprachigen Raums häufig differenziert betrachtet (Reinmann-Rothmeier und Mandl, 2002). Eine differenzierte Betrachtung der beiden Begriffe liefert Dillenbourg et al. (1995). Demnach handelt es sich um *Kooperation*, wenn eine Problemlösung durch die Arbeitsteilung unter den Mitgliedern einer Gruppe erreicht wird, wobei jedes Mitglied für einen Teil der Problemlösung verantwortlich ist. Bei *Kollaboration* wird die Problemlösung hingegen durch gemeinsames Engagement der Gruppenmitglieder bei koordiniertem Vorgehen zur Problemlösung erreicht. Als Beispiel kann eine Gruppenarbeit mit der Aufgabe, mehrere Textabschnitte zusammenzufassen, betrachtet werden. Bei einem kooperativen Vorgehen wäre zunächst jeweils ein Gruppenmitglied für die Zusammenfassung genau eines Textabschnittes verantwortlich. In einem zweiten Schritt würden dann die einzelnen

The screenshot shows the Moodle course page for "WS16/17 - Kurs 01697 Einführung in Mensch-Computer-Interaktion". The left sidebar contains a navigation menu with sections like "Navigation", "Startseite", "Meine Lernumgebungen", "Kurse", "Personen", and "Aktivitäten". The main content area displays course information, including the title, description, and various course activities such as forums, announcements, events, and activities.

Abbildung 2.2: Kursseite des Kurses „Einführung in Mensch-Computer-Interaktion“
(FernUniversität in Hagen, 2016)

Teilergebnisse zu einem Ergebnis zusammengefasst. Bei einem kollaborativen Vorgehen wiederum würden alle Gruppenmitglieder die einzelnen Textabschnitte gemeinsam zusammenfassen.

Dem kooperativen Lernen werden positive Effekt zugesprochen, unter anderem bezüglich der Lernmotivation (Dillenbourg, 1999; Reinmann-Rothmeier und Mandl, 1995). So wird beispielsweise beim Lernen durch Lehren, einer speziellen Form des kooperativen Lernens, die Motivation der Beteiligten erhöht. Gleichzeitig werden sowohl beim Lehrenden als auch beim Lernenden effektive Lernprozesse in Gang gesetzt (Reinmann-Rothmeier und Mandl, 1995).

Reinmann-Rothmeier und Mandl (2002) beschreiben auch, welche Probleme bei netzbasierten Umgebungen entstehen können. So können neben technischen Problemen auch Widerstände seitens der Nutzer vorliegen. Oftmals ist es schwer, die Gruppenmitglieder dazu zu bringen, aktiv an den netzbasierten Szenarien teilzunehmen. Auch besteht das Problem, dass in netzbasierten Umgebungen weniger individuelles als kollektives Wissen geteilt wird als in Face-to-Face-Situationen. Aufgrund der fehlenden sozialen Hinweisreize und der Anonymität kann „es leicht zu unkontrollierter Kommunikation und heftigen Gefühlsausbrüchen, dem sog. Flaming, kommen“ (Reinmann-Rothmeier und Mandl, 2002).

2.3 Hypermedia

Bevor mit der genauen Konzeption und Implementation des Hyperaudio-Plugins begonnen werden kann, werden *Hypermedia* im Allgemeinen betrachtet. Dabei soll zunächst eine Begriffsklärung durchgeführt werden. Aus diversen Erfahrungen mit *Hypermedia* können Rückschlüsse für das zu entwickelnde Moodle-Plugin und die zugrundeliegende Interpretation von *Hyperaudio* gezogen werden.

2.3.1 Grundbegriffe

Zu Beginn wird zunächst der Begriff *Multimedia* betrachtet. Unter *Multimedia* wird die Bereitstellung von Informationen mittels verschiedener Formate, beispielsweise Text, Audio, Bilder oder Video, bezeichnet (Mayer, 2009; Moos und Marroquin, 2010).

Der Begriff *Hypermedia* wurde das erste Mal von Ted Nelson 1965 verwendet (Nelson, 1965). In seinem Paper beschreibt er detailliert, was er sich unter einem *Hypertext* vorstellt. Hierunter versteht er ein Dokument bestehend aus geschriebenen oder bildhaften Inhalten, welche in solch einer komplexen Art und Weise miteinander verbunden sind, dass sie nicht mehr auf Papier dargestellt werden können. Es kann Zusammenfassungen, Karten über die Inhalte und deren Zusammenhänge, Annotationen, Ergänzungen oder Anmerkungen von Wissenschaftlern, die das Dokument begutachtet haben, enthalten. Nelson beschreibt das Kriterium für den Präfix *hyper* damit, dass diese Objekte nicht durch eine Konvertierung in ein einfaches lineares Medium, wie beispielsweise einen String umgewandelt werden können. Der wesentliche Punkt ist also, dass es sich beim Arbeiten mit *Hypermedia* um ein nicht-lineares Vorgehen handelt.

Genauer betrachtet stellt das, was Nelson (1965) sich als *Hypertext* vorgestellt hatte, nach Nielsen (2013) bereits eine Form von *Hypermedia* dar. Auch wenn die beiden Begriffe *Hypertext* und *Hypermedia* synonym verwendet werden können, werden bei strikter Betrachtung bei *Hypertext* ausschließlich Texte miteinander verbunden, während bei *Hypermedia* auch andere Medien eingebunden werden können. Gemeinsam haben beide Arten jedoch, dass der Betrachter keinen linearen Weg vorgegeben hat, sondern von einem Knoten (Node) zum anderen springen kann und sich somit seinen Weg selbst aussucht. Demnach stellt *Hypermedia* eine nicht-lineare Variante von *Multimedia* dar.

Nach dieser Logik handelt es sich bei *Hyperaudio* in seiner klassischen Form eigentlich um reine Audiosequenzen, die miteinander verknüpft sind, wobei der Zuhörer selbst entschieden kann, in welcher Reihenfolge er diese abspielt (Zumbach und Kroeber, 2006).

2.3.2 Lernen mit Hypermedia

Wissenschaftler beschäftigen sich schon seit vielen Jahren damit, festzustellen, welche Effekte der Einsatz von *Multimedia*, *Hypertext* und *Hypermedia* auf den Lernerfolg von Lernenden hat.

Mayer (2009) konnte am Ende seiner Untersuchungen positive Effekte beim Einsatz von *Multimedia* nachweisen. So schnitten Studenten bei einem Transfertest besser ab, wenn sie mit Text und Bildern lernten, als wenn sie ausschließlich mit Text lernten. Mayer (2009) knüpft dieses Ergebnis aber an folgende Prinzipien, welche beim Einsatz von *Multimedia* befolgt werden müssen:

- Kohärenz: irrelevante Wörter, Töne und Bilder sollten vermieden werden
- Signalisieren: essentielle Inhalte sollten hervorgehoben sein
- Redundanz: Text sollte ausschließlich auditiv statt auditiv und visuell in Multimediapräsentationen wiedergegeben werden
- Räumliche Nähe: zusammengehörige Texte und Bilder sollten nah beieinander statt weit entfernt auf der Seite beziehungsweise dem Bildschirm dargestellt werden
- Zeitliche Nähe: zusammengehörige Texte und Bilder sollten gleichzeitig statt nacheinander auf der Seite beziehungsweise dem dargestellt werden
- Segmentierung: temporeiche, komplexe Multimedia–Lektionen sollten in benutzerfreundlichen Stücken statt im Ganzen präsentiert werden
- Vorbereitung: der Lernende sollte die Begrifflichkeiten und Merkmale der Lektion kennen
- Modalität: Text sollte auditiv statt visuell wiedergegeben werden

- Personalisierung: Texte sollten in dialogorientierterem Stil statt in formellem Stil wiedergegeben werden
- Stimme: Text sollte von einer freundlichen menschlichen Stimme statt einer Computerstimme wiedergegeben werden

Im Gegenzug verweisen Moos und Marroquin (2010) auf Arbeiten, nach denen eine Herausforderung bei *Multimedia* und somit auch bei *Hypermedia* darin besteht, dass die kognitive Aufnahmekapazität der Studierenden überschritten werden kann, wenn Informationen sowohl aus einem Text als auch aus einem Diagramm entnommen werden sollen (Mayer und Moreno; van Merriënboer und Ayres, nach Moos und Marroquin (2010)). Dies beruht auf der Annahme der Cognitive Load Theory, welche dem Arbeitsgedächtnis nur eine begrenzte Kapazität zuspricht (Sweller; van Merriënboer und Sweller, nach Moos und Marroquin (2010)).

Hypertext bietet zwar Vorteile, da der Studierende den Lernweg bestimmen kann, der am besten auf seine Bedürfnisse angepasst ist. Auf der anderen Seite ist hierzu aber eine ausreichende Vorkenntnis in dem entsprechenden Lernbereich notwendig, um die Entscheidung treffen zu können, wie dieser Weg aussehen soll. Des Weiteren wirkt sich auch ein fehlendes Interesse des Studierenden negativ auf die Effektivität der *Hypertext*-Lernumgebung aus (Lawless und Kulikowich, nach Moos und Marroquin (2010)).

Es ist nun also nicht verwunderlich, dass *Hypermedia* als Verschmelzung von *Multimedia* und *Hypertext* ebenfalls einige Herausforderungen mit sich bringt (Moos und Marroquin, 2010). Scott und Schwartz (nach Moos und Marroquin (2010)) fordern für das Lernen mit *Hypermedia* eine Balance zwischen effektiver Navigation und Inhaltsverständnis. Dies soll durch Prozesse zur Überwachung des eigenen Lernfortschritts erreicht werden, doch Untersuchungen haben ergeben, dass viele Studierende Schwierigkeiten dabei haben, diese Prozesse korrekt anzuwenden (Moos und Marroquin, 2010).

Donker und Blenn (2007) weisen auf die Probleme bei der Verwendung von *Hyperaudio* hin. Hier besteht die Herausforderung darin, dem Hörer die Hyperlinks innerhalb einer *Hyperaudio*-Anwendung sinnvoll darzustellen. Bei ihrer Untersuchung kamen Donker und Blenn (2007) zu dem Ergebnis, „dass sowohl eine Verdeutlichung von Links durch das Voranstellen eines Tons als auch die Variante durch das Ändern des Alters des Sprechers zu guten Ergebnissen hinsichtlich des Erkennens des Links führen.“ Die genannten vorangestellten Töne werden auch als *Audio Cues* bezeichnet.

2.3.3 Repräsentation von Kurseinheiten im Hyperaudio-Format

Nachdem nun die Begrifflichkeiten um *Hypermedia*, sowie den Begriff *Hyperaudio* im eigentlichen Sinne beleuchtet und entsprechende Studien betrachtet wurden, wird nun darauf eingegangen, welche Rückschlüsse daraus für diese Arbeit gezogen werden können.

Um eine auditive Repräsentation der Kurseinheiten umsetzen zu können, müssen deren Inhalte entsprechend auditiv aufbereitet werden. Inhalte, die nicht auditiv umgesetzt werden können, können in visueller Darstellung zum entsprechenden Zeitpunkt an den Audioinhalt annotiert werden.

In Ergänzung zu Zumbach und Kroeber (2006) wird in dieser Arbeit unter *Hyperaudio* ein Audioinhalt verstanden, der mittels der Erweiterung durch Kommunikations- und Interaktionsmöglichkeiten (vgl. Abschnitt 1.3) sowie visuelle Inhalte um Multimedia- und Hypermedia-Elemente ergänzt wird. Ein Hyperaudio-Dokument im Sinne dieser Arbeit besteht aus auditiven Inhalten und annotierten visuellen Inhalten. Bei den Hyperaudio-Dokumenten handelt es sich somit eigentlich um Multimedia-Dokumente. Erst die Kommunikations- und Interaktionsmöglichkeiten führen dazu, dass der *Hypermedia*-Aspekt erfüllt wird.

Der Vorteil dieser Betrachtungsweise von *Hyperaudio* liegt darin, dass die Herausforderungen, die in Verbindung mit *Hypermedia* bzw. *Hypertext* normalerweise auftreten, nicht besonders prägnant sind. Im zu entwickelnden Moodle-Plugin wird dem Studierenden in erster Linie ein linearer Audioinhalt

2 Grundlagen

vorgespielt, der um Multimedia-Elemente ergänzt wird. Erst durch den Einsatz der Kommunikations- und Interaktionsmöglichkeiten kommen die herausfordernden Elemente von *Hypermedia* ins Spiel. Dementsprechend stellt das Hyperaudio-Plugin einen guten Kompromiss zwischen den verschiedenen Lehrmethoden dar.

2.4 Zusammenfassung

Zusammenfassung schreiben

3 Analyse

Für die Konzeption und Implementierung des Hyperaudio-Plugins ist zunächst eine Analyse notwendig. Basierend auf einer Zielgruppenanalyse sollen Anforderungen an das Plugin bestimmt und priorisiert werden. Darauf aufbauend werden Möglichkeiten und Technologien zur Umsetzung des Hyperaudio-Plugins analysiert.

3.1 Zielgruppe

Im ersten Schritt soll die Zielgruppe anhand von *Personas* und deren *User Stories* festgelegt werden. Dies soll dann die Grundlage für die Definition der Anforderungen im nächsten Abschnitt darstellen.

3.1.1 Personas

Unter *Personas* werden fiktive Benutzer verstanden, für welche das Programm, in unserem Fall das Moodle-Plugin, designt wird (Cooper et al., 2004). Jeder Persona wird eine Rolle im Zusammenhang mit der Anwendung zugewiesen. Darüber hinaus wird die Persona ausreichend beschrieben, damit sich leicht in die Person hineinversetzt werden kann (Cohn, 2004). Dieses Vorgehen hilft dabei, möglichst authentische *User Stories* zu generieren, ohne auf echte Benutzer zurückgreifen zu müssen. Darüber hinaus ist festzustellen, dass die realen Anwender zwar Problemstellungen identifizieren können, aber nicht unbedingt in der Lage sind, Anforderungen zur Lösung dieser Probleme zu formulieren (Cooper et al., 2004). Für die Analyse und Konzeption des Hyperaudio-Plugins wird mit folgenden Personas gearbeitet:

Prof. Dr. Karolin Schröder ist verantwortlich für den Kurs „Einführung in die Wirtschaftsinformatik“. In diesem Kurs werden bereits erfolgreich Hyperaudio-Dokumente eingesetzt. Nachdem Prof. Dr. Schröder mit dem Start des nächsten Semesters überarbeitete Kurseinheiten anbietet, müssen nun auch die zugehörigen Hyperaudio-Dokumente auf die Notwendigkeit einer Überarbeitung hin überprüft werden. Die veralteten Hyperaudio-Dokumente müssen dann durch neuere Versionen ersetzt werden.

Dr. Julian Schmidt ist wissenschaftlicher Mitarbeiter und Betreuer für den Kurs „Marketing“, der ebenfalls das Lernen mittels Hyperaudio-Dokument anbietet. Dr. Julian Schmidt ist unter anderem für die Betreuung dieser verantwortlich und ist derjenige, der hier Rede und Antwort steht.

Laura Ebert ist 35 Jahre alt, verheiratet und hat drei Kinder. Sie geht halbtags ihrem Beruf als Anwendungsentwicklerin nach, zu dem sie 30 Minuten mit dem Bus pendelt. Neben der Arbeit kümmert sie sich zusammen mit ihrem Mann um den Haushalt und die Kinder. Laura studiert in Teilzeit den Bachelorstudiengang Informatik im ersten Semester. Das Semester hat erst vor einigen Wochen begonnen und sie entdeckt gerade Moodle für sich. Hierbei ist sie auf die Hyperaudio-Dokumente gestoßen und hat sich fest vorgenommen, sich im Laufe des Semesters mit deren Hilfe

3 Analyse

mit den Lerninhalten auseinanderzusetzen.

Max Lustig ist 24 Jahre alt, ledig und hat sich nach einer abgeschlossenen Ausbildung zum Informatikkaufmann dazu entschlossen, neben dem Beruf zu studieren. Zur Arbeit kommt er in wenigen Minuten zu Fuß. Viel Zeit verbringt er jedoch im Fitnessstudio mit Kraft- und Ausdauertraining. Er absolviert ein Vollzeitbachelorstudium in Wirtschaftsinformatik und befindet sich kurz vor der Prüfungsphase zum Ende des dritten Semesters. Max möchte sich nun auf die Klausur des Moduls „Investition und Finanzierung (BWL II)“, welche in zwei Wochen stattfindet, intensiv vorbereiten. Im Laufe des Semesters hat er bereits ausgiebig die neuen Hyperaudio-Dokumente zum Erreichen des Lernziels genutzt.

3.1.2 User Stories

Unter Zuhilfenahme der entwickelten Personas werden im nächsten Schritt *User Stories* formuliert. *User Stories* beschreiben, wie die klassischen *Use Cases*, Anforderungen an ein Softwaresystem. Diese sind dabei im Vergleich wesentlich oberflächlicher und ungenauer formuliert als *Use Cases* (Wirdemann, 2017).

Erst im Laufe der Entwicklung werden *User Stories* konkreter und dienen am Ende dazu, das Entwicklungsergebnis zu validieren. Beim Erstellen von *User Stories* ist zu beachten, dass sogenannte *Epic*s, das sind *User Stories* mit sehr großem Umfang, wenn möglich in kleinere *User Stories* aufgesplittet werden. Unter anderem ist zu beachten, dass die *User Stories* keine Abhängigkeiten untereinander aufweisen und dass deren Erfüllung überprüfbar ist. *User Stories* können im *Connextra Format* festgehalten werden, welches wie folgt aufgebaut ist (Cohn, 2004):

Ich als (Rolle) möchte (Funktion), um (Nutzen).

Mit den Unterschieden und Einsatzzwecken von Personas und Rollen beschäftigt sich Constantine (2006). Grundsätzlich ist demnach festzustellen, dass Personas eher aus dem *User-centered design* (Norman und Draper, 1986), Benutzerrollen dagegen aus dem *Usage-centered design* (Constantine, 1996) motiviert sind. Während sowohl Personas als auch Benutzerrollen durchaus nützlich sind, um ein Verständnis von den Nutzern eines Systems zu erhalten, unterscheiden sie sich nach Constantine (2006) in ihrer Philosophie und Relevanz für das Interaktionsdesign. Im Gegensatz zu Personas, die wie bereits im vorangegangenen Abschnitt 3.1.1 beschrieben dazu dienen, die Nutzersicht anhand möglichst realer Personen darzustellen, sollen Benutzerrollen in einer wesentlich technischeren Sicht ein abstrahiertes Modell für die Art und Weise, in der Nutzer mit dem System interagieren, bilden (Constantine, 2006).

Für die weitere Analyse soll das Beste beider Welten vereinbart werden. Um die *User Stories* möglichst anschaulich zu halten, werden sie anhand der vorgestellten Personas definiert. Gleichzeitig wird eine Aufteilung in Benutzerrollen vorgenommen, die die Grundlage für die Ableitung von Anforderungen aus den *User Stories* bilden soll.

Für die Konzeption des Hyperaudio-Plugins ergeben sich im Wesentlichen zwei Benutzerrollen wie in Abbildung 3.1 dargestellt. *Administrierende* sind diejenigen Anwender, die Lerninhalte in Form von Hyperaudio-Dokumenten bereitstellen und verwalten. Diese Rolle kann nur von Lehrenden eingenommen werden. Die Gruppe der *Nutzenden* kann derweil aus Lehrenden sowie Studierenden bestehen, die mit Hyperaudio-Dokumenten interagieren möchten.

Für das Angebot einer Hyperaudio-Lernumgebung lassen sich folgende *User Stories* festhalten:

US-1: Als Administrierende möchte Prof. Dr. Karolin Schröder ein neues Hyperaudio-Dokument in ihrem Kurs „Einführung in die Wirtschaftsinformatik“ zur Verfügung stellen, um den Studie-



Abbildung 3.1: Benutzerrollen

renden neue Lerninhalte bereitzustellen.

- US-2: Als Administrierende möchte Prof. Dr. Karolin Schröder Hyperaudio-Dokumente aus ihrem Kurs „Einführung in die Wirtschaftsinformatik“ löschen können, um veraltete Informationen zu entfernen.
- US-3: Als Administrierende möchte Prof. Dr. Karolin Schröder Hyperaudio-Dokumente aus ihrem Kurs „Einführung in die Wirtschaftsinformatik“ im Sommersemester in den darauffolgenden Kurs im Wintersemester übernehmen, um diese nicht erneut erstellen zu müssen.
- US-4: Als Administrierende möchte Prof. Dr. Karolin Schröder Hyperaudio-Dokumente anderer Kurse in ihren Kurs „Einführung in die Wirtschaftsinformatik“ übernehmen, um auf die hervorragende Arbeit anderer Lehrender zurückgreifen zu können, da sich die Themen mit ihrem Kurs überschneiden.
- US-5: Als Administrierende möchte Prof. Dr. Karolin Schröder Erkenntnisse daraus gewinnen, wie die Hyperaudio-Dokumente des Kurses „Einführung in die Wirtschaftsinformatik“ von Studierenden genutzt werden, um Verbesserungspotenzial auszumachen.
- US-6: Als Administrierender möchte Dr. Julian Schmidt ein vorhandenes Hyperaudio-Dokument in dem von ihm betreuten Kurs „Marketing“ überarbeiten, um einen Fehler zu beseitigen.
- US-7: Als Nutzende möchte Prof. Dr. Karolin Schröder die bereits vorhandenen Hyperaudio-Dokumente aus ihrem Kurs „Einführung in die Wirtschaftsinformatik“ wiedergeben, um diese auf ihre Richtigkeit zu überprüfen.
- US-8: Als Nutzende möchte Prof. Dr. Karolin Schröder die Kommentare zu einem Hyperaudio-Dokument lesen und beantworten können, um auf Fragen von Studierenden einzugehen.
- US-9: Als Nutzende möchte Prof. Dr. Karolin Schröder eine Notiz zu einem Hyperaudio-Dokument machen, um ihren Gedanken festzuhalten und später darauf zurückgreifen zu können.
- US-10: Als Nutzender möchte Dr. Julian Schmidt eine gefundene Erklärungslücke in einem Hyperaudio-Dokument durch einen Kommentar zum entsprechenden Zeitpunkt schließen, um eventuellen Fragen der Studierenden vorzukommen.
- US-11: Als Nutzende möchte Laura Ebert mittels Hyperaudio-Dokument lernen, um die Zeit während Haushaltsarbeiten, wie dem Bügeln, Kochen oder Putzen, und dem Pendeln sinnvoller zu nutzen.
- US-12: Als Nutzende möchte Laura Ebert erfahren, welche Hyperaudio-Dokumente in den von ihr belegten Kursen angeboten werden, um herauszufinden, mit welchen Mitteln sie sich auf die anstehenden Prüfungen vorbereiten kann.
- US-13: Als Nutzende möchte Laura Ebert einen Kommentar verfassen, um dem Kursbetreuer und den anderen Studierenden eine Frage zu stellen.
- US-14: Als Nutzende möchte Laura Ebert eine Markierung setzen, wenn eine klausurrelevante Thema-

3 Analyse

tik erklärt wird. Bei der Prüfungsvorbereitung möchte sie anhand dieser Markierungen diejenigen Themen erkennen, mit welchen sie sich besonders intensiv beschäftigen möchte.

- US-15: Als Nutzende möchte Laura Ebert eine Markierung löschen, da sie den markierten Lerninhalt inzwischen beherrscht. Anhand der übrigen Markierungen möchte sie schnell erkennen, wo für sie noch Lernbedarf besteht.
- US-16: Als Nutzende möchte Laura Ebert eine Notiz erstellen, um ein Beispiel zu dem genannten Sachverhalt festzuhalten, sodass sie die Thematik beim nächsten Mal einfacher nachvollziehen kann.
- US-17: Als Nutzende möchte Laura Ebert die Wiedergabe eines Hyperaudio-Dokuments beenden und am nächsten Tag automatisch an derselben Stelle fortsetzen können, um das Lernen schnell wiederaufnehmen zu können.
- US-18: Als Nutzende möchte Laura Ebert die Hyperaudio-Angebote mit ihrem Smartphone in Anspruch nehmen, um auch die Zeit während des Pendelns zum Lernen nutzen zu können.
- US-19: Als Nutzender möchte Max Lustig eine alte Notiz bearbeiten, um einen Schreibfehler zu korrigieren.
- US-20: Als Nutzender möchte Max Lustig eine alte Notiz löschen, da er inzwischen Lernfortschritte gemacht hat und auf diese Notiz verzichten kann.
- US-21: Als Nutzender möchte Max Lustig schnell erkennen welche Inhalte im Hyperaudio-Dokument behandelt werden, um eine Erklärung eines bestimmten Themas zu finden.
- US-22: Als Nutzender möchte Max Lustig nach Textinhalten in Kommentaren suchen können, um schnell Erklärungen zu finden.
- US-23: Als Nutzender möchte Max Lustig die Kommentare nach Erstellungsdatum sortieren können, um sich einen Überblick über die neuesten Aktionen zu verschaffen.
- US-24: Als Nutzender möchte Max Lustig die Kommentare und persönlichen Notizen zu den Annotationszeitpunkten zuordnen können, um diese bei der Wiedergabe verfolgen zu können.
- US-25: Als Nutzender möchte Max Lustig öffentliche Kommentare und persönliche Notizen getrennt betrachten können, um die öffentliche Diskussion verfolgen beziehungsweise die eigenen Anmerkungen isoliert betrachten zu können.
- US-26: Als Nutzender möchte Max Lustig auf Kommentare antworten können, um sich mit den Studierenden und Lehrenden auszutauschen.
- US-27: Als Nutzender möchte Max Lustig erkennen, welche Hyperaudio-Dokumente er zuletzt abgespielt hat, um seinen Lernfortschritt im Auge zu behalten.
- US-28: Als Nutzender möchte Max Lustig besonders hilfreiche Hyperaudio-Dokumente als Favoriten speichern, um diese schnell als solche identifizieren zu können.
- US-29: Als Nutzender möchte Max Lustig die Markierung als Favorit entfernen können, wenn der Inhalt für ihn nicht mehr von Interesse ist.
- US-30: Als Nutzender möchte Max Ebert auf seinem Tablet Zugang zu Hyperaudio-Dokumenten haben, um die Zeit auf dem Laufband gleichzeitig zum Lernen nutzen zu können.

3.2 Anforderungsdefinition

Basierend auf den *Personas*, Rollen und *User Stories* können nun die Anforderungen für das Hyperaudio-Plugin definiert werden. Hierbei werden aus einer oder mehreren *User Stories* jeweils eine oder mehrere Anforderungen abgeleitet und zugleich mit einer Priorität versehen. Für die Priorisierung stehen die drei Prioritätsstufen *niedrig*, *mittel* und *hoch* zur Verfügung. Bei der Priorisierung der Anforderungen

soll die Zielsetzung aus Kapitel 1 als Orientierung dienen.

3.2.1 Anforderungen der Administrierenden

Es wird mit der Definition der Anforderungen der Administrierenden begonnen, welche in Tabelle 3.1 festgehalten werden.

Aus US-1 kann die Anforderung des Erstellens von Hyperaudio-Dokumenten abgeleitet werden. Dass bestehende Hyperaudio-Dokumente auch bearbeitet und gelöscht werden können sollen, ergibt sich aus US-6 und US-2. Zusammen stellen diese Anforderungen die Grundfunktionalitäten für die alternative Repräsentation der Lerninhalte dar und werden dementsprechend mit der Prioritätsstufe *hoch* versehen.

Aus US-3 und US-4 ergibt sich die Anforderung, dass Hyperaudio-Dokumente in einen anderen Kurs übernommen werden können sollen, sei es der gleiche Kurs im nächsten Semester oder ein anderer Kurs. Diese Anforderung zählt nicht zu den Grundfunktionalitäten, kann das Verwalten von Hyperaudio-Dokumenten jedoch vereinfachen und wird daher mit der Prioritätsstufe *mittel* bewertet.

Der Wunsch, Erkenntnisse aus der Nutzung von Hyperaudio-Dokumenten durch die Studierenden zu erhalten (US-5), schlägt sich in der Anforderung nach statistischen Auswertungsmöglichkeiten nieder. Für diese Anforderung wird die Priorität *niedrig* vergeben, da es sich um ergänzende Metainformationen handelt.

Tabelle 3.1: Anforderungen der Administrierenden

Nr.	Anforderung	Priorität
1	Erstellen eines Hyperaudio-Dokuments	hoch
2	Bearbeiten eines Hyperaudio-Dokuments	hoch
3	Löschen eines Hyperaudio-Dokuments	hoch
4	Übernahme eines Hyperaudio-Dokuments in einen anderen Kurs	mittel
5	Statistische Auswertungen über die Nutzung der Hyperaudio-Dokumente	niedrig

3.2.2 Anforderungen der Nutzenden

Die aus den *User Stories* der Nutzenden abgeleiteten Anforderungen sind in Tabelle 3.2 festgehalten. Die Priorisierung wird dabei nach folgenden Kriterien vorgenommen:

- *hoch*: Basisfunktionalität zum Erreichen der Zielsetzung der Arbeit
- *mittel*: Verbesserung der Interaktion mit Hyperaudio-Dokumenten und Annotationen
- *niedrig*: Vereinfachter Zugriff auf Hyperaudio-Dokumente

Aus US-7, US-11, US-18 und US-30 ergibt sich die grundlegende Anforderung, Hyperaudio-Dokumente abspielen zu können. US-11, US-18 und US-30 führen zudem zur Anforderung der Audio Cues, mithilfe derer auf annotierte Zusatzinhalte hingewiesen wird. Auch diese zählen neben der Wiedergabe zu den Basisfunktionalitäten, da erst dadurch das Ziel der größeren zeitlichen Flexibilität beim Lernen erreichbar wird (vgl. Abschnitte 1.3 und 2.3.2). Um Inhalte schnell auffinden zu können, wie in US-21 gefordert, ist eine Übersicht über annotierte Zusatzinhalte nützlich.

Basierend auf US-8, US-10, US-13, US-26 und US-24 lässt sich die Anforderung an eine Kommentarfunktion ableiten, die über Möglichkeiten zum Erstellen, Anzeigen und Beantworten von Kommentaren verfügen muss. An dieser Stelle kann die Interaktion durch eine Suchfunktion innerhalb der

3 Analyse

Kommentare verbessert werden (vgl. US-22). In den *User Stories* US-9, US-16, US-19, US-20 und US-24 werden Wünsche bezüglich einer Notizfunktion formuliert. Diese lässt sich aufschlüsseln in die Anforderungen zum Erstellen, Anzeigen, Bearbeiten und Löschen von Notizen. Die Notizfunktion spiegelt das Ziel der Erhaltung typischer Nutzerinteraktionen mit textuellen Lernmedien wieder und kann das Lernen für die Studierenden erleichtern (Scutter et al., 2010). Ähnlich sind die Anforderung zum Erstellen, Anzeigen und Löschen von persönlichen Markierungen aus US-14 und US-15 zu bewerten. Obwohl Markierungen, ebenso wie die Notizfunktion, zu den Basisfunktionalitäten des Plugins gezählt werden können, wird die Markierungsfunktion mit der Priorität *mittel* versehen. Das Herabsetzen der Priorität wird dadurch begründet, dass eine Markierung im Wesentlichen einer Notiz ohne textuellen Inhalt entspricht und durch die höher priorisierte Notizfunktion abgebildet werden kann. Aus US-23, US-24 und US-25 ergibt sich zudem der Wunsch nach Filter- und Sortiermöglichkeiten.

Der Wunsch nach einer Favoritenfunktion für Hyperaudio-Dokumente ergibt sich aus US-28 und US-29. Daraus resultieren die Anforderungen, Favoriten zu setzen, anzuzeigen und zu löschen. Übersichten über Hyperaudio-Dokumente werden in US-12 und US-27 gefordert: im ersten Fall eine Übersicht über alle Hyperaudio-Dokumente der belegten Kurse und im zweiten Fall eine Übersicht über die zuletzt abgespielten Hyperaudio-Dokumente. Sowohl die Favoritenfunktion als auch die Übersichten stellen eine reine Optimierung der Navigation dar und verhelfen somit zu einem vereinfachten Zugriff auf Hyperaudio-Dokumente. US-17 bringt die Anforderung für eine Funktion zum Fortsetzen unterbrochener Wiedergaben bei folgenden Aufrufen in Moodle hervor. Auch diese Funktion vereinfacht die Navigation zum gewünschten Hyperaudio-Dokument beziehungsweise dessen Inhalt.

Das Verlangen, Hyperaudio-Dokumente auch auf einem Smartphone oder Tablet nutzen zu können, wie in US-18 und US-30 beschrieben, resultiert in der Anforderung zur Unterstützung von mobilen Endgeräten. Da es sich um eine Verbesserung der Interaktionsmöglichkeiten handelt, wird der Anforderung die Priorität *mittel* zugewiesen.

3.3 Möglichkeiten der Moodle Plugin-Entwicklung

Für den Betrieb von Moodle werden ein Webserver, eine MySQL- oder PostgreSQL-Datenbank und PHP vorausgesetzt (Moodle, 2018d). Moodle unterstützt die Verwendung von JavaScript und die Einbindung von Thirdparty-Frameworks (Wild, 2017). Dies gepaart mit den Möglichkeiten durch den Einsatz von PHP bietet bei der Entwicklung ausreichend Möglichkeiten um das gewünschte Plugin umzusetzen.

Des Weiteren wird die Plugin-Entwicklung von Moodle mithilfe der sogenannten *Core APIs* (Application Programming Interfaces) unterstützt. Moodle bietet für fast jeden Anwendungszweck eine passende API, welche über ein Objekt angesprochen werden kann (Wild, 2017). So kann beispielsweise unter Verwendung der DML-API (Data Manipulation Language) und dem dazugehörigen Objekt \$DB Zugriffe auf die Datenbank vorgenommen werden.

Zu Beginn der Entwicklung eines Moodle-Plugins steht jedoch die Frage, um welche Art von Plugin es sich handelt. Es werden nun diejenigen Plugin-Typen beschrieben, welche für das Hyperaudio-Plugin infrage kommen (Moodle, 2017b).

Media Player

Mit diesem Plugin-Typ kann Moodle um alternative Player für Audio- und Videoformate, aber auch für andere Medien (z.B. Diagramme, Formeln, etc.) ergänzt werden (Moodle, 2017a). Player beziehen sich aber stets auf das reine Abspielen von Dateien oder Links zu externen Medieninhalten, wie zum Beispiel ein Link zu einem *Youtube*-Video.

Tabelle 3.2: Anforderungen der Nutzenden

Nr.	Anforderung	Priorität
1	Wiedergabe von Hyperaudio-Dokumenten	hoch
2	Hinweise auf die Darstellung von annotierten Zusatzinhalten	hoch
3	Übersicht über annotierte Zusatzinhalte	mittel
4	Kommentarfunktion bei Hyperaudio-Dokumenten	mittel
4.1	Erstellen von Kommentaren	hoch
4.2	Anzeigen von Kommentaren	hoch
4.3	Antworten auf Kommentare	hoch
4.4	Suchfunktion innerhalb der Kommentare	mittel
5	Notizfunktion bei Hyperaudio-Dokumenten	mittel
5.1	Erstellen von Notizen	hoch
5.2	Anzeigen von Notizen	hoch
5.3	Bearbeiten von Notizen	hoch
5.4	Löschen von Notizen	hoch
6	Markierungsfunktion bei Hyperaudio-Dokumenten	mittel
6.1	Erstellen von Markierungen	mittel
6.2	Anzeigen von Markierungen	mittel
6.3	Löschen von Markierungen	mittel
7	Filter- und Sortermöglichkeiten	mittel
8	Favoritenfunktion für Hyperaudio-Dokumente	niedrig
8.1	Erstellen von Favoriten	niedrig
8.2	Anzeigen von Favoriten	niedrig
8.3	Löschen von Favoriten	niedrig
9	Übersicht über alle Hyperaudio-Dokumente der belegten Kurse	niedrig
10	Übersicht über die zuletzt abgespielten Hyperaudio-Dokumente	niedrig
11	Funktion zum Fortsetzen unterbrochener Wiedergaben bei folgenden Aufrufen in Moodle	niedrig
12	Unterstützung von mobilen Endgeräten	mittel

Blöcke

Blöcke dienen dazu, Kursseiten um zusätzliche Informationen anzureichern, welche dann in der rechten oder linken Spalte angeheftet werden können (siehe Abbildung 2.2). Blöcke können auch „angeheftet“ werden (Moodle, 2018b). Diese Anheftung kann auf verschiedene Bereiche erfolgen, beispielsweise in der gesamten Moodle-Umgebung, auf der Seite des Benutzerprofils oder der Startseite (Moodle, 2015b). Blöcke sind also nicht dazu geeignet größere Inhalte darzustellen. Dementsprechend kommt dieser Plugin-Typ nicht für unser Hyperaudio-Plugin in Frage. Es wäre aber durchaus denkbar, dass mithilfe von Blöcken die Anforderungen 8.2, 9 und 10 umgesetzt werden könnten.

Ressourcen

Mittels eines Plugins dieses Typs ist es möglich, dem Studierenden Inhalte zu präsentieren, wobei dieses Plugin keinerlei Eingabe oder Interaktion von Seiten des Studierenden erwartet und ausschließlich dazu dient, Informationen darzustellen (Wild, 2017).

Aktivitäten

Neben der Darstellung von Inhalten erwarten Aktivitäten im Gegensatz zu Ressourcen eine Art von Interaktion durch den Studierenden. Dies kann beispielsweise in Form eines Quiz, bei dem der Nutzer Antworten anhaken muss, oder in Form eines Forums, in dem der Studierende Beiträge schreibt, geschehen (Wild, 2017). Aufgrund des interaktiven Charakters stellen Aktivitäten den richtigen Plugin-Typ für das Hyperaudio-Plugin dar.

3.4 Aktueller Stand der Technik

Bevor mit der Konzeption und Implementierung des Hyperaudio-Plugins begonnen werden kann, ist der aktuelle Stand der Technik bezüglich der Zielsetzung dieser Arbeit zu betrachten. Hierbei werden im ersten Schritt bereits etablierte Plattformen für die Bereitstellung von Audio- und Videoinhalten begutachtet. Im zweiten Schritt werden dann vorhandene Technologien für die Umsetzung innerhalb von Moodle untersucht.

3.4.1 Etablierte Audio- und Video-Plattformen

Mit dem Hintergrund, eine nach DIN EN ISO 9241 erwartungskonforme Software gestalten zu wollen, erfolgt nun zunächst eine Analyse von etablierten Systemen zur Wiedergabe von Audio- und Videoinhalten mit integrierten Kommunikationsmöglichkeiten. Da menschliches Handeln stark durch erlernte Verhaltensmuster geprägt ist, empfiehlt es sich, bei der Gestaltung von Interaktionen auf bekannte Verfahren zurückzugreifen, um den kognitiven Aufwand zum Erlernen der Bedienmöglichkeiten gering zu halten und somit eine Konzentration auf die wesentlichen Inhalte zu ermöglichen (Rampl, Hansjörg, 2007).

SoundCloud

„Als weltweit größte Musik- und Audio-Plattform“ (SoundCloud, o.D.) bietet *SoundCloud* Künstlern eine Plattform, um ihre Musik einem breiten Publikum anzubieten. Charakteristisch für *SoundCloud* ist das Design des Players (siehe Abbildung 3.2). Zum einen wird hier die Waveform des Musikstückes angezeigt und zum anderen werden gleichzeitig mittels Thumbnails Kommentare an ebenjener Stelle des Stücks visualisiert, zu der kommentiert wurde. Beim Abspielen des Musikstücks werden die annotierten Kommentare zum jeweiligen Zeitpunkt eingeblendet. Zusätzlich bietet der Player auch durch Mouseover-Effekte auf den Thumbnails die Möglichkeit, die annotierten Kommentare zu lesen. Durch einen Klick auf das entsprechende Thumbnail kann direkt auf den Kommentar geantwortet werden. Unterhalb des Players befindet sich der Eingabebereich, um eigene Kommentare zu verfassen. Diese werden zu dem Zeitpunkt gespeichert, zu dem der Kommentar begonnen wurde. Wiederum unterhalb des Eingabebereichs befindet sich ein Bereich für die Anzeige der Kommentare. Sie werden chronologisch nach Erstellungsdatum, mit dem neusten Kommentar an oberster Stelle, dargestellt. Antworten auf Kommentare werden durch eine leichte Einrückung gekennzeichnet.



Abbildung 3.2: Player der Musik- und Audio-Plattform *SoundCloud* (SoundCloud, 2015)

Youtube

Im Bereich der Videoplattformen gilt *Youtube* als die mit Abstand am weitesten verbreitete Videoplattform in Deutschland (Statista, 2016). Zum Abspielen der von den Nutzern hochgeladenen Videos setzt *Youtube* auf den HTML5-Player⁴ (siehe Abbildung 3.3).



Abbildung 3.3: Player der Video-Plattform *Youtube* (Youtube, 2015)

In (Google, 2018a,b) werden Möglichkeiten aufgezeigt, Informationen an ein Video zu annotieren. Die Informationen können Verweise auf andere Videos, Playlists und Kanäle, eine Abstimmung oder einen Link zu einer Webseite beinhalten. Einem Video können insgesamt maximal ein Abspann und bis zu fünf Infokarten mittels des integrierten Webeditors angeheftet werden. Bei Infokarten kann nur der jeweilige Startzeitpunkt für die Anzeige frei festgelegt werden, die Dauer wird durch *Youtube* vorgegeben. Abbildung 3.4 zeigt, wie eine solche Infokarte im Player dargestellt wird. Ein Abspann kann hingegen nur während der letzten fünf bis 20 Sekunden eines Videos angezeigt werden.

[Infokarten Beschreibung überarbeiten](#)

3.4.2 Technologien für den Einsatz in Moodle

Im zweiten Schritt wird sich nun der Analyse bestehender Komponenten zugewendet, die als Basis für das Moodle-Plugin dienen könnten. Ziel ist es festzustellen, ob bereits Technologien existieren, mit deren Hilfe die Idee des Hyperaudio-Plugins umgesetzt werden kann oder ob zumindest Teile davon - unter entsprechender Beachtung der Lizenzierung - sinnvoll wiederverwendet werden können. Im Zuge dessen wird so vorgegangen, dass die einzelnen vorhanden Technologien auf diesem Gebiet

⁴HTML = HyperText Markup Language

3 Analyse



Abbildung 3.4: Anzeige der Infokarte (Youtube, 2015)

mit ihren Funktionen vorgestellt werden. Dabei wird deren Relevanz für die Umsetzung der Plugins begutachtet.

VideoJS Player

Bei dem *VideoJS Player*⁵ handelt es sich um eine Open-Source-Bibliothek zum Abspielen von Videos und stellt damit einen HTML5-Video-Player zur Verfügung. Der *VideoJS Player* ist bereits als Standard-Plugin für die Wiedergabe von Audio- und Video-Dateien in Moodle integriert. Wie der Name schon erkennen lässt, handelt es sich hierbei um eine JavaScript-Bibliothek. Der *VideoJS Player* beschränkt sich in seiner Ausgangsversion ausschließlich auf das Abspielen von Audio- und Video-Dateien und bietet bis auf ein optionales Fallback auf den Adobe FlashPlayer keine weiteren Funktionen. Die Funktionalität des *VideoJS Player* kann aber über Plugins erweitert werden. Es existieren bereits zahlreiche solcher Plugins. Hier sei vor allem das Plugin *videojs-wavesurfer*⁶ genannt, welches das *wavesurfer.js*-Framework (siehe Abschnitt 3.4.2) in den *VideoJS Player* integriert. Dank der Unterstützung von Plugins ist es durchaus denkbar, den Player mittels Plugin beispielsweise um Buttons zum Erstellen von Kommentaren oder persönlichen Notizen zu erweitern. Auch wäre es denkbar, mittels eines Plugins die annotierten Kommentare zu visualisieren. Grundsätzlich stellt der *VideoJS Player* somit eine gute Ausgangslage für einen Hyperaudio-Player dar.

H5P

Mit *H5P* und dem bereits vorhanden Plugin für Moodle⁷ ist es möglich, verschiedene Arten von interaktiven Lerninhalten zu gestalten. Dabei handelt es sich um eine Sammlung von interaktiven Komponenten, darunter *Course Presentation*, *Timeline* und *Interactive Video*. *Course Presentation* bietet die Möglichkeit, interaktive Präsentationen zu gestalten. *Timeline* kann genutzt werden um Inhalte anhand eines Zeitstrahls darzustellen. *Interactive Video* ermöglicht, ähnlich wie *Course Presentation*, die Interaktion während des Abspielens eines Videos (siehe Abbildung 3.5). Besonders erwähnenswert ist, dass sich die interaktiven Inhalte bei *H5P* innerhalb der Weboberfläche erstellen lassen. Es wäre also denkbar, eine eigene interaktive Komponente zu entwickeln, welche es ermöglicht, Hyperaudio-Dokumente als interaktiven Lerninhalt zu erstellen und abzuspielen sowie eine Kommentarfunktion zu integrieren.

⁵GitHub-Projekt, Apache-Lizenz 2.0: <http://videojs.com/>; <https://github.com/videojs>

⁶GitHub-Projekt, MIT Lizenz: <https://github.com/collab-project/videojs-wavesurfer>

⁷GitHub-Projekt, GNU General Public License v2.0: <https://github.com/h5p/h5p-moodle-plugin>

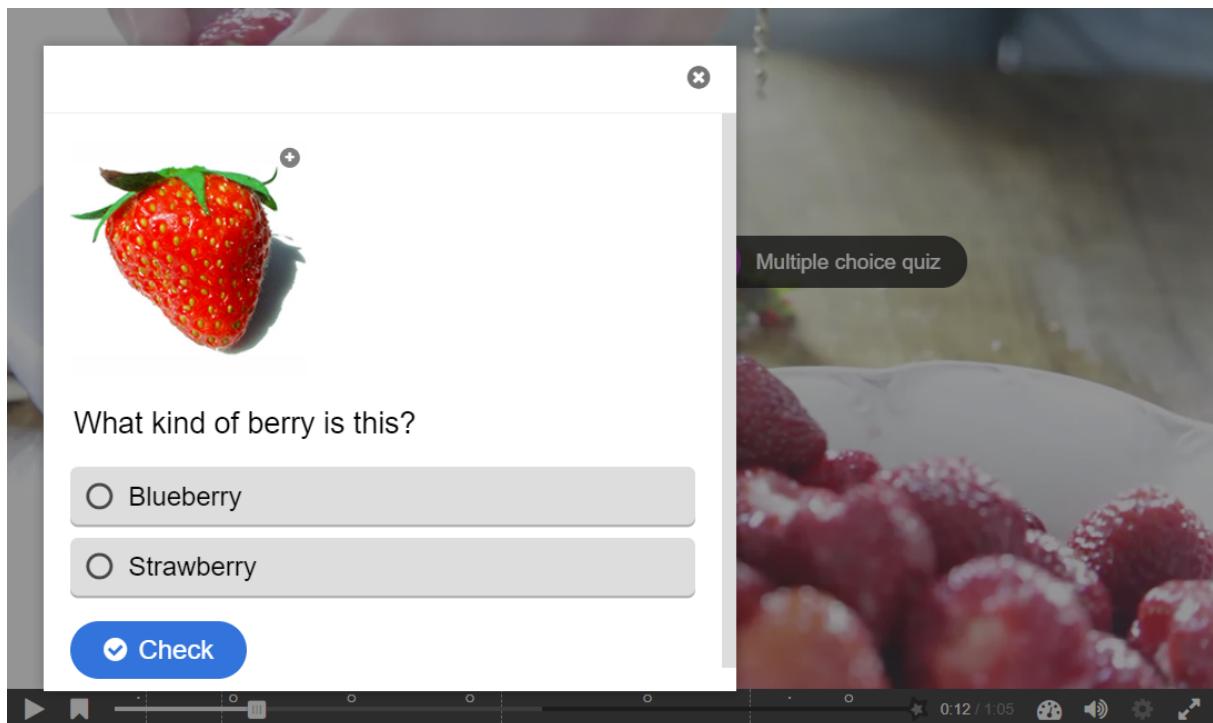


Abbildung 3.5: Auszug aus einem *Interactive Video* (H5P, 2013)

Popcorn.js

Die Mozilla Corporation bietet mit *Popcorn.js*⁸ eine Bibliothek an, welche neben einer standardisierten Steuerung von Medieninhalten aus verschiedenen Quellen auch die zeitabhängige Annotation von Inhalten mittels Plugins ermöglicht. Hier wäre also auch eine Entwicklung eines Plugins denkbar, mit welchem Hyperaudio-Dokumente wie gewünscht wiedergegeben werden können. Die Wartung für die Bibliothek wurde seitens Mozilla zwar eingestellt, das Projekt steht aber weiterhin auf GitHub zur Verfügung. Obwohl das Projekt nicht mehr weiterentwickelt wird, kann es durch die vorhandenen Steuerungsmöglichkeiten und das Plugin-System ein geeignetes Grundgerüst für die Entwicklung des Moodle-Plugins darstellen.

wavesurfer.js

Bei *wavesurfer.js*⁹ handelt es sich um ein JavaScript-Framework, welches es ermöglicht, die Wellenform zu der abgespielten Audio-Datei visualisieren zu lassen (siehe Abbildung 3.6). Diese Basisfunktionalität wurde durch Weiterentwicklungen um nützliche Funktionen erweitert. Auf zwei dieser Weiterentwicklungen wird im Folgenden eingegangen.

Der *audio-annotator*¹⁰ stellt eine auf dem *wavesurfer.js*-Framework basierende Weiterentwicklung dar, welche es mittels Weboberfläche ermöglicht, Annotationen in Form von Text an eine Audio-Datei anzuhafte. Es erweitert *wavesurfer.js* also um die Möglichkeit, Annotationen an eine Datei anzuhafte und bietet gleichzeitig noch eine Oberfläche, um ebendiese Annotationen vorzunehmen.

Beim *BAT - BMAT Annotation Tool*¹¹ handelt es sich um eine Entwicklung basierend auf dem im Zusammenhang von *audio-annotator* erweiterten Frameworks *wavesurfer.js*. Es ermöglicht, ebenso

⁸GitHub-Projekt, MIT Lizenz: <https://github.com/mozilla/popcorn.js>

⁹GitHub-Projekt, BSD-3-Clause: <https://wavesurfer-js.org>; <https://github.com/katspaugh/wavesurfer.js>

¹⁰GitHub-Projekt, BSD-2-Clause: <https://github.com/CrowdCurio/audio-annotator>

¹¹GitHub-Projekt, GNU General Public License 3: <https://wavesurfer-js.org>; <https://github.com/BlaiMelendezCatalan/BAT>

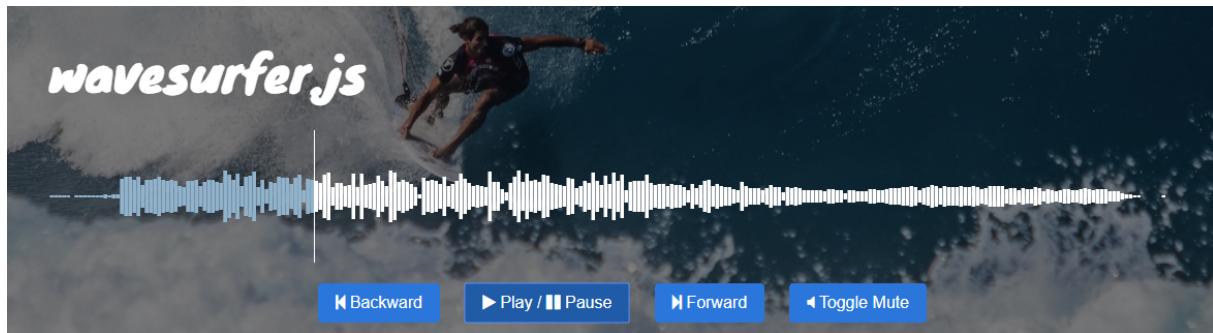


Abbildung 3.6: Der Audio-Player auf der *wavesurfer.js*-Webseite(H5P, o.D.)

wie *audio-annotator*, dem Benutzer mittels Weboberfläche Annotationen an einer Audio-Datei vorzunehmen. Somit bietet *BAT* logischerweise dieselben Vorzüge wie bereits der *audio-annotator*. Im Vergleich zum *audio-annotator* stellt *BAT* jedoch ein weiterentwickelteres Framework dar.

Das *wavesurfer.js* Framework - speziell mit seinen Weiterentwicklungen - bietet einige Funktionen, die für das Abspielen von Hyperaudio-Dokumenten nützlich sein könnten. Zusätzlich bietet es auch die Funktion, die entsprechenden Annotationen in einer Weboberfläche an die Audio-Dateien anzuheften. Grundsätzlich lässt sich feststellen, dass *wavesurfer.js* und seine Ableger im Vergleich zu den zuvor betrachteten Entwicklungen einen wesentlich unausgereifteren Eindruck hinterlassen.

timesheets.js

*timesheets.js*¹² ist ebenfalls ein JavaScript-Framework, welches analog zu *audio-annotator* und *BAT* die Annotation zusätzlicher Inhalte ermöglicht. Leider befindet sich das Framework aktuell nicht mehr in der Entwicklung. Aufgrund der Ähnlichkeit zu den *wavesurfer.js*-Ablegern und der eingestellten Entwicklung können hier zwar Ideen übernommen werden, als Basis für das zu entwickelnde Moodle-Plugin ist dieses Framework jedoch nicht geeignet.

Zusammenfassend ist festzustellen, dass für die Entwicklung des Plugins für Hyperaudio-Dokumente vor allem *VideoJS Player*, *H5P* und *Popcorn.js* die vielversprechendsten bestehenden Entwicklungen darstellen, da diese bereits einen hohen Entwicklungsstand haben. Unter Anbetracht der benötigten Funktionen stellen aber speziell der *VideoJS Player* und *Popcorn.js* eine gute Basis dar, da diese mit ihrem Kernelement als Player und durch die integrierten Plugin-Systeme für die Entwicklung von Multimedia-Elementen ausgelegt sind. Bei *H5P* müsste die Playerfunktion mit der dazugehörigen Erweiterung für Hyperaudio-Dokumente von Grund auf entwickelt werden, um eine entsprechende interaktive Komponente für Hyperaudio-Dokumente bereitzustellen zu können. Letztlich scheint *Popcorn.js* die beste Grundlage für die Entwicklung des Plugins darzustellen, da hier auch die Steuerung der Medieninhalte bereits von Grund auf ausgeprägt implementiert sind, woraus bei der Umsetzung einiger Funktionen großer Nutzen gezogen werden kann. Des Weiteren ist *Popcorn.js* als JavaScript-Bibliothek mühelos als Thirdparty-Framework in Moodle integrierbar.

3.5 Zusammenfassung

Im ersten Schritt wurden Personas als mögliche Nutzer der Anwendung entwickelt. Anhand der Personas konnte die Aufteilung der Zielgruppe in Nutzende und Administrierende abgeleitet werden. Personas und Rollen wurden daraufhin zurate gezogen, um User Stories zu formulieren. Die daraus

¹²ehemaliges GitHub-Projekt, MIT Lizenz: <http://wam.inrialpes.fr/timesheets>

resultierenden rollenspezifischen Anforderungen wurden definiert und im Hinblick auf die Zielsetzung der Arbeit in drei Prioritätsstufen kategorisiert, welche die Basis für das Vorgehen im Implementierungsprozess bilden. Als Vorbereitung für die Implementierung dient ebenfalls die genauere Betrachtung der Möglichkeiten zur Plugin-Entwicklung in Moodle. Als Rahmenbedingung kann festgehalten werden, dass das Hyperaudio-Plugin primär als Aktivitäten-Plugin umzusetzen ist und dass Übersichten und Favoritenfunktion innerhalb von Blöcken realisiert werden können. Bei der Betrachtung des aktuellen Stands der Technik im Allgemeinen sowie im Bezug auf die Entwicklung des Plugins wurde entschieden, dass *Popcorn.js* als Grundlage für das Hyperaudio-Plugin dienen soll, da dieses mit seinen vorhanden Steuerungs- und Annotationsmöglichkeiten ein gutes Grundgerüst darstellt und problemlos in Moodle integriert werden kann.

4 Konzept

Mit den Erkenntnissen des vorherigen Kapitels kann sich nun der Konzeption des Hyperaudio-Plugins zugewendet werden. Dabei werden zu Beginn die Zusammenhänge der Komponenten von Hyperaudio-Dokument und Annotationen analysiert und deren Zusammenhänge festgehalten. Diese Zusammenhänge können durch eine Schnittstellendatei abgebildet werden, deren Format in Abschnitt 4.2 definiert wird. Anschließend wird der Datenbankentwurf vorgenommen. Im Anschluss kann sich der Gestaltung der Benutzeroberfläche des Plugins gewidmet werden.

4.1 Zusammenhänge der Komponenten der Hyperaudio-Anwendung

Basierend auf der Definition eines Hyperaudio-Dokuments aus Abschnitt 2.3.3 und der in Abschnitt 3.2 erarbeiteten Anforderungen werden die Zusammenhänge der medialen Komponenten weiter analysiert. Hierbei soll vor allem geklärt werden, wie die einzelnen Komponenten von Hyperaudio-Dokument und Annotationen zusammenhängen und welche Möglichkeiten dadurch gegeben beziehungsweise nicht gegeben sind.

4.1.1 Komponenten

Im Mittelpunkt eines Hyperaudio-Dokuments steht eine Audio-Datei. Inhaltlich kann es sich hierbei beispielsweise um einen Vorlesungsvortrag handeln. Man könnte sich auch vorstellen, dass ein Hyperaudio-Dokument aus mehreren aneinander gereihten Audio-Dateien besteht. Dies würde an der grundsätzlichen Problemstellung jedoch nichts ändern und kann im Nachhinein jederzeit als Erweiterung umgesetzt werden. Aus diesem Grund wird in dieser Arbeit nur ein Plugin für ein Hyperaudio-Dokument bestehend aus einer Audio-Datei entwickelt.

Neben dieser zentralen Audio-Datei besteht das Hyperaudio-Dokument aus mehreren Zusatzinhalten, wobei es sich um Bilder, Graphen, Tabellen usw. handeln kann. Entscheidend ist aber, dass diese Zusatzinhalte immer nur eine rein grafische Darstellung verkörpern. Videos mit Ton sind somit beispielsweise nicht als Zusatzinhalt verwendbar, reine Animationen ohne Ton sind aber durchaus möglich.

Als besondere, nämlich externe Komponente, sind die Kommentare zu nennen. Diese gehören nicht zum eigentlichen Hyperaudio-Dokument, sollen aber mit diesem verknüpft werden. Es wird drei verschiedene Arten von Kommentaren geben, nämlich öffentliche Kommentare, persönliche Notizen und Markierungen. Innerhalb der öffentlichen Kommentare muss noch zwischen den Original-Kommentaren und den Antworten auf diese unterschieden werden.

4.1.2 Zusammenhänge

Diese Zusammenhänge der soeben genannten Komponenten sind im UML-Diagramm in Abbildung 4.1 ersichtlich. Zunächst werden die Zusammenhänge zwischen der Audio-Datei und den Zusatzinhalten betrachtet. Nach dem Master-Slave-Prinzip werden Zusatzinhalte der Audio-Datei untergeordnet. Zu jedem beliebigen Zeitpunkt innerhalb der Abspieldauer der Audio-Datei kann maximal ein Zusatzinhalt gleichzeitig annotiert werden. Es sind also auch Phasen möglich, zu denen keinerlei

Zusatzinhalt dargestellt wird. Das Zeitfenster für die Annotation soll mittels einer Start- und Endzeit pro Zusatzinhalt definiert werden, wobei nur die Minuten und Sekunden anzugeben sind. Bei dem Zeitfenster sollte natürlich bedacht werden, dass dieses nicht zu kurz sein sollte. Zwar soll, sobald ein Zusatzinhalt im Player des Hyperaudio-Dokuments angezeigt wird, ein entsprechender *Audio Cue* abgespielt werden, dennoch können bereits einige Sekunden vergehen bis der Studierende seinen Blick dem Player zuwendet.

Auch die Kommentare stehen als externe Komponente in einer gewissen Art und Weise im Zusammenhang mit der Audio-Datei. Dies ergibt sich daraus, dass Kommentare zu einem bestimmten Zeitpunkt innerhalb der Audio-Datei erfasst werden. Während Antworten auf Original-Kommentare erfasst werden können, sind Antworten auf Antworten nicht möglich.

Zwischen Kommentaren und Zusatzinhalten gibt es jedoch keinen direkten Zusammenhang. Solche Zusammenhänge ergeben sich alleine aus den Zeitpunkten der Annotationen. Zusatzinhalte können wiederum in keinem Zusammenhang mit einem anderen Zusatzinhalt stehen.

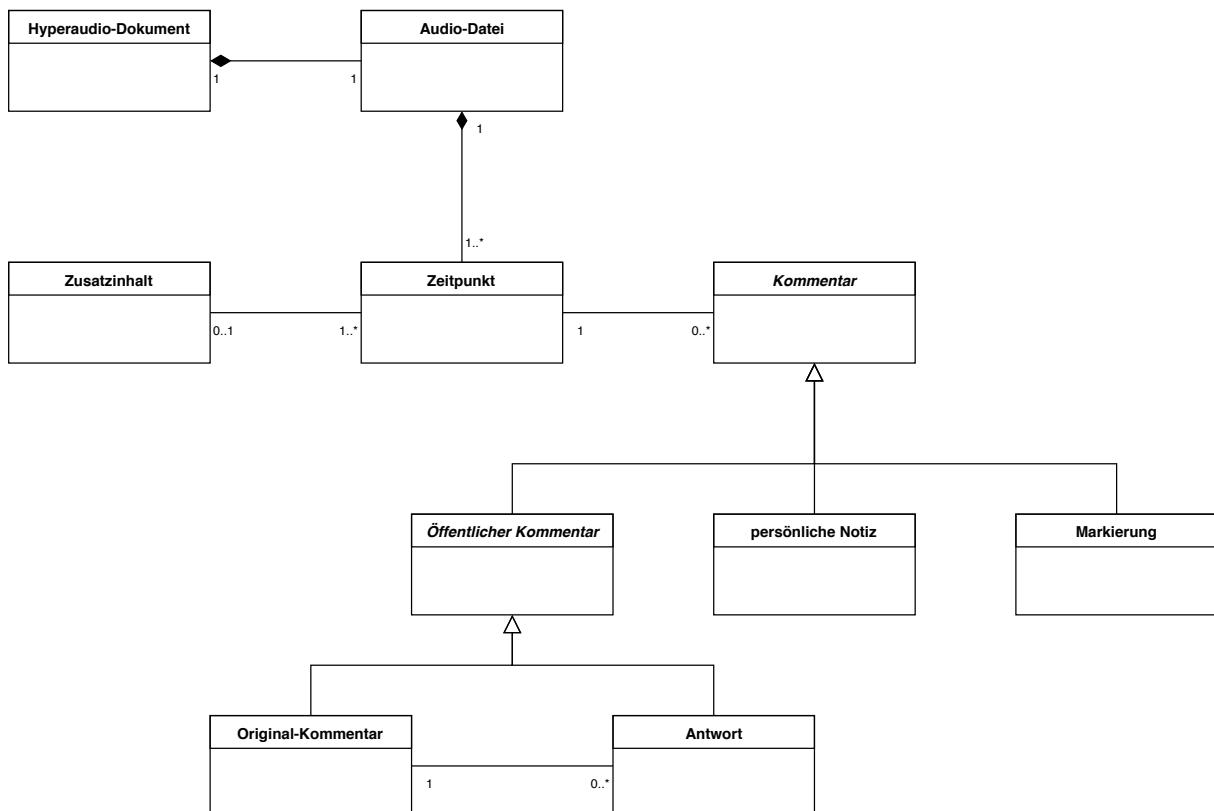


Abbildung 4.1: Zusammenhänge der Komponenten

4.2 Definition des Schnittstellenformats für Hyperaudio-Dokumente

Um ein Hyperaudio-Dokument zu erstellen, können mehrere Dateien hochgeladen werden. Verpflichtend ist das Bereitstellen einer Audio-Datei. Sollen Zusatzinhalte angezeigt werden, muss neben den Zusatzinhalten selbst auch eine Konfigurationsdatei hochgeladen werden. Darin wird festgehalten, zu welchem Zeitpunkt welcher Zusatzinhalt annotiert werden soll. Darüber hinaus ist es mittels der Konfigurationsdatei möglich Metadaten (Name und Beschreibung des Zusatzinhalts sowie die betroffene Kurseinheit und die zugehörigen Seiten) für die einzelnen Zusatzinhalte anzufügen. Aufgrund des Einsatzes von PHP und JavaScript innerhalb der Moodle Plugin-Entwicklung bietet sich der Einsatz von JSON (JavaScript Object Notation) an. JSON wird direkt durch JavaScript unterstützt

4 Konzept

und bietet im Vergleich zu XML (Extensible Markup Language) erhebliche Geschwindigkeitsvorteile (Nurseitov et al., 2009). Mittels der JSON-Datei sollen folgende Informationen der Zusatzinhalte übertragen werden:

- Dateiname
- Name des Zusatzinhaltes
- Beschreibung des Zusatzinhaltes
- Kurseinheit
- Betroffene Seiten innerhalb der Kurseinheit
- Startzeitpunkt der Annotation
- Endzeitpunkt der Annotation

Entscheidend ist dabei vor allem der Dateiname. Anhand des Dateinamens kann anschließend die Zuordnung der weiteren Informationen zu der entsprechenden Audio-Datei in der Datenbank vorgenommen werden. Eine beispielhaft befüllte Konfigurationsdatei ist in Auflistung 4.1 dargestellt.

```
1 {
2   "additional_contents": [
3     "additional_content": [
4       {"filename": "Abbildung_1_4.png",
5        "name": "Abbildung 1.4",
6        "course_unit": 1,
7        "page": "31",
8        "description": "Ein kooperativer Editor zur Visualisierung von gemeinsam zu
lernenden Vokabeln.",
9        "begin": "5",
10       "end": "10"},,
11       {"filename": "Abbildung_1_5.png",
12       "name": "Abbildung 1.5",
13       "course_unit": "1",
14       "page": "32",
15       "description": "Verschiedene Komponenten des Papierprototyps.",
16       "begin": "15",
17       "end": "25"}]
18   ]
19 }
20 }
```

Auflistung 4.1: Beispielhafte Konfigurationsdatei

JSON Beispieldaten Zeitpunkte korrigieren

JSON-Datei um Autor ergänzen

4.3 Datenbankentwurf

Um die dem Plugin zugrundeliegende Datenbank zu gestalten, wird auf die Erkenntnisse aus Abschnitt 4.1.2 zurückgegriffen. Das Ergebnis ist dem ER-Diagramm in Abbildung 4.2 zu entnehmen.

Jede der Datenbanktabellen verfügt über einen Primärschlüssel (*id*). Darüber hinaus wird zu jedem Eintrag gespeichert, wann dieser erstellt (*timecreated*) und zuletzt bearbeitet (*timemodified*) wurde. Für das Abspeichern von Dateien stellt Moodle die Tabelle *files* bereit (Moodle, 2018c). Dort kann die Datei abgelegt und an anderer Stelle darauf referenziert werden.

Im Mittelpunkt des Hyperaudio-Plugins steht die Tabelle *hyperaudio*. Diese repräsentiert das Hyperaudio-Dokument und die Audio-Datei aus Abbildung 4.1. Zunächst wird die ID des Moodle-Kurses (*course*) abgelegt, dem das Hyperaudio-Dokument zugeordnet ist. Während die Audio-Datei selbst in der bereits erwähnten Tabelle *files* zu finden ist, wird in der Tabelle *hyperaudio* deren Dateiname (*audiofile*) festgehalten. Zum Hyperaudio-Dokument können außerdem Name und Ersteller in den Spalten *name* und *author* hinterlegt werden. Eine optionale Beschreibung kann entsprechend dem de-facto-Standard der Moodle-Plugin-Entwicklung über *introformat* und *intro* hinzugefügt werden (Moodle, 2016).

Bei der Tabelle *hyperaudio_config* handelt es sich um eine Tabelle, in welcher die in Abschnitt 4.2 beschriebene Konfigurationsdatei gespeichert wird (*file* enthält den Dateinamen als Referenz auf die *files*-Tabelle). Daneben wird nur noch der Fremdschlüssel *hyperaudio_id* auf die Tabelle *hyperaudio* als Zuordnung zum Hyperaudio-Dokument benötigt.

Zur Ablage der annotierten Zusatzinhalte dient die Tabelle *additional_content*. Auch hier steht der Name der Datei in der Spalte *file* als Referenz auf die *files*-Tabelle. Ebenso dient der Fremdschlüssel *hyperaudio_id* zur Verknüpfung des Zusatzinhalts mit der Audio-Datei. Des Weiteren werden folgende Metainformationen zum Zusatzinhalt abgespeichert:

- Name des Zusatzinhalts (*name*)
- optional: Beschreibung (*description*)
- optional: Kurseinheit (*course_unit*)
- optional: Seitenangabe (*page*)
- Startzeitpunkt der Annotation innerhalb des Hyperaudio-Dokuments (*begin*)
- Endzeitpunkt der Annotation innerhalb des Hyperaudio-Dokuments (*end*)

Die Tabelle *hyperaudio_comments* dient der Speicherung der vier in Abbildung 4.1 modellierten Kommentararten. Der Zusammenhang zum Hyperaudio-Dokument wird analog per Fremdschlüssel *hyperaudio_id* hergestellt. Neben dem textuellen Kommentar (*commenttext*), werden auch der Verfasser in Form der *userid*, die Art des Kommentars (*comment_type*¹³) und der Annotationszeitpunkt (*timeannotated*) gespeichert. Für den Fall, dass es sich um einen Antwortkommentar handelt, wird in der Spalte *comment_id* die Referenz auf den Original-Kommentar festgehalten.

4.4 Gestaltung der Benutzeroberfläche

Damit den Lehrenden und Studierenden die im vorherigen Abschnitt beschriebenen Nutzungsszenarien möglichst leicht fallen, wenden wir uns nun der Gestaltung der Benutzeroberfläche zu. „Das Design der Benutzeroberfläche stellt einen zentralen Aspekt für die Gebrauchstauglichkeit eines Softwareprodukts dar“ (Oppermann, 2002, S. 1). Einen dementsprechend hohen Stellenwert wollen wir der Benutzeroberfläche unseres Moodle-Plugins zuschreiben. Bei der Gestaltung der Benutzeroberfläche gehen wir wie bereits bei der Analyse in Kapitel 3 vor und teilen die Benutzeroberfläche in Teilbereiche auf. Im ersten Schritt betrachten wir zunächst die Seite eines Hyperaudio-Dokuments innerhalb eines Kurses. Danach widmen wir uns der Administrationsseite eines Hyperaudio-Dokuments innerhalb eines Kurses. Im letzten Schritt wenden wir uns den verschiedenen Integrationsmöglichkeiten innerhalb der allgemeinen Moodle-Oberfläche zu.

Generell erfolgen alle Entscheidungen bezüglich der Oberfläche auf Basis von Skizzen. Diese wurden mittels des Programms *Balsamiq Mockups 3.5.15*¹⁴ erstellt. Anhand der Skizzen können Vor- und Nachteile der verschiedenen Designansätze schnell erkannt und auf Grund dessen sachliche Entscheidungen getroffen werden.

¹³COMMENT, NOTE oder MARK

¹⁴<https://balsamiq.com/>

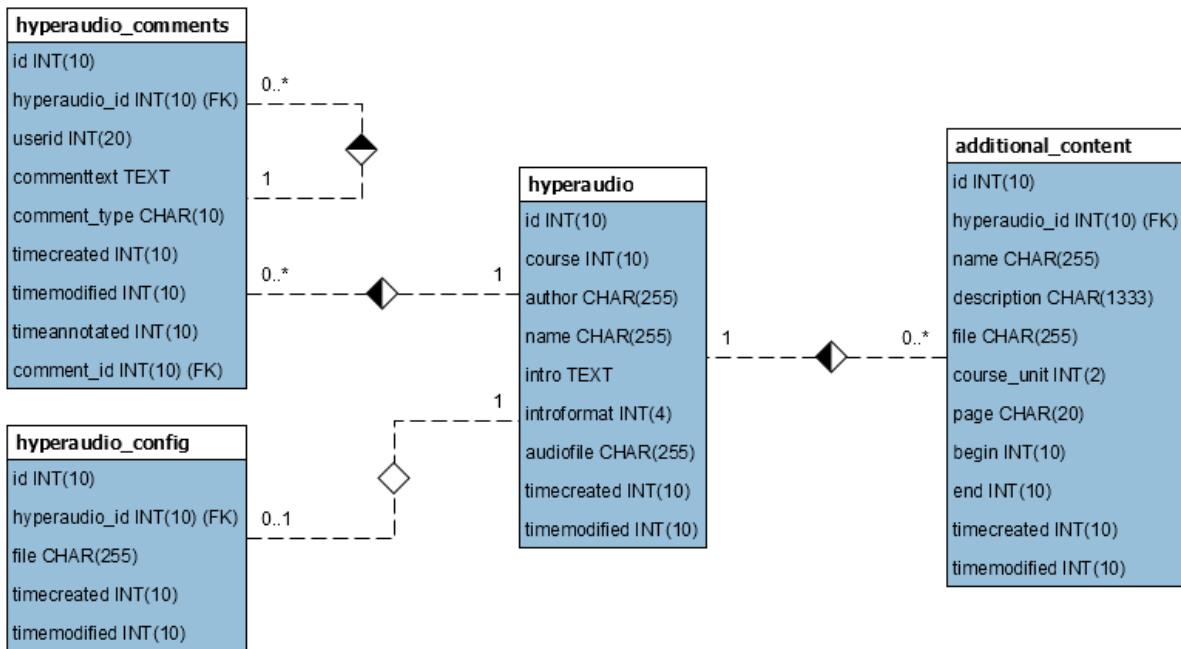


Abbildung 4.2: ER-Diagramm der Datenbank des Moodle-Plugins

Die Seite eines Hyperaudio-Dokuments lässt sich grob, wie bereits in Abbildung ?? dargestellt, in die Bereiche Player, Galerie und Kommentarsektion aufteilen. Wir werden nun zunächst für jeden dieser Bereiche verschiedene Designs diskutieren und uns dann für eines entscheiden. Danach erfolgt die Entscheidung über die Anordnung dieser Bereiche auf der Seite eines Hyperaudio-Dokuments.

Player

Beim Player für Hyperaudio-Dokumente müssen, neben den üblichen Mediensteuerungselementen, gleich mehrere zusätzliche Elemente visualisiert werden. Zum einen müssen zu den entsprechenden Zeitpunkten die annotierten Zusatzinhalte dargestellt werden. Auf der anderen Seiten sollen auch die annotierten öffentlichen Kommentare, persönlichen Notizen und Markierungen veranschaulicht werden. Dem Wunsch, direkt über den Player öffentliche Kommentare, persönlichen Notizen und Markierungen erstellen und in letzterem Fall sogar löschen zu können, muss auch Sorge getragen werden.

Der Player für Hyperaudio-Dokumente wird, wie in Abbildung 4.3a zusehen ist, als Videoplayer umgesetzt. Somit werden die Zusatzinhalte an Stelle eines Videos dargestellt. Die persönlichen Notizen und Markierungen werden innerhalb der Abspielleiste mittels unterschiedlich gefärbter Kreisen illustriert. In diesem Fall sollen die roten Kreise Markierungen und der blaue Kreis eine persönliche Notiz widerstrengen. Unterhalb der Mediensteuerung ist ein Bereich zu finden, in dem die Kommentare grafisch sichtbar gemacht werden sollen. Hierfür wird jedes Hyperaudio-Dokument in die gleiche fixe Anzahl an Zeitfenstern aufgeteilt. Diese Zeitfenster werden durch senkrecht orientierte Balken dargestellt, deren Höhe für die Anzahl der zu diesem Zeitfenster erfassten Kommentare stehen soll. Unter dem Bereich für die Kommentare befindet sich eine Eingabemaske, mit welcher öffentliche Kommentare und persönliche Notizen erfasst werden können. Das Erstellen und Löschen von Markierungen soll mittels Rechtsklick auf die entsprechende Stelle innerhalb der Abspielleiste in einem dazugehörigen Kontextmenü umgesetzt werden. Dies ist in Abbildung 4.3a mittels der beiden Mauszeiger, den Pfeilen und den entsprechenden Buttons symbolisiert.

Unterscheidung nicht nur durch Farbe (Barrierefreiheit)

In einer zweiten Variante des Players wird die Visualisierung der persönlichen Notizen von der Abspielleiste in den Bereich der Kommentare verschoben. Wie in Abbildung 4.3b ersichtlich, wird der Balken für den Zeitraum, in dem die persönlichen Notiz liegt, zu einem gewissen Teil blau eingefärbt. Dadurch wird nebenbei das Handling der Punkte in der Abspielleiste vereinheitlicht, da es hier nur noch die Markierungen mit Interaktionsmöglichkeit gibt.

Galerie

Die Galerie soll dazu dienen, einen Überblick über die vorhanden Zusatzinhalte zu bieten. Die Zusatzinhalte stellen alle einen grafischen Inhalt dar. Dementsprechend kann jeder Zusatzinhalt durch ein kleines Vorschaubild repräsentiert werden. Eine weitere Grundfunktionalität einer Galerie ist die vergrößerte Anzeige der in der Vorschau dargestellten Inhalte, die auch in unserer Galerie zur Verfügung stehen soll. Beim Erstellen des Designs muss zusätzlich auch die Anforderung der Rückkopplung zum Player bedacht werden (siehe Abschnitt ??).

Die einfachste Umsetzung der Galerie ist eine Darstellung der Zusatzinhalte in einem einfachen Grid mit Scrollbalken, wie es in Abbildung 4.4a zu sehen ist. Zusätzlich wird das Grid um zwei Buttons für eine vergrößerte Ansicht des Zusatzinhalts sowie für die Rückkopplung zum Player ergänzt. Um eine dieser beiden Aktionen auszuführen, müsste also der gewünschte Zusatzinhalt markiert und der entsprechende Button betätigt werden.

Diese Variante hat den Vorteil, dass besonders viele Zusatzinhalte gleichzeitig angezeigt werden können. Auf der anderen Seite erhalten wir aber keinerlei Informationen zu den Zusatzinhalten. In der zweiten Variante, bei dem das Grid um einen Bereich für Details ergänzt wurde, kann man zumindest die Details des ausgewählten Zusatzinhaltes einsehen. Diese in Abbildung 4.4b erkennbaren Details sind natürlich von den vorhandenen Metadaten abhängig. Nachteil ist in diesem Fall aber, dass, durch den Bereich für die Details, bei gleicher Größe der Galerie weniger Zusatzinhalte zur selben Zeit dargestellt werden können. Das führt dazu, dass die Verwendung des Scrollbalkens häufiger notwendig wird.

Bei einer Darstellung der Zusatzinhalte als Kacheln, wie in Abbildung 4.4c zu sehen, können gleichzeitig für alle vorhandenen Zusatzinhalte die Details angezeigt werden. Durch diese Art der Darstellung passen jedoch noch weniger Zusatzinhalte auf die gleiche Fläche.

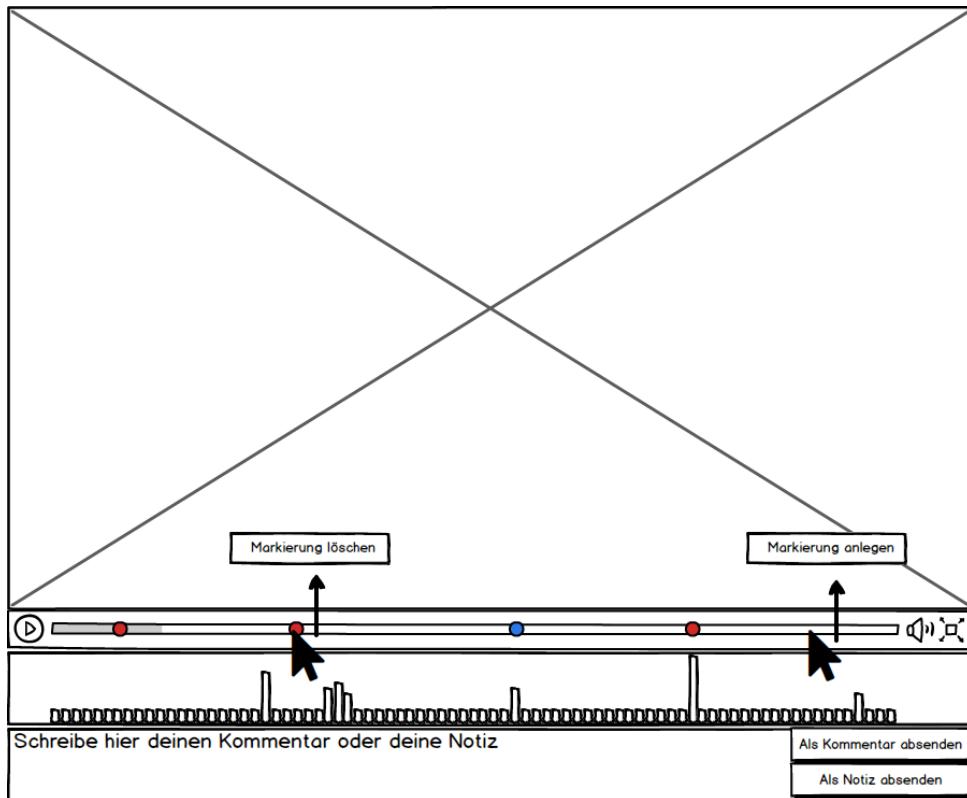
Eine besonders schicke Art der Darstellung wäre die des Cover Flows, bekannt aus verschiedenen Musikplayern. In Abbildung 4.4d ist zu erkennen, dass auch hier ausschließlich Details des aktuell ausgewählten Zusatzinhaltes sichtbar sind. Des Weiteren hat diese Darstellungsweise den großen Nachteil, dass auch nicht auf einen Blick alle verfügbaren Zusatzinhalte ersichtlich sind. Dies erschwert das Durchsuchen der Zusatzinhalte ungemein. Somit ist diese Art der Darstellung zwar schön anzusehen, aber nicht sonderlich gebrauchstauglich im Zusammenhang dieser Arbeit.

Letztlich stellt sich die optimierte Variante der Kachel Darstellung aus Abbildung 4.4e als beste Lösung heraus. Die Optimierung besteht daraus, dass die beiden Buttons obsolet gemacht werden. Dies kann zum einen erreicht werden, indem die vergrößerte Darstellung durch einen Klick auf die Abbildung des Zusatzinhaltes ausgelöst wird. Zum anderen bietet der Bereich der Details noch ausreichend Platz, um hier die Funktion zur Rückkopplung an den Player einzufügen. Durch diese Verbesserungen wird nicht nur mehr Platz geschaffen, sondern auch die Benutzerfreundlichkeit erhöht, indem der Vorgang zum Anzeigen der vergrößerten Ansicht beziehungsweise des Springens an den entsprechenden Zeitpunkt jeweils um einen Klick reduziert wurde.

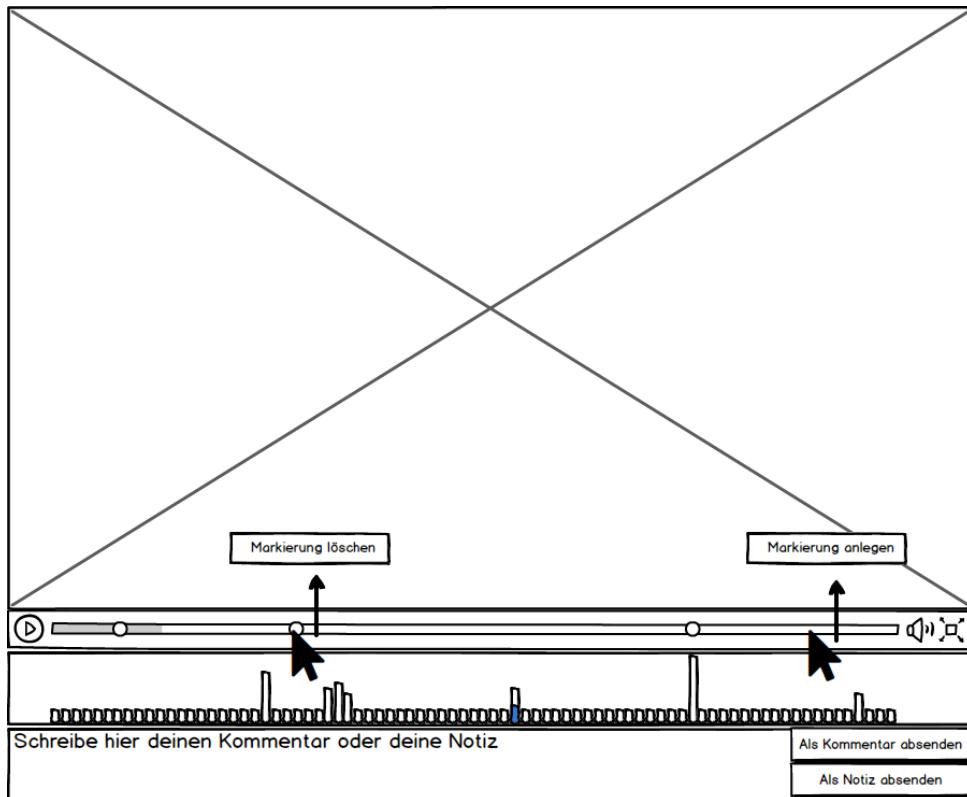
Kommentarsektion

Die Kommentarsektion ist für die Anzeige der öffentlichen Kommentare sowie der persönlichen Notizen zuständig. Zusätzlich muss eine Suchmaske auf Basis der Anforderungen aus Abschnitt ?? in

4 Konzept

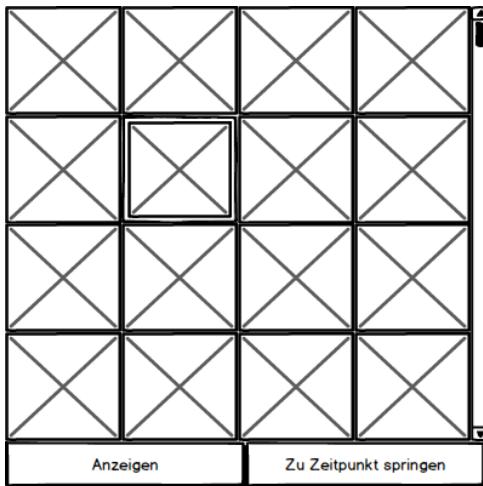


a) Erste Version

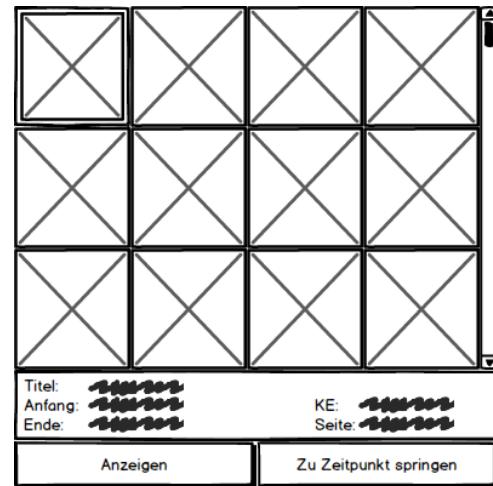


b) Finale Version

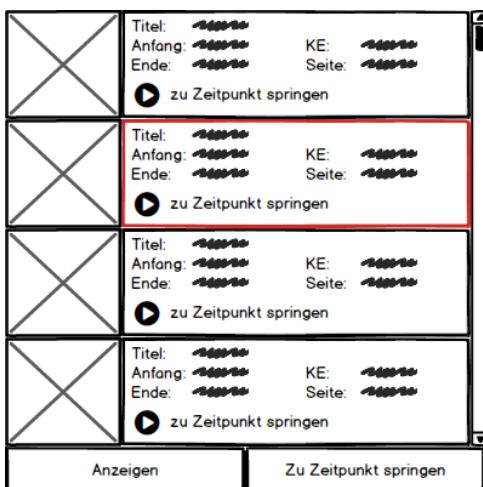
Abbildung 4.3: Benutzeroberfläche - Player



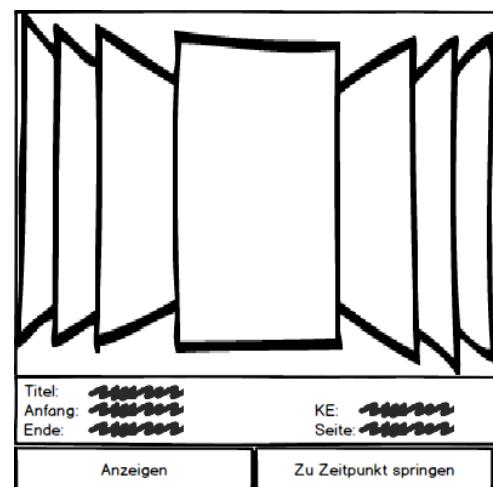
a) Galerie als einfaches Grid



b) Galerie als Grid mit Bereich für Details



c) Galerie mit Darstellung in Kachelform



d) Galerie als Cover Flow



e) Finale Version der Galerie

Abbildung 4.4: Benutzeroberfläche - Galerie

die Oberfläche integriert werden.

Abbildung 4.5a zeigt eine erste Version der Kommentarsektion. Neben der Suchmaske im Kopfbereich befinden sich zwei Checkboxen. Diese ermöglichen es dem Betrachter nach öffentlichen Kommen-

4 Konzept

taren und persönlichen Notizen zu filtern. Im benachbarten Dropdown-Menü kann die Grundlage der Sortierung bestimmt werden. Die Sortierung kann nach Erstellungsdatum beziehungsweise nach Zeitpunkt der Annotation innerhalb des Hyperaudio-Dokuments erfolgen. Unterhalb dieser Funktionen befindet sich die Anzeige der Kommentare und Notizen. Sowohl bei Kommentaren als auch bei Notizen wird neben dem Erstellungsdatum auch der Annotationszeitpunkt festgehalten. Dieser wird als Link umgesetzt, sodass bei einem Klick die Rückkopplung an den Player erfolgen kann. Bei Kommentaren gibt es nach Betätigung der *Antworten*-Schaltfläche noch eine zusätzliche Eingabemaske zum Verfassen von Antworten. Persönliche Notizen werden durch ein Schloss-Symbol hinter dem Erstellungsdatum visualisiert. Zusätzlich befinden sich noch jeweils zwei Buttons zum Bearbeiten und Löschen auf der rechten Seite einer Notiz.

Im nochmals verbesserten Design der Kommentarsektion, welches in Abbildung 4.5b abgebildet ist, werden die Antworten auf Kommentare eingerückt dargestellt. Diese Darstellung führt zu einer besseren Übersichtlichkeit und ist auch aus anderen modernen Anwendungen bekannt.

Zusammenführen der Elemente

Im ersten Schritt führen wir die jeweils favorisierten Elemente in ein Layout zusammen. Dabei orientieren wir uns zunächst an unserer grobe Skizze aus Kapitel 3. Wie nun in Abbildung 4.6a zu erkennen ist, ist die Kommentarsektion so in die Breite gezogen, dass das Lesen der Inhalte unangenehm werden kann. Aus diesem Grund wird in der finalen Version (siehe Abbildung 4.6b) die Breite der Kommentarsektion auf die Breite des Players beschränkt. Dies hat zeitgleich zur Folge, dass der nun vorhandene freie Platz für die Galerie verwendet werden kann. Spätestens hiermit wird der Nachteil der gewählten Darstellungsweise der Galerie egalisiert, da nun ausreichend viele Zusatzinhalte ohne die Verwendung des Scrollbalkens eingesehen werden können.

4.5 Zusammenfassung

...

Kommentare und Notizen

(Suche) Kommentare Notizen Sortieren nach: Zeitpunkt ▾

Erstellt am 22.07.2018 - 16:13 [Zeitpunkt: 00:15](#)

Erstellt am 25.07.2018 - 22:11

[Antworten](#)

Erstellt am 01.06.2018 - 09:34 [Zeitpunkt: 01:27](#)

a) Erste Version

Kommentare und Notizen

(Suche) Kommentare Notizen Sortieren nach: Zeitpunkt ▾

Erstellt am 22.07.2018 - 16:13 [Zeitpunkt: 00:15](#)

Erstellt am 25.07.2018 - 22:11

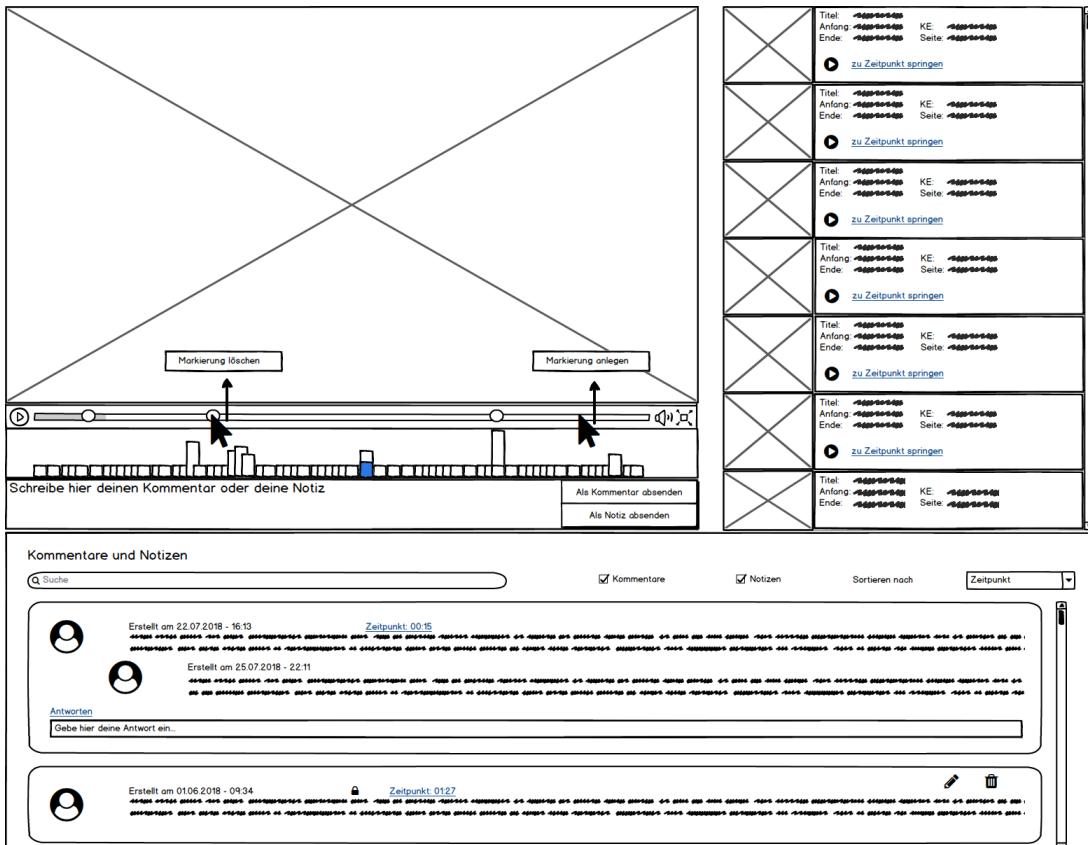
[Antworten](#)

Erstellt am 01.06.2018 - 09:34 [Zeitpunkt: 01:27](#)

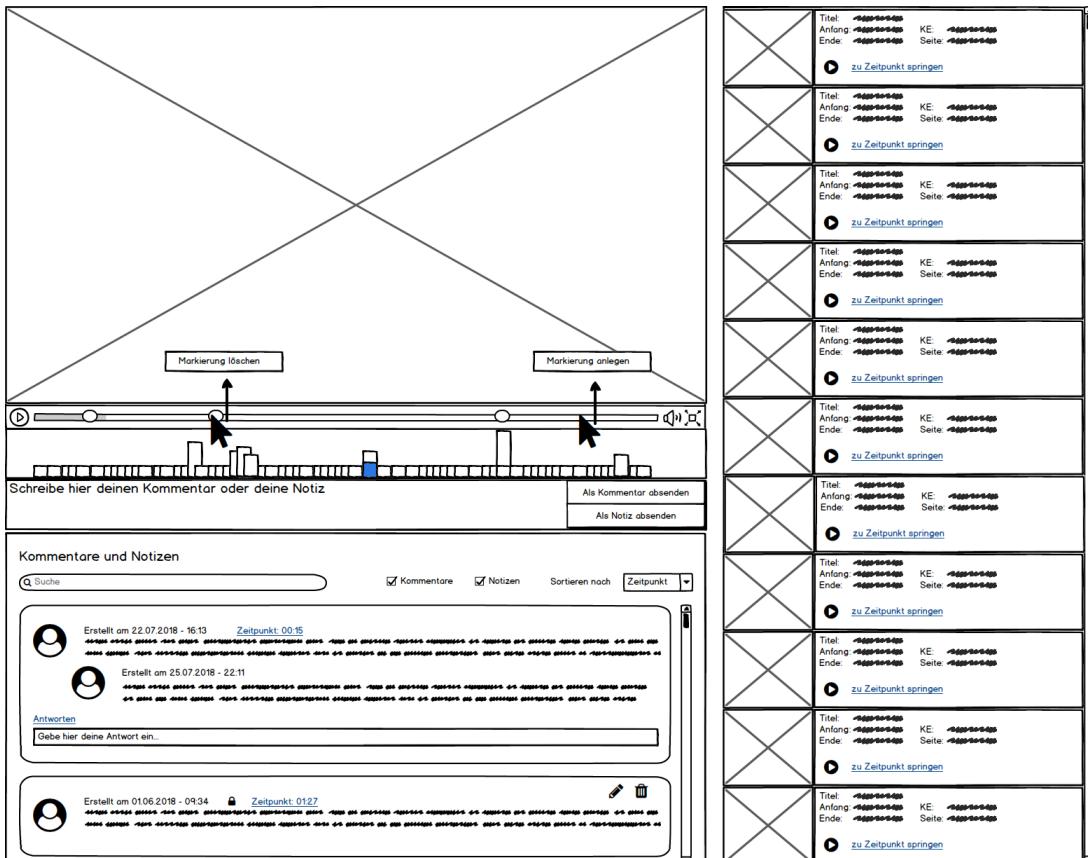
b) Finale Version

Abbildung 4.5: Benutzeroberfläche - Kommentarsektion

4 Konzept



a) Erste Version



b) Finale Version

Abbildung 4.6: Benutzeroberfläche - Layout der Seite für Hyperaudio-Dokumente

5 Implementierung

Einleitung

5.1 Architektur des Moodle-Plugins

Bei der Implementierung des Hyperaudio-Plugins ist die durch Moodle vorgegebene Architektur von Plugins zu beachten (Moodle, 2016). Diese besteht stets aus vorgegebenen Dateien und Ordnern, wobei die jeweilige Anzahl von der Art des zu entwickelnden Plugins abhängig ist. Darüber hinaus bestimmt die Art des Plugins auch den zu wählenden Speicherort.

Bei Activity Plugins, wie dem Plugin für Hyperaudio-Dokumente, ist als Speicherort der Ordner **/mod** vorgeben. In diesem Ordner muss ein Unterordner mit dem Namen des Plugins angelegt werden, in diesem Fall **hyperaudio**, in welchem alle Plugin-Dateien abgelegt werden. Eine Übersicht über die Ordnerstruktur des Hyperaudio-Plugins findet sich in Abbildung 5.1.

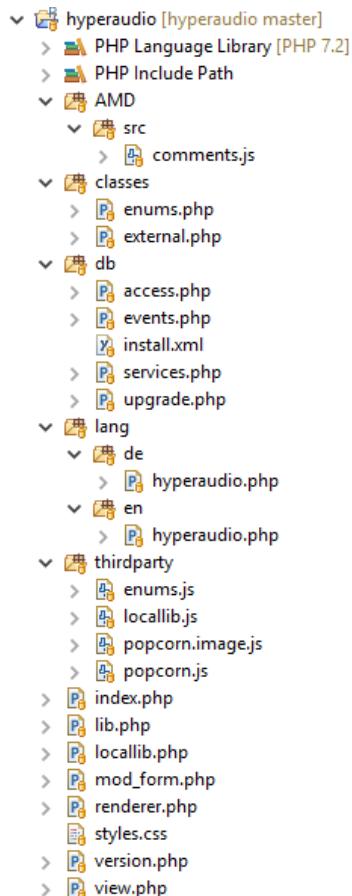


Abbildung 5.1: Ordnerstruktur des Hyperaudio-Plugins

Ordnerstruktur aktualisieren

5 Implementierung

Im Ordner **/hyperaudio/backup** werden die Dateien abgelegt, welche Anwendung finden, wenn ein Backup oder eine Wiederherstellung eines Kurses vorgenommen wird.

Der Ordner **/hyperaudio/db** beherbergt die Dateien **access.php**, **events.php**, **install.xml** und **upgrade.php**. Die Datei **access.php** dient zur Steuerung der Berechtigungen innerhalb des Moodle-Plugins, wobei den verschiedenen Moodle-Rollen verschiedene Rechte für die einzelnen Funktionen zugewiesen werden können. In der **events.php** können Beobachter eingerichtet werden, welche auf bestimmte Ereignisse warten. Bei der Installation des Plugins wird die **install.xml** zur Erstellung der Datenbanktabellen für das Plugin verwendet. Es ist mindestens eine Tabelle mit dem Namen des Plugins anzulegen. Sollten die Datenbanktabellen nach Veröffentlichung des Plugins um Spalten erweitert werden, so kommt die Datei **upgrade.php** zum Einsatz. Hierin werden die notwendigen Schritte für einen Versionsabgleich definiert.

Im Ordner **/hyperaudio/lang** wird die Sprachlokalisierung vorgenommen. Für jede Sprache wird innerhalb des **lang**-Ordners ein eigener Unterordner angelegt. Darin befindet sich jeweils eine PHP-Datei, in welcher die Übersetzungen definiert werden. Der Name dieser Datei entspricht wiederum dem Namen des Plugins.

Das Icon, welches für das Plugin verwendet werden soll, muss im Ordner **/hyperaudio/pix** mit dem Dateinamen **icon.gif** abgelegt werden und sollte eine Auflösung von 16x16 Pixel besitzen.

Im Ordner **/hyperaudio** liegen darüber hinaus die Dateien **lib.php**, **mod_form.php**, **index.php**, **view.php** und **version.php**. Die **lib.php** dient dazu, Standardfunktionen von Moodle zu überschreiben, wobei `add_instance`, `update_instance` und `delete_instance` als essenzielle Funktionen zu nennen sind. Mit diesen Funktionen wird das Anlegen, Aktualisieren und Löschen von Instanzen des Plugins ermöglicht. Zum Anlegen und Aktualisieren wird in der **mod_form.php** die dazugehörige Maske festgelegt. Die **index.php** dient der Auflistung aller Instanzen eines Plugins innerhalb eines Kurses. Je nach Umsetzung kann der Inhalt dieser Auflistung unterschiedlich viele Informationen zu den Instanzen bereitstellen. Auch ist es beispielsweise anhand von Berechtigungen aus der **access.php** möglich gewisse Informationen nur bestimmten Usern anzuzeigen. Die erste Datei, die beim Öffnen der Aktivität geladen wird, ist die **view.php**, welche dementsprechend vornehmlich der Anzeige der Inhalte dient. In der **version.php** wird die Version des Plugins gepflegt. Erhöht sich die Versionsnummer in der **version.php**, wird der automatische Upgradeprozess von Moodle für das Plugin ausgelöst.

Neben diesen vorgegebenen Dateien kommen üblicherweise noch weitere Dateien bei der Entwicklung eines Moodle-Plugins zum Einsatz (Wild, 2017). Dazu gehört beispielsweise die **locallib.php**, in welcher üblicherweise alle plugineigenen PHP-Funktionen deklariert werden. Auch ist es Usus, die eigentliche Darstellung der Plugin-Inhalte innerhalb eines Kurses von der **view.php** in eine **renderer.php** zu verlagern. Dort können verschiedene Renderer-Klassen, welche durch Moodle bereitgestellt werden, für die eigenen Bedürfnisse überschrieben werden. Anpassungen optischer Natur können durch CSS (Cascading Style Sheets) in der **styles.css** vorgenommen werden. Eigene JavaScript-Module, welche beispielweise beim Laden der **view.php** automatisch aufgerufen werden, sind im Verzeichnis **/hyperaudio/AMD** (Asynchronous Module Definition) abzulegen.

5.2 Iterative Entwicklung

Die Entwicklung des Plugins wird in iterativer Form durchgeführt. In jeder Iteration soll das Plugin nur um einige wenige Funktionalitäten erweitert werden. Jede Iteration soll mit einem lauffähigen Plugin abgeschlossen werden. So kann direkt das Ergebnis betrachtet werden und gegebenenfalls in der nächsten Iteration nochmals angepasst werden (Stephan Augsten, 2018). Die Reihenfolge, in welcher die Funktionalitäten umgesetzt werden, leitet sich aus der Priorisierung der Anforderungen aus Abschnitt 3.2 ab.

5.2.1 Speichern und Abspielen einer Audio-Datei

In der ersten Iteration wird zunächst die grundlegende Struktur des Plugins erstellt (vgl. Abschnitt 5.1). Ziel der ersten Iteration soll es sein, eine Audio-Datei speichern und wiedergeben zu können.

Dazu wird in der Maske zum Anlegen und Aktualisieren von Instanzen des Hyperaudio-Plugins (**mod_form.php**) neben dem obligatorischen Namens-Feld noch ein Element zum Hinzufügen einer Audio-Datei angelegt. Auflistung 5.1 zeigt einen Ausschnitt des Codes, der in der Funktion `definition` der Klasse `mod_hydraudio_mod_form`, die von der Klasse `moodleform_mod` erbt, ergänzt werden muss.

```

1 $mform = $this->_form;
2 $mform->addElement('text', 'name', get_string('hydraudio_mod_form_name',
3   'hydraudio'));
4 $mform->setType('name', PARAM_TEXT);
5 $mform->addRule('name', get_string('error_wrong_hydraudio_name_input',
6   'hydraudio'), 'required');
7 $mform->addElement('filemanager', 'audiofile', get_string('hydraudiodata',
8   'hydraudio'), null,
9   array(
10    'subdirs' => 0,
11    'maxbytes' => 0,
12    'areamaxbytes' => 10485760,
13    'maxfiles' => 1,
14    'accepted_types' => array('audio')
15  )
16 );
17 $mform->addRule('audiofile', get_string('required'), 'required');
```

Auflistung 5.1: Ausschnitt der **mod_form.php** in der 1. Iteration

Der `_form` der `moodleform_mod` können durch `addElement` neue Form-Elemente hinzugefügt werden. Mithilfe der Funktionen `setType` und `addRule` können den Elementen Datentypen und Regeln zugewiesen werden, die bei Auswertung der Form automatisch validiert werden. Zum Hochladen von Dateien kann der `filemanager` eingesetzt werden. Mithilfe eines Arrays können dabei Einschränkungen für Anzahl und Eigenschaften der hochzuladenden Dateien festgelegt werden. In diesem Fall darf maximal eine Datei hinzugefügt werden, die vom Typ `audio` sein muss. Die Funktion `get_string` dient im Allgemeinen der Darstellung der lokalisierten Bezeichnungen.

Um die Daten aus der Form in der Datenbank speichern und später wieder löschen zu können, muss auch die **lib.php** bearbeitet werden. Dazu dienen die bereits erwähnten Funktionen `add_instance`, `update_instance` und `delete_instance`. Beispielhaft wird in Auflistung 5.2 die Funktion `add_instance` zum Hinzufügen eines neuen Hyperaudio-Dokuments betrachtet.

```

1 function hydraudio_add_instance($data) {
2   global $DB;
3
4   $cmid = $data->coursemodule;
5   $context = context_module::instance($cmid);
6
7   $draftitemid_audiofile = $data->audiofile;
8   unset($data->audiofile);
9
10  $now = time();
11  $data->timecreated = $now;
12  $data->timemodified = $now;
13
14  $data->id = $DB->insert_record('hydraudio', $data);
15
16  hydraudio_update_audiofile($data->id, $context, $draftitemid_audiofile);
```

5 Implementierung

```
17     return $data->id;
18 }
19 }
```

Auflistung 5.2: Ausschnitt der **lib.php** in der 1. Iteration

Der Parameter `$data` enthält bereits die in der Form eingegebenen Daten. Das Attribut `audiofile` enthält nicht die Audio-Datei selbst, sondern die ID der *draft file area* und soll im ersten Schritt nicht in der Tabelle `hyperaudio` abgespeichert werden (vgl. Zeilen 7-8). Vor dem Speichern wird noch der aktuelle Zeitstempel hinterlegt (vgl. Zeilen 10-12). Mithilfe der Funktion `$DB->insert_record` kann das `$data`-Objekt mit seinen Attributen in der Tabelle `hyperaudio` abgelegt werden. Im Nachhinein sorgt die in der **locallib.php** definierte Funktion `hyperaudio_update_audiofile` dafür, dass die Audio-Datei in der `files`-Tabelle abgespeichert und in der `hyperaudio`-Tabelle korrekt referenziert wird (vgl. Auflistung 5.3).

```
1 function hyperaudio_update_audiofile($hyperaudiooid, $context, $draftitemid) {
2     global $DB;
3
4     file_save_draft_area_files($draftitemid, $context->id, 'mod_hyperaudio',
5         'audiofile', $hyperaudiooid);
6     $fs = get_file_storage();
7     $files = $fs->get_area_files($context->id, 'mod_hyperaudio', 'audiofile',
8         $hyperaudiooid, 'itemid, filepath, filename', false);
9
10    $file = reset($files);
11    $DB->set_field('hyperaudio', 'audiofile', $file->get_filename(), array(
12        'id' => $hyperaudiooid
13    ));
14 }
```

Auflistung 5.3: Ausschnitt der **locallib.php** in der 1. Iteration

Wie bereits in Abschnitt 5.1 angedeutet, übernimmt die **renderer.php** die Anzeige der Hyperaudio-Inhalte (siehe Auflistung 5.5). Die **view.php** dagegen reduziert sich auf wenige Zeilen (vgl. Auflistung 5.4). Der Plugin-Renderer wird hier benutzt, um Header, Hauptinhalte und Footer anzuzeigen.

```
1 $output = $PAGE->get_renderer('mod_hyperaudio');
2 echo $output->header();
3 echo $output->display($hyperaudio, $context);
4 echo $output->footer();
```

Auflistung 5.4: Ausschnitt der **view.php** in der 1. Iteration

In der Funktion `display` der Klasse `mod_hyperaudio_renderer`, die von der Klasse `plugin_renderer_base` erbt, werden die HTML-Inhalte erzeugt. Dabei handelt es sich in der 1. Iteration um einen Container, der ein `<audio>`-Element beinhaltet. Als Quelle wird im `<source>`-Element eine URL (Uniform Resource Locator) angegeben, die zuvor mit Moodle-Standardmitteln erzeugt wurde und auf die in der Datenbank abgelegte Audio-Datei verweist.

```
1 $audio_fileinfo = array(
2     'component' => 'mod_hyperaudio',
3     'filearea' => 'audiofile',
4     'itemid' => $hyperaudio->id,
5     'contextid' => $context->id,
6     'filepath' => '/',
7     'filename' => $hyperaudio->audiofile
8 );
9
10 $audiofileurl = moodle_url::make_pluginfile_url(
```

```
11     $audio_fileinfo['contextid'], $audio_fileinfo['component'],
12     $audio_fileinfo['filearea'], $audio_fileinfo['itemid'],
13     $audio_fileinfo['filepath'], $audio_fileinfo['filename']);
14 $audio_url = $audiofileurl->get_scheme() . '://' . $audiofileurl->get_host() .
15     $audiofileurl->get_path();
16 if ($audiofileurl->get_port()) {
17     $audio_url .= ':' . $audiofileurl->get_port();
18 }
19 $output = '<div id="hyperaudio" data-hyperaudio_id="'. $hyperaudio->id.'">';
20 $output .= '<audio id="hyperaudio_audio" controls style="width:800px;">' .
21     '<source src="'. $audio_url . '" />' .
22     '</audio>';
23 $output .= '</div>';
24
25 echo $output;
```

Auflistung 5.5: Ausschnitt der `renderer.php` in der 1. Iteration

Auf die beschriebene Art und Weise lässt sich ein Hyperaudio-Dokument, das vorläufig allein aus einer Audio-Datei besteht, speichern und mithilfe des HTML5-Audio-Players wiedergeben.

5.2.2 Speichern und Anzeige von Zusatzinhalten

Bei der zweiten Iteration wird das Plugin um die Möglichkeit zum Speichern und zeitabhängigen Anzeigen der Zusatzinhalte erweitert. Für das Speichern wird analog zu Abschnitt 5.2.1 vorgegangen. Es wird ein Filemanager (**mod_form**) ergänzt, welcher das Hochladen von beliebig vielen Bilddateien erlaubt. Zusätzlich wird die **locallib.php** um die Funktionen `hyperaudio_update_additional_content` und `hyperaudio_delete_additional_content` erweitert. Die Löschfunktion dient dazu zu verhindern, dass beispielsweise beim Ergänzen von Zusatzinhalten, die bereits vorhanden Zusatzinhalte erneut abgespeichert werden. Dementsprechend wird diese Funktion, wie in Auflistung 5.6 zu sehen, innerhalb der Funktion `hyperaudio_update_additional_content` aufgerufen. Die Zeilen 19 bis 24 dienen dazu die Metadaten des Zusatzinhaltes festzuhalten, in dieser Iteration werden diese noch manuelle mit Beispieldaten befüllt. Die Funktion `hyperaudio_update_additional_content` wird dann entsprechend bei den Funktionen `add_instance` und `update_instance` innerhalb der **lib.php** ergänzt.

```
1 function hyperaudio_update_additional_content($hyperaudiooid, $context, $draftitemid,
2     $configfile) {
3     global $DB;
4     hyperaudio_delete_additional_content($hyperaudiooid, $context);
5
6     file_save_draft_area_files($draftitemid, $context->id, 'mod_hyperaudio',
7         'additional_content', $hyperaudiooid);
8     $fs = get_file_storage();
9
10    $files = $fs->get_area_files($context->id, 'mod_hyperaudio', 'additional_content',
11        $hyperaudiooid, 'itemid, filepath, filename',
12        false);
13    $counter=1;
14    $begin=0;
15    $end=10;
16    foreach ($files as $file) {
17        $additional_content = new \stdClass();
18        $additional_content->file = $file->get_filename();
19        $additional_content->hyperaudio_id = $hyperaudiooid;
20
21        $additional_content->name = 'Name: '.$counter;
22
23        $DB->insert('mod_hyperaudio_additional_content', $additional_content);
24    }
25}
```

5 Implementierung

```
20 $additional_content->course_unit = 'Kurseinheit: '.$counter;
21 $additional_content->page = 'Seite: '.$counter;
22 $additional_content->description = 'Beschreibung: '.$counter;
23 $additional_content->begin = $begin;
24 $additional_content->end = $end;
25
26 $now = time();
27 $additional_content->timecreated = $now;
28 $additional_content->timemodified = $now;
29
30 $additional_content_id = $DB->insert_record('additional_content',
31 $additional_content);
32
33 $counter++;
34 $begin+=15;
35 $end+=15;
36 }
```

Auflistung 5.6: Ausschnitt der **locallib.php** in der 2. Iteration

Nachdem nun die Zusatzinhalte samt der Metadaten gespeichert sind, kann sich der zeitabhängigen Darstellung dieser beim Abspielen der Audio-Datei zugewendet werden. An dieser Stelle findet nun das JavaScript-Framework *Popcorn.js* seine Anwendung, da dies wie bereits in Abschnitt 3.4 beschrieben, das zeitabhängige Annotieren von Inhalten ermöglicht. Um *Popcorn.js* und dessen *Images*-Plugin nutzen zu können, werden beide JavaScript Dateien in dem Ordner **/hyperaudio/thirdparty** abgelegt. Anschließend werden diese in die **view.php** eingebunden (siehe Auflistung 5.7).

```
1 $PAGE->requires->js('mod/hyperaudio/thirdparty/popcorn.js', true);  
2 $PAGE->requires->js('mod/hyperaudio/thirdparty/popcorn.image.js', true);
```

Auflistung 5.7: Ausschnitt der `view.php` in der 2. Iteration

Der eigentliche Einsatz von *Popcorn.js* findet im Renderer statt. Hierfür wird der Code aus der ersten Iteration um ein neues `div` erweitert, in welchem die Zusatzinhalte angezeigt werden sollen. Darauf werden zunächst alle zugehörigen Zusatzinhalte geladen und ein Link zu diesem generiert (Zielen 9 bis 32 in Abbildung 5.8). Beim Aufruf des *Popcorn.js Images*-Plugins werden in Zeile 35 die Metadaten über Beginn und Ende der Anzeige sowie der generierte Link verwendet, um den Zusatzinhalt zum gewünschten Zeitpunkt im dafür erstellten `div` darzustellen.

```

18
19 $additional_content_fileinfo = array(
20   'component' => 'mod_hyperaudio',
21   'filearea' => 'additional_content',
22   'itemid' => $hyperaudio->id,
23   'contextid' => $context->id,
24   'filepath' => '/',
25   'filename' => $additional_content->file
26 );
27
28 $additional_content_fileurl = moodle_url::make_pluginfile_url(
29   $additional_content_fileinfo['contextid'], $additional_content_fileinfo['component'],
30   $additional_content_fileinfo['itemid'], $additional_content_fileinfo['filepath'],
31   $additional_content_fileinfo['filename']);
32 $additional_content_url = $additional_content_fileurl->get_scheme() . '://'.
33   $additional_content_fileurl->get_host() . $additional_content_fileurl->get_path
34   ();
35 if ($additional_content_fileurl->get_port()){
36   $additional_content_url .= ':' . $additional_content_fileurl->get_port();
37 }
38 $output .= 'popcorn.image({start: '.$additional_content->begin.', end: '.
39   $additional_content->end.', href: "javascript:void(0);", src: "'.
40   $additional_content_url.'", target: "hyperaudio_additional_content"});';
41 echo $output;

```

Auflistung 5.8: Ausschnitt der **renderer.php** in der 2. Iteration

Mit diesen Erweiterungen wurde das Ziel dieser Iteration erreicht. Es ist nun möglich Zusatzinhalte abzuspeichern und diese unter Verwendung von *Popcorn.js* zeitabhängig zur Audi-Datei darstellen zu lassen

5.2.3 Einbindung der Konfigurationsdatei

Während zum aktuellen Stand die Metadaten der Zusatzinhalte noch manuell im Code gepflegt werden, sollen dies mittels dieser Iteration aus einer Konfigurationsdatei ausgelesen werden. Am Anfang dieser Iteration steht erneut die Erweiterung der **mod_form** um einen weiteren Filemanger. Die Speicherung verläuft analog zu der der Zusatzinhalte, mit der Einschränkung, dass nur eine Konfigurationsdatei im JSON-Format gespeichert werden kann. In diesem Zuge wird also erneut die **lib.php** entsprechend erweitert. Innerhalb der **lib.php** werden die Funktionen `hyperaudio_update_hyperaudio_config` und `hyperaudio_delete_hyperaudio_config` ergänzt. Des Weiteren wird, wie in Auflistung 5.9 zu erkennen, die Funktion `hyperaudio_update_additional_content` erweitert. Es ist ersichtlich, dass zunächst in Zeile 4 die Inhalte der Konfigurationsdatei ausgelesen werden, um diese Metadaten dann beim Abspeichern der Zusatzinhalte zu verwenden.

warum ist hier die delete Funktion notwendig?

```

1 function hyperaudio_update_additional_content($hyperaudiooid, $context, $draftitemid,
2   $configfile) {
3   global $DB;
4   $additional_contents_data = parse_hyperaudio_config($hyperaudiooid, $context,
5   $configfile);

```

5 Implementierung

```
5      hyperaudio_delete_additional_content($hyperaudiooid, $context);
6
7      file_save_draft_area_files($draftitemid, $context->id, 'mod_hyperaudio', 'additional_content', $hyperaudiooid);
8      $fs = get_file_storage();
9
10     $files = $fs->get_area_files($context->id, 'mod_hyperaudio', 'additional_content',
11         $hyperaudiooid, 'itemid, filepath, filename',
12         false);
13     foreach ($files as $file) {
14         $additional_content = new \stdClass();
15         $additional_content->file = $file->get_filename();
16         $additional_content->hyperaudio_id = $hyperaudiooid;
17
18         $additional_content_data = $additional_contents_data[$additional_content->file];
19
20         $additional_content->name = $additional_content_data->name;
21         $additional_content->course_unit = $additional_content_data->course_unit;
22         $additional_content->page = $additional_content_data->page;
23         $additional_content->description = $additional_content_data->description;
24         $additional_content->begin = $additional_content_data->begin;
25         $additional_content->end = $additional_content_data->end;
26
27         $now = time();
28         $additional_content->timecreated = $now;
29         $additional_content->timemodified = $now;
30
31         $additional_content_id = $DB->insert_record('additional_content',
32             $additional_content);
33     }
33 }
```

Auflistung 5.9: Ausschnitt der **locallib.php** in der 3. Iteration

Nachdem nun also die Zusatzinhalte mit den korrekten Metadaten versorgt sind, können diese im Renderer für die gesteuerte Darstellung mittels *Popcorn.js* verwendet werden. Somit werden dank dieser Iteration die Zusatzinhalte nun zu den Zeitpunkten dargestellt, welche in der Konfigurationsdatei festgelegt sind.

5.2.4 Speichern und Anzeige von Kommentaren

Nachdem in den vorherigen Iterationen die Pflege und das Abspielen von Hyperaudio-Dokumenten im Fokus lag, konzentriert sich diese Iteration auf die Kommentarfunktion. Zu diesem Zweck wird im ersten Schritt der Renderer um entsprechende Elemente ergänzt, welche in Auflistung 5.10 zu sehen sind. Bei den Elementen handelt es sich zum einen um ein Input-Feld mit dazugehörigen Submit-Button. Diese Elemente dienen dem Erstellen von Kommentaren. Das Element `<div id="hyperaudio_comments"></div>` soll der Anzeige der Kommentare dienen.

Problem mit » lösen

```
1 $output .=
2     '<div>
3         <label>'.get_string('mod_hyperaudio_renderer_comment', 'hyperaudio').':
4             <input type="text" id="hyperaudio_comment" name="comment" autocomplete="off"/>
5             <button type="button" class="comment_submit" data-comment_type="'.CommentType
6                 :Comment.'">'.get_string('mod_hyperaudio_renderer_submit_comment', 'hyperaudio'
7             ) . '</button>
8     </div>'
```

```

6     </label>
7     </div>';
8 $output .= '<div id="hyperaudio_comments"></div>';

```

Auflistung 5.10: Ausschnitt der **renderer.php** in der 4. Iteration

Das Speichern als auch das Anzeigen der Kommentare wird unter dem Einsatzes von Webservices, welche durch ein JavaScript-Modul ([/hyperaudio/AMD/src/comments.js](#)) angesprochen werden, umgesetzt. Hierfür werden zusätzlich die **services.php** ([/hyperaudio/db](#)) und **external.php** ([/hyperaudio/classes](#)) für die Bereitstellung der Webservices benötigt.

In der **services.php** werden zunächst die benötigten Webservices beschrieben (siehe Auflistung 5.11). Wie zu erkennen ist, wird hier lediglich auf die **external.php** verwiesen, in welcher der eigentliche Code des Webservices erstellt werden muss.

```

1 $functions = array(
2     'mod_hyperaudio_save_comment' => array(
3         'classname'    => 'mod_hyperaudio_external',
4         'methodname'   => 'save_comment',
5         'classpath'    => 'mod/hyperaudio/classes/external.php',
6         'description'  => 'Save comment',
7         'type'          => 'write'
8     ),
9     'mod_hyperaudio_load_comments' => array(
10        'classname'   => 'mod_hyperaudio_external',
11        'methodname'  => 'load_comments',
12        'classpath'   => 'mod/hyperaudio/classes/external.php',
13        'description' => 'Load comments',
14        'type'         => 'read'
15    )
16 );

```

Auflistung 5.11: **services.php** in der 5. Iteration

In der **external.php** werden in der `mod_hyperaudio_external` extends, welche von der Klasse `external_api` erbt (vgl. Auflistung 5.12), die nötigen Funktionen definiert. Beim Speichern von Kommentaren sind dies die Funktionen `save_comment_parameters`, `save_comment_is_allowed_from_ajax`, `save_comment` und `save_comment_returns`. In der ersten Funktion werden die Parameter, welche benötigt werden definiert. In der zweiten Funktion wird festgelegt, ob die Funktion durch AJAX (Asynchronous JavaScript and XML) aufgerufen werden darf, dies ist für das geplante Vorgehen innerhalb des Hyperaudio-Plugins nötigt. In der Funktion `save_comment` wird die eigentliche Funktionalität des Webservices programmiert. In diesem Fall also das Speichern des Kommentars in der Datenbank. In der vierten Funktion werden die Rückgabeparamter des Webser-
vices beschrieben. Im Fall der Funktion `load_comments_returns` also die Kommentare, welche angezeigt werden sollen.

```

1 class mod_hyperaudio_external extends external_api {
2
3     public static function load_comments_parameters() {
4         return new external_function_parameters(
5             array(
6                 'hyperaudio_id' => new external_value(PARAM_INT, 'hyperaudio_id')
7             )
8         );
9     }
10
11    public static function load_comments_is_allowed_from_ajax() {
12        return true;
13    }

```

5 Implementierung

```
14
15     public static function load_comments($hyperaudio_id) {
16         global $CFG, $DB, $USER;
17
18         require_once ($CFG->dirroot . '/mod/hyperaudio/locallib.php');
19         require_once ($CFG->dirroot . '/mod/hyperaudio/classes/enums.php');
20
21         $comments = $DB->get_records_sql(
22             'SELECT comments.id, comments.comment_type, comments.commenttext, comments.
23             timeannotated, comments.timecreated, user.username, comments.comment_id'.
24             ' FROM mdl_hyperaudio_comments comments'.
25             ' INNER JOIN mdl_user user ON comments.userid = user.id'.
26             ' LEFT JOIN mdl_hyperaudio_comments original_comment ON original_comment.id =
27             comments.comment_id'.
28             ' WHERE comments.hyperaudio_id = ?'.
29             ' AND (comments.comment_type = ? OR (comments.comment_type =? AND comments.
30             userid = ?))'.
31             ' ORDER BY IFNULL(original_comment.timecreated, comments.timecreated)*POWER
32             (10, 11) + comments.timecreated',
33             array($hyperaudio_id, CommentType::Comment, CommentType::Note, $USER->id)
34         );
35
36         $result_comments = array();
37         foreach ($comments as $comment) {
38             $result_comment = array(
39                 'id' => $comment->id,
40                 'username' => $comment->username,
41                 'date' => date('d.m.Y H:i', $comment->timecreated),
42                 'time' => format_time_annotated($comment->timeannotated),
43                 'text' => $comment->commenttext,
44                 'comment_type_display' => CommentType::get_comment_type($comment->
45                 comment_type),
46                 'comment_type' => $comment->comment_type,
47                 'comment_id' => $comment->comment_id
48             );
49             $result_comments[] = $result_comment;
50         }
51
52         return $result_comments;
53     }
54
55     public static function load_comments_returns() {
56         return new external_multiple_structure(
57             new external_single_structure(
58                 array(
59                     'id' => new external_value(PARAM_INT, 'id'),
60                     'username' => new external_value(PARAM_TEXT, 'username'),
61                     'date' => new external_value(PARAM_TEXT, 'date'),
62                     'time' => new external_value(PARAM_TEXT, 'time'),
63                     'text' => new external_value(PARAM_TEXT, 'text'),
64                     'comment_type_display' => new external_value(PARAM_TEXT, 'comment_type_display'),
65                     'comment_type' => new external_value(PARAM_TEXT, 'comment_type'),
66                     'comment_id' => new external_value(PARAM_INT, 'comment_id')
67                 )
68             )
69         );
70     }
71
72     public static function save_comment_parameters() {
73         return new external_function_parameters(
74             array(
75                 'comment' => new external_value(PARAM_TEXT, 'comment'),
76             )
77         );
78     }
79
80     public static function update_comment_parameters() {
81         return new external_function_parameters(
82             array(
83                 'comment' => new external_value(PARAM_TEXT, 'comment'),
84                 'comment_id' => new external_value(PARAM_INT, 'comment_id')
85             )
86         );
87     }
88
89     public static function delete_comment_parameters() {
90         return new external_function_parameters(
91             array(
92                 'comment_id' => new external_value(PARAM_INT, 'comment_id')
93             )
94         );
95     }
96
97     public static function get_comment_type_parameters() {
98         return new external_function_parameters(
99             array(
100                 'comment_type' => new external_value(PARAM_TEXT, 'comment_type')
101             )
102         );
103     }
104
105     public static function get_comment_type_display_parameters() {
106         return new external_function_parameters(
107             array(
108                 'comment_type' => new external_value(PARAM_TEXT, 'comment_type')
109             )
110         );
111     }
112
113     public static function get_comment_text_parameters() {
114         return new external_function_parameters(
115             array(
116                 'comment_id' => new external_value(PARAM_INT, 'comment_id')
117             )
118         );
119     }
120
121     public static function get_comment_username_parameters() {
122         return new external_function_parameters(
123             array(
124                 'comment_id' => new external_value(PARAM_INT, 'comment_id')
125             )
126         );
127     }
128
129     public static function get_comment_date_parameters() {
130         return new external_function_parameters(
131             array(
132                 'comment_id' => new external_value(PARAM_INT, 'comment_id')
133             )
134         );
135     }
136
137     public static function get_comment_time_parameters() {
138         return new external_function_parameters(
139             array(
140                 'comment_id' => new external_value(PARAM_INT, 'comment_id')
141             )
142         );
143     }
144
145     public static function get_comment_time_annotated_parameters() {
146         return new external_function_parameters(
147             array(
148                 'comment_id' => new external_value(PARAM_INT, 'comment_id')
149             )
150         );
151     }
152
153     public static function get_comment_original_comment_parameters() {
154         return new external_function_parameters(
155             array(
156                 'comment_id' => new external_value(PARAM_INT, 'comment_id')
157             )
158         );
159     }
160
161     public static function get_comment_original_comment_time_parameters() {
162         return new external_function_parameters(
163             array(
164                 'comment_id' => new external_value(PARAM_INT, 'comment_id')
165             )
166         );
167     }
168
169     public static function get_comment_original_comment_time_annotated_parameters() {
170         return new external_function_parameters(
171             array(
172                 'comment_id' => new external_value(PARAM_INT, 'comment_id')
173             )
174         );
175     }
176
177     public static function get_comment_original_comment_userid_parameters() {
178         return new external_function_parameters(
179             array(
180                 'comment_id' => new external_value(PARAM_INT, 'comment_id')
181             )
182         );
183     }
184
185     public static function get_comment_original_comment_username_parameters() {
186         return new external_function_parameters(
187             array(
188                 'comment_id' => new external_value(PARAM_INT, 'comment_id')
189             )
190         );
191     }
192
193     public static function get_comment_original_comment_text_parameters() {
194         return new external_function_parameters(
195             array(
196                 'comment_id' => new external_value(PARAM_INT, 'comment_id')
197             )
198         );
199     }
200
201     public static function get_comment_original_comment_type_parameters() {
202         return new external_function_parameters(
203             array(
204                 'comment_id' => new external_value(PARAM_INT, 'comment_id')
205             )
206         );
207     }
208
209     public static function get_comment_original_comment_type_display_parameters() {
210         return new external_function_parameters(
211             array(
212                 'comment_id' => new external_value(PARAM_INT, 'comment_id')
213             )
214         );
215     }
216
217     public static function get_comment_original_comment_type_annotation_parameters() {
218         return new external_function_parameters(
219             array(
220                 'comment_id' => new external_value(PARAM_INT, 'comment_id')
221             )
222         );
223     }
224
225     public static function get_comment_original_comment_annotation_parameters() {
226         return new external_function_parameters(
227             array(
228                 'comment_id' => new external_value(PARAM_INT, 'comment_id')
229             )
230         );
231     }
232
233     public static function get_comment_original_comment_annotation_annotation_parameters() {
234         return new external_function_parameters(
235             array(
236                 'comment_id' => new external_value(PARAM_INT, 'comment_id')
237             )
238         );
239     }
240
241     public static function get_comment_original_comment_annotation_annotation_annotation_parameters() {
242         return new external_function_parameters(
243             array(
244                 'comment_id' => new external_value(PARAM_INT, 'comment_id')
245             )
246         );
247     }
248
249     public static function get_comment_original_comment_annotation_annotation_annotation_annotation_parameters() {
250         return new external_function_parameters(
251             array(
252                 'comment_id' => new external_value(PARAM_INT, 'comment_id')
253             )
254         );
255     }
256
257     public static function get_comment_original_comment_annotation_annotation_annotation_annotation_annotation_parameters() {
258         return new external_function_parameters(
259             array(
260                 'comment_id' => new external_value(PARAM_INT, 'comment_id')
261             )
262         );
263     }
264
265     public static function get_comment_original_comment_annotation_annotation_annotation_annotation_annotation_annotation_parameters() {
266         return new external_function_parameters(
267             array(
268                 'comment_id' => new external_value(PARAM_INT, 'comment_id')
269             )
270         );
271     }
272
273     public static function get_comment_original_comment_annotation_annotation_annotation_annotation_annotation_annotation_annotation_parameters() {
274         return new external_function_parameters(
275             array(
276                 'comment_id' => new external_value(PARAM_INT, 'comment_id')
277             )
278         );
279     }
280
281     public static function get_comment_original_comment_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_parameters() {
282         return new external_function_parameters(
283             array(
284                 'comment_id' => new external_value(PARAM_INT, 'comment_id')
285             )
286         );
287     }
288
289     public static function get_comment_original_comment_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_parameters() {
290         return new external_function_parameters(
291             array(
292                 'comment_id' => new external_value(PARAM_INT, 'comment_id')
293             )
294         );
295     }
296
297     public static function get_comment_original_comment_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_parameters() {
298         return new external_function_parameters(
299             array(
300                 'comment_id' => new external_value(PARAM_INT, 'comment_id')
301             )
302         );
303     }
304
305     public static function get_comment_original_comment_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_parameters() {
306         return new external_function_parameters(
307             array(
308                 'comment_id' => new external_value(PARAM_INT, 'comment_id')
309             )
310         );
311     }
312
313     public static function get_comment_original_comment_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_parameters() {
314         return new external_function_parameters(
315             array(
316                 'comment_id' => new external_value(PARAM_INT, 'comment_id')
317             )
318         );
319     }
320
321     public static function get_comment_original_comment_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_parameters() {
322         return new external_function_parameters(
323             array(
324                 'comment_id' => new external_value(PARAM_INT, 'comment_id')
325             )
326         );
327     }
328
329     public static function get_comment_original_comment_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_parameters() {
330         return new external_function_parameters(
331             array(
332                 'comment_id' => new external_value(PARAM_INT, 'comment_id')
333             )
334         );
335     }
336
337     public static function get_comment_original_comment_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_parameters() {
338         return new external_function_parameters(
339             array(
340                 'comment_id' => new external_value(PARAM_INT, 'comment_id')
341             )
342         );
343     }
344
345     public static function get_comment_original_comment_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_parameters() {
346         return new external_function_parameters(
347             array(
348                 'comment_id' => new external_value(PARAM_INT, 'comment_id')
349             )
350         );
351     }
352
353     public static function get_comment_original_comment_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_parameters() {
354         return new external_function_parameters(
355             array(
356                 'comment_id' => new external_value(PARAM_INT, 'comment_id')
357             )
358         );
359     }
360
361     public static function get_comment_original_comment_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_parameters() {
362         return new external_function_parameters(
363             array(
364                 'comment_id' => new external_value(PARAM_INT, 'comment_id')
365             )
366         );
367     }
368
369     public static function get_comment_original_comment_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_parameters() {
370         return new external_function_parameters(
371             array(
372                 'comment_id' => new external_value(PARAM_INT, 'comment_id')
373             )
374         );
375     }
376
377     public static function get_comment_original_comment_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_parameters() {
378         return new external_function_parameters(
379             array(
380                 'comment_id' => new external_value(PARAM_INT, 'comment_id')
381             )
382         );
383     }
384
385     public static function get_comment_original_comment_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_parameters() {
386         return new external_function_parameters(
387             array(
388                 'comment_id' => new external_value(PARAM_INT, 'comment_id')
389             )
390         );
391     }
392
393     public static function get_comment_original_comment_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_annotation_parameters() {
394         return new external_function_parameters(
395             array(
396                 'comment_id' => new external_value(PARAM_INT, 'comment_id')
397             )
398         );
399     }
399 }
```

```

71     'hyperaudio_id' => new external_value(PARAM_INT, 'hyperaudio_id'),
72     'timeannotated' => new external_value(PARAM_INT, 'timeannotated'),
73     'comment_type' => new external_value(PARAM_TEXT, 'comment_type')
74   )
75 }
76 }
77
78 public static function save_comment_is_allowed_from_ajax() {
79   return true;
80 }
81
82 public static function save_comment($comment, $hyperaudio_id, $timeannotated,
83   $comment_type) {
84   global $CFG, $DB, $USER;
85
86   require_once ($CFG->dirroot . '/mod/hyperaudio/classes/enums.php');
87
88   $hyperaudio_comment = new \stdClass();
89   $hyperaudio_comment->hyperaudio_id = $hyperaudio_id;
90   $hyperaudio_comment->userid = $USER->id;
91   $hyperaudio_comment->timeannotated = $timeannotated;
92   $hyperaudio_comment->comment_type = $comment_type;
93
94   if ($comment_type == CommentType::Comment || $comment_type == CommentType::Note)
95   {
96     $hyperaudio_comment->commenttext = $comment;
97   }
98
99   $now = time();
100  $hyperaudio_comment->timecreated = $now;
101  $hyperaudio_comment->timemodified = $now;
102
103  $DB->insert_record('hyperaudio_comments', $hyperaudio_comment);
104
105  return 0;
106 }
107
108 public static function save_comment_returns() {
109   return new external_value(PARAM_INT, '0 = ok, 1 = error');
110 }

```

Auflistung 5.12: **external.php** in der 5. Iteration

Listing anpassen, speziell SQL-Statement auf die nur Kommentare-Variante kürzen

Auf die mittels **services.php** und **external.php** bereitgestellten Webservice wird letztlich in der **comments.js** zugegriffen, um die im Renderer definierten Elemente zu befüllen, beziehungsweise deren Inhalte auszulesen. Die Funktion zum Speichern eines Kommentars ist exemplarisch in Auflistung 5.13 dargestellt. Zunächst werden die im Renderer erstellten Elemente zum Erstellen eines Kommentars ausgelesen, dann wird mittels eines AJAX-Calls der Webservice `mod_hyberaudio_save_comment` aufgerufen und die benötigten Parameter übergeben. Hierbei ist hervorzuheben, dass der Zeitpunkt zu dem der Kommentar an das Hyperaudio-Dokument annotiert wird über die Funktion `currentTime` des *Popcorn.js*-Frameworks ermittelt wird. Bei der Funktion zum Darstellen der Kommentare wird analog vorgegangen, nur dass mittels des Webservices die Kommentare für das betroffene Hyperaudio-Dokumente aufgerufen werden und dann innerhalb des entsprechenden Elements des Renderers dargestellt werden.

```

1 function save_comment(comment_type) {
2   var comment = $("#hyperaudio_comment").val();

```

5 Implementierung

```
3 var hyperaudio_id = get_hyperaudio_id();
4 var popcorn = Popcorn("#hyperaudio_audio");
5 var timeannotated = popcorn.currentTime();
6 timeannotated = parseInt(timeannotated);
7
8 var promises = ajax.call([
9   methodname: 'mod_hyperaudio_save_comment',
10  args:{
11    'comment': comment,
12    'hyperaudio_id': hyperaudio_id,
13    'timeannotated': timeannotated,
14    'comment_type': comment_type
15  }
16 ]]);
17 promises[0].done(function(data) {
18   show_comments();
19 });
20 }
```

Auflistung 5.13: Ausschnitt der **comments.js** in der 5. Iteration

Sprache des Listings anpassen

Mit dem Abschluss dieser Iteration wurde die Möglichkeit geschaffen Kommentar zu erstellen und den Zeitpunkt innerhalb des Hyperaudio-Dokuments festzuhalten, zu dem der Kommentar erstellt wurde. Im gleichen Zuge wurde auch die Darstellung dieser Kommentare ermöglicht.

5.2.5 Antworten auf Kommentare

Während bereits Kommentare verfasst und angezeigt werden können, fehlt noch die Möglichkeit auf diese zu antworten. Diese Funktionalität wird in dieser Iteration ergänzt. Zu diesem Zweck wird ein neuer Webservices zum Speichern von Antworten erstellt und der Webservice zum Darstellen der Kommentare um die Antworten erweitert. Demzufolge wird in der **services.php** eine Beschreibung für den neuen Webservice zum Erstellen von Antworten ergänzt und in der **external.php** werden die vier Funktionen analog zum Webservice zum Speichern Kommentaren ergänzt. Bei der **comments.js** bedarf es ebenfalls nur kleinere Anpassungen, unter anderem um die Anzeige eines Antworten Buttons und eines Textfeldes bei dessen Betätigung zu ermöglichen. Die hierfür notwendige Erweiterung innerhalb der Funktion `show_comments` ist in der Auflistung 5.14 zu sehen.

```
1 if (this.comment_type === CommentType.Comment && !this.comment_id) {
2   output = output + '<div class="comment_answer comment_actions" data-comment_id="'+
3     this.id+'>';
4   output = output + '<a href="javascript:void(0);" class="comment_answer_link
5     comment_link" onclick="show_answer_input(this)" data-comment_id="'+this.id+'">' +
6     string_answer + '</a>';
7   output = output + '<input type="text" class="comment_answer_text
8     comment_hidden_element" data-comment_id="'+this.id+'"/>';
9   output = output + '<button type="button" class="comment_answer_button
10    comment_hidden_element" data-comment_id="'+this.id+'>' + string_answer + '<
11    button>';
12   output = output + '</div>';
```

Auflistung 5.14: Ausschnitt der **comments.js** in der 6. Iteration

Die Darstellung der Antworten und deren Zuordnung zu den dazugehörigen Kommentaren wird durch die in Auflistung 5.15 ersichtlichen Anpassung des SQL-Statements `load_comments`-Funktion innerhalb der **external.php** erreicht.

```

1 $comments = $DB->get_records_sql(
2   'SELECT comments.id, comments.comment_type, comments.commenttext, comments.
3     timeannotated, comments.timecreated, user.username, comments.comment_id'.
4   ' FROM mdl_hyperaudio_comments comments'.
5   ' INNER JOIN mdl_user user ON comments.userid = user.id'.
6   ' LEFT JOIN mdl_hyperaudio_comments original_comment ON original_comment.id =
7     comments.comment_id'.
8   ' WHERE comments.hyperaudio_id = ?'.
9   ' AND (comments.comment_type = ? OR (comments.comment_type =? AND comments.userid
10    = ?))'.
11   ' ORDER BY IFNULL(original_comment.timecreated, comments.timecreated)*POWER(10,
12    11) + comments.timecreated',
13   array($hyperaudio_id, CommentType::Comment, CommentType::Note, $USER->id)
14 );

```

Auflistung 5.15: Ausschnitt der **external.php** in der 6. Iteration

Durch die Erweiterungen und Anpassungen innerhalb dieser Iteration ist es nun möglich auf Kommentare zu antworten und diese Antwort bei dem zugehörigen Kommentar darzustellen.

SQL-Statement auf Kommentare und Antworten kürzen

Abkürzung SQL muss schon früher kommen

5.2.6 Notizen

Nachdem durch die vorangegangenen Iterationen nun die Funktionen für die Kommunikation zwischen Studierenden und Lehrenden geschaffen wurden, zielt diese Iteration auf das Erstellen und Darstellen von Notizen ab.

Im ersten Schritt wird hierfür in der **renderer.php** die Erweiterung um einen weiteren dedizierten Button entsprechend der Auflistung 5.16 vorgenommen.

```

1 $output .=
2   '<div>
3     <label>' . get_string('mod_hyperaudio_renderer_comment', 'hyperaudio') .':
4       <input type="text" id="hyperaudio_comment" name="comment" autocomplete="off"/>
5       <button type="button" class="comment_submit" data-comment_type="'
6         .CommentType
7         ::Comment.'">' . get_string('mod_hyperaudio_renderer_submit_comment', 'hyperaudio')
8       ) . '</button>
9       <button type="button" class="comment_submit" data-comment_type="'
10        .CommentType
11        ::Note.'">' . get_string('mod_hyperaudio_renderer_submit_note', 'hyperaudio') .'
12      button>
13     </label>
14   </div>';

```

Auflistung 5.16: Ausschnitt der **renderer.php** in der 7. Iteration

Nachdem diese Anpassung vorgenommen wurde, können zum Speichern der Notizen die gleichen Webservices und Funktionen, wie zum Speichern der Kommentare verwendet werden. Dies röhrt daher, dass Notizen als eine bestimmte Art von Kommentaren angesehen wird (vgl. Abschnitt 4.1.2). Zum Darstellen der Notizen wird, wie bereits in der vorherigen Iteration für die Antworten, das SQL-Statement in der `load_comments` der **external.php** angepasst.

Da Notizen im Gegensatz zu Kommentaren geändert und gelöscht werden können sollen, müssen hierfür entsprechende Webservices bereitgestellt werden (siehe Auflistung 5.17).

```

1 'mod_hyperaudio_save_note_edit' => array(

```

5 Implementierung

```
2     'classname' => 'mod_hyperaudio_external',
3     'methodname' => 'save_note_edit',
4     'classpath' => 'mod/hyperaudio/classes/external.php',
5     'description' => 'Save note edit',
6     'type' => 'write'
7   ),
8   'mod_hyperaudio_delete_comment' => array(
9     'classname' => 'mod_hyperaudio_external',
10    'methodname' => 'delete_comment',
11    'classpath' => 'mod/hyperaudio/classes/external.php',
12    'description' => 'Delete comment',
13    'type' => 'write'
14 )
```

Auflistung 5.17: Ausschnitt der **services.php** in der 7. Iteration

Wie bereits am Namen des Webservices erkennbar, wird dieser theoretisch in der Lage sein, alle Arten von Kommentaren löschen zu können. Der dazugehörig Button wird jedoch nur bei Notizen dargestellt. Dies wird mittels der Bedingung aus Auflistung 5.18 innerhalb der **comments.js** erreicht.

```
1 else if (this.comment_type === CommentType.Note) {
2   output = output + '<div class="comment_note comment_actions" data-comment_id="'+
3     this.id+'>';
4   output = output + '<a href="javascript:void(0);" class="comment_note_edit_link
5     comment_link" onclick="show_note_edit_input(this)" data-comment_id="'+this.id+''
6     >' + string_edit + '</a>';
7   output = output + '<a href="javascript:void(0);" class="comment_note_delete_link
8     comment_link" data-comment_id="'+this.id+'>' + string_delete + '</a>';
9   output = output + '<input type="text" class="comment_note_edit_text
10     comment_hidden_element" data-comment_id="'+this.id+'"/>';
11   output = output + '<button type="button" class="comment_note_edit_button
12     comment_hidden_element" data-comment_id="'+this.id+'>' + string_edit + '<
13     button>';
14   output = output + '</div>';
15 }
```

Auflistung 5.18: Ausschnitt der **comments.js** in der 7. Iteration

Entsprechend der Webservices werden innerhalb der **external.php** Funktionen zum Editieren von Notizen und dem Löschen von Kommentaren erstellt. Damit sind alle nötigen Schritte durchgeführt um Notizen erstellen, anzeigen und löschen zu können.

5.2.7 Audio Cues

Das Abspielen der Audio Cues im Moment der Darstellung eines Zusatzinhaltes wird durch eine Anpassung an dem verwendeten *Popcorn.js*-Plugins *Images* erreicht. Zu diesem Zweck wird der Code aus Zeile 6 bis 7 aus Auflistung 5.19 ergänzt. Mit diesen Zeilen wird ein Audio Element erstellt und abgespielt. Diese Anpassung wird innerhalb der Funktion `start` vorgenommen, also in der Funktion, die ausgeführt wird, wenn die Darstellung des Zusatzinhalts beginnen soll.

```
1 start: function( event, options ) {
2   options.anchor.style.display = "inline";
3   if ( options.trackedContainer ) {
4     options.trackedContainer.start();
5   }
6   var audioElement = document.createElement('audio');
7   audioElement.setAttribute('src', '../hyperaudio/ding.mp3');
8   audioElement.play();
9 }
```

Auflistung 5.19: Ausschnitt der **popcorn.image.js** in der 8. Iteration

5.2.8 Galerie der Zusatzinhalte

...

5.2.9 Zeitabhängige Visualisierung der Kommentare

...

5.2.10 Markierungen

...

5.3 Zusammenfassung

...

6 Evaluation

...

Entwicklung von Kurseinheit zu Hyperaudio -> Rückschluss auf Tetrade der Medieneffekte

7 Zusammenfassung und Ausblick

...

Lehrend möchten außerdem den Inhalt ihrer bestehenden Kurse vollständig als Audio/Hyperaudio abilden können. Formeln, Tabellen, Bilder,...Sie möchten bei der Konvertierung/Produktion wenig Arbeit haben. Sie möchten Inhalte nachträglich editieren (auch Audio?!).

Ausblick:

Anforderungen die nicht umgesetzt wurden

mobile Endgeräte

<https://developers.google.com/resonance-audio/>

verschiedene Audio-Cues, über Config-JSON konfigurierbar

nachträgliche Änderungen

Versionierung

Konvertierung von Kurseinheit zu Hyperaudio

Unterstützung für Chromecast/Apple TV

Löschen von Kommentaren durch Administratoren

A Erster Teil des Anhangs

Modulnummer	Modul	Fakultät	Kursnummer	Kurs	Moodle im Einsatz
31001	Einführung in die Wirtschaftswissenschaft	Fakultät für Wirtschaftswissenschaft	40500	Einführung in die Betriebswirtschaftslehre	ja
31001	Einführung in die Wirtschaftswissenschaft	Fakultät für Wirtschaftswissenschaft	40501	Einführung in die Volkswirtschaftslehre	ja
31011	Externes Rechnungswesen	Fakultät für Wirtschaftswissenschaft	00029	Jahresabschluss	ja
31011	Externes Rechnungswesen	Fakultät für Wirtschaftswissenschaft	00034	Grundzüge der betrieblichen Steuerlehre	ja
31011	Externes Rechnungswesen	Fakultät für Wirtschaftswissenschaft	00046	Buchhaltung	ja
31021	Investition und Finanzierung	Fakultät für Wirtschaftswissenschaft	40520	Investition	ja
31021	Investition und Finanzierung	Fakultät für Wirtschaftswissenschaft	40525	Finanzierung	ja
31031	Internes Rechnungswesen und funktionale Steuerung	Fakultät für Wirtschaftswissenschaft	40530	Grundbegriffe und Systeme der Kosten- und Leistungsrechnung	ja
31031	Internes Rechnungswesen und funktionale Steuerung	Fakultät für Wirtschaftswissenschaft	40531	Grundlagen der Leistungserstellung	ja
31031	Internes Rechnungswesen und funktionale Steuerung	Fakultät für Wirtschaftswissenschaft	40532	Einführung in das Marketing	ja
31041	Theorie der Marktwirtschaft (Makroökonomik)	Fakultät für Wirtschaftswissenschaft	00049	Theorie oder Marktwirtschaft	ja
31051	Makroökonomik	Fakultät für Wirtschaftswissenschaft	40550	Makroökonomik I (Dateikurs und Studienbrief)	ja
31051	Makroökonomik	Fakultät für Wirtschaftswissenschaft	40551	Makroökonomik II (Dateikurs und Studienbrief)	ja
31071	Einführung in die Wirtschaftsinformatik	Fakultät für Wirtschaftswissenschaft	00008	Einführung in die Wirtschaftsinformatik	ja
31101	Grundlagen der Wirtschaftsmathematik und Statistik	Fakultät für Wirtschaftswissenschaft	40600	Grundlagen der Analysis und Linearen Algebra	ja
31101	Grundlagen der Wirtschaftsmathematik und Statistik	Fakultät für Wirtschaftswissenschaft	40601	Grundlagen der Statistik	ja
31201	Algorithmische Mathematik	Fakultät für Mathematik und Informatik	01142	Algorithmische Mathematik	ja
31221	Einführung in die objektorientierte Programmierung	Fakultät für Mathematik und Informatik	20022	Einführung in die objektorientierte Programmierung	nein
31231	Einführung in die technischen und theoretischen Grundlagen der Informatik	Fakultät für Mathematik und Informatik	20046	Einführung in die technische und theoretische Informatik	nein
31231	Einführung in die technischen und theoretischen Grundlagen der Informatik	Fakultät für Mathematik und Informatik	20047	Betriebssysteme und Rechnernetze für Wirtschaftsinformatiker	nein
31241	Einführung in Internet-Technologien und Informationssysteme	Fakultät für Mathematik und Informatik	01671	Datenbanken I	nein
31241	Einführung in Internet-Technologien und Informationssysteme	Fakultät für Mathematik und Informatik	01873	Daten- und Dokumentenmanagement im Internet	nein
31241	Einführung in Internet-Technologien und Informationssysteme	Fakultät für Mathematik und Informatik	01866	Sicherheit im Internet I	nein
31241	Einführung in Internet-Technologien und Informationssysteme	Fakultät für Mathematik und Informatik	01843	Einführung in wissensbasierte Systeme	nein
31251	Betriebliche Informationssysteme	Fakultät für Mathematik und Informatik	01770	Betriebliche Informationssysteme	nein
31751	Modellierung von Informationssystemen	Fakultät für Wirtschaftswissenschaft	41750	Grundlagen der Modellierung betrieblicher Informationssysteme	ja
31751	Modellierung von Informationssystemen	Fakultät für Wirtschaftswissenschaft	00817	Datenmodellierung und Datenbanksysteme	ja
31751	Modellierung von Informationssystemen	Fakultät für Wirtschaftswissenschaft	00818	Objektorientierte Systemanalyse	ja
31751	Modellierung von Informationssystemen	Fakultät für Wirtschaftswissenschaft	00825	Anwendungssysteme und Geschäftsprozessmodellierung	ja
31771	Informationsmanagement	Fakultät für Wirtschaftswissenschaft	41760	Informationsmanagement	ja

Abbildung A.1: Verwendung von Moodle in den Pflichtmodulen des Bachelorstudiengangs Wirtschaftsinformatik (Sommersemester 2018)

B Zweiter Teil des Anhangs

...

Literaturverzeichnis

- Mike Cohn. *User stories applied: For agile software development*. Addison-Wesley Professional, 2004.
- Larry Constantine. Users, Roles, and Personas. *The Persona Lifecycle*, pages 498–519, 2006.
- Larry L. Constantine. Usage-Centered Software Engineering: New Models, Methods and Metrics. In *Proceedings of the 1996 International Conference on Software Engineering: Education and Practice (SE:EP '96)*, SEEP '96, pages 2–, Washington, DC, USA, 1996. IEEE Computer Society. ISBN 0-8186-7379-6. URL <http://dl.acm.org/citation.cfm?id=829500.829892>.
- Alan Cooper et al. *The inmates are running the asylum:[Why high-tech products drive us crazy and how to restore the sanity]*. Sams Indianapolis, 2004.
- Pierre Dillenbourg. *Collaborative learning: Cognitive and computational approaches. advances in learning and instruction series*. ERIC, 1999.
- Pierre Dillenbourg, Michael J Baker, Agnes Blaye, und Claire O'Malley. The evolution of research on collaborative learning, 1995.
- Hilko Donker und Norbert Blenn. Gestaltung von hyperlinks in einer hyperaudio-encyklopädie. In *Mensch & Computer*, pages 139–148, 2007.
- FernUniversität in Hagen. Das Kursportal – helpdesk.
https://wiki.fernuni-hagen.de/helpdesk/index.php/Das_Kursportal, 2014. abgerufen am 28.07.2018.
- FernUniversität in Hagen. Kurs: WS16/17 - Kurs 01697 Einführung in Mensch-Computer-Interaktion.
<https://moodle-wrm.fernuni-hagen.de/course/view.php?id=949>, 2016. abgerufen am 15.08.2018.
- FernUniversität in Hagen. Zusammenarbeiten: Moodle, BSCW, Virtual Classroom, News, Chat - Leistungsangebot - ZMI - FernUniversität in Hagen. http://www.fernuni-hagen.de/zmi/produkte_service/kommunikationstools.shtml, 2018a. abgerufen am 28.07.2018.
- FernUniversität in Hagen. Moodle – helpdesk.
<https://wiki.fernuni-hagen.de/helpdesk/index.php/Moodle>, 2018b. abgerufen am 28.07.2018.
- FernUniversität in Hagen. Daten-Zahlen-Fakten - Uni Intern - FernUniversität in Hagen.
<http://www.fernuni-hagen.de/arbeiten/statistik/daten/index.shtml>, 2018c. abgerufen am 01.06.2018.
- FernUniversität in Hagen. Virtueller Studienplatz – helpdesk.
https://wiki.fernuni-hagen.de/helpdesk/index.php/Virtueller_Studienplatz, 2018d. abgerufen am 28.07.2018.
- Google. Videos mit einem Abspann versehen - Computer - Hilfe für YouTube.
https://support.google.com/youtube/answer/6388789?visit_id=1-636685303009655859-2954709875&p=end_screens&hl=de&rd=1, 2018a. abgerufen am 28.07.2018.
- Google. Infokarten in Videos hinzufügen - Computer - Hilfe für YouTube.
<https://support.google.com/youtube/answer/6140493?hl=de>, 2018b. abgerufen am 28.07.2018.
- H5P. Interactive Video | H5P. <https://h5p.org/interactive-video>, 2013. abgerufen am 15.08.2018.
- H5P. wavesurfer.js. <https://wavesurfer-js.org/>, o.D. abgerufen am 15.08.2018.
- Michael Kerres und Thomas Jechle. Didaktische Konzeption des telelernens. *Information und Lernen mit Multimedia und Internet*, 3:267–281, 2002.
- R.E. Mayer. *Multimedia Learning*. Cambridge University Press, 2009. ISBN 9780521514125. URL <https://books.google.de/books?id=5g0AM1CHysgC>.
- Marshall McLuhan. Laws of the Media. *ETC: A Review of General Semantics*, pages 173–179, 1977.

- Moodle. Aufbau einer Moodle-Site – MoodleDocs. https://docs.moodle.org/35/de/Aufbau_einer_Moodle-Site, 2015a. abgerufen am 28.07.2018.
- Moodle. Block settings - MoodleDocs. https://docs.moodle.org/35/en/Block_settings, 2015b. abgerufen am 18.08.2018.
- Moodle. Was ist Moodle – MoodleDocs. https://docs.moodle.org/35/de/Was_ist_Moodle, 2015c. abgerufen am 28.07.2018.
- Moodle. Activity Modules - MoodleDocs. https://docs.moodle.org/dev/Activity_modules, 2016. abgerufen am 15.08.2018.
- Moodle. Media players - MoodleDocs. https://docs.moodle.org/dev/Media_players, 2017a. abgerufen am 18.08.2018.
- Moodle. Plugin types - MoodleDocs. https://docs.moodle.org/dev/Plugin_types#List_of_Moodle_plugin_types, 2017b. abgerufen am 28.07.2018.
- Moodle. About Moodle - MoodleDocs. https://docs.moodle.org/35/en/About_Moodle, 2018a. abgerufen am 28.07.2018.
- Moodle. Blocks - MoodleDocs. <https://docs.moodle.org/35/en/Blocks>, 2018b. abgerufen am 18.08.2018.
- Moodle. File API - MoodleDocs. https://docs.moodle.org/dev/File_API, 2018c. abgerufen am 15.08.2018.
- Moodle. Installing Moodle - MoodleDocs. https://docs.moodle.org/35/en/Installing_Moodle, 2018d. abgerufen am 18.08.2018.
- Moodle. Moodle.org: Moodle Statistics. <https://moodle.net/stats/>, 2018e. abgerufen am 28.07.2018.
- Daniel C Moos und Elizabeth Marroquin. Multimedia, hypermedia, and hypertext: Motivation considered and reconsidered. *Computers in Human Behavior*, 26(3):265–276, 2010.
- Theodor H Nelson. Complex information processing: a file structure for the complex, the changing and the indeterminate. In *Proceedings of the 1965 20th national conference*, pages 84–100. ACM, 1965.
- Jakob Nielsen. *Multimedia, Hypertext und Internet: Grundlagen und Praxis des elektronischen Publizierens*. Springer-Verlag, 2013.
- Donald A. Norman und Stephen W. Draper. *User Centered System Design; New Perspectives on Human-Computer Interaction*. L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 1986. ISBN 0898597811.
- Nurzhan Nurseitov, Michael Paulson, Randall Reynolds, und Clemente Izurieta. Comparison of json and xml data interchange formats: a case study. *Caine*, 9:157–162, 2009.
- Reinhard Oppermann. User-interface design. In *Handbook on information technologies for education and training*, pages 233–248. Springer, 2002.
- Rampl, Hansjörg. ISO 9241:10 Erwartungskonformität. <http://www.handbuch-usability.de/erwartungskonformitaet.html>, 2007. abgerufen am 30.07.2018.
- Gabi Reinmann-Rothmeier und Heinz Mandl. Kooperation. lernen im team. *Grundlagen der Weiterbildung*, 6(2): 65–68, 1995.
- Gabi Reinmann-Rothmeier und Heinz Mandl. Analyse und förderung kooperativen lernens in netzbasierten umgebungen. *Zeitschrift für Entwicklungspsychologie und Pädagogische Psychologie*, 34(1):44–57, 2002.
- Maren Schmidt-Kassow, Marie Deusser, Christian Thiel, Sascha Otterbein, Christian Montag, Martin Reuter, Winfried Banzer, und Jochen Kaiser. Physical exercise during encoding improves vocabulary learning in young female adults: a neuroendocrinological study. *PloS one*, 8(5):e64172, 2013.
- Sheila Scutter, Ieva Stupans, Tim Sawyer, und Sharron King. How do students use podcasts to support learning? *Australasian journal of educational technology*, 26(2), 2010.
- SoundCloud. Victorious von Panic! At The Disco | Kostenlos hören auf SoundCloud. <https://soundcloud.com/panicatthedisco/panic-at-the-disco-victorious>, 2015. abgerufen am 29.07.2018.
- SoundCloud. Info zu SoundCloud auf SoundCloud. <https://soundcloud.com/pages/contact>, o.D. abgerufen am 28.07.2018.
- Statista. Videoplattformen - Reichweite in Deutschland 2016 | Statistik.

<https://de.statista.com/statistik/daten/studie/209329/umfrage/fuehrende-videoportale-in-deutschland-nach-nutzeranteil/>, 2016. abgerufen am 28.07.2018.

Statistisches Bundesamt. Wie die Zeit vergeht - Ergebnisse zur Zeitverwendung in Deutschland - 2012/2013.
https://www.destatis.de/DE/PresseService/Presse/Pressekonferenzen/2015/zeitverwendung/Pressebroschüre_zeitverwendung.pdf?__blob=publicationFile, 2015. abgerufen am 27.07.2018.

Stephan Augsten. Was ist iterative Entwicklung?
<https://www.dev-insider.de/was-ist-iterative-entwicklung-a-707340/>, 2018. abgerufen am 19.08.2018.

Mark S Tremblay, Allana G LeBlanc, Michelle E Kho, Travis J Saunders, Richard Larouche, Rachel C Colley, Gary Goldfield, und Sarah Connor Gorber. Systematic review of sedentary behaviour and health indicators in school-aged children and youth. *International Journal of Behavioral Nutrition and Physical Activity*, 8(1):98, 2011.

I. Wild. *Moodle 3.x Developer's Guide*. Packt Publishing, 2017. ISBN 9781786469540. URL
<https://books.google.de/books?id=EHg5DwAAQBAJ>.

Ralf Wirdemann. *Scrum mit User Stories*. Carl Hanser Verlag GmbH Co KG, 2017.

Youtube. Panic! At The Disco: Victorious [OFFICIAL VIDEO] - YouTube.
<https://www.youtube.com/watch?v=AUCHk0lxF44>, 2015. abgerufen am 28.07.2018.

Joerg Zumbach und Christiane Kroeber. Learning with Hyperaudio: Cognitive Load and Knowledge Acquisition in Non-Linear Auditory Instruction. *Avoiding simplicity confronting complexity. Advances in studying and designing (computer-based) powerful learning environments*, pages 359–170, 2006.

Abbildungsverzeichnis

Schematischer Aufbau einer Moodle Seite	14
Kursseite des Kurses „Einführung in Mensch-Computer-Interaktion“ (FernUniversität in Hagen, 2016)	15
Benutzerrollen	21
Player der Musik- und Audio-Plattform <i>SoundCloud</i> (SoundCloud, 2015)	27
Player der Video-Plattform <i>Youtube</i> (Youtube, 2015)	27
Anzeige der Infokarte (Youtube, 2015)	28
Auszug aus einem <i>Interactive Video</i> (H5P, 2013)	29
Der Audio-Player auf der <i>wavesurfer.js</i> -Webseite(H5P, o.D.)	30
Zusammenhänge der Komponenten	33
ER-Diagramm der Datenbank des Moodle-Plugins	35
Benutzeroberfläche - Player	38
Benutzeroberfläche - Galerie	39
Benutzeroberfläche - Kommentarsektion	40
Benutzeroberfläche - Layout der Seite für Hyperaudio-Dokumente	42
Ordnerstruktur des Hyperaudio-Plugins	43
Verwendung von Moodle in den Pflichtmodulen des Bachelorstudiengangs Wirtschaftsinformatik (Sommersemester 2018)	61

Tabellenverzeichnis

Anforderungen der Administrierenden	23
Anforderungen der Nutzenden	25

Verzeichnis der Auflistungen

4.1 Beispielhafte Konfigurationsdatei	34
5.1 Ausschnitt der mod_form.php in der 1. Iteration	45
5.2 Ausschnitt der lib.php in der 1. Iteration	45
5.3 Ausschnitt der locallib.php in der 1. Iteration	46
5.4 Ausschnitt der view.php in der 1. Iteration	46
5.5 Ausschnitt der renderer.php in der 1. Iteration	46
5.6 Ausschnitt der locallib.php in der 2. Iteration	47
5.7 Ausschnitt der view.php in der 2. Iteration	48
5.8 Ausschnitt der renderer.php in der 2. Iteration	48
5.9 Ausschnitt der locallib.php in der 3. Iteration	49
5.10 Ausschnitt der renderer.php in der 4. Iteration	50
5.11 services.php in der 5. Iteration	51
5.12 external.php in der 5. Iteration	51
5.13 Ausschnitt der comments.js in der 5. Iteration	53
5.14 Ausschnitt der comments.js in der 6. Iteration	54
5.15 Ausschnitt der external.php in der 6. Iteration	55
5.16 Ausschnitt der renderer.php in der 7. Iteration	55
5.17 Ausschnitt der services.php in der 7. Iteration	55
5.18 Ausschnitt der comments.js in der 7. Iteration	56
5.19 Ausschnitt der popcorn.image.js in der 8. Iteration	56

Verzeichnis der Algorithmen

Abkürzungsverzeichnis

AJAX	Asynchronous JavaScript and XML
AMD	Asynchronous Module Definition
API	Application Programming Interface
BAT	BMAT Annotation Tool
CSS	Cascading Style Sheets
DML	Data Manipulation Language
HTML	HyperText Markup Language
JSON	JavaScript Object Notation
LMS	Learningmanagementsystem
LVU	Lernraum Virtuelle Universität
Moodle	Modular Object-Oriented Dynamic Learning Environment
PHP	PHP: Hypertext Preprocessor
URL	Uniform Resource Locator
VU	Virtuelle Universität
XML	Extensible Markup Language

Eidesstattliche Erklärung

Ich erkläre hiermit, dass ich diese Bachelorarbeit selbstständig verfasst, noch nicht anderweitig für Prüfungszwecke vorgelegt, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe.

Hagen, den

Datum

Michael Lämmermann