communicated by:
Lehr- und Forschungsgebiet Informatik 9

Prof Dr. Ulrik Schroeder

Diese Arbeit wurde vorgelegt am Lehr- und Forschungsgebiet Informatik 9
The present work was submitted to Learning Technologies Research Group

Webbasiertes Analysetool zum Ersetzen von Medienelementen

Web-based analytical tool for replacing media content

Bachelorarbeit
Bachelor-Thesis

von / presented by

Aufdermauer, Patrick
356057

Lubna Ali Msc.

Prof. Dr.-Ing. Ulrik Schroeder

Prof. Dr. rer. nat. Horst Lichter

Aachen, 15. Mai 2020

Inhaltsverzeichnis

I. Einleitung & Einführung **5**

1. Einleitung und Einführung **7**
1.1. Idee von freier Bildung 7
1.2. Warum wird die Idee von OER verfolgt? 8
1.3. Verfügbare Werkzeuge 9
1.4. Motivation und Ziel dieser Arbeit 10

2. Über Open Educational Ressources (OER) **11**

II. Bedienung des Tools **13**

3. Bedienung **15**

III. Technologien & Technische Umsetzung **25**

4. Versionverwaltung mit Git **27**

5. Angular Frontend **29**
5.1. Allgemein 29
5.2. Konfigurationsdateien 30
5.3. Klassen 31
5.4. Services 31

6. Express Backend **33**
6.1. Allgemein 33
6.2. Klassen 33
6.3. Anforderungen an den Datenbankadapter 34

7. MySQL Datenbank **39**
7.1. Warum MySQL? 39
7.2. Theoretische Struktur und Semantik 39
7.2.1. Struktur 40
7.2.2. Tabellen 43
7.3. SQL-Queries für MySQL Lösung 44

8. Office OpenXML Format **49**
8.1. Folie hinzufügen 49

8.2. Referenz zur Quelle neben das Bild setzen 58
8.3. Text mit Hyperlink erstellen 61
8.4. Kommentar hinzufügen 62
8.5. Arbiträre Daten einbetten 66
8.5.1. Vorbereitung 67
8.5.2. Daten speichern 67

9. Docker **69**
9.1. Frontend Container 69
9.2. Backend Container 70
9.3. Datenbank Container 71
9.4. Komposition mit *docker-compose.yml* 71

IV. Konklusion **73**

10. Konklusion **75**

Anhänge **77**

A. Abkürzungen **77**

B. Literaturverzeichnis **79**

Abstrakt

Open Educational Ressources (OER) beschreibt eine Idee von *freier* Bildung. *Frei* bedeutet in diesem Kontext, dass Bildungsmaterialien gewisse lizenzrechtliche Bedingungen erfüllen müssen. Dies führt zu einer Rechtsunsicherheit bei Menschen, die OER erstellen. Überdies bedeutet das Erstellen von OER einen großen Aufwand, da die Lizenzen der verwendeten Inhalte geprüft und korrekt zitiert werden müssen. Dies ist auch nötig, wenn bereits kreiertes Material überarbeitet werden soll. Dies würde also nicht viel Zeit oder Arbeit sparen. Es gibt Werkzeuge, die dabei helfen, OER Materialien von Grund auf neu zu erstellen. Eine Möglichkeit, bestehendes Lehrmaterial möglichst automatisiert zu konvertieren, existiert nicht und wäre an dieser Stelle wünschenswert.

Daher soll im Rahmen dieser Bachelorarbeit eine Webapplikation (im folgenden Tool genannt) entwickelt werden, welches einen Benutzer dabei unterstützen soll, PowerPoint-Folien mit möglichst wenig Eingriffen von seiner Seite zu konvertieren. Zunächst wird die Idee von OER ein wenig vertieft. Dann wird die Bedienung des entwickelten Tools und seiner Funktionen erläutert. Darauf folgt eine technische Erklärung zu den verwendeten Programmen, Konzepten und Technologien während der Entwicklung. Danach werden Einschränkungen, die während der Entwicklungsphase entdeckt wurden, aufgezeigt und Ausblicke auf zukünftig mögliche Funktionen gegeben. Zuletzt setzte ich mich damit auseinander, inwieweit das von mir entwickelte Tool bereits dem Anspruch einer echten Unterstzützung gerecht würde.

Abstract

Open Educational Ressources (OER) is an idea of *free* education. In this context, *free* means that there are certain requirements to the legal aspects of materials that have been published for the purpose of education. Meeting those requirements is a struggle for people that want to create OER-materials but are uncertain about how to abide by the legal obligations. Additionally, creating material that is OER-conform usually comes with a huge load of work, because all the licenses of other people's content that is used need to be checked for conformity with the OER requirements. Furthermore, they also ought to be cited in a correct and accepted manner. Even if the materials are ones that already exist, checking the licenses and correctly citing the owner would also be mandatory. So it would not save much time nor really decrease the amount of work that needs to be done. There are some tools available that aid in creating materials initially suited for OER. A possibility to, as much as possible, automatically convert existing materials to make them meet the OER requirements does not seem to be around, and would be very much appreciated.

Hence, in this bachelor thesis, we will develop a web application (hereafter referred to as tool) which supports a user when converting a PowerPoint presentation, with as little intervention by the user as possible. First, I will talk about some more details regarding OER. Then I will guide through the use of the tool and explain its functions. Next, the technical details of the implementation, concepts, and technologies will be provided. Thereafter, I will discuss limitations that came up during development and I will give a brief prospect of functionalities that will likely be added in the future. In a final step, I will inspect how well the developed tool copes with arbitrary learning material and in what degree it satisfies my own requirements to assist people in creating OER materials, when compared to some other available tools.

Teil I

Einleitung & Einführung

Kapitel 1 Einleitung und Einführung

1.1. Idee von freier Bildung

Open Educational Ressources (OER) (dt. freie, offene Bildungsmaterialien) können helfen, den Menschen auf der Welt den Zugang zu hochwertigen und freien Bildungsmaterialien zu erleichtern. Nach Wiley et al. müssen OER-Ressourcen so lizenziert sein, dass die folgenden fünf Anforderungen erfüllt werden [64, 82]. Sie werden auch "The 5 R's of Openness" [64] genannt:

- Retain (dt. aufbewahren):
  Es muss erlaubt sein, sich Kopien anzulegen
- Reuse (dt. Wiederverwendung):
  Es muss erlaubt sein, den Inhalt zu benutzen, zu verwenden und jemandem zu zeigen, z.B. den Studierenden einer Vorlesung
- Revise (dt. ändern):
  Es muss erlaubt sein, den Inhalt anzupassen oder zu verändern, z.B. durch Übersetzen in eine andere Sprache
- Remix (dt. neu mischen):
  Es muss erlaubt sein, den Inhalt (oder den bereits veränderten Inhalt) mit anderen Inhalten neu zu kombinieren, um auf diese Weise einen neuen Inhalt zu erzeugen
- Redistribute (dt. weiterverbreiten):
  Es muss erlaubt sein, den originalen Inhalt, den veränderten Inhalt oder neu zusammengestellten Inhalt zugänglich zu machen (so, dass eine echte Kopie angefertigt werden kann, siehe *Retain*)

Darüber hinaus wird im deutsch-sprachigen Raum unter OER noch folgendes verstanden [78]:

- Sie sind kostenlos verfügbar
- Sie können ohne Gebühren benutzt werden (also geändert, zusammengefügt und weiterverbreitet werden)
- Sie sind mithilfe kostenloser, freier Software einsehbar und änderbar
- Sie unterstützen freies, offenes Lehren und Lernen

Darüber hinaus hebt Steven Ovadia in [84] die Wichtigkeit offener Dateiformate für die Zukunft von OER hervor. Offene Dateiformate können gewährleisten, dass der Inhalt verändert beziehungsweise geöffnet und damit angesehen werden kann. Dann können Hersteller von Software mit proprietären Speicherformaten nicht dafür sorgen, dass nur noch deren Software die gespeicherten Informationen lesen und bearbeiten können.

Als Beispiel lässt sich hier das *.docx* Format von Microsofts Textverarbeitungsprogramm *Word* heranziehen. Als dieser Text verfasst wurde, können auch freie nicht-Microsoft Anwendungen wie z.B. *LibreOffice* [70] Dokumente im *.docx* Format öffnen. Ginge nun Microsoft hin und unterbände das öffnen von *.docx* Dateien durch Programme, die nicht von Mircosoft stammen oder keine Lizenzierung gewählt haben, könnten Benutzer, die vorher *LibreOffice* genutzt hatten, ihre Dateien nun nicht mehr öffnen, ansehen oder bearbeiten. Daher sind offene Lizenzen nach dem Vorbild von "The 5 Rs of Openness" [64] nicht genügend, wenn das Bearbeiten der Dateien technisch nicht problemlos möglich ist bzw. nicht gewährleistet wird, dass dies auch in der Zukunft der Fall ist.

Diese Anforderungen stellen Lehrende an Hochschulen, die ihre Lernmaterialien OER-konform bereitstellen wollen, vor Herausforderungen. Daher fällt ihnen das Erstellen und Nutzen von OER schwer [75]. Eine dieser Herausforderungen ist das erstellen neuer OER Materialien aus bereits bestehendem Lehrmaterial, anstatt OER-konforme Lehrinhalte von Grund auf neu zu erstellen. Dies wird in Sektion 1.3 aufgegriffen werden.

1.2. Warum wird die Idee von OER verfolgt?

Nach Yuan et al. [86] befassen sich Institutionen unter anderem aus den folgenden Gründen mit OER:

- Das Teilen von Wissen entspricht der wissenschaftlichen Tradition
- Institutionen, besonders die von Steuerzahlern finanzierten, sollten das staatliche Geld zu Ihrem Vorteil nutzen, indem sie ihre Materialien unentgeltlich zur Verfügung stellen und auch zur Arbeit damit motivieren
- Durch Teilen mit anderen und durch Wiederverwendung durch andere kann die Qualität erhöht werden
- Es ist gut für das öffentliche Bild der Institutionen, wenn Sie sich an OER Projekten beteiligen, was auch neue Studierende anlockt
- Freies Teilen der Materialien beschleunigt die Erstellung von neuen Lernmaterialien und verschafft der Einrichtung einen guten Überblick über intern oder extern eingesetzes Material

Überdies legt der Bericht auch Anreize von Privatpersonen im akademischen Umfeld dar, sich mit OER zu beschäftigen [86], unter anderem:

- Auch sie finden, dass das Teilen von Wissen im Einklang mit der wissenschaftlichen Tradition steht
- Persönliche, nicht-monetäre Profite, wie z.B. höheres Ansehen und eventuell damit einhergehend ein "Egoboost" [sic] [86]

Zusätzlich kann ein Angebot an OER-Plattformen neue Studierende anlocken und zur Wahl der Universität beitragen. So gaben Studierende am Massachusetts Institute of Technology (MIT) bekannt, dass das dort online angebotene OpenCourseWare (OCW)-Projekt einen signifikanten Einfluss auf ihre Institutswahl hatte [76, 80].

1.3. Verfügbare Werkzeuge

Zurzeit existieren verschiedene Werkzeuge, die Lehrenden bei der Erstellung von OER-Materialien unterstützen, z.B. Tutory [66], H5P [28] oder ein Lizenhinweisgenerator [40].

Tutory ist ein Generator für Arbeitsblätter und unterstützt Lehrkräfte beim erstellen von eben solchen. Der Dokumenteneditor basiert auf einem Bausteinsystem, so dass sich Bilder, Absätze oder Aufgaben einzeln Erstellen und auf Arbeitsblätter verteilen lassen. Außerdem können Lehrende so rechtssicher arbeiten, da die von Tutory bereitgestellen Inhalte unter einer freien Lizenz zur Verfügung stehen. Zum Beispiel können die Bilder aus den frei Lizenzierten Werken von Openclipart, Flickr, Pixabay und der Wikipedia (bzw. den Wikimedia Commons) gewählt werden. Es ist auch möglich, verschiedene Schwierigkeitsgrade eines Blattes zu erstellen, um so besser auf die Leistungsstände der Unterwiesenen einzugehen. Diese Dokumente können auf Tutory-Servern gespeichert werden, so dass alle Dokumente an einem Ort zu finden sind. Eine Volltextsuche wird auch angeboten, mit der alle in Tutory gesicherten Dokumente durchsucht werden können.
Obgleich Tutory den Lehrenden ein starkes Werkzeug zur Dokumenterzeugung an die Hand gibt und zugleich auch noch Rechtssicherheit schafft, besteht das Problem, dass sich mit dieser Applikation nur Arbeitsblätter von Grund auf neu erstellen lassen. Konvertieren von bereits bestehendem Material wäre also nur möglich, wenn es komplett nachgebaut werden würde. Erstellen von Folien, ähnlich wie z.B. PowerPoint-Folien, ist gar nicht möglich.

H5P ist ein HTML5 Anwendungsgenerator. Mithilfe von H5P können Lehrende interaktive Applikationen entwickeln, wie z.B. ein Videoquiz, bei dem ein laufendes Video pausiert und eine Frage eingeblendet wird. Auch hier wird ein Baukastensystem angeboten. Zur Auswahl stehen verschiedene Typen der Anwendung (z.B. Tutorial oder Beispiel) welche dann aus Inhalten wie Mehrfachauswahlfragen, Lückentexten oder Präsentationen bestehen können. Am Ende können die so erstellen Applikationen dann auf einem H5P-Server gehostet werden. Es ist aber auch möglich, die Applikation als Plugin für WordPress, Drupal oder Moodle zu exportieren.
Im Gegensatz zu Tutory ist keine Rechtssicherheit gegeben, da die genutzten Bilder und Videos dem Benutzer überlassen sind und auch aus nicht-freien Quellen (bzw. zu nicht-freien Zwecken) eingebunden werden können. Außerdem setzt das hosten von H5P-Applikationen auf einem eigenen Server via WordPress, Drupal oder Moodle eine gewisse Expertise voraus, welche nicht von jeder Lehrkraft verlangt werden kann. Letztendlich hat H5P das gleiche Problem, wie Tutory: konvertieren von Lehrmaterial, welches bereits existiert, ist nur indirekt möglich, indem auf hier wieder das ganze Dokument oder die ganze Präsentation nachgebaut wird.

Der Lizenhinweisgenerator erzeugt, wie der Name es bereits vermuten lässt, zu einem Bild die Hinweise gemäß der TULLU-Regel [50]. Wenn dem Generator der Verweis zu einem Wikipediaartikel übergeben wird, werden die in dem Artikel verwendeten Bilder angezeigt. Nach Auswahl eines Bildes wird der Benutzer durch einen kurzen Prozess geführt. Hierin wird gefragt, ob das Bild Online oder in einem Printmedium veröffentlicht wird, ob das Bild einzeln stehen soll oder Teil einer Sammlung ist und ob das ausgewählte Bild noch bearbeitet wird oder in seiner ursprünglichen Form verwendet werden soll. Nachdem dieser Prozess abgeschlossen ist, wird der Hinweistext zur Lizenz in den Varianten *Text* (wie er in einem Browser gerendert würde), Plain Text (z.B. für Auszeichnungssprachen) oder als HTML-Quellcode (zur Einbettung in Webseiten) angezeigt. Zu

Bildern, welche nicht in den Wikimedia Commons liegen, kann kein Hinweistext erzeugt werden.
Der Lizenzhinweisgener unterscheidet sich zu Tutory und H5P in der Weise, dass der Lizenzhinweisgenerator keine Dokumente erstellt. Er dient lediglich als Unterstützung, wenn manuell unfreie Dokumente befreit oder freie Dokumente erstellt werden. Dass nur Bilder verwendet werden können, welche in der Wikimedia Commons liegen, ist ebenfalls eine große Einschränkung.

Deshalb ist es aktuell nicht möglich, bestehendes Lehrmaterial automatisiert in OER-konforme Dateien zu konvertieren. Stattdessen müssen die Lizenzen von Mediendateien händisch geprüft und Dateien gegebenenfalls manuell ersetzt werden. Es könnten auch die genannten Werkzeuge genutzt werden, hier muss jedoch z.B. eine komplette PowerPoint-Präsentation nachgebaut werden.
Hier fehlt bisher ein Werkzeug, welches Lehrdateien, die bereits existieren und genutzt werden, in eine OER-konforme Version konvertiert.

1.4. Motivation und Ziel dieser Arbeit

Es wurden Gründe für die Verwendung von OER-Material aufgeführt, aber der Arbeitsaufwand und die Rechtsunsicherheit können abschreckend wirken.

Um Menschen zu Unterstützung, die Lehrmaterialien konvertieren möchten, soll ein webbasiertes Analysetool entwickelt werden, welches eine Datei mit Lehrmaterial einliest und Bildelemente sowie mögliche Metadaten verarbeitet. Nicht OER-konforme Abbildungen sollen durch geeignete Alternativen ersetzt werden. Das Webtool soll möglichst automatisch funktionieren und aus bereits bestehendem Lehrmaterial ein OER konformes Ergebnis erstellen. Dazu muss das Tool in der Lage sein, auf die verwendeten Bilder zuzugreifen und diese bei Bedarf auszutauschen.
Zur Demonstration der Machbarkeit soll das Tool eine Powerpoint-Präsentationsdatei im Office Open Extensible Markup Language (OOXML) Format [19, 52] entgegennehmen und diese verarbeiten. Es müssen die Folien manipuliert und neue Folien angehängt werden können. Das Resultat soll schließlich wieder als PowerPoint-Datei im OOXML Format gespeichert werden können.
Das Tool soll aus einem Backend und einem Frontend bestehen. Das Frontend wird die Anwendung sein, mit der der Benutzer interagiert und welches das Konvertieren der Lehrmaterialien vornimmt. Das Backend zeichnet auf, welche konkreten Substitutionsoperationen stattfinden, also welche ursprünglichen Bilder in der Präsentation durch passende Bilder aus den zur Verfügung stehenden Quellen ersetzt und welche Bilder als OER deklariert werden. Damit diese Daten gespeichert werden können, wird eine persistente Speicherlösung benötigt, welche durch eine MySQL Datenbank realisiert werden soll. Später kann auf diese gespeicherten Daten zurückgegriffen werden, damit weiteres Lehrmaterial schneller konvertiert werden kann.

Kapitel 2 Über OER

Im Jahr 1998 prägte David Wiley den Begriff *open content* (zu dt. *Offener Inhalt*) als Analogon zu der *open source/free software* (sinngemäß *Offener Quellcode/Freie Software*) Bewegung und wollte damit zeigen, dass sich jene Prinzipien auch im Bereich der Bildung einbetten lassen [85]. Das Konzept von *Offen(-heit)* basiert auf der Idee, dass Wissen mittels des Internets verbreitet und verteilt werden sollte, sodass es letztlich allen Menschen zugute kommt [86].
Ebenfalls angetan von den Ideen der *Freien Software*-Bewegung [76], gab das MIT 2001 einen 10-Jahres-Plan bekannt [81]. Das MIT rief damit das OpenCourseWare (OCW)-Projekt ins Leben. Es war das erklärte Ziel, praktisch alle am MIT angebotenen Kurse und deren Materialien unentgeltlich zur Verfügung zu stellen [74]. Davon erhoffte sich die Institution unter anderem auch einen positiven Einfluss auf die Studierenden vor Ort. Tatsächlich gaben 96% der Studierenden an, welche das MIT OCW nutzten, einen positiven oder extrem positiven Effekt auf ihr Studienerlebnis gespürt zu haben [76]. Für viele Studierende war der Einfluss des MIT OCW auf ihre Universitätswahl sogar maßgeblich [76, 80].
Das MIT OCW-Projekt zog mit seinen Bemühungen um freie Bildung Aufmerksamkeit auf sich, doch auch andere Institutionen leisteten ihren Beitrag. So veröffentlichte Creative Commons (CC) seinen ersten Satz an freien Lizenzen in 2002 [76] und die *William and Flora Hewlett Foundation* bemühte sich um finanzielle Hilfen für OER-Projekte [74, 76, 80] und avancierte dabei zum größten Antreiber der OER-Bewegung [74]. Die *William and Flora Hewlett Foundation* sponserte u.a. das United Nations Educational, Scientific and Cultural Organization (UNESCO) *Forum on the impact of Open Courseware for Higher Education in Developing Countries* (dt. *Forum über die Auswirkungen von Open Courseware auf höhere Bildungsinstitutionen in Entwicklungsländern*) [80]. Dort drückten Teilnehmer ihren Wunsch aus, gemeinschaftlich eine universale Bildungsresource zu entwickeln, die der Menschheit als ganzes zugute kommt und als *Offene Bildungsmaterialien* (Open Educational Ressources (OER)) bezeichnet werden solle, als Anlehnung an *Offene Inhalte* (*open content*) [80].
Das MIT OCW-Projekt inspirierte auch andere Insitutionen, sich mit ähnlichen OER-Projekten zu befassen. So existiert heute das *Connexions Project*, welches sich auf eine weltweit aktive Community stützt und im Jahr 2007 ca. 3.755 Module in 197 Kursen in verschiedenen Sprachen, darunter Englisch, Chinesisch, Italienisch, Japanisch, Portugisisch, Spanisch und Thai, zugänglich machte [74]. Aber auch andere Universitäten wie die Utah State University (USU) machten große Teile ihrer Vorlesungen und Module öffentlich zugänglich und ist heute ebenfalls ein großer Vorantreiber der Idee von freier Bildung [74].

Um der stetig wachsenden Zahl an OCW-Projekten Herr zu werden und diese in gewissen Weisen zu vereinheitlichen, wurde das OpenCourseWare (OCW) Consortium konstituiert, welches nun über 120 Mitglieder zählt und Institutionen unter anderem bei der Realisation und Pflege von OCW-Projekten hilft und zusätzlich versucht, andere Institutionen von der Idee und den Vorzügen freien Wissens in Form von OER zu überzeugen [74].
Daneben gibt es noch weitere Projekte, die sich den Zielen der OER-Bewegung verschrieben haben. Darunter ist zum Beispiel das *Internet Archive* [34], welches 1996 ins Leben gerufen wurde. Erst als reines Webarchiv konzipiert, welches es ermöglicht, Webseiten in einem Zustand aus der Vergangenheit zu betrachten, führt es heute auch eine Sammlung über Bücher, Audiodateien, Videos, Bildern und Software [33] und setzt sich seinerseits zur Erhaltung des freien Internets und der Verbreitung gemeinfreier Werke ein [8]. Die Internetenzyklopädie *Wikipedia* [69] oder das frei zugängliche Literaturarchiv für gemeinfreie Werke *Project Gutenberg* [51] sind weitere namhafte Verfechter der Idee von OER.

Teil II

Bedienung des Tools

Kapitel 3 Bedienung

In diesem Kapitel wird anhand einer Beispielpräsentation die Bedienung des Tools erläutert. Die Beispielpräsentation lässt sich hier [39] herunterladen. Wir gehen dabei Schritt für Schritt vor. Zum Zeitpunkt dieser Ausarbeitung liegt das Frontend des Tools in der Version 1.1.1 vor. Die hier dargestellen Abbildungen können mit einer älteren Version des Tools erstellt worden sein (z.B. 1.1.0, wie in Abb. 3.3 deutlich erkennbar). Die neueren Versionen beinhalten lediglich Bugfixes und ändern nichts an der Bedienung des Tools.

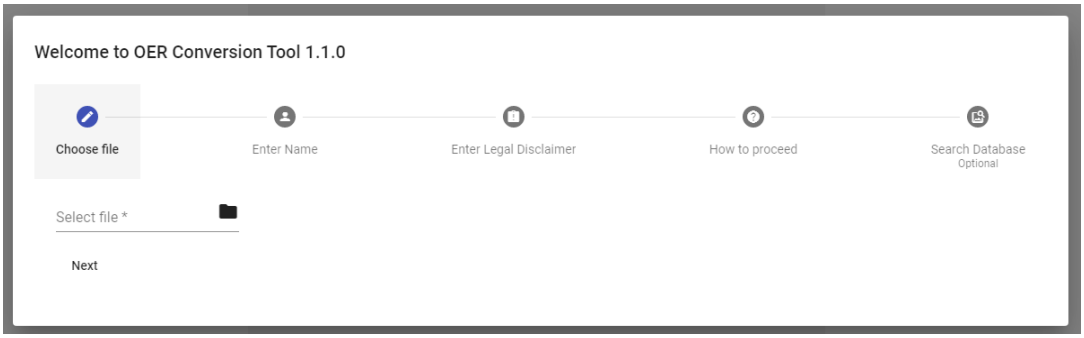

Abbildung 3.1.: Dialog Nach dem ersten Aufruf, welches im ersten Schritt eine Präsentationsdatei entgegen nimmt

Hier [49] lässt sich die Anwendung ausprobieren. Wenn das Tool zum ersten mal aufgerufen wird, präsentiert es sich wie in Abbildung 3.1. Es erscheint eine Dialogbox, in welcher wir durch mehrere Schritte geführt werden. Im ersten Schritt wird eine Schaltfläche angeboten, mit der eine Präsentationsdatei gewählt werden kann. Dazu klickt man auf das Ordnersymbol oder den freien Platz links davon. Dann öffnet sich das Dateiauswahlfenster des Betriebssystem und es kann eine Datei gewählt werden. Dieses Fenster filtert Dateien heraus, welche nicht auf *.pptx* enden, wie in Abbildung 3.2 gezeigt.

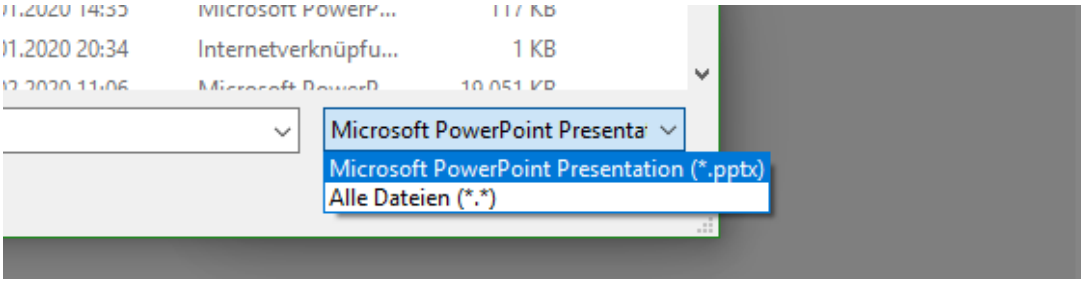

Abbildung 3.2.: Wenn unterstützt, blendet der Dateibrowser nicht unterstützte Dateien aus

Im nächsten Schritt können wir einen Namen wählen (Abbildung 3.3). Dieser ist standardmäßig mit *unknown* ausgefüllt. Der Name wird in der Vorlage des Hauftungsausschlusses verwendet. Er wird auch verwendet, wenn wir später ein Bild bearbeiten, um der TULLU-Regel [50] gerecht zu werden. Außerdem kann dieser Name automatisch als Autor eines Bildes genommen werden, wenn das Bild als OER deklariert wird.

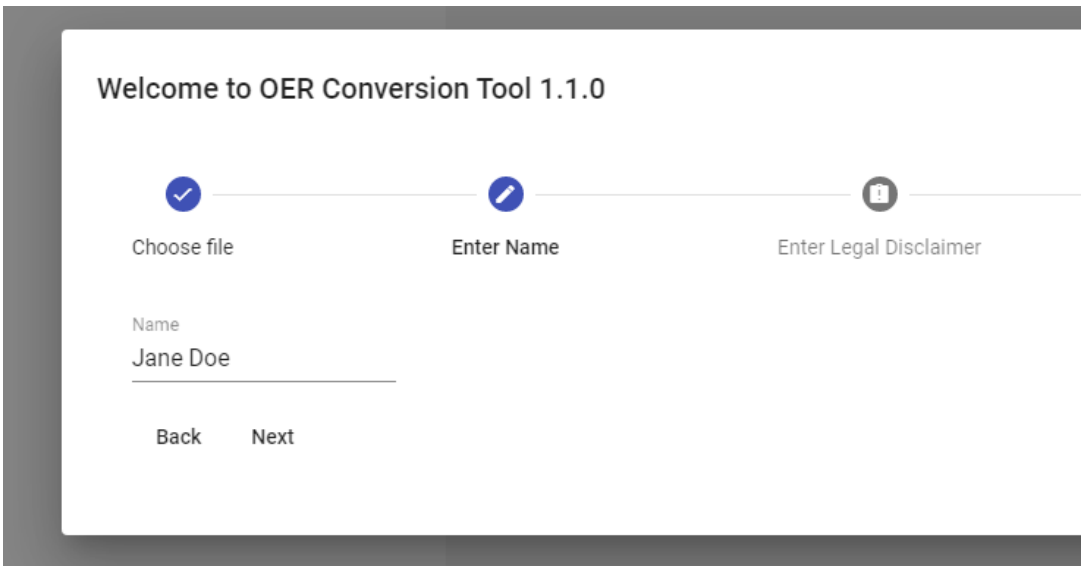

Abbildung 3.3.: Der Name wird später an mehreren Stellen eingesetzt werden

Im darauffolgenden Schritt wird der Text für den Haftungsauschluss eingegeben (siehe Abbildung 3.4). Dabei ist das Textfeld mit einer Vorlage ausgefüllt, in die der Name des ursprüngliche Erstellers, unser im vorigen Schritt gewählter Name und der Name sowie die Version des Tools eingefügt werden. Dieser Text kann nach belieben verändert werden und dient lediglich als grobe Idee, wie ein Hauftungsausschlusstext lauten könnte.

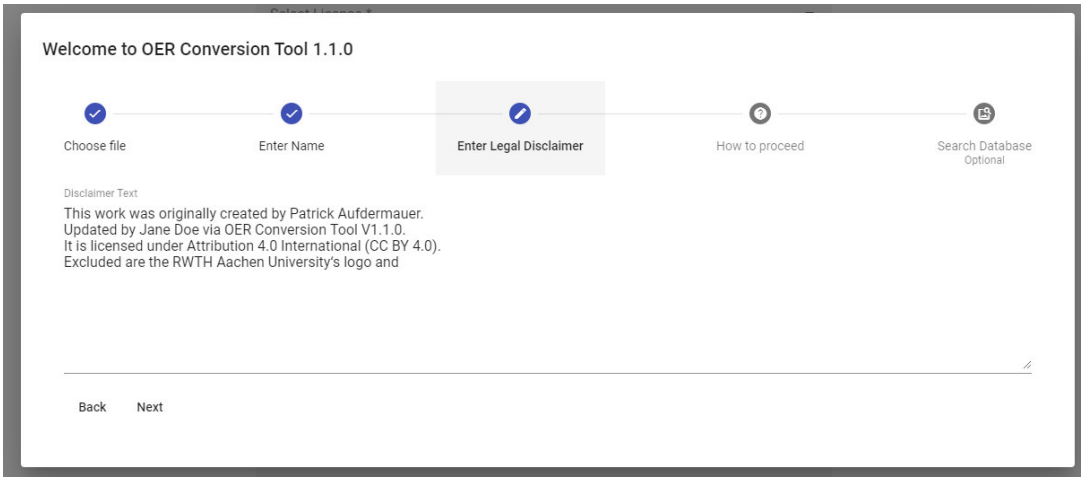

Abbildung 3.4.: Dieser Text wird auf der Haftungsausschlussfolie am Ende der konvertierten Präsentation eingefügt

Danach werden wir gefragt, ob die Datenbank nach den Bildern aus der Präsentation durchsucht werden soll, wie in Abbildung 3.5 zu sehen. Dieser Vorgang ist optional. Durch einen Klick auf die Schaltfläche *No, thanks* gelangen wir zur Standardansicht des

Tools. Möchte man das Angebot, die Datenbank zu durchsuchen, in Anspruch nehmen, führt ein Klick auf den Knopf *Yes, search database* zum nächsten Schritt.

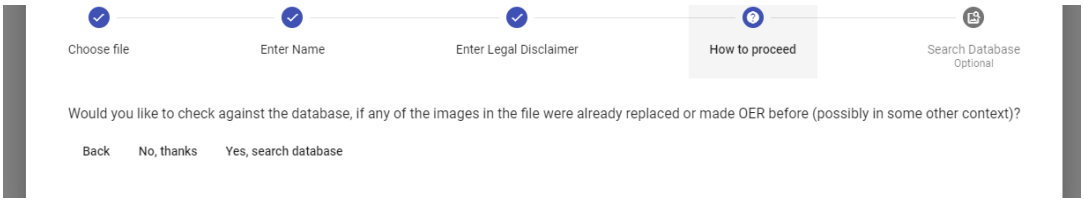

Abbildung 3.5.: Das durchsuchen der Datenbank ist optional und muss nicht vollzogen werden

Wenn die Datenbank durchsucht wird, werden die Checksummen der in der Präsentation enthaltenen Bilder mit den in der Datenbank gespeicherten Operationen verglichen. Es wird geprüft, ob ein Bild bereits in einer anderen Datei durch ein neues Bild ersetzt wurde oder ob dieses Bild zuvor schon als OER deklariert wurde. Wenn das der Fall ist, öffnet sich beim Klick auf ein Bild, welches der Datenbank bekannt ist, ein Dialogfenster (siehe dazu die Abbildungen 3.12 und 3.13). In diesem Dialogfenster werden Bilder vorgeschlagen, durch die das angeklickte Bild in der Vergangenheit bereits ersetzt wurde. Falls dieses Bild vorher bereits als OER deklariert wurde, werden hier die Daten für eine erneute Deklarierung als OER vorgeschlagen. In Abbild 3.6 ist der Vorgang beendet und ein Klick auf die Schaltfläche *Okay!* führt zur Standardansicht des Tools.

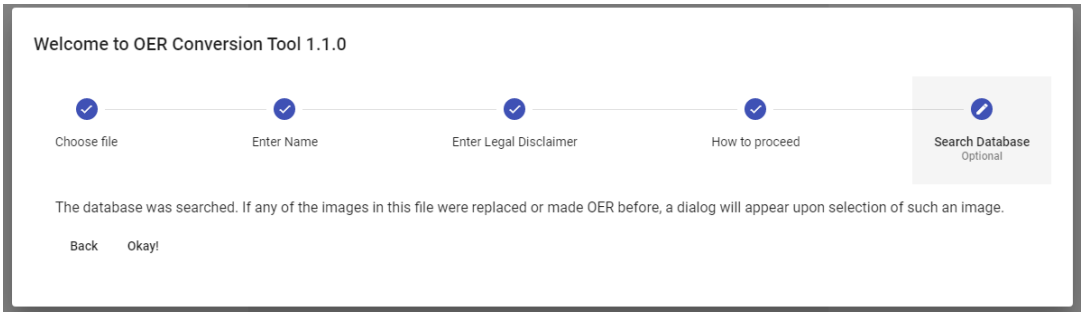

Abbildung 3.6.: Die Datenbank wurde fertig durchsucht

Nachdem man zur Standardansicht gelangt ist, werden die Bilder in den unterstützten Bildformaten JPEG, Portable Network Graphics (PNG) und Graphics Interchange Format (GIF) in der linken Spalte angezeigt (Abbildung 3.7).

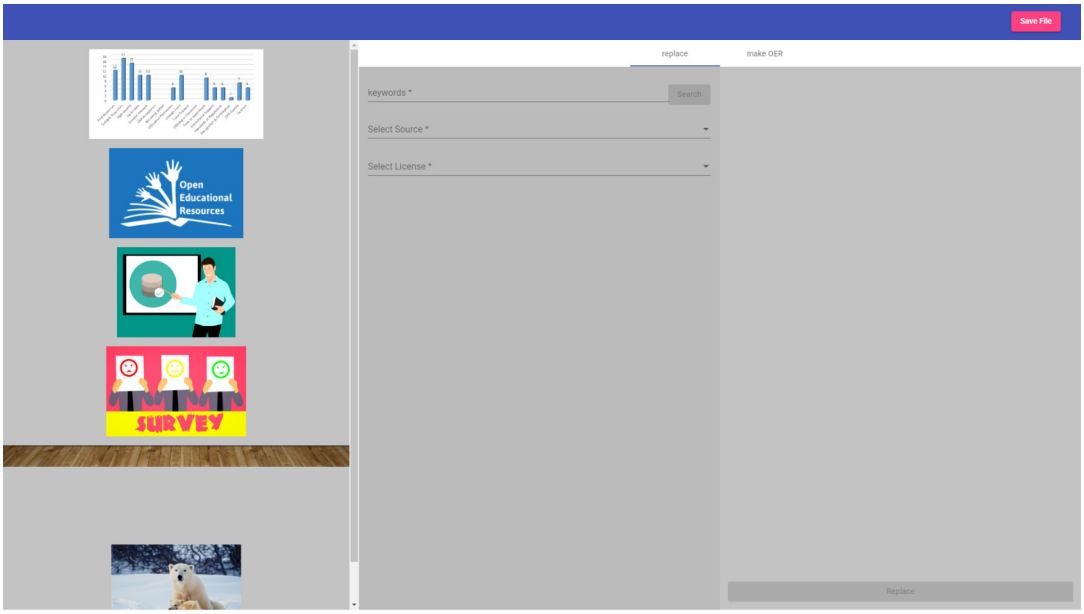

Abbildung 3.7.: Die extrahierten Bilder aus der Beispielpräsentation

Mit einem Klick auf ein Bild erscheint es in vergrößerter Ansicht in der rechten Spalte. In der mittleren Spalte befindet sich das Suchfeld für die Suche mithilfe externer Suchmaschinenangebote.

Bisher werden die *CC Image Search* [9, 14] und die *Flickr Public Image Search* [15, 63] unterstützt. Jeweils eine der beiden Suchmaschinen kann im Aufklappmenü *Select Source* (das zweite Element von oben in der mittleren Spalte in Abbildung 3.7) ausgewählt werden.
Die *CC Image Search* wurde gewählt, da sie sich vieler verschiedener Quellen bedient [10], wie z.B. den *Wikimedia Commons* [68], *DeviantArt* [21] oder dem *Metropolitan Museum of Art* [29].
Obwohl *Flickr* ebenfalls von der *CC Image Search* durchsucht wird, wurde es als Backup implementiert, falls die *CC Image Search* nicht erreichbar sein sollte. Eine Situation, die während der Implementierung auftrat.
Da es sich bei dem Tool nur um eine Demonstration handelt, wurde auf das Hinzufügen weiterer Möglichkeiten verzichtet.

In das oberste Eingabefeld *keywords* werden die Suchbegriffe eingegeben, mit denen nach Bildern gesucht werden soll. Im Aufklappmenü *Select License* (das dritte Element in der mittleren Spalte in Abb. 3.7) kann die Lizenz gewählt werden, unter der die Ergebnisse veröffentlicht worden sein sollen.
Zuerst werden wir nun das Eisbärenbild durch ein OER-konformes Pendant ersetzen. Wir wählen als Suchmaschine die *CC Image Search* und als Lizenz *CC-BY*.

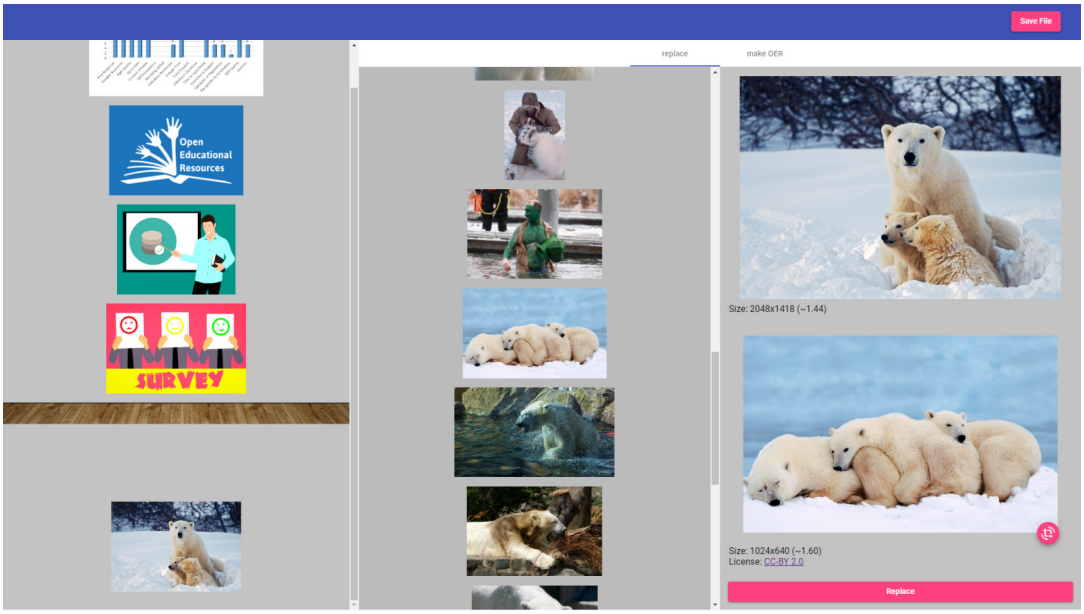

Abbildung 3.8.: In der rechten Spalte werden die beiden Bilder gegenüber gestellt

In Abbildung 3.8 ist oben rechts das Eisbärenbild aus der Präsentation zu sehen, darunter ein Bild welches wir aus den Ergebnissen der *CC Image Search*-Suche ausgewählt haben. Jeweils unter den Bildern sind weitere Informationen eingeblendet. Unter dem Bild, welches ersetzt wird, ist die Auflösung sowie das Seitenverhältnis zu sehen. Unter dem Bild, welches ersetzen wird, ist neben der Auflösung und dem Seitenverhältnis auch noch die verwendete Lizenz zu sehen, unter der das Bild veröffentlicht wurde. Ein Klick auf den Namen der Lizenz führt zu den dazugehörigen Lizenzbedingungen.
In unserem Beispiel besitzt das Bild aus der Präsentation ein Seitenverhältnis von ungefähr 1,44, während das Bild, welches wir stattdessen nutzen wollen, ein Seitenverhältnis von ungefähr 1,6 aufweist. Klicken wir nun auf die Schaltfläche *Replace*, so öffnet sich ein Dialogfenster (Abbildung 3.9), da der Unterschied zwischen den Seitenverhältnissen die Toleranz überschreitet (sie ist zurzeit durch Konfigurationsdateien auf dem Server festgelegt) und macht uns auf die Problematik aufmerksam.

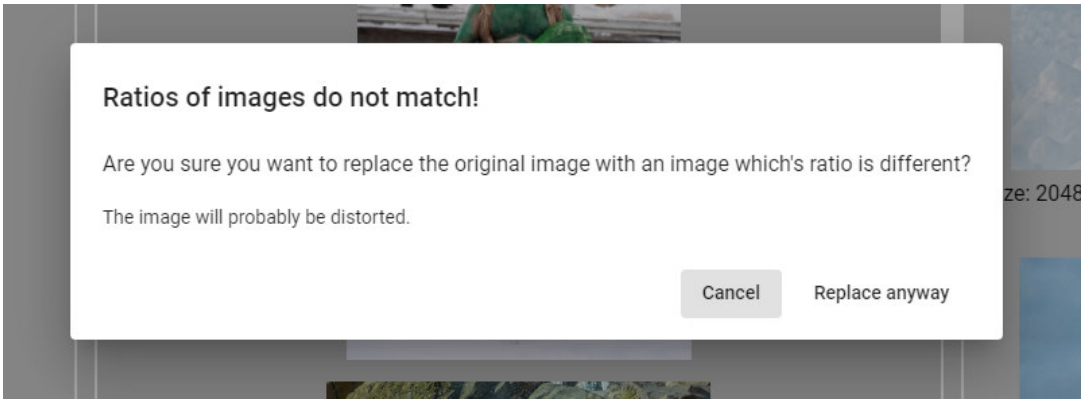

Abbildung 3.9.: Die Bilder besitzen unterschiedliche Seitenverhältnisse

Wir könnten durch einen Klick auf *Replace anyway* bei unserer Auswahl bleiben, doch wenn das neue Bild ein anderes Seitenverhältnis aufweist als das Bild, welches ersetzt werden soll, führt dies zu einer Verzerrung des Bildes durch stauchen oder strecken. Da

wir dies nicht wollen, klicken wir zunächst auf *Cancel*. Nun klicken wir, in Abbildung 3.10 vergrößert dargestellt, auf den rosafarbenen Knopf in der rechten unteren Ecke des Bildes. Dieser öffnet den Bildeditor.

Abbildung 3.10.: Ein klick auf das Rosa Symbol öffnet den Bildeditor

Der Bildeditor ist in Abbildung 3.11 zu sehen. Der Bildeditor ermöglicht es, Bilder zu drehen (durch Klick auf einen der beiden Kreisförmigen Pfeile unten links) oder einen Teil des Bildes auszuschneiden (dargestellt durch das Rechteck auf dem Bild). Die Größe und Position des Ausschnitts kann nach Belieben geändert werden, das Seitenverhältnis ist allerdings auf das Seitenverhältnis des Bildes fixiert, welches wir ersetzen wollen. Durch einen Klick auf *Ok* bestätigen wir unseren Bildausschnitt.

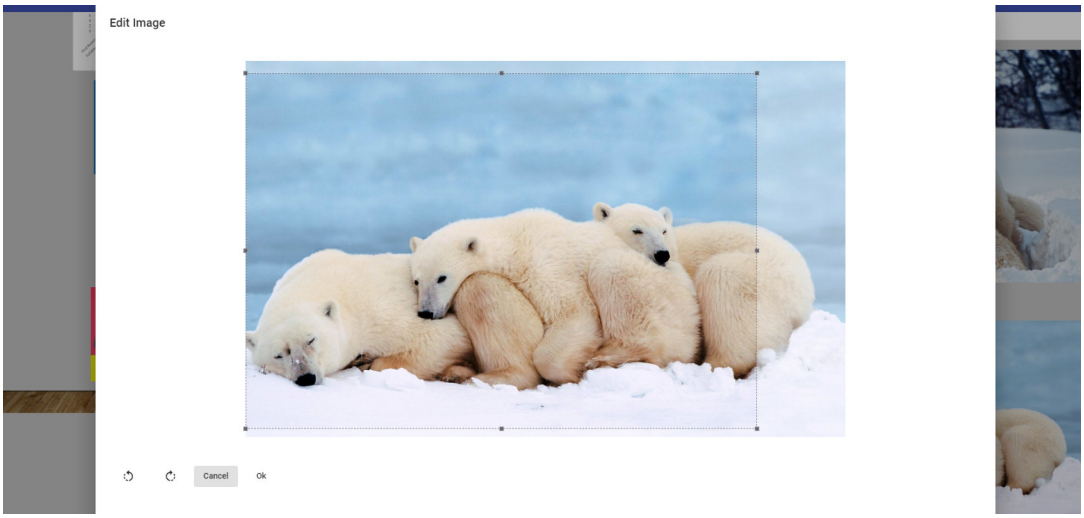

Abbildung 3.11.: Der Bildeditor

Nun wird in der rechten Spalte der Ausschnitt des Bildes anstelle des Originals angezeigt. Wären wir mit unserem Bildausschnitt unzufrieden, könnten wir den Editor erneut öffnen und können wieder das Originalbild bearbeiten. Klicken wir nun auf den Button *Replace*, wird der Dialog über falsche Seitenverhältnisse natürlich nicht mehr angezeigt, da unser Ausschnitt nun das gewünschte Verhältnis aufweist. Außerdem verschwindet das Bild von den Eisbären aus der linken Spalte und ist somit nicht erneut auswählbar. Die gewählte Ersetzungsoperation wird allerdings nicht direkt ausgeführt, sondern nur vermerkt.

Als nächstes möchten wir uns dem OER-Logo (die weißen Hände auf blauem Hintergrund in Abb. 3.8) annehmen. Bei einem Klick darauf präsentiert sich uns der in Abbildung 3.12 gezeigte Dialog.

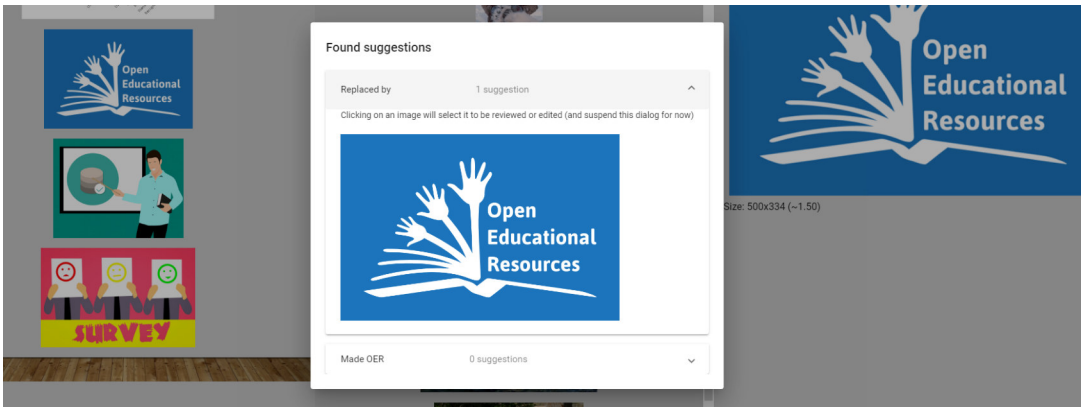

Abbildung 3.12.: Ein Klick auf das Bild in dem Dialog lässt es in der rechten Spalte zum Vergleich mit dem OER-Logo aus unserer Präsentation auftauchen

Offenbar wurde das Logo bereits in einer anderen Datei durch das hier angezeigte Bild ersetzt. Durch einen Klick auf den grauen Hintergrund können wir den Dialog schließen und selbst nach einem Bild suchen, falls wir mit dem vorgeschlagenen Bild nicht zufrieden sind. Wir sind allerdings mit dem Vorschlag einverstanden und akzeptieren ihn mittels einem Klick auf das vorgeschlagene Bild. Das Bild wird dann in der rechten Spalte dem OER-Logo aus unserer Präsentation gegenüber gestellt. Auf Wunsch kann das Bild nun noch verändert werden. Wir möchten allerdings keine Änderungen an dem Bild vornehmen und klicken wieder auf die *Replace* Schaltfläche. Daraufhin wird die Ersetzungsoperation vermerkt und das OER-Logo verschwindet aus der linken Spalte.

Nun möchten wir das Balkendiagramm (oberstes Bild in der linken Spalte in Abb. 3.8) als OER deklarieren, da wir wissen von wem es stammt und dass es nicht in den Quellen der *CC Image Search* veröffentlich wurde. Auch bei diesem Bild öffnet sich wieder der Vorschlagsdialog, zu sehen in Abbildung 3.13.

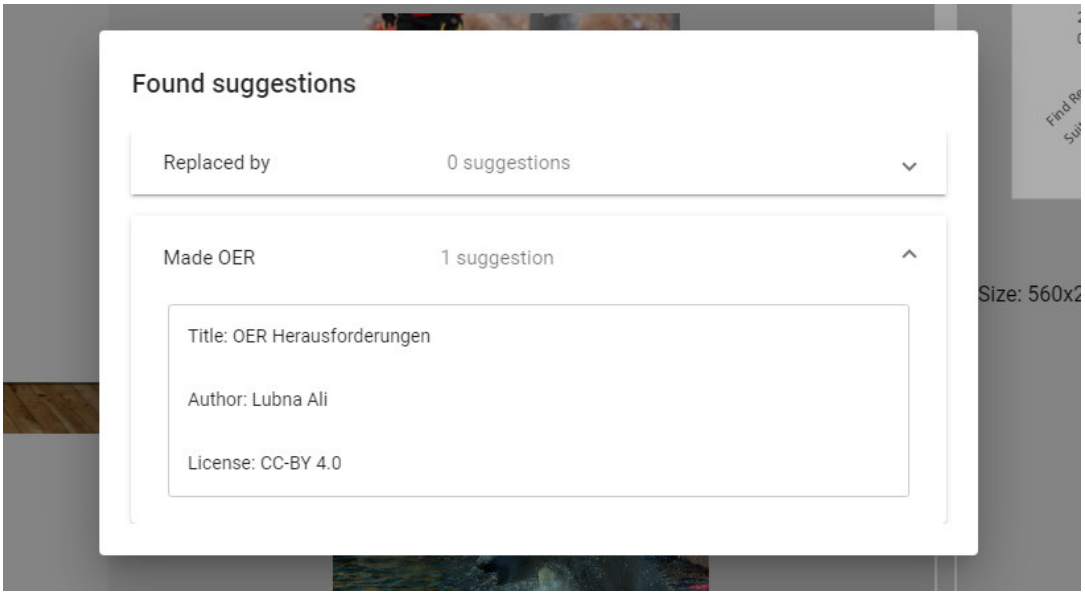

Abbildung 3.13.: Die OER-relevanten Informationen zu dem Balkendiagramm sind dem Tool bereits bekannt

Anscheinend wurde das Balkendiagramm bereits in einer anderen Präsentation verwendet und als OER deklariert. Mit einem Klick auf das gräulich umrandete Element erklären wir uns mit diesem Vorschlag einverstanden, das Balkendiagramm verschwindet sofort aus der linken Spalte und die entsprechende Operation wird vermerkt.

Auch das Survey Bild möchten wir als OER-deklarieren, da wir wissen dass es von Mohamed Mohamed [sic] Mahmoud Hassan gemeinfrei publiziert wurde [48]. Bei einem Klick auf das Bild öffnet sich der Vorschlagsdialog dieses mal nicht, anscheinend ist dieses Bild dem Tool noch nicht bekannt. Also müssen wir die Informationen manuell eingeben. Dazu klicken wir oben im Reiter auf *make OER*.

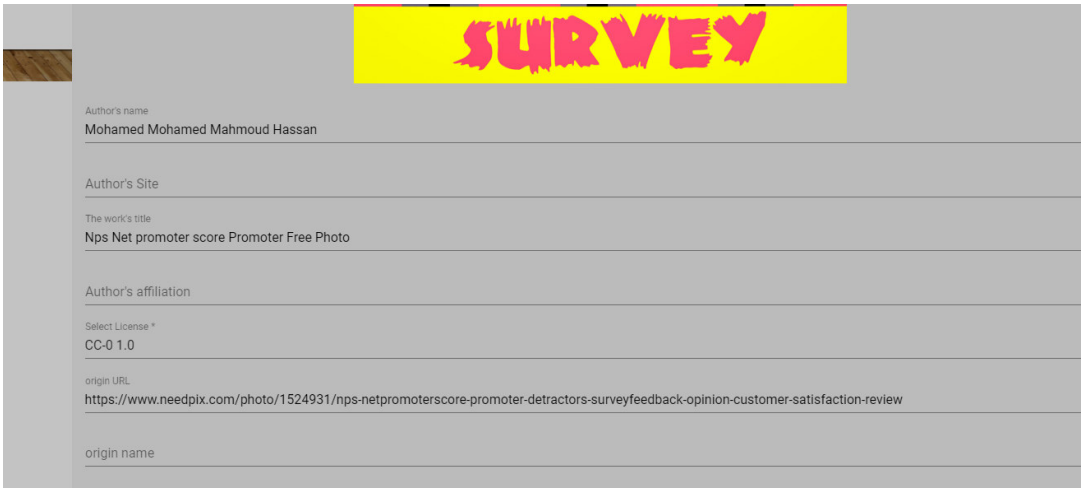

Abbildung 3.14.: Auch wenn das Bild gemeinfrei ist, haben wir hier Informationen über den Autor und das Bild eingetragen

In Abbildung 3.14 sehen wir die eingetragenen Informationen. Obgleich das Bild gemeinfrei ist und daher keine Angaben gemacht werden müssen, haben wir hier als

Informationen den Namen des Autors, den Titel des Bildes sowie den Link zur Quelle angegeben. Durch einen Klick auf Button *Make OER* am unteren Rand vermerken wir diese Operation und das Bild verschwindet aus der linken Spalte.

An dieser Stelle schließen wir nun das Konvertieren dieser Präsentation ab (da es sich nur um ein Beispiel handelt) und klicken auf den Button *Save File* in der oberen rechten Ecke. Daraufhin beginnt das Tool, die vorgemerkten Operationen tatsächlich auszuführen und die Quellenfolie(n) sowie die Haftungsausschlussfolie hinzuzufügen. Im Anschluss müssen alle Dateien, die für die Präsentationsdatei relevant sind, wieder gepackt werden. Der Fortschritt dieses Vorgangs wird uns angezeigt und ist in Abbildung 3.15 zu sehen.

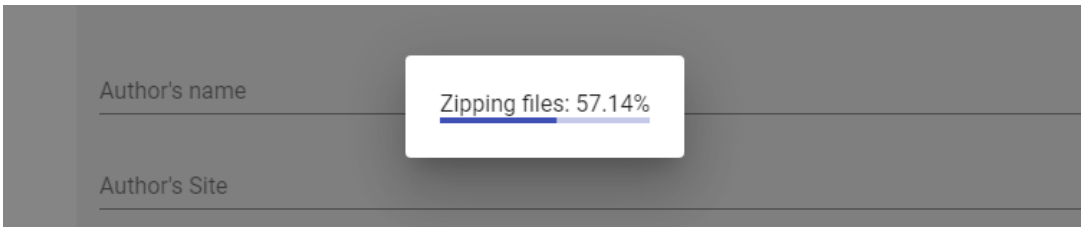

Abbildung 3.15.: Der Fortschritt des Packens aller für die Präsentation benötigten Dateien

Ist der Vorgang abgeschlossen, werden uns zwei Dateien zum Download angeboten, zuerst die konvertierte Präsentation und im Anschluss ein Beleg im Comma-separated values (CSV)-Format [61]. Dieser Beleg enthält Informationen über die ersetzten und als OER deklarierten Bilder dieser Sitzung.

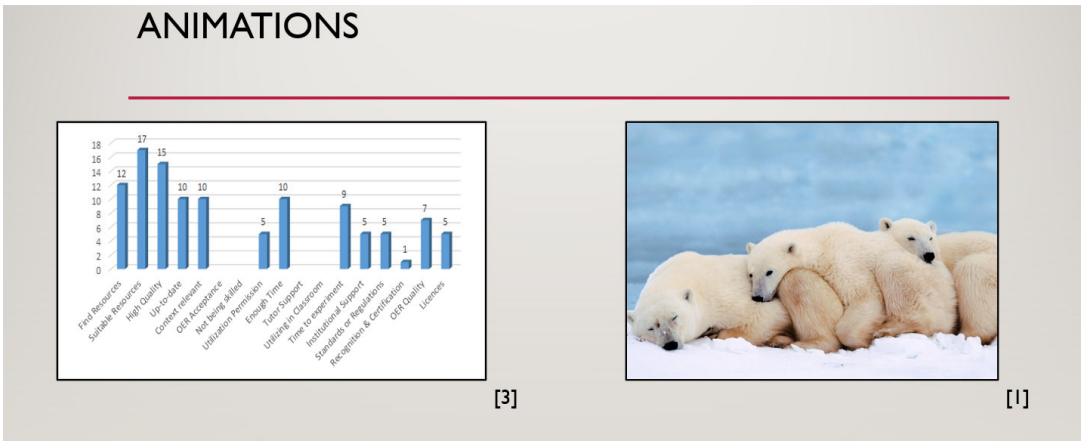

Abbildung 3.16.: Den Bildern wurden Referenzen zu den Quellen auf der Quellenfolie hinzugefügt

In der resultierenden Präsentation finden wir nun Referenzen neben den Bildern (siehe Abb. 3.16) zu den entsprechenden Quellen auf der Quellenfolie (zu sehen in Abb. 3.17). Obwohl das neue Bild des OER-Logos unter der Lizenz CC-BY veröffentlich wurde, gab uns die Suche keinen Autor zurück. Hier muss manuell nachgeholfen werden.

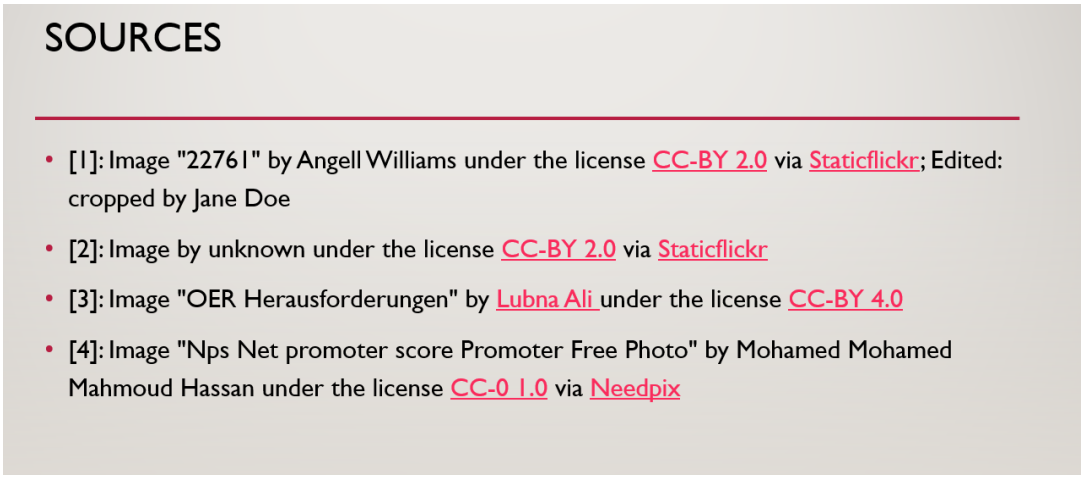

Abbildung 3.17.: Angaben der Quellen gemäß der TULLU-Regel

Die letzte Folie der Präsentation, zu sehen in Abb. 3.18, enthält nun unseren Haftungsausschluss (wie in Abb. 3.4 spezifiziert) sowie das zu unserer Lizenz passende Logo.

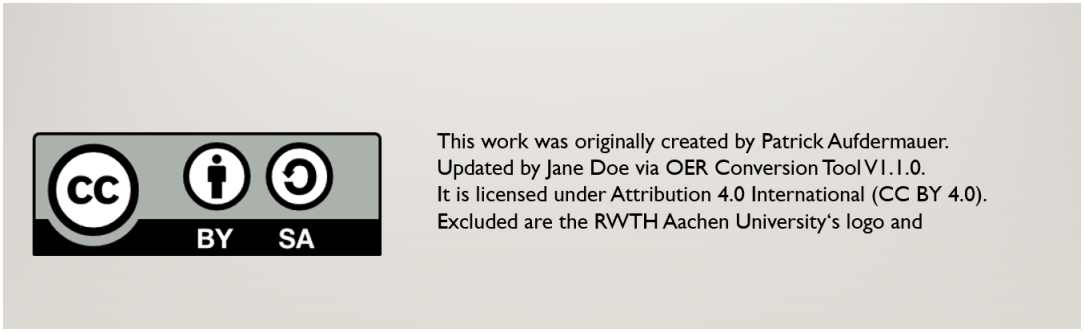

Abbildung 3.18.: Unser Haftungsausschlusstext nebst dem Logo der Lizenz

Haftungsausschluss: Das Bild mit den Bretterdielen stammt aus den Powerpoint-Vorlagen, die die Firma Microsoft durch das Programm Powerpoint zur Verfügung stellt. Die Rechte an dem Bild liegen also nicht bei mir.
Die Rechte an dem Bild, auf dem eine Person auf einer weißen Tafel auf ein Symbol auf grünlichem Hintergrund zeigt, liegen bei ***mohamed_hassan*** *[17], welcher das Bild unter der* ***Pixabay License*** *[55] veröffentlichte.*
Die Rechte an dem ersetzten OER Logo liegen bei ***Jonathasmello*** *[27], welcher das Bild unter der Lizenz* ***CC BY 3.0*** *veröffentlichte.*
Die Rechte an dem ersetzen Eisbärenbild liegen bei ***KeithSzafranski*** *[56], welcher das Bild* ***Lizenzfrei*** *veröffentlichte.*

Teil III

Technologien & Technische Umsetzung

Kapitel 4 Versionverwaltung mit Git

Um diese Bachelorarbeit zu implementieren, wurde die GitLab [65] Instanz der RWTH Aachen [58] genutzt. Diese setzt auf die Versionsverwaltungssoftware Git [25] und stellt darüber hinaus weitere Werkzeuge zur Verfügung, unter anderem einen Issue-Tracker, Webspeicherplatz zum hosten eigener Webseiten und eine Container Registry [32]. Außderdem untersützt GitLab das Konzept von *Continuous Integration*. Damit ist es möglich, automatisch auf Änderungen zu reagiren, welche in das Repository gepusht werden. So wird z.B. jedes mal die Dokumentation sowohl des Frontends als auch des Backends neu generiert, wenn Änderungen in den *Master*-Branch gepusht werden. Der Webspeicher wird dazu genutzt, die Dokumentation des Frontends [6] und des Backends [5] zu hosten. Die Container Registry wird in Kapitel 9.4 noch von Bedeutung sein.

Kapitel 5 Angular Frontend

5.1. Allgemein

Das Frontend des Tools baut auf dem Angular Framework [1] zusammen mit der Angular CLI [2] auf. Als Design wurde das Angular Material Design [3] genutzt, da es von Haus aus unterstützt wird und mit einigen modernen Komponenten daherkommt. Angular selbst, wie das Backend auch, basiert auf der Laufzeitumgebung Node.js [47].
Das gesamte Frontend ist in TypeScript [67] geschrieben. TypeScript ist eine echte Obermenge von JavaScript, d.h. jede valide JavaScript Datei ist auch in TypeScript valide. TypeScript stellt allerdings weitere Konstrukte wie mitunter Klassen und Typisierung bereit. TypeScript Code kann allerdings nicht im Browser ausgeführt werden und muss zunächst zu JavaScript transpiliert werden. Dies übernimmt die Angular CLI.
Für die visuelle Darstellung kommen HTML5 Elemente zum Einsatz, die zum Teil durch die vom Angular Framework und dem Angular Material Design bereitgestellten Module, Komponenten und Elemente ergänzt werden. Als Sprache für die Stylesheets wurde Sass [62] gewählt (Dateiendung *.scss*), da es sich analog zu JavaScript und TypeScript verhält: Sass ist eine echte Obermenge von CSS, welche erweiterte Konstrukte wie *mixins* und *functions* einführt und eine wesentlich kürzere Syntax (im Vergleich zu CSS) aufweist. Zudem ist auch wieder jede valide CSS-Datei auch eine valide Sass-Datei. Es ist anzumerken, dass es noch eine Variante von Sass gibt: Sie hat die Dateiendung *.sass* und verzichtet auf geschwungene Klammern und arbeitet stattdessen mit Einrückungen (ähnlich Verhält sich YAML [73] zu JSON [37]). Ebenso wie TypeScript muss Sass zunächst in CSS transpiliert werden, da Browser Sass nicht verstehen würden. Auch diese Aufgabe übernimmt die Angular CLI.
Da Node.js allerdings letztlich eine Software ist, welche auf einem Server läuft und nicht im Browser beim Client, muss die Anwendung am Ende gebaut werden. Im Bauprozess werden genau die Abhängigkeiten, die das Frontend gegenüber Node.js besitzt, in eine eigenständige JavaScript-Datei geschrieben, welche dann im Browser ausgeführt werden können. Der Rest des Frontends wird (wie zuvor erwähnt) zu JavaScript und CSS transpiliert und mit ausgeliefert. Dazu stellt die Angular CLI einige Befehle bereit, welche entweder unterstützen oder die ganze Arbeit übernehmen.

Während der Recherchen für OOXML Werkzeuge stieß ich auf das recht vielversprechend aussehende Projekt *Open XML SDK for JavaScript by Eric White* [26, 53]. Leider stellte sich heraus, dass einige Abhängigkeiten veraltet waren. Ich habe es nicht geschafft, sie durch neuere Versionen oder ähnliche Pakete zu ersetzen. Daher war es mir im Großen und Ganzen nicht gelungen, mir dieses Software Development Kit (SDK) zunutze zu machen. In der Folge mussten die grundlegenden Manipulationen an OOXML Dokumenten selbst implementiert werden.

Aus technischer Sicht liest das Frontend eine Powerpointdatei im *.pptx* Format ein. Solche Dateien sind im Grunde einfache *zip*-Dateien. Das Frontend öffnet sie mittels der Bibliothek JSZip [38]. Es extrahiert die in */ppt/media* (innerhalb der *.zip*-Datei) enthaltenen Bilder. Während der Benutzer des Frontends die gewünschten Operationen auswählt (siehe Kapitel 3), werden die Operationen in einer Warteschlange vermerkt. Sobald aus der Sicht des Nutzers das Konvertieren abgeschlossen ist (er also alle gewünschten Änderungen vorgenommen hat), wird die Warteschlange abgearbeitet. Wenn die Warteschlange erfolgreich abgearbeitet wurde, wird ein Beleg erstellt. Im Anschluss werden die konvertierte Präsentation und der Beleg zum Download angeboten und währenddessen werden die ausgeführten Operationen an das Backend übermittelt.
Die Dokumentation des Quellcodes des Frontends ist hier [6] zu finden.

Die wichtigsten Klassen, Services und Konfigurationsdateien werden in den folgenden Kapiteln vorgestellt.

5.2. Konfigurationsdateien

Die Konfigurationsdateien dienen dazu, Parameter und Eigenschaften für das Frontend festzulegen. Durch den Bauvorgang gehen diese Dateien in arbiträren JavaScript-Code auf und lassen sich daher nicht mehr so einfach anpassen. Die Konfigurationsdateien befinden sich in */src/config/*.

MainConfig

In dieser Datei wird der Name des Tools festgelegt. Es kann auch angegeben werden, ob die Versionsnummer im Titel der Webseite, in dem das Tool ausgeführt wird, angezeigt werden soll. Ebenso werden die unterstützen Medientypen und deren Dateiendung festgelegt. Hier wird auch eingestellt, wie viele Nachkommastellen vom Seitenverhältnis eines Bildes angezeigt wird, ebenso wie die Toleranz, bevor der Dialog erscheint, der den Benutzer über ungleiche Seitenverhältnisse aufkärt (siehe Abb. 3.9). Es wird hier der Name der Variable festgelegt, unter der die Globally Unique Identifier (GUID) der Präsentation innerhalb derselben (siehe Kapitel 8.5) gespeichert werden soll. Außerdem werden hier die Adresse des Backends angegeben sowie die Vorlage für den Haftungsausschlusstext definiert.

ExternalImageSearchApiConfig

Hier werden die verfügbaren Suchmaschinen definiert. Für jede Suchmaschine muss der Name angegeben werden, wie er im Frontend angezeigt wird, ein Kürzel, unter dem es Toolintern verarbeitet wird und natürlich die Adresse zu der dazugehörigen Application Programming Interface (API). Falls nötig, muss hier auch der *publicKey* angegeben werden (Flickr erfordert einen solchen Schlüssel).

SupportedImportFileTypesConfig

Hier wird definiert, welche Dateien untersützt werden. Dazu wird hier die Dateiendung sowie der offizielle Multipurpose Internet Mail Extension (MIME)-Type angegeben.

5.3. Klassen

FileHandlerAbstract

Die Klasse *FileHandlerAbstract* in */src/classes/file-handler.abstract.ts* ist eine abstrakte Klasse, welche die notwendigsten Funktionen vorgibt, die zur Bearbeitung einer Datei benötigt werden. Grundlegende Funktionen, wie die Rückgabe des Beleges oder die GUID der Datei, oder das Vermerken von Operatioen sind hierin implementiert.

PptxHandler

Die Klasse *PptxHandler* in */src/classes/pptx-handler.ts* erbt von *FileHandlerAbstract*. Sie implentiert die von der abstrakten Klasse geforderten Funktionen für *.pptx* Dateien.

JsZipHelper

Die Klasse *JsZipHelper* in */src/classes/js-zip-helper.ts* abstrahiert die durch JSZip [38] zur Verfügung gestellten Funktionen.

OpenXmlPresentationJs

Die Klasse *OpenXmlPresentationJs* in */src/assets/open-xml-presentation-js.ts* kümmert sich um alle OOXML bezogenen Funktionalitäten. Diese Klasse fügt Folien hinzu, bereitet Text für Folien vor (indem z.B. Hyperlinks und Zeilenumbrüche eingefügt werden), kann Mediendateien extrahieren und mittels *Custom Data*(mehr dazu in Kapitel 8.5) gespeicherte Daten zurückgeben oder darin speichern. Außerdem abstrahiert sie diese Low-Level Funktionen, so dass *PptxHandler* mit ihrer Hilfe direkt die Quellenfolien und die Haftungsausschlussfolie erstellen kann. Außerdem können Layouts ausgewählt und kreiert werden. Einige der Low-Level Funktionen dieser Klasse sind in Kapitel 8 beschrieben.

5.4. Services

Services sind ein Konstrukt in Angular. Ein Service ist im Grunde eine Klasse, welche nach dem Singleton-Konzept arbeitet und an jeder Stelle des Frontends eingebunden werden kann.

BackendService

Dieser Service abtrahiert die nötigsten Funktionen zur Kommunikation mit dem Backend.

HttpService

Dieser Service abstrahiert HTTP-Funktionen, formatiert Anfragenkörper in das richtige Format, setzt die korrekten Anfragenheader und kümmert sich um die Weiterleitung von Antworten. Außerdem findet das Behandeln von HTTP-Fehlern hier statt.

SessionService

Im *SessionService* werden Informationen bezüglich der aktuellen Sitzung festgehalten. Zum Beispiel wird hier der Name des Nutzers, die GUID der aktuell geladenen Datei aber auch die extrahierten Bilder gespeichert.

Kapitel 6 Express Backend

6.1. Allgemein

Das Backend ist eine Node.js Anwendung und nutzt das Express Framework [23], um eine REST [79] API zur Verfügung zu stellen. Mittels des Express Frameworks lassen sich Routen und HTTP Methoden verknüpfen, die Anfragenkörper einsehen oder Antworten zum Client schicken. Um mit der MySQL Datenbank zu kommunizieren, wird das Node Package Manager (NPM) Paket *mysql2* [45] verwendet.
Das Framework nutzt die Middleware cors [24], um Cross-Origin Resource Sharing (CORS) [16] zu konfigurieren. Middlewares sind im Grunde Funktionen, welche sich zwischen zwei andere Funktionen quetschen. Sie fangen die Ausgabe ab und erzeugen eine neue Eingabe an die nächste Funktion. In unserem Falle wird die HTTP-Anfrage abgegriffen und dann werden dieser Anfrage noch zusätzliche Header vorangestellt, denn sonst würden moderne Browser die Antwort des Backends nicht mehr an das Frontend leiten. Diese Schutzmaßnahme ist gedacht, um das nachladen von Schadcode zu unterbinden. Da wir aber möchten, dass die Antwort des Backends das Frontend auch erreicht, muss unsere Anfrage die CORS-Header aufweisen, damit sie nicht gleich blockiert wird.
Das Backend besteht im wesentlichen, abgesehen von Helferklassen, Konfigurationsdateien, Interfacedeklarationen und der */src/server.ts*, welche die REST API mit dem offenen Port verbindet, aus den drei Dateien */src/rest-api.app.ts*, */src/classes/backend.ts* und */src/classes/mysql.ts*. Diese werden im nächsten Kapitel 6.2 erläutert.
Die Quellcodedokumentation des Backends lässt sich hier [5] finden.

6.2. Klassen

Mysql

Die Klasse *Mysql* in */src/classes/mysql.ts* dient als Wrapper für *mysql2*, in der nochmal einige Funktionen gebündelt und vereinfacht wurden. Sie ermöglicht die Nutzung von *Prepared Statements* und kann nach ausgeführter Query die Verbindung automatisch beenden. Außerdem kann ein Callback in eine Transaktion eingebettet werden. Sollte eine Aktion innerhalb dieser Transaktion fehlschlagen, so werden alle bis dahin getätigten Änderungen wieder rückgängig gemacht. Die in dieser Klasse verwendeten SQL Queries werden in 7.3 dargelegt.

Backend

Die Klasse *Backend* in */src/classes/backend.ts* stellt eine Art Adapter zwischen der REST API und der Datenbank dar. Alle benötigten Interaktionen mit der Datenbank werden hier abstrahiert. Sollte in Zukunft ein anderes System als MySQL genutzt werden, so

muss nur Klasse *Backend* angepasst werden.

RestApiApp

Die Klasse *RestApiApp* in */src/rest-api.app.ts* lauscht auf eingehende Anfragen und bindet CORS ein. Wenn eine Anfrage den Server erreicht, wird geprüft, ob der Pfad, an den die Anfrage sich richtet, gültig ist und ob die Methode erlaubt ist. Wenn beides der Fall ist, so greift die Klasse auf verschiedene Funktionen aus der Klasse *Backend* zurück. Wenn alles erfolgreich abgearbeitet wurde, schickt *RestApiApp* einen 200er Statuscode [31] und entweder das Ergebnis zurück, wenn eine Ressource angefragt wurde, oder nur den Statuscode zurück, wenn eine Ressource erfolgreich erstellt wurde. Eine ausführlichere Dokumentation lässt sich hier [4] finden.
Falls ein Fehler auftritt, wird entsprechend der Situation ein 400er oder 500er Code gesendet, zusammen mit einer Nachricht, dass ein Fehler aufgetreten ist und die Anfrage nicht bearbeitet werden konnte.

6.3. Anforderungen an den Datenbankadapter

In diesem Kapitel werden die Funktionen behandelt, die ein Adapter zwischen dem Backend und einer Speicherlösung besitzen muss. Es handelt sich dabei um die Semantik der Funktionen und deren korrekte Benennung. Alle Funktionen sind asynchroner Natur. Daher müssen sie alle mit dem Modifier *async* ausgestattet werden. Der Übersichthalber wird bei den Rückgabewerten ignoriert, dass sie in einem Promise gewrapt sind. Dies erschließt sich nämlich bereits aus der Asynchronität der Funktionen.
Die konkreten Funktionen bzw. Queries, die für den MySQL-Adapter implementiert wurden, sind in Kapitel 7.3 zu finden.

Folgende Funktionen muss ein Adapter nach außen hin aufweisen können:

createReplacedImage

Argumente:

checksum: string

Die MD5 Checksumme des zu speichernden Bildes

Rückgabewert:

number

Die zugewiesene *id*, wenn das Objekt erfolgreich in der Datenbank angelegt wurde

Beschreibung:

Hiermit wird ein neues Bild erzeugt, welches durch ein anderes ersetzt werden soll

createReplacingImage

Argumente:

sourceApi: string

Die API, über die das Bild gefunden wurde

sourceUrl: string

Der Quellpfad des Bildes

licenseId: number
Die *id* der verwendeten Lizenz

width: number
Die Breite des Bildes

height: number
Die Höhe des Bildes

checksum: string
Die MD5 Checksumme des zu speichernden Bildes

authorUserName?: string
Optional Der Benutzername des Autors, der das Bild veröffentlichte

authorRealName?: string
Optional Der echte Name des Autors, der das Bild veröffentlichte

archiveUrl?: string
Optional Der Pfad zu einer durch das WebArchive gespeicherten Version

Rückgabewert:

number
Die zugewiesene *id*, wenn das Objekt erfolgreich in der Datenbank angelegt wurde

Beschreibung:

Hiermit wird ein neues Bild erzeugt, welches ein anderes Bild ersetzen soll

createEditedMedia

Argumente:

width: number
Die neue Breite des Mediums

height: number
Die neue Höhe des Mediums

dataUrlBase64: string
Das veränderte Medium als Data URL Objekt

checksum: string
Die MD5 Checksumme des zu speichernden Mediums

derivedFromId: number
Die *id* des Mediums, aus dem das veränderte Medium hervor ging

Rückgabewert:

number
Die zugewiesene *id*, wenn das Objekt erfolgreich in der Datenbank angelegt wurde

Beschreibung:

Hiermit wird ein neues Medium erzeugt, welches durch Zuschnitt, Zoom oder Ähnlichem aus einem anderen Medium hervorgegangen ist

getLicenseByNameAndVersion

Argumente:

name: string
Der Name der Lizenz, die gesucht werden soll

version: string
Die Version der Lizenz, die gesucht werden soll

Rückgabewert:

number
Die *id* der Lizenz mit dem Namen *name* und der Version *version*

Beschreibung:

Sucht anhand des Namens und der Version nach einer Lizenz und gibt ihre *id* zurück

encapsulateAsTransaction<T>

Argumente:

encapsulatedSequence: (rollback?: () => void) => Promise<T>
Eine Funktion mit der Angegebenen Signatur

Rückgabewert:

T|void
Der Rückgabewert der übergebenen Funktion, die enkapsuliert wurde, oder gar nichts

Beschreibung:

Üblicherweise greift die übergebene Funktion mehrmals auf die Datenbank zu und möchte im Fehlerfall alle Änderungen rückgängig machen. Um dies zu gewährleisten, müssen alle Datenbankoperationen in einer Transaktion stattfinden

linkReplacedImageToReplacingImage

Argumente:

replacedImageId: number
Die *id* des Bildes, was durch ein anderes Bild ersetzt wird

replacingImageId: number
Die *id* des Bildes, was ein anderes Bild ersetzt

Rückgabewert:

number
Die zugewiesene *id*, wenn die Beziehung erfolgreich in der Datenbank angelegt wurde

Beschreibung:

Hiermit wird eine neue Beziehung zwischen zwei Bildern erzeugt

wasMediaIdReplacedByMediaIdAlready

Argumente:

replacedMediaId: number
Die *id* des Mediums, was durch ein anderes Medium ersetzt wurde

replacingMediaId: number
Die *id* des Mediums, was ein anderes Medium ersetzt hat

Rückgabewert:

boolean

true, falls die Beziehung bereits bekannt war und *false* wenn nicht

Beschreibung:

Prüfe, ob die Beziehung zwischen den beiden Medien bereits bekannt ist

getReplacementMedia

Argumente:

checksum: string

Die MD5 Checksumme des ersetzenden Mediums

Rückgabewert:

IReplacementMediaResponse[]

Ein Array von Mediaobjekten, welche die gesuchte Checksumme aufweisen, da z.B. in der Theorie bei MD5 Kollisionen entstehen können. Für genauere Infos sollte die Dokumentation [6] konsultiert werden

Beschreibung:

Suche nach einem ersetzenden Medium anhand seiner Checksumme

getIdOfReplacedImageByChecksum

Argumente:

checksum: string

Die MD5 Checksumme des ersetzten Bildes

Rückgabewert:

number

Die *id* des gesuchten Bildes

Beschreibung:

Erhalte die *id* eines gesuchten ersetzten Bildes

getIdOfReplacingImageByChecksum

Argumente:

checksum: string

Die MD5 Checksumme des ersetzenden Bildes

Rückgabewert:

number

Die *id* des gesuchten Bildes

Beschreibung:

Erhalte die *id* eines gesuchten ersetzenden Bildes

getIdOfEditedVariantByChecksumAndDerivingFrom

Argumente:

checksum: string

Die MD5 Checksumme des veränderten Mediums

derivedFromId: number

Die *id* des Mediums, aus der die veränderte Version hervorging

Rückgabewert:

number

Die *id* der gesuchten veränderten Version eines Mediums

Beschreibung:

Finde die *id* einer veränderten Version eines Mediums anhand der Checksumme der veränderten Version und der *id* des originalen Mediums

Kapitel 7 MySQL Datenbank

7.1. Warum MySQL?

Wie bereits in 1.4 erwähnt, wird als persistente Speicherlösung auf eine MySQL [43] Datenbank gesetzt. Zur Erinnerung: in dieser Bachelorarbeit geht es um die Demonstration der Realisierbarkeit eines Werkzeugs, welches bestehende Lehrmaterialien mit möglichst wenig Aufwand befreien kann. Die Geschwindigkeit und Effizienz der Datenbak spielt daher eine untergeordnete Rolle. Deshalb fiel die Wahl auf MySQL, da ich bereits über mehrjährige Erfahrung im Umgang damit verfüge und mir daher mehr Zeit für die Implementierung des eigentlichen Tools blieb. Ich möchte damit nicht ausdrücken, dass MySQL eine unperformante Wahl darstellt. Es wurden lediglich keine Vergleiche mit anderen Datenbankverwaltungssystemen hinsichtlich der Effizienz bzw. Geschwindigkeit durchgeführt.
Außerdem gibt es eine große und breite Unterstützung für MySQL. So existiert z.B. ein NPM Paket [45], welches einen Großteil der Kommunikation mit dem MySQL-Datenbankserver abstrahiert. Dies erleichterte mir die Implementierung des Backends. Desweiteren existiert ein Plugin [20] für die Integrated Development Environment (IDE) meiner Wahl, PhpStorm [54], welches mir nicht nur die Einsicht und Manipulation der Datenbank erlaubt, ohne weitere Software zu benötigen, sondern auf der Grundlage des Plugins auch Codevervollständigung beim schreiben von Queries bietet.
Die Entscheidung, auf MySQL zu setzen, fiel daher augfrund der Vertrautheit mit der Technologie und ihrer breiten Unterstützung und nicht wegen empirischer Aspekte.

7.2. Theoretische Struktur und Semantik

In diesem Kapitel wird ein Blick auf die Struktur und Semantik der Datenbank geworfen. Dazu betrachten wir zunächst in der folgenden Sektion 7.2.1 ein Entity Relationship (ER)-Diagramm und gehen dann auf die Bedeutung von Entitäten und deren etwaige Relation zueinander ein, sowie auf die Bedeutung der Attribute.
In 7.2.2 betrachten wir die sich aus der Struktur ergebenden Tabellen der Datenbank und deren Spalten. Dort wird unter anderem erläutert, weshalb ich mich für genau diese Tabellen und genau diese Datentypen für die Spalten entschieden habe.
Danach werden in 7.3 die Queries gezeigt, die von der aktuellen Implementierung genutzt werden.

7.2.1. Struktur

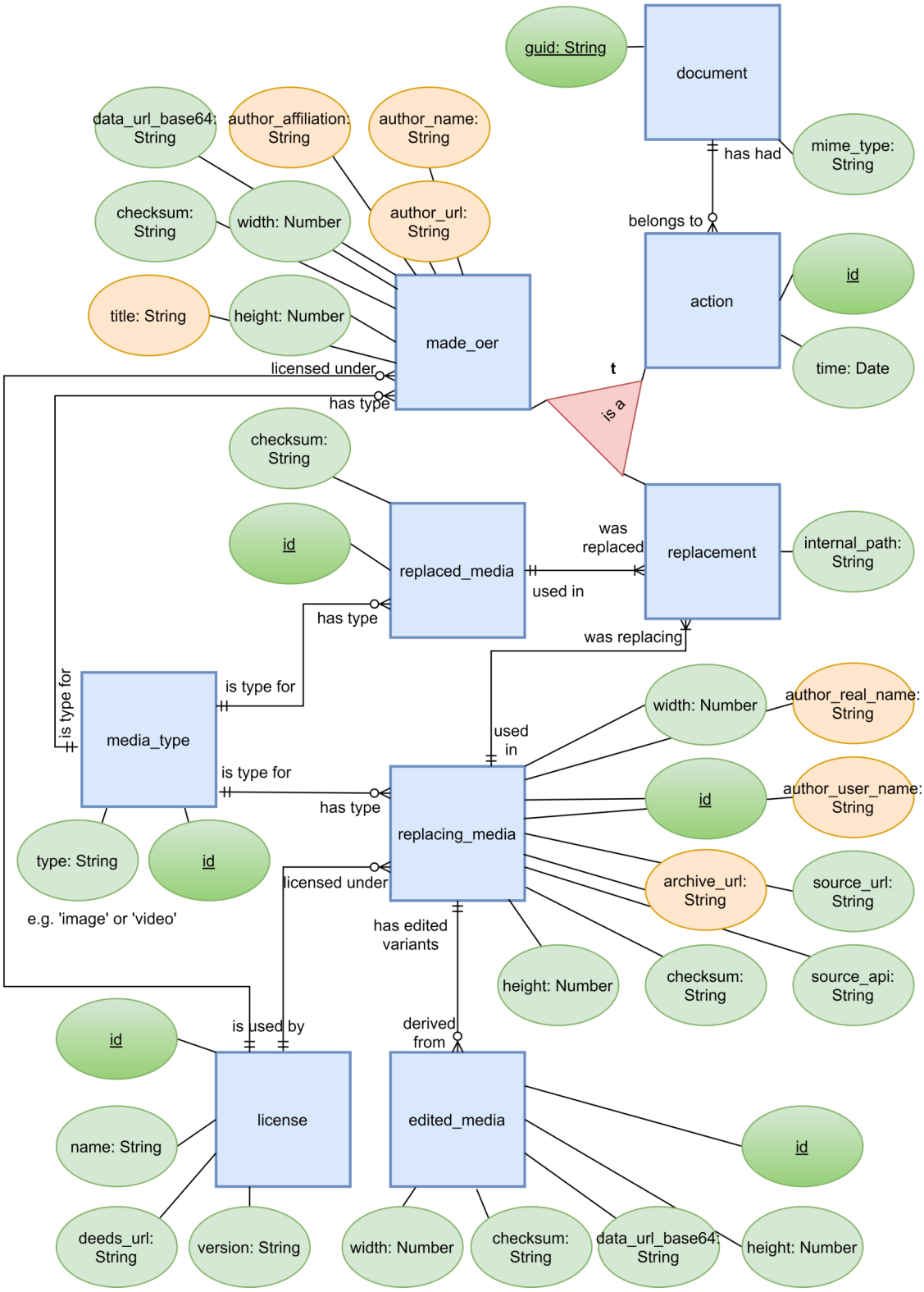

Abbildung 7.1.: Die Datenbankstruktur als ER-Diagramm [77] in Krähenfuß-Notation [42]

In Abbildung 7.1 ist die Struktur der Datenbank als ER-Diagramm [77] dargestellt. Das Diagramm bedient sich der Krähenfuß-Notation [42], um die Relation zwischen den Entitäten (blaue Quadrate) zu beschreiben. Das Indexattribut einer Entität ist mit einem kräftigeren grün hinterlegt, zusätzlich ist der Name unterstrichen. Alle Indexattribute in unserer Datenbank sind eine numerische ID. Deshalb ist dort kein Typ für den Wert vermerkt, wie bei den anderen Attributen. Alle anderen Attribute sind mit einem Datentypen versehen. Dieser Datentyp bezieht sich auf die Typen, welche von TypeScript genutzt werden [7]. Attribute, welche grün hinterlegt sind, dürfen nicht leer bleiben. Die orange hinterlegten Attribute hingegen schon. Das rote Dreieck zeigt eine *isA*-Relation an. Mehr dazu in dem Abschnitt, in dem die Entität *action* behandelt wird.

Die Entität *replaced_media* steht für eine Mediendatei, welches durch mindestens eine andere Mediendatei mit geeigneter Lizenzierung ersetzt wurde. Es kann davon ausgegangen werden, dass das ersetzte Medium unfrei war. Daher wurde darauf verzichtet, das Medium selbst in der Datenbank abzuspeichern. Stattdessen wird nur eine Checksumme gespeichert, welche wir durch Anwendung von Message-Digest Algorithm 5 (MD5) [60] auf die Data URL [18] des zu ersetzenden Mediums erhalten. Auf diese Weise können wir prüfen, ob genau dieses (mutmaßlich unfreie) Medium bereits in einer anderen Lehrdatei ersetzt wurde und können somit genau die Ersatzmedien vorschlagen, die bereits zuvor das (unfreie) Medium ersetzt haben. Da MD5 in der Theorie Kollisionen verursachen kann, wurde darauf verzichtet, die Spalte *checksum unique* zu machen. Logischerweise hat jedes ersetzte Medium genau einen Mediatypen in *media_types* und wird durch mindestens ein Element aus *replacement* referenziert, d.h. es wurde mindestens einmal durch ein anderes Medium ersetzt. Ein Medium kann aber auch durch mehrere Medien ersetzt worden sein. Wird das gleiche Medium mehrmals ersetzt, so wird nicht jedes mal ein neues Item in *replaced_media* generiert.

Ein Element in *replacing_media* beschreibt eine Mediendatei, welche ein anderes Medium ersetzt bzw. ersetzt hat. Ein solches Element hat eine Reihe von Eigenschaften. Darunter hat es eine Breite und eine Höhe sowie wieder eine MD5 [60] Checksumme. Auch hier wurde wieder darauf verzichtet, die Spalte *checksum unique* zu machen. Anhand dieser Checksumme lassen sich auch hier Elemente schnell vergleichen. Außerdem besitzt ein solches Element Informationen darüber, wo es herkommt (*source_url*) und über welche API (*source_api*) es gefunden wurde. Es besitzt auch die optionalen Eigenschaften *archive_url*, *author_real_name* und *author_user_name*. *author_real_name* gibt den echten Namen und *author_user_name* den Benutzernamen des Autors an, unter dem das Medium veröffentlicht wurde. Damit das Medium auch dann noch verfügbar ist, wenn es gelöscht wurde, die Adresse sich geändert hat oder ein persistenter Nachweis der Lizenz erwünscht ist, kann in *archive_url* ein Verweis auf eine durch das Internet Archive [34] erstellte Adresse abgelegt werden. Genau wie Elemente aus *replaced_media* haben auch die Elemente in *replacing_media* jeweils nur genau einen Mediatypen in *media_types*. Außerdem sind diese Elemente genau einer Lizenz in *license* zugeordnet und können beliebig viele bearbeitete Versionen in *edited_media* besitzen. Ersetzt das gleiche Medium nun zusätzlich noch andere Medien, so wird kein neuer Eintrag in *replacing_media* erstellt.

Die Entität *edited_media* steht für veränderte Varianten eines Ursprungsmediums, zum Beispiel durch Zuschneiden oder Zoomen. Für ein solches Element werden wieder die

Höhe, die Breite und eine Checksumme gespeichert. Dieses mal wird das gesamte (veränderte) Medium in *data_url_base64* gespeichert, und zwar wieder als Data URL [18]. Jedes Medium in *edited_media* ist genau einem Medium in *replacing_media* zugeordnet.
In *license* sind die Lizenzen hinterlegt. Jede Lizenz besitzt einen Namen (*name*), eine Version (*version*) und eine Adresse, an der die Lizenzbedingungen (*deeds_url*) zu finden sind. Jede Lizenz ist beliebig vielen Elementen in *replacing_media* und *made_oer* zugeordnet.

Die Entität *media_type* beschreibt eine Art Datentypen für ein Medium. In der Eigenschaft *type* wird eine Stringrepräsentation gespeichert, z.B. *image* oder *video*. Jeder Medientyp ist sowohl beliebig vielen Medien in *replaced_media* und *replacing_media* als auch beliebig vielen Medien in *made_oer* zugeordnet.
Der Medientyp hätte direkt als String in *replaced_media* und *replacing_media* gespeichert werden können. Darauf wurde jedoch verzichtet, da in der Datebank das Suchen nach Zahlenwerten schneller erfolgt als das Suchen nach Strings. Dies folgt daher, dass Zahlen direkt miteinander verglichen werden können, also nur eine Vergleichsoperation benötigen, während bei Strings Buchstabe für Buchstabe miteinander verglichen wird und somit mindestens so viele Vergleichsoperation benötigt werden, wie der String lang ist. Deshalb ist es performanter, die Medientypen separat in *media_type* abzuspeichern und die, in Bezug auf die Anzahl der Elemente in *replaced_media* und *replacing_media* vergleichsweise wenigen Elemente nach dem String zu durchsuchen. Da jeder Medientyp nur ein mal abgespeichert wird, kann die Suche aufhören, sobald die zugehörige *id* gefunden wurde. Dann können die Elemente in *replaced_media* und *replacing_media* effizient anhand ihres Medientypens, welcher nun als Zahl vorliegt, gefiltert werden.

Die Aktionen des Tools werden hier als *action* gespeichert. Dabei ist jede *action* entweder vom Typ *made_oer* oder *replacement*. Jede *action* erhält eine *id* und eine Zeit (*time*) im Datumsformat, zu der die *action* dem Backend bekannt gemacht wurde. Jede Aktion ist genau einem Dokument zugeordnet, in dem sie ausgeführt wurde. Der Einfachheit halber wird dieses Feld automatisch mit dem aktuellen Datum gefüllt, wenn ein solches Element in der Datenbank angelegt wird.

Elemente in *replacement* erhalten keine eigene *id* sondern beziehen ihre von der *action*-Entität. Sie verweisen auf ein Medium in *replaced_media*, welches im Zuge der Aktion ersetzt wurde und auf eines in *replacing_media*, welches nun das neue Medium darstellt. Außerdem wird der interne Pfad (innerhalb des zu befreienden Dokuments) des ersetzten Mediums festgehalten (*internal_path*).

Die Entität *made_oer* beschreibt Medien, welche nicht durch ein anderes Medium ersetzt wurden, sondern unter einer bestimmten Lizenz vom Benutzer des Tools veröffentlicht wurde. Dabei kann es sich um selbst geschossene Motive, um eigens kreierte Graphen oder Ähnlichem handeln. Für solche Elemente wird auf jeden Fall die Höhe (*height*), Breite (*width*), eine Checksumme (*checksum*) und die entsprechende Data URL gespeichert. Wie bereits zuvor, wurde darauf verzichtet, die Spalte *checksum unique* zu machen, da MD5 in der Theorie Kollisionen verursachen kann. Außerdem ist ein Element aus *made_oer* genau einem Medientypen und genau einer Lizenz zugeordnet. Auch diese Elemente erhalten ihre *id* via der *action*-Entität. Auch wenn je nach gewählter Lizenz einige der folgenden Angaben Pflicht sind, sind für die Datenbank der Name des Autors (*author_name*), ein Link zu seinem Webauftritt (*author_url*), die Zugehörigkeit z.B. zu

einem Institut (*author_affiliation*) und der Titel des Mediums (*title*) optionale Attribute.

Die Dokumente, welche durch das Tool bereinigt werden, werden in *document* gespeichert. Das Feld *guid* speichert die GUID, welche alle vom Tool befreiten Dokumente erhalten und fungiert in diesem Falle als *Primärschlüssel* der Tabelle. Zusätzlich wird noch der MIME-Type in *mime_type* gespeichert. Dies macht es uns später möglich, z.B nach Präsentationen oder Textdokumenten zu filtern. Jedes Dokument kann beliebig vielen Aktionen zugeordnet sein.

Im nächsten Abschnitt werden die konkreten Tabellen vorgestellt, die sich aus dem Diagramm in Abbildung 7.1 ergeben.

7.2.2. Tabellen

Nachdem in Kapitel 7.2 die theoretische Struktur erläutert wurde, beschäftigt sich dieses Kapitel mit den resultierenden Tabellen und deren Spalten.

Es werden mindestens neun Tabellen für die neun Entitäten in der Abbildung 7.1 benötigt. Da sowohl *replaced_media*, *replacing_media* als auch *made_oer* genau einen Medientypen besitzen, kann diese Information direkt in der jeweiligen Tabelle gespeichert werden. Daher entfällt die Notwendigkeit einer Tabelle, welche einen Medientypen mit einem Medium verknüpft. Ähnliches trifft auf die Relationen *license~replacing_media*, *license~made_oer* und *replacing_media~edited_media* zu. Da jeweils jedes Element in *replacing_media* und *made_oer* unter genau einer Lizenz veröffentlich wurde und jedes Element in *edited_media* von genau einem Medium abstammt, kann auch hier der Verweis auf die *id* direkt in die Tabellen eingetragen werden und es erübrigen sich weitere Tabellen, die die Relationen festhalten. Ebenso können die *id*s von den Elementen aus *replaced_media* und *replacing_media* direkt in *replacement* gespeichert werden und auch die *id* eines Dokumentes kann direkt in einer *action* festgehalten werden.
Es ist also möglich, auf zusäzliche Tabellen, die Beziehungen festhalten, zu verzichten. Daher ergeben sich die folgenden neun Tabellen mitsamt Spalten und Datentypen:

edited_media
(*id*: int, width: int, height: int, data_url_base64: longtext, checksum: text, derived_from: int→replacing_media.id)

license
(*id*: int, name: text, version: text, deeds_url: text)

media_type
(*id*: int, type: text)

replaced_media
(*id*: int, checksum: text, is_type: int→media_types.id)

replacing_media
(*id*: int, source_url: text, source_api: text, width: int, height: int, checksum: text, is_type: int→media_type.id, licensed_under: int→license.id, author_real_name?: text, author_user_name?: text, archive_url?: text)

action
(*id*: int, time: datetime, document: int→document.guid)

replacement
(*id*: int →action.id, internal_path: text, id_replaced_media: int→replaced_media.id, id_replacing_media: int→replacing_media.id)

made_oer
(*id*: int →action.id, data_url_base64: longtext, checksum: text, width: int, height: int, is_type: int→media_type.id, licensed_under: int→license.id, author_name?: text, author_url?: text, author_affiliation?: text, title?: text)

document
(*guid*: text, mime_type: text)

Dabei steht „*id*“ für den Primärschlüssel, „col1: int→table1.col2“ für einen Fremdschlüssel, welcher in *col1* einen *Integer* einträgt, der auf die Spalte *col2* in der Tabelle *table1* verweist und „col3?: int“, dass die Spalte *col3* leer bleiben darf und ansonsten den Datentyp *int* erwartet.
Für die Spalten *data_url_base64* wurde der Datentyp *longtext* gewählt, da er $2^{32} - 1$ Byte [44] halten kann, was ungefähr 4, 29 GB entspricht. Dies ist von Bedeutung, da später eventuell auch Videodateien ersetzt oder als eigenes OER veröffentlicht werden sollen.

7.3. SQL-Queries für MySQL Lösung

Die Queries für die MySQL Datenbank werden alle mittels *prepared statements* ausgeführt, um einer MySQL-Injection vorzukommen. Die Beispielhafte Query besitzt Platzhalter, nämlich das Fragezeichen, wie nachfolgend gezeigt

```
SELECT * FROM ex_table WHERE ex_row1 = ? AND ex_row2 = ?;
```

und anschließend wird ein Array, wie z.B.

```
["string", 42]
```

auf die Platzhalter gemappt. Dabei entspricht die Reihenfolge und Position der Platzhalter exakt der Reihenfolge und Position der Substitutionswerte in dem Array. Wenn im weiteren Verlaufe dieses Kapitels die Query vorgestellt werden, werde ich im Allgemeinen auf die Unprozessierten Queries mit ihren Platzhaltern eingehen.
Die fertigen Queries bzw. die Substitutionswerte sind dann entweder Situationsabhängig oder erschließen sich aus dem Kontext. Sollte dies nicht unbedingt gegeben sein, werde ich auf die Werte gesondert eingehen. Auch lassen sich die Semantiken von Spalten erschließen, z.B. bei *join*-Operationen über mehrere Tabellen, in dem die Semantik der entsprechenden Spalte in seiner ursprünglichen Tabelle untersucht wird. Da die Semantiken bereits alle in 7.2.2 erläutert wurden, werde ich in diesem Kapitel nicht mehr gesondert darauf eingehen.

Die *id* eines Medientypens bekommen

Die Query dazu lautet

```
SELECT * FROM media_type WHERE type = ? LIMIT 1;
```

Die Query ist simpel und fragt lediglich nach der numerischen *id* für einen gegebenen Medientypen, wie z.B.

```
["image"]
```

Bisher werden keine weiteren Medientypen unterstützt.

Ein ersetztes Medium anlegen

Um ein ersetztes Bild zu vermerken, muss eine neue Zeile in *replaced_media* angelegt werden. Dies geschieht mittels

```
INSERT INTO replaced_media (checksum, is_type) VALUES (?, ?);
```

wobei *checksum* die errechnete Checksumme und *is_type* die *id* des entsprechenden Medientyps ist.

Ein Lizenzobjekt anhand seines Namens und seiner Version bekommen

Um eine Lizenz samt ihren Eigenschaften zu erhalten, muss die Query

```
SELECT * FROM license WHERE name = ? AND version = ? LIMIT 1;
```

ausgeführt werden. Ungeachtet der Limitierung auf einen Datensatz, sollte auch sonst nur ein passender Datensatz in der Datenbank existieren. Da Prozesse, die diese Query absenden, auch mit nur einem einzelnen Datensatz in der Antwort rechnen, wird das Ergebnis sicherheitshalber eingeschränkt. Dies hat auch den Vorteil, dass die Suche beendet wird und nicht fortgeführt werden muss, sobald ein Datensatz gefunden wurde, was Ressourcen spart.

Ein ersetzendes Medium anlegen

Dazu muss via

```
INSERT INTO replacing_media (source_url, author_real_name, source_api,
    author_user_name, is_type, licensed_under, height, width, checksum,
     archive_url) VALUES (?,?,?,?,?,?,?,?,?,?);
```

eine neue Zeile in *replacing_media* erstellt werden.

Prüfen, ob zwei Medien miteinander in Beziehung stehen

Um zu prüfen, ob ein gegebenes ersetztes Medium bereits durch ein gegebenes ersetzendes Medium ersetzt worden ist, kann die Query

```
SELECT id_replaced_media as id FROM replacement WHERE id_replaced_media
     = ? AND id_replacing_media = ? LIMIT 1;
```

ausgeführt und auf die Anzahl der Resultate geprüft werden. Offensichtlich existiert die Beziehung bereits, wenn ein Datensatz zurückgegeben wird. Ist das Resultat hingegen leer, so ist diese Beziehung nicht bekannt und kann bei Bedarf mit der nächsten Query angelegt werden.

Ersetzungsaktion bekannt machen

Um eine Ersetzungsaktion der Datenbank bekannt zu machen, muss zuerst mittels der Query

```
INSERT INTO action (document) VALUES (?);
```

eine Aktion erzeugt werden. Durch auslesen der *insertionId* erhalten wir die zugewiesene *id* der eben erzeugten Aktion. Die Details der Ersetzung werden dann durch

```
INSERT INTO replacement (id, id_replaced_media, id_replacing_media,
    internal_path) VALUES (?,?, ?,?);
```

übergeben. Dabei ist die *id* die *id* der vorher erstellten Aktion.

Bekannt machen, dass ein Medium als OER deklariert wurde

Um der Datenbank das neue OER Medium bekannt zu machen, muss zunächst wieder mittels der Query

```
INSERT INTO action (document) VALUES (?);
```

eine Aktion erzeugt werden. Durch auslesen der *insertionId* erhalten wir die zugewiesene *id* der eben erzeugten Aktion. Diese *id* verwenden wir nun für *id*, wenn wir folgende Query

```
INSERT INTO made_oer (id, data_url_base64, checksum, width, height,
    is_type, licensed_under, author_name, author_url,
    author_affiliation, title) VALUES (?,?,?,?,?,?,?,?,?,?,?);
```

ausführen.

Ersetzendes Medium anhand der Checksumme erhalten

Das ersetzende Medium wird anhand seiner Checksumme und der *id* des gewünschten Medientypens gefunden. Da die Werte für ein solches Objekt über mehrere Tabellen verteilt sind, müssen einige JOIN-Operationen durchgeführt werden:

```
SELECT
  replacing_media.source_api,
  replacing_media.source_url,
  replacing_media.author_real_name,
  replacing_media.author_user_name,
  replacing_media.width,
  replacing_media.height,
  license.name as license_name,
  license.version as license_version,
  license.deeds_url as license_deeds_url,
  edited_media.width as edited_width,
  edited_media.height as edited_height,
  edited_media.data_url_base64 as edited_data_url_base64
FROM replaced_media
JOIN replacement ON replaced_media.id = replacement.id_replaced_media
JOIN replacing_media ON
  replacing_media.id = replacement.id_replacing_media
JOIN license ON replacing_media.licensed_under = license.id
JOIN media_type ON replacing_media.is_type = media_type.id
LEFT JOIN
  edited_media ON replacing_media.id = edited_media.derived_from
WHERE
  replaced_media.checksum = ? AND media_type.type = ?;
```

Verändertes Medium erstellen

Um eine veränderte Version eines Mediums zu erstellen, muss ein neuer Eintrag in der Tabelle *edited_media* angelegt werden. Dazu führt man die Query

```
INSERT INTO edited_media (width, height, data_url_base64, checksum,
    derived_from) VALUES (?,?,?,?,?);
```

aus.

id eines anhand der Checksumme und des Typs gesuchtem ersetztem Mediums

Die Query

```
SELECT id FROM replaced_media WHERE checksum = ? AND is_type = ? LIMIT
    1;
```

liefert die *id* des betreffenden Mediums. Auch hier wird nach dem ersten gefundem Datensatz die Suche abgebrochen und das Resultat zurückgegeben, um Ressourcen zu sparen und immer maximal ein Ergebnis liefern zu können.

id eines anhand der Checksumme und des Typs gesuchtem ersetzendem Mediums

Analog zu der vorigen Query muss hierzu

```
SELECT id FROM replacing_media WHERE checksum = ? AND is_type = ? LIMIT
     1;
```

ausgeführt werden.

id eines anhand der Checksumme und des Ursprungsmediums gesuchtem verändertem Mediums

Völlig analog zu den beiden vorherigen Queries muss dazu

```
SELECT id FROM edited_media WHERE checksum = ? AND derived_from = ?
    LIMIT 1;
```

ausgeführt werden.

Ein Dokument erstellen

Um ein Dokument zu erstellen, werden die Strings für eine GUID und für den MIME-Type erwartet. Durch ausführen von

```
INSERT INTO document (guid, mime_type) VALUES (?,?);
```

wird das Dokument angelegt.

Prüfen, ob ein Dokument bereits existiert

Da die GUID einmalig sein muss, muss zuvor geprüft werden, ob ein Dokument zu einer gegebenen GUID bereits existiert, bevor ein solches Dokument erzeugt werden könnte. Durch prüfen, ob die Antwort von

```
SELECT * FROM document WHERE guid = ? LIMIT 1;
```

keine Zeilen aus der Tabelle *document* enthält, kann darauf geschlossen werden, dass ein solches Dokument noch nicht in der Datenbank existiert.

Mittels Checksumme nach einem OER Medium suchen

Auch hier müssen wieder einige JOIN Operationen durchgeführt werden. Durch

```
SELECT
  made_oer.id as id,
  data_url_base64,
  checksum,
  width,
  height,
  media_type.type as media_type,
  author_name,
  author_url,
  author_affiliation,
  title,
  license.name as license_name,
  license.version as license_version,
  license.deeds_url as license_deeds
FROM made_oer
JOIN license on made_oer.licensed_under = license.id
JOIN media_type on made_oer.is_type = media_type.id
WHERE checksum = ?;
```

erhält man das Medium (oder mehrere), falls es existiert.

Prüfen, ob Medium anhand der Checksumme bereits ersetzt wurde

Um Ersetzungsvorschläge für ein Medium zu erhalten, kann geprüft werden ob dieses Medium in der Vergangenheit bereits durch ein anderes Medium ersetzt wurde. Dazu kann die Query

```
SELECT
  replacing_media.source_api,
  replacing_media.source_url,
  replacing_media.author_real_name,
  replacing_media.author_user_name,
  replacing_media.width,
  replacing_media.height,
  license.name as license_name,
  license.version as license_version,
  license.deeds_url as license_deeds_url,
  edited_media.width as edited_width,
  edited_media.height as edited_height,
  edited_media.data_url_base64 as edited_data_url_base64
FROM replaced_media
JOIN replacement ON replaced_media.id = replacement.id_replaced_media
JOIN replacing_media ON
  replacing_media.id = replacement.id_replacing_media
JOIN license ON replacing_media.licensed_under = license.id
JOIN media_type ON replacing_media.is_type = media_type.id
LEFT JOIN
  edited_media ON replacing_media.id = edited_media.derived_from
WHERE
  replaced_media.checksum = ? AND
  media_type.type = ?;
```

ausgeführt werden.

Kapitel 8 Office OpenXML Format

In diesem Kapitel wird erklärt, welche Dateien verändert oder angelegt werden müssen, um eine neue Folie an das Ende der Präsentation anzuhängen, um Referenzen zu Quellenangaben neben den Bildern zu platzieren, um arbiträre Daten einzubetten und um einen Kommentar zu einer neuen Folie zu erstellen.

8.1. Folie hinzufügen

Die Reihenfolge der in diesem Kapitel beschriebenen Schritte ist nicht unbedingt die einzig mögliche. Allerdings hat diese genutzte Reihenfolge zur Vermeidung von Mehrfachberechnungen beigetragen.
Die folgenden Kapitel legen dar, welche Änderungen an der jeweiligen Datei vorgenommen werden müssen, um eine neue Folie an das Ende der Präsentation anzuhängen.

/docProps/app.xml

```
1  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2  <Properties ...>
3    ⋮
4    <Slides>69</Slides>
5    ⋮
6    <HeadingPairs>
7      <vt:vector size="6" baseType="variant">
8        ⋮
9        <vt:variant>
10         <vt:i4>69</vt:i4>
11       </vt:variant>
12     </vt:vector>
13   </HeadingPairs>
14   <TitlesOfParts>
15     <vt:vector size="84" baseType="lpstr">
16       ⋮
17       <vt:lpstr>Titel einer Folie</vt:lpstr>
18     </vt:vector>
19   </TitlesOfParts>
20 </Properties>
```

Listing 8.1: Relevante Elemente der Datei */docProps/app.xml* mit Beispielinhalt

In Listing 8.1 sind die Elemente abgebildet, welche manipuliert werden müssen.
Der Inhalt des Knotens `<Slides>` in Zeile 4 muss um eins erhöht werden.

Außerdem muss auch der Inahlt des Knotens in Zeile 10 um eins erhöht werden. Typischerweise sollten die Knoten in Zeile 4 und 10 den gleichen Wert tragen. Der Knoten `<vt:variant>` in Zeile 9 ist das letzte Kind des Knotens `<vt:vector>` in Zeile 7, welches wiederum ein Kind des Knotens `<HeadingPairs>` in Zeile 6 ist.
Nun wird der neuen Folie ein Titel zugeordnet. Dazu wird dem Knoten `vt:vector` in Zeile 15 ein neues Kindelement `<vt:lpstr>` angehangen, welches als Inhalt den neuen Folientitel zugeordnet bekommt. Da sich die Anzahl der Kindelemente nun geändert hat, muss zusätzlich die Eigenschaft *size* von `<vt:vector>` in Zeile 15 um eins erhöht werden.
Das Beispiel aus Listing 8.1 sieht nun wie folgt aus:

```
1  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2  <Properties ...>
3    ⋮
4    <Slides>70</Slides>
5    ⋮
6    <HeadingPairs>
7      <vt:vector size="6" baseType="variant">
8        ⋮
9        <vt:variant>
10         <vt:i4>70</vt:i4>
11       </vt:variant>
12     </vt:vector>
13   </HeadingPairs>
14   <TitlesOfParts>
15     <vt:vector size="85" baseType="lpstr">
16       ⋮
17       <vt:lpstr>Titel einer Folie</vt:lpstr>
18       <vt:lpstr>Titel der neuen Folie</vt:lpstr>
19     </vt:vector>
20   </TitlesOfParts>
21 </Properties>
```

Listing 8.2: Das Beispiel nach der Modifikation

/docProps/core.xml

```
1 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2 <cp:coreProperties ...>
3   <dc:title>Beispielhafter Titel</dc:title>
4   <dc:creator>Dr. Frankenstein</dc:creator>
5   <cp:lastModifiedBy>Dr. Frankenstein's Monster</cp:lastModifiedBy>
6   <cp:revision>95</cp:revision>
7   <dcterms:created xsi:type="dcterms:W3CDTF">2016-10-30T15:30:41Z</
       dcterms:created>
8   <dcterms:modified xsi:type="dcterms:W3CDTF">2018-10-22T09:13:48Z</
       dcterms:modified>
9 </cp:coreProperties>
```

Listing 8.3: Relevante Elemente der Datei */docProps/core.xml* mit Beispielinhalt

Änderungen in dieser Datei sind optional und nicht zwingend vonnöten. `<dc:title>` in Zeile 3 enthält den Namen der Präsentation. `<dc:creator>` benennt den Author der Präsentation. `<cp:lastModifiedBy>` gibt den Namen des Benutzers an, welcher die letzten Änderungen an der Präsentation vorgenommen hat. In Zeile 6 zählt `<cp:revision>` mit, wie oft die Datei bereits nach Änderungen gespeichert wurde. `<dcterms:created>` und `<dcterms:modified>` halten fest, wann die Datei erstellt respektive zuletzt modifiziert wurde. Der Inhalt, also das Datum, wird im ISO 8601 Format [35] angegeben.

/ppt/_rels/presentation.xml.rels

```
1  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2  <Relationships xmlns="...">
3    ⋮
4    <Relationship Id="rId76" Target="tableStyles.xml" Type="..."/>
5    <Relationship Id="rId71" Target="notesMasters/notesMaster1.xml" Type=
       "..."/>
6    <Relationship Id="rId74" Target="viewProps.xml" Type="..."/>
7    <Relationship Id="rId73" Target="presProps.xml" Type="..."/>
8    <Relationship Id="rId72" Target="handoutMasters/handoutMaster1.xml"
       Type="..."/>
9    <Relationship Id="rId59" Type="http://schemas.openxmlformats.org/
       officeDocument/2006/relationships/slide" Target="slides/slide58.
       xml"/>
10   <Relationship Id="rId70" Type="http://schemas.openxmlformats.org/
       officeDocument/2006/relationships/slide" Target="slides/slide69.
       xml"/>
11   <Relationship Id="rId75" Target="theme/theme1.xml" Type="..."/>
12   <Relationship Id="rId1" Target="slideMasters/slideMaster1.xml" Type="
       ..."/>
13   ⋮
14 </Relationships>
```

Listing 8.4: Die Datei */ppt/_rels/presentation.xml.rels* mit Beispielinhalt

Diese Datei führt Referenzen zu allen benutzten Notizen, Handouts, Folien, Tabellen und Themes, sowie zu einigen speziellen Dateien, welche Metainformationen über die Darstellung der Präsentation speichern. Da eine neue Folie eingefügt werden soll, muss ein neues `<Relationship>` Element als Kind des `<Relationships>` Elements in Zeile 2 eingefügt werden. Dieses neue `<Relationship>` erhält den gleichen Wert für die Eigenschaft *Type* wie das Element in Zeile 9. In dem Beispiel in Listing 8.4 ist die Folie 69 die letzte Folie. Das neue `<Relationship>` Element erhält nun die Eigenschaft *Id="rId71"*, denn diese *id* muss der Nachfolger der *id* der letzten Folie sein. Dies hat zur Folge, dass alle anderen *id*s in diesem Dokument, welche mindestens den Wert *rId71* zugewiesen bekommen haben, um eins hoch gezählt werden müssen. Die Reihenfolge der einzelnen `<Relationship>` Knoten hat keine Relevanz. Das in Listing 8.4 gezeigte Beispiel hat am Ende wie folgt auszusehen:

```
1  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2  <Relationships xmlns="...">
3    ⋮
4    <Relationship Id="rId77" Target="tableStyles.xml" Type="..."/>
```

```
5   <Relationship Id="rId72" Target="notesMasters/notesMaster1.xml" Type=
        "..."/>
6   <Relationship Id="rId75" Target="viewProps.xml" Type="..."/>
7   <Relationship Id="rId74" Target="presProps.xml" Type="..."/>
8   <Relationship Id="rId73" Target="handoutMasters/handoutMaster1.xml"
        Type="..."/>
9   <Relationship Id="rId59" Type="http://schemas.openxmlformats.org/
        officeDocument/2006/relationships/slide" Target="slides/slide58.
        xml"/>
10  <Relationship Id="rId70" Type="http://schemas.openxmlformats.org/
        officeDocument/2006/relationships/slide" Target="slides/slide69.
        xml"/>
11  <Relationship Id="rId76" Target="theme/theme1.xml" Type="..."/>
12  <Relationship Id="rId1" Target="slideMasters/slideMaster1.xml" Type="
        ..."/>
13  ⋮
14  <Relationship Id="rId71" Type="http://schemas.openxmlformats.org/
        officeDocument/2006/relationships/slide" Target="slides/slide70.
        xml"/>
15 </Relationships>
```

Listing 8.5: Die Datei */ppt/_rels/presentation.xml.rels*, nachdem in Zeile 14 ein neuer `<Relationship>` Knoten eingefügt und die *id*s angepasst wurden

/[Content_Types].xml

Diese Datei enthält, ähnlich wie in 8.1, Referenzen zu sämtlichen genutzten Dateien. Darüber hinaus sind auch Informationen über Dateitypen sowie die Folienlayouts enthalten.

```
1  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2  <Types xmlns="http://schemas.openxmlformats.org/package/2006/content-
       types">
3    <Default Extension="png" .../>
4    <Default Extension="jpeg" .../>
5    <Default Extension="rels" .../>
6    <Default Extension="xml" .../>
7    <Default Extension="jpg" .../>
8    <Default Extension="mp4" .../>
9    <Override PartName="/ppt/presentation.xml" .../>
10   <Override PartName="/ppt/slideMasters/slideMaster1.xml" .../>
11   <Override PartName="/ppt/slides/slide1.xml" ContentType="application/
         vnd.openxmlformats-officedocument.presentationml.slide+xml"/>
12   ⋮
13   <Override PartName="/ppt/slides/slide69.xml" ContentType="application
         /vnd.openxmlformats-officedocument.presentationml.slide+xml"/>
14   <Override PartName="/ppt/notesMasters/notesMaster1.xml" .../>
15   <Override PartName="/ppt/handoutMasters/handoutMaster1.xml" .../>
16   <Override PartName="/ppt/commentAuthors.xml" .../>
17   <Override PartName="/ppt/presProps.xml" .../>
18   <Override PartName="/ppt/viewProps.xml" .../>
19   <Override PartName="/ppt/theme/theme1.xml" .../>
20   <Override PartName="/ppt/tableStyles.xml" .../>
```

```
21   <Override PartName="/ppt/slideLayouts/slideLayout1.xml" ContentType="
        application/vnd.openxmlformats-officedocument.presentationml.
        slideLayout+xml"/>
22   ⋮
23   <Override PartName="/ppt/slideLayouts/slideLayout8.xml" ContentType="
        application/vnd.openxmlformats-officedocument.presentationml.
        slideLayout+xml"/>
24   <Override PartName="/ppt/theme/theme2.xml" .../>
25   <Override PartName="/ppt/theme/theme3.xml" .../>
26   <Override PartName="/ppt/notesSlides/notesSlide1.xml" .../>
27   ⋮
28   <Override PartName="/ppt/notesSlides/notesSlide45.xml" .../>
29   <Override PartName="/ppt/comments/comment1.xml" ... />
30   ⋮
31   <Override PartName="/ppt/comments/comment30.xml" ... />
32   <Override PartName="/docProps/core.xml" .../>
33   <Override PartName="/docProps/app.xml" .../>
34 </Types>
```

Listing 8.6: Die Datei */[Content_Types].xml*, mit beispielhaftem Inhalt

In dieser Datei ist die Reihenfolge der Kinder von `<Type>` wichtig. Zuerst kommen `<Default>` Knoten, welche Informationen über die verwendeten Dateitypen beinhalten. Danach kommt das erste `<Override>` Element, welches die Datei *presentation.xml* referenziert. Dann kommen die Folienmaster in aufsteigender Reihenfolge. Nach den Folienmastern kommen die Folien. In dem Beispiel muss nach der Zeile 13 ein neues Element eingefügt werden, welches die neue Folie referenziert. Dazu wird

```
<Override PartName="/ppt/slides/slide70.xml" ContentType="application/
    vnd.openxmlformats-officedocument.presentationml.slide+xml"/>
```

zwischen die Zeilen 13 und 14 geschoben. Danach folgen die Notizenmaster, dann die Handout, jeweils in aufsteigender Reihenfolge. Dann werden *commentAuthors.xml*, *presProps.xml*, *viewProps.xml*, *theme1.xml* und die *tableStyles.xml* referenziert. Darauf folgen die Referenzen zu den Folienlayouts. Falls die neue Folie auf eines der von PowerPoint bereitgestellten Layouts zurückgreift, muss hier nichts verändert werden. In dem Falle, dass ein neues Layout angelegt und genutzt werden soll, muss in Listing 8.6 nach Zeile 22 noch

```
<Override PartName="/ppt/slideLayouts/slideLayout9.xml" ContentType="
    application/vnd.openxmlformats-officedocument.presentationml.
    slideLayout+xml"/>
```

eingefügt werden. Danach folgen weitere Themes, falls welche genutzt werden. Anschließend folgen die Notizen der einzelnen Folien, wieder in aufsteigender Reihenfolge. Es ist nicht nötig, hier einen neuen Knoten für die neue Folie zu erzeugen, da diese Datei bei Bedarf automatisch von PowerPoint erzeugt wird, wenn Notizen zu der neuen Folie erstellt werden. Nun werden die Kommentare in aufsteigender Reihenfolge referenziert. Als letztes werden noch die *core.xml* und die *app.xml* referenziert.

/ppt/slideMasters/slideMaster1.xml

Änderungen an dieser Datei sind nur nötig, wenn ein neues Folienlayout erstellt wurde. Falls kein neues Layout und stattdessen nur eine neue Folie angelegt wird, dann kann diese Datei ignoriert werden.
Wenn ein neues Folienlayout erstellt werden soll, so muss dieses einem Folienmaster zugewiesen werden. Die Datei */ppt/slideMasters/slideMaster1.xml* existiert immer, auch in einer gerade neu erzeugten, leeren Präsentation. Daher wird das Layout mit diesem Master assoziiert. Um ein Layout mit einem Master zu assoziieren, wird das Betreffende Folienlayout in dieser Datei referenziert.

```
1 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2 <p:sldMaster xmlns:p="http://schemas.openxmlformats.org/presentationml
    /2006/main" preserve="1" ...>
3   ⋮
4   <p:sldLayoutIdLst>
5     <p:sldLayoutId id="2147483729" r:id="rId1"/>
6     ⋮
7     <p:sldLayoutId id="2147483736" r:id="rId8"/>
8   </p:sldLayoutIdLst>
9   ⋮
10 </p:sldMaster>
```

Listing 8.7: Die Datei */ppt/slideMasters/slideMaster1.xml*, mit beispielhaftem Inhalt

Erstelle nun einen neuen Knoten `<p:sldLayoutId>` und hänge ihn als neues Kindelement an `<p:sldMaster>` aus Zeile 2 an. Es erhält die Eigenschaften *id* und *r:id*. Da diese beiden Eigenschaften fortlaufend durchnummeriert werden, ergeben sich für unser Beispiel aus Listing 8.7 die Eigenschaften *id="2147483737"* und *r:id="rId9"*. Nun sähe das Beispiel aus Listing 8.7 folgends aus:

```
1 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2 <p:sldMaster xmlns:p="http://schemas.openxmlformats.org/presentationml
    /2006/main" preserve="1" ...>
3   ⋮
4   <p:sldLayoutIdLst>
5     <p:sldLayoutId id="2147483729" r:id="rId1"/>
6     ⋮
7     <p:sldLayoutId id="2147483736" r:id="rId8"/>
8     <p:sldLayoutId id="2147483737" r:id="rId9"/>
9   </p:sldLayoutIdLst>
10  ⋮
11 </p:sldMaster>
```

Listing 8.8: Die Datei */ppt/slideMasters/slideMaster1.xml*, nachdem das Folienlayout in Zeile 8 eingefügt wurde

/ppt/slideMasters/slideMaster1.xml.rels

Auch an dieser Datei sind Änderungen nur nötig, wenn ein neues Folienlayout erstellt wurde. Falls kein neues Layout und stattdessen nur eine neue Folie angelegt wird, dann kann diese Datei ignoriert werden.
Wenn ein neues Folienlayout erstellt werden soll, so muss dieses einem Folienmaster zugewiesen werden. Die Datei */ppt/slideMasters/slideMaster1.xml* existiert immer, auch in einer gerade neu erzeugten, leeren Präsentation. Daher wird das Layout mit diesem Master assoziiert. Um ein Layout mit einem Master zu assoziieren, wird das Betreffende Folienlayout in dieser Datei referenziert.

```
1  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2  <Relationships ... >
3    <Relationship Id="rId8" Target="../slideLayouts/slideLayout8.xml"
       .../>
4    <Relationship Id="rId3" Target="../slideLayouts/slideLayout3.xml"
       .../>
5    <Relationship Id="rId7" Target="../slideLayouts/slideLayout7.xml"
       .../>
6    <Relationship Id="rId2" Target="../slideLayouts/slideLayout2.xml"
       .../>
7    <Relationship Id="rId1" Target="../slideLayouts/slideLayout1.xml"
       .../>
8    <Relationship Id="rId6" Target="../slideLayouts/slideLayout6.xml"
       .../>
9    <Relationship Id="rId5" Target="../slideLayouts/slideLayout5.xml"
       .../>
10   <Relationship Id="rId10" Target="../media/image1.jpeg" .../>
11   <Relationship Id="rId4" Target="../slideLayouts/slideLayout4.xml"
       .../>
12   <Relationship Id="rId9" Target="../theme/theme1.xml" .../>
13 </Relationships>
```

Listing 8.9: Die Datei */ppt/slideMasters/slideMaster1.xml.rels*, mit beispielhaftem Inhalt

In dieser Datei muss ein Knoten `<Relationship>` angelegt werden. Dieses wird als neues Kind an `<Relationships>` aus Zeile 2 angehängt. Die Reihenfolge der Kinder spielt keine Rolle, allerdings muss bei der Vergabe der *Id*s darauf geachtet werden. Die *Id* erhält als Wert den Nachfolger der *Id* der letzten Folie. In dem Beispiel in Listing 8.9 würde dem neuen Element die Eigenschaft *Id="rId9"* zugewiesen werden. Alle anderen *Id*s größer oder gleich *rId9* müssen um eins inkrementiert werden. Es muss also der Knoten

```
<Relationship Id="rId9" Type="http://schemas.openxmlformats.org/
   officeDocument/2006/relationships/slideLayout" Target="../
   slideLayouts/slideLayout9.xml"/>
```

erstellt werden.
Es ist naheliegend, dass die *Id* sich aus der Nummer des neuen Folienlayouts ableiten lässt. Da ich keinen Nachweis dafür gefunden habe und solche Annahmen während der Entwicklung auch an anderen Stellen widerlegt wurden, halte ich den gewählten Ansatz für den sichersten.
Am Ende sieht die unser Beispiel so aus:

```
1  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2  <Relationships ... >
3    <Relationship Id="rId8" Target="../slideLayouts/slideLayout8.xml"
       .../>
```

```
4   <Relationship Id="rId3" Target="../slideLayouts/slideLayout3.xml"
        .../>
5   <Relationship Id="rId7" Target="../slideLayouts/slideLayout7.xml"
        .../>
6   <Relationship Id="rId2" Target="../slideLayouts/slideLayout2.xml"
        .../>
7   <Relationship Id="rId1" Target="../slideLayouts/slideLayout1.xml"
        .../>
8   <Relationship Id="rId6" Target="../slideLayouts/slideLayout6.xml"
        .../>
9   <Relationship Id="rId5" Target="../slideLayouts/slideLayout5.xml"
        .../>
10  <Relationship Id="rId11" Target="../media/image1.jpeg" .../>
11  <Relationship Id="rId4" Target="../slideLayouts/slideLayout4.xml"
        .../>
12  <Relationship Id="rId10" Target="../theme/theme1.xml" .../>
13  <Relationship Id="rId9" Target="../slideLayouts/slideLayout9.xml"
        .../>
14 </Relationships>
```

Listing 8.10: Die Datei */ppt/slideMasters/slideMaster1.xml.rels*, nachdem ein neues `<Relationship>` Element hinzugefügt wurde

/ppt/presentation.xml

```
1  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2  <p:presentation ...>
3    <p:sldMasterIdLst>
4      <p:sldMasterId id="2147483728" r:id="rId1"/>
5    </p:sldMasterIdLst>
6    <p:notesMasterIdLst>
7      <p:notesMasterId r:id="rId71"/>
8    </p:notesMasterIdLst>
9    <p:handoutMasterIdLst>
10     <p:handoutMasterId r:id="rId72"/>
11   </p:handoutMasterIdLst>
12   <p:sldIdLst>
13     <p:sldId id="268" r:id="rId2"/>
14     <p:sldId id="270" r:id="rId3"/>
15     ⋮
16     <p:sldId id="333" r:id="rId51"/>
17     <p:sldId id="334" r:id="rId52"/>
18     ⋮
19     <p:sldId id="311" r:id="rId69"/>
20     <p:sldId id="312" r:id="rId70"/>
21   </p:sldIdLst>
22   ⋮
23 </p:presentation>
```

Listing 8.11: Beispielhafter Inhalt von */ppt/presentation.xml*. Beachte, dass *334* die höchste *id* ist, obwohl das Element nicht das letzte Kind ist

In dieser Datei muss die neue Folie der gesamten Präsentation bekannt gemacht werden. Dazu wird ein neues `<p:sldId>` Element erstellt und dem Knoten `<p:sldIdLst>` aus Zeile 12 als Kindelement angehängt. Die Reihenfolge der Kindsknoten spielt hier aber wohl keine Rolle. Dem neuen `<p:sldId>` Element wird die Eigenschaft *r:id="rId71"* zugewiesen, da diese fortlaufend vergeben werden. Dies führt wieder dazu, dass auch hier alle vorkommenden *r:id*s größer oder gleich *rId71* um eins erhöht werden müssen. In der Praxis bedeutet das, dass die Kinder von `<p:sldMasterIdLst>`, `<p:notesMasterIdLst>` und `<p:handoutMasterLst>` geprüft werden müssen. Die Eigenschaft *id* ergibt sich aus dem Nachfolger der höchsten sich in Verwendung befindenden *id* unter den `<p:sldId>` Elementen. Es sei Angenommen, *334* ist die höchste *id* im Beispiel in Listing 8.11. Dann ergibt sich für das neue `<p:sldId>` Element die Eigenschaft *id="335"*. Das Beispiel sieht nach den Änderungen aus wie folgt:

```
1  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2  <p:presentation ... >
3    <p:sldMasterIdLst>
4      <p:sldMasterId id="2147483728" r:id="rId1"/>
5    </p:sldMasterIdLst>
6    <p:notesMasterIdLst>
7      <p:notesMasterId r:id="rId72"/>
8    </p:notesMasterIdLst>
9    <p:handoutMasterIdLst>
10     <p:handoutMasterId r:id="rId73"/>
11   </p:handoutMasterIdLst>
12   <p:sldIdLst>
13     <p:sldId id="268" r:id="rId2"/>
14     <p:sldId id="270" r:id="rId3"/>
15     ⋮
16     <p:sldId id="333" r:id="rId51"/>
17     <p:sldId id="334" r:id="rId52"/>
18     ⋮
19     <p:sldId id="311" r:id="rId69"/>
20     <p:sldId id="312" r:id="rId70"/>
21     <p:sldId id="335" r:id="rId71"/>
22   </p:sldIdLst>
23   ⋮
24 </p:presentation>
```

Listing 8.12: Geänderter Inhalt von */ppt/presentation.xml*

/ppt/slides/slide#NUMBER#.xml & /ppt/slides/_rels/slide#NUMBER#.xml.rels

Diese Dateien enthalten den Extensible Markup Language (XML) Code für die Folie selbst. Der Inhalt dieser Dateien hängt davon ab, wie die Folie aussehen soll. Die Nummer dieser Datei lässt sich aus der Anzahl der Folien ableiten. Aus den Beispielen zuvor lässt sich ableiten, dass die Dateinamen in diesem Falle */ppt/slides/slide70.xml* respektive */ppt/slides/_rels/slide70.xml.rels* sind. Falls diese Datei ein neues Folienlayout nutzen soll, so muss dieses in */ppt/slides/_rels/slide70.xml.rels* referenziert werden.
Da der Inhalt der Folie und das Layout zeitgleich an die Funktion *appendChild()* der Klasse *OpenXmlPresentationJs* übergeben wird, ist die Nummer des neuen Layouts noch nicht bekannt. Zu diesem Zwecke kann auf den Platzhalter *__NEXT_SLIDE_LAYOUT_NUMBER__* zurückgegriffen werden. Dieser wird, sobald

die Nummer während der Bearbeitung der Dateien bekannt wird, durch diese ersetzt. Das Problem könnte auch anders gelöst werden: Es könnte z.B. eine Funktion *getCurrentNumberOfSlideLayouts()* nach außen hin verfügbar gemacht werden, welche, wie der Name impliziert, die Anzahl der zurzeit existierenden Folienlayouts zurückgibt. Da allerdings in der Theorie Wettlaufsituationen beim Zugriff auf die Präsentation entstehen können, halte ich es für weniger Fehleranfällig, die genaue Nummer erst beim erstellen der Datei festzulegen. Während die in diesem Kapitel genannten Dateien sequenziell verändert werden, werden etwaige Zugriffe von außen in einer Warteschlange hintangestellt. Dadurch werden Wettlaufsituationen und daraus resultierende Komplikationen vermieden.
Da diese Dateien noch nicht existieren, müssen diese angelegt werden.

/ppt/slideLayouts/slideLayout#NUMBER#.xml & /ppt/slideLayouts/_rels/slideLayout#NUMBER#.xml.rels

Offensichtlich müssen diese Dateien nur angelegt werden, falls ein neues Folienlayout erstellt werden soll. Diese Dateien enthalten den XML Code für das Folienlayout. Auch hier hängt der Inhalt der Dateien davon ab, wie die Folie aussehen soll. Die Nummer dieser Datei lässt sich aus der Anzahl der Folienlayouts ableiten. Aus den Beispielen zuvor lässt sich ableiten, dass die Dateinamen in diesem Falle */ppt/slideLayouts/slideLayout9.xml* respektive */ppt/slideLayouts/_rels/slideLayout9.xml.rels* sind. Da diese Dateien noch nicht existieren, müssen diese angelegt werden.

8.2. Referenz zur Quelle neben das Bild setzen

In diesem Abschnitt wird erläutert, wie neben Bildern eine kleine Textbox eingefügt wird, welche als Referenz zur genutzten Quelle dient.
Zuerst müssen die Folien ausfindig gemacht werden, die das Bild, welches wir ersetzt haben, beinhalten. Nehmen wir dazu an, das Bild *image6.png* sei ersetzt worden. Nun müssen alle Dateien am Ort */ppt/slides/_rels/* überprüft werden, ob sie an einer Stelle auf *image6.png* verweisen.

```
1 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2 <Relationships ... >
3   ⋮
4   <Relationship Id="rId3" Target="../media/image6.png" .../>
5   ⋮
6   <Relationship Id="rId7" Target="../media/image6.png" .../>
7   ⋮
8 </Relationships>
```

Listing 8.13: Eine beispielhafte */ppt/slides/_rels/slide7.xml.rels*, in der zwei mal auf das Bild *image6.png* verwiesen wird

Nehmen wir nun an, es gäbe eine Folie 7, welche unser ersetztes Bild *image6.png* zwei mal nutzt. Dann würden wir eine Datei ähnlich dem Beispiel in Listing 8.13 erwarten können. Dort ist zu sehen, dass zwei mal auf unser Bild *image6.png* verwiesen wird. Da es zwei Verweise gibt, existieren auch zwei *Id*s, in diesem Falle *rId3* und *rId7*. Diese *Id*s gilt es nun in der dazugehörigen Foliendatei ausfindig zu machen. Im Falle unseres Beispieles wäre dies die Datei */ppt/slides/slide7.xml*. Natürlich kann das Bild auch zusätzlich noch auf anderen Folien in Verwendung sein. Zur Demonstration der Schritte, die nötig sind, um das gewünschte Textfeld neben einem Bild einzufügen, beschränken wir uns hier darauf, dass das Bild *image6.png* nur auf Folie 7 vorkommt.

```
1  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2  <p:sld ... >
3    <p:cSld>
4      <p:spTree>
5        ⋮
6        <p:pic>
7          ⋮
8          <p:blipFill>
9            <a:blip r:embed="rId7" ... >
10             ⋮
11           </a:blip>
12         ⋮
13         </p:blipFill>
14         <p:spPr bwMode="auto">
15           <a:xfrm>
16             <a:off x="10668000" y="259090"/>
17             <a:ext cx="1368152" cy="1368152"/>
18           </a:xfrm>
19           ⋮
20         </p:spPr>
21       </p:pic>
22     </p:spTree>
23     ⋮
24   </p:cSld>
25   ⋮
26 </p:sld>
```

Listing 8.14: Beispielhafter Ausschnitt der */ppt/slides/slide7.xml*, welcher die relevanten Informationen zu dem Bildelement mit *Id="rId7"* zeigt

In der Datei */ppt/slides/slide7.xml* suchen wir nach einem Element `<p:pic>`, welches einen Nachfolger `<p:blip>` mit der Eigenschaft *r:embed="rId7"* hat. In unserem Falle wäre es der Knoten in Zeile 6, denn er besitzt den gesuchten Nachfolger in Zeile 9. Nun wird von dem `<p:pic>` das Nachfolgeelement `<a:off>` gesucht. Es gibt nur ein solches Element. Wir finden es in Zeile 16. die Eigenschaften *x="10668000"* und *y="259090"* verraten uns den Offset der oberen linken Ecke des Bildes zu einem relativen Ankerpunkt. Nun suchen wir nach dem Nachbarknoten `<a:ext>` in Zeile 17. An dieser Stelle ist anzumerken, dass es Fälle gibt, wo das `<p:pic>` Element mehrere solcher `<a:ext>` Elemente besitzt. Dennoch gibt es immer nur genau einen solchen Knoten als Nachbarn unseres `<a:off>` Elementes. Deshalb ist es sicherer, nach dem Nachbarn zu suchen. Dieser Nachbar speichert

in seinen Eigenschaften *cx="1368152"* und *cy="1368152"* die Höhe (*cy*) und die Breite (*cx*) des Bildes. Wir kommen später wieder auf diese Informationen zurück.

```
1  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2  <p:sp ... >
3    ⋮
4    <p:txBody>
5      ⋮
6      <a:p>
7        <a:r>
8          <a:rPr lang="de-DE" dirty="0"/>
9          <a:t>[1]</a:t>
10       </a:r>
11     </a:p>
12   </p:txBody>
13 </p:sp>
```

Listing 8.15: Beispielhafter Ausschnitt des `<p:sp>` Elements

Zunächst wird ein Element `<p:sp>` mit entsprechender Struktur benötigt. Da es eine vergleichsweise umfangreiche Struktur benötigt, ist in Listing 8.15 nur ein Ausschnitt zu sehen.
Das Tool baut die Knoten nicht alle einzeln nach, sondern liest einen Vorlagenstring ein und parst ihn. Dieser Vorlagenstring wurde durch erstellen eines Textfeldes mit arbiträrem Inhalt in PowerPoint gewonnen. Die Knoten hingegen einzeln per Code zu erstellen hat keine Vorteile und der einzige Nachteil ist, dass das Nachbauen, aufgrund der größe der Struktur, viel Zeit in Anspruch genommen hätte. Dies ist der einzige Grund, weshalb ich diesen Ansatz nicht gewählt habe.
In Zeile 9 von Listing 8.15 ist der Inhalt der Textbox zu sehen, in diesem Falle wäre *[1]* die Referenz zur Quelle des Bildes.
So wie zuvor auch, durchsuchen wir unser neues `<p:sp>` Element nach dem Nachfolger `<a:off>`. Nun benötigen wir die zuvor erhaltenen Informationen über die Position und die Größe des Bildes. Die Offsetwerte ergeben sich aus den Offsetwerten des Bildes und dessen Größe. Wir errechnen den Wert für *x* durch *10668000* (Offset X des Bildes) + *1368152* (Breite des Bildes) und den für *y* durch *259090* (Offset Y des Bildes) + *1368152* (Höhe des Bildes). D.h., wir geben unserem `<p:sp>` Element die Eigenschaften *x="12036152"* und *y="1627242"*. Dies platziert das Textelement an die rechte untere Ecke des Bildes.
Nun wird das `<p:sp>` Element als Nachbar von `<p:pic>` hinzugefügt. Es ist wichtig, dass es als Nachbar hinzugefügt wird, da sonst nicht gewährleistet ist, dass das Textelement und das Bild den gleichen relativen Ankerpunkt verwenden. Ist dies nicht der Fall, und `<p:sp>` wird nicht als Nachbar von `<p:pic>` angehängt, so kann es passieren, dass das Textfeld irgendwo anders auf der Folie erscheint, aber nicht an der rechten unteren Ecke des Bildes.
Diese Schritte müssen dann für jedes Vorkommen eines Bildes wiederholt werden. In unserem Beispiel müssten die oben aufgeführten Schritte nun auch nochmal für ein `<p:pic>` Element, welches einen Nachfolger `<p:blip>` mit der Eigenschaft *r:embed="rId3* hat, ausgeführt werden.

8.3. Text mit Hyperlink erstellen

Es ist möglich, einen Text oder einen Teil des Textes mit einer Verlinkung, einem Hyperlink, auszustatten. Dieser kann z.B. auf eine andere Folie in der Präsentation, eine Datei oder eine Webadresse verweisen. Betrachte das folgende beispielhafte `<a:p>` (Paragraph) Element

```
1 <a:p>
2   <a:r>
3     <a:rPr lang="de-DE" dirty="0"/>
4     <a:t>Unter CC-0 veröffentlicht</a:t>
5   </a:r>
6 </a:p>
```

Listing 8.16: Paragraph mit einfachem Text ohne Link

ähnlich dem `<a:p>` Element aus 8.15 in Zeile 6. Nun soll der Name *CC-0* mit einem Link zu den Lizenzbestimmungen versehen werden. Um das zu erreichen, muss zunächst bekannt sein, auf welcher Folie der Text platziert wird. Nehmen wir an, wir möchten den Text auf Folie 12 platzieren. Dann muss in der Datei */ppt/slides/_rels/slide12.xml.rels* ein neues `<Relationship>` element erstellt werden. Die *Id*s werden aufsteigend vergeben. Wenn nun also nur ein `<Relationship>` Element vorhanden ist, sähe die Datei nach dem Hinzufügen so aus:

```
1 <?xml version="1.0" encoding="UTF-8" standalone="true"?>
2 <Relationships xmlns="http://schemas.openxmlformats.org/package/2006/
    relationships">
3   <Relationship TargetMode="External" Target="https://creativecommons.
      org/publicdomain/zero/1.0/" Type="http://schemas.openxmlformats.
      org/officeDocument/2006/relationships/hyperlink" Id="rId2"/>
4   <Relationship Target="../slideLayouts/slideLayout12.xml" Type="http:
      //schemas.openxmlformats.org/officeDocument/2006/relationships/
      slideLayout" Id="rId1"/>
5 </Relationships>
```

Listing 8.17: Neues `<Relationship>` Element

Die Reihenfolge der einzelnen Knoten ist nicht von Belang. Nun müssen wir noch das Paragraphelement aus Listing 8.15 wie folgt bearbeiten

```
1 <a:p>
2   <a:r>
3     <a:rPr lang="de-DE" dirty="0"/>
4     <a:t>Unter </a:t>
5   </a:r>
6   <a:r>
7     <a:rPr lang="de-DE" dirty="0">
8       <a:hlinkClick r:id="rId2"/>
9     </a>
10    <a:t>CC-0</a:t>
11  </a:r>
12  <a:r>
13    <a:rPr lang="de-DE" dirty="0"/>
14    <a:t> veröffentlicht</a:t>
15  </a:r>
16 </a:p>
```

Listing 8.18: Der Name der Lizenz führt nun zu den Bedingungen

Da hier nur ein Teil des Textes ein Hyperlink werden soll, müssen die anderen Teile des Textes in jeweis separierten `<a:r>` Elementen untergebracht werden. Der zu verlinkende Teil bekommt in Zeile acht ein neues `<a:hlinkClick>` Kindelement zugewiesen. Dieses besitzt die Eigenschaft *r:id="rId2"*, wobei dies die *rId* des zuvor erstellten `<Relationship>` Elements aus Listing 8.17 ist.
Soll statt einer Webadresse eine Folie, sagen wir die siebte, verlinkt werden, so müsste das `<Relationship>` Element so

```
<Relationship Target="slide7.xml" Type="http://schemas.openxmlformats.
    org/officeDocument/2006/relationships/slide" Id="rId2"/>
```

aussehen und das `<a:hlinkClick>` Element müsste zu

```
<a:hlinkClick r:id="rId2" action="ppaction://hlinksldjump"/>
```

geändert werden.
Nun haben wir einen Paragraphen, dessen Text einen Hyperlink besitzt, und können ihn einem `<p:txBody>` Element als Kind anhängen, wie z.B. in Listing 8.15, damit dieser sichtbar wird.

8.4. Kommentar hinzufügen

In diesem Kapitel wird erläutert, welche Dateien manipuliert werden müssen, um einer Folie Kommentare hinzuzufügen. Der Einfachheit halber wählen wir eine Folie, von der wir wissen, dass sie keine Kommentare enthält. Dies können wir annehmen, da das Tool zu dem Zeitpunkt, als es noch geplant war, den Beleg als Kommentar abzuspeichern, nur zu der Disclaimer-Folie Kommentare anlegen sollte. Da diese Folie vom Tool selbst angelegt wurde, sind zu ihr noch keine Kommentare vorhanden, daher war diese Einschränkung in Ordnung.
Kommentare werden in PowerPoint in einer separaten Ansicht angezeigt, sind nicht selbst auf der Folie zu sehen (es wird ein Sprechblasensymbol angezeigt) und während einer Präsentation sehen weder das Publikum noch der oder die Vortragende den Inhalt des Kommentares oder die dazugehörige Sprechblase.

/[Content_Types].xml

```
1 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2 <Types ... >
3   ⋮
4   <Override PartName="/ppt/presProps.xml" ... />
5   ⋮
6   <Override PartName="/ppt/comments/comment3.xml" ... />
7   <Override PartName="/docProps/core.xml" ... />
8   <Override PartName="/docProps/app.xml" ... />
9 </Types>
```

Listing 8.19: Beispielinhalt von */[Content_Types].xml*

In dieser Datei muss eine Referenz zur Datei, welche Informationen über den Author enthält, und eine Referenz zu der neuen Kommentardatei */ppt/comments/comment#NUMBER#.xml* eingefügt werden. Die *#NUMBER#* ist am einfachsten zu erhalten, indem die Dateien am Ort */ppt/comments/* gezählt werden und man diese Zahl um eins erhöht.

Prüfe nun, ob die Datei */ppt/commentAuthors.xml* bereits referenziert wird. Dazu suchen wir nach einem `<Override>` Element mit der Eigenschaft *PartName=/ppt/commentAuthors.xml"*. Falls wir fündig werden, müssen wir uns nicht um eine Referenz dieser Datei kümmern. In dem Falle, dass wir kein solches Element finden, müssen wir eines Anlegen. Dazu wird der Knoten

```
<Override PartName="/ppt/commentAuthors.xml" ContentType="application/
    vnd.openxmlformats-officedocument.presentationml.commentAuthors+xml
    "/>
```

vor der Referenz zu */ppt/presProps.xml* eingefügt. In dem Beispiel aus Listing 8.19 müsste er vor dem Knoten aus der vierten Zeile eingefügt werden.
Nun müssen wir ersteinmal in Erfahrung bringen, wie viele Kommentardateien es bereits gibt. Dazu könnten wir entweder die Anzahl an `<Override>` Elementen zählen, die auf eine Datei an dem Ort */ppt/comments/* verweisen, oder wir zählen direkt die Anzahl der Dateien an dem Ort. Dem obigen Beispiel lässt sich entnehmen, dass zurzeit drei Kommentardateien existieren. Dies bedeutet, dass wir den Knoten

```
<Override PartName="/ppt/comments/comment4.xml" ContentType="
    application/vnd.openxmlformats-officedocument.presentationml.
    comments+xml"/>
```

erstellen müssen. Dieser wird nach dem letzten Verweis auf eine Kommentardatei und vor dem Verweis auf die */docProps.core.xml* eingefügt. In unserem Beispiel muss das Element also zwischen Zeile sechs und sieben eingefügt werden.
Nach der Bearbeitung sieht die Datei */[Content_Types].xml* wie folgt aus:

```
1  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2    <Types ... >
3    ⋮
4    <Override PartName="/ppt/commentAuthors.xml" ... />
5    <Override PartName="/ppt/presProps.xml" ... />
6    ⋮
7    <Override PartName="/ppt/comments/comment3.xml" ... />
8    <Override PartName="/ppt/comments/comment4.xml" ... />
9    <Override PartName="/docProps/core.xml" ... />
10   <Override PartName="/docProps/app.xml" ... />
11 </Types>
```

Listing 8.20: Der geänderte Inhalt von */[Content_Types].xml*

/ppt/commentAuthors.xml

Falls die Datei im vorigen Schritt erst noch referenziert werden musste, dann existiert die Datei noch nicht. Lege dazu die Datei */ppt/commentAuthors.xml* mit folgendem Grundgerüst an:

```
1 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2 <p:cmAuthorLst xmlns:a="http://schemas.openxmlformats.org/drawingml
    /2006/main" xmlns:r="http://schemas.openxmlformats.org/
    officeDocument/2006/relationships" xmlns:p="http://schemas.
    openxmlformats.org/presentationml/2006/main">
3 </p:cmAuthorLst>
```

Listing 8.21: Grundgerüst von */ppt/commentAuthors.xml*

Nun füge folgenden Knoten

```
<p:cmAuthor id="X" name="NAME OF THE AUTHOR" initials="SOMETHING LIKE '
    TA'" lastIdx="Y" clrIdx="0">
  <p:extLst>
    <p:ext uri="{19B8F6BF-5375-455C-9EA6-DF929625EA0E}">
      <p15:presenceInfo xmlns:p15="http://schemas.microsoft.com/office/
          powerpoint/2012/main" userId="None" providerId="None"/>
    </p:ext>
  </p:extLst>
</p:cmAuthor>
```

als neues Kind dem Knoten `<p:cmAuthorLst>` hinzu. Das *X* ist die nächste freie *id*. Der erste Autor in dieser Liste erhält *id="1"*.
Die Eigenschaft *lastIdx="1"* verweist auf den zuletzt abgegebenen Kommentar. Wenn wir die Datei */ppt/commentAuthors.xml* selbst erstellt haben, dann können wir davon ausgehen dass es noch keine Kommentare gibt. Dann ist $Y = 1$. Ansonsten muss nach dem höchsten Wert für die Eigenschaft *lastIdx* gesucht werden. Dieser wird dann um eins inkrementiert und dem *Y* zugewiesen. Nehmen wir nun an, in unserem Beispiel aus Listing 8.20 gäbe es bereits einen Autor, der bisher vier Kommentare abgegeben hat. Dann würde dieses Beispiel, nachdem wir einen neuen Autor hinzugefügt haben, so aus:

```
1  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2  <p:cmAuthorLst ... >
3    <p:cmAuthor id="1" name="Batman" initials="BM" lastIdx="4" clrIdx="0"
       >
4      <p:extLst>
5        <p:ext uri="{19B8F6BF-5375-455C-9EA6-DF929625EA0E}">
6          <p15:presenceInfo userId="None" providerId="None" .../>
7        </p:ext>
8      </p:extLst>
9    </p:cmAuthor>
10   <p:cmAuthor id="2" name="Robin" initials="RO" lastIdx="5" clrIdx="0">
11     <p:extLst>
12       <p:ext uri="{19B8F6BF-5375-455C-9EA6-DF929625EA0E}">
13         <p15:presenceInfo userId="None" providerId="None" .../>
14       </p:ext>
15     </p:extLst>
16   </p:cmAuthor>
17 </p:cmAuthorLst>
```

Listing 8.22: */ppt/commentAuthors.xml* nachdem der neue Autor *Robin* hinzugefügt wurde

/ppt/_rels/presentation.xml.rels

Falls die Datei */ppt/commentAuthors.xml* bereits existierte, so kann diese Sektion übersprungen und mit der nächsten fortgefahren werden.

```
1  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2  <Relationships ...>
3    <Relationship Id="rId3" Target="slides/slide2.xml" .../>
4    ⋮
5    <Relationship Id="rId4" Target="presProps.xml" .../>
```

```
6 </Relationships>
```

Listing 8.23: */ppt/_rels/presentation.xml.rels*, als die Präsenation noch keine Kenntnis von */ppt/commentAuthors.xml* hatte

Anderenfalls muss */ppt/commentAuthors.xml* der Präsentation bekannt gemacht werden. Füge dafür den Knoten

```
<Relationship Id="rId4" Type="http://schemas.openxmlformats.org/
    officeDocument/2006/relationships/commentAuthors" Target="
    commentAuthors.xml"/>
```

dem Element `<Relationships>` als neues Kind hinzu. Die Reihenfolge der Kinder ist unerheblich. Auch hier folgen die *Id*s einer Ordnung. Unser neuer Knoten erhält die *Id* des Elementes mit der Eigenschaft *Target="presProps.xml"*. Nun muss bei allen Elementen, mit Ausnahme unseres neuen Knotens, die *Id* um eins erhöht werden, wenn diese gleich oder größer der *Id* unseres neuen Knotens. Das Beispiel aus Listing 8.23 sieht nun so aus:

```
1 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2 <Relationships ...>
3   <Relationship Id="rId3" Target="slides/slide2.xml" .../>
4   ⋮
5   <Relationship Id="rId5" Target="presProps.xml" .../>
6   <Relationship Id="rId4" Target="commentAuthors.xml" .../>
7 </Relationships>
```

Listing 8.24: Unser Knoten wurde in der sechsten Zeile der Datei */ppt/_rels/presentation.xml.rels* hinzugefügt

/ppt/comments/comment#NUMBER#.xml

In Listing 8.20 haben wir einen Verweis auf */ppt/comments/comment4.xml* erstellt. Daher ergibt sich für *#NUMBER#* der Wert 4 und wir legen die Datei */ppt/comments/comment4.xml* an. Die Datei wird mit

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<p:cmLst xmlns:a="http://schemas.openxmlformats.org/drawingml/2006/main
    " xmlns:r="http://schemas.openxmlformats.org/officeDocument/2006/
    relationships" xmlns:p="http://schemas.openxmlformats.org/
    presentationml/2006/main">
  <p:cm authorId="A" dt="2020-03-12T07:38:42.585" idx="B">
    <p:pos x="-2147483647" y="0"/>
    <p:text>YOUR COMMENT</p:text>
  </p:cm>
</p:cmLst>
```

gefüllt, wobei das `<p:cm>` Element der eigentliche Kommentar ist [71]. Die Eigenschaft *dt* gibt an, wann dieser Kommentar verfasst wurde und wird im W3C XML Schema datetime datatype [71, 72] präsentiert. Der Wert *A* für die Eigenschaft *authorId* ergibt sicht aus der *Id*, welche dem Autor *Robin* in 8.4 zugewiesen wurde. Analog ist *B* von *idx* die Zahl, welche dort der Eigenschaft *lastIdx* zugeteilt wurde.

Das Element `<p:pos>` in der vierten Zeile gibt die relative Lage zur oberen linken Ecke der Folie an [57]. Hier wird der Kommentar weit außerhalb des Sichtfeldes zur linken Seite der Folie platziert. Da das Symbol für den Kommentar nicht höher liegt als die Folie hoch ist, ist die vertikale Scrollleiste nicht von der platziertung betroffen. Allerdings wird nun die horizontale Scrollleiste eingeblendet. Dies hindert nicht daran, die Folie zu bearbeiten oder zu betrachten.

/ppt/slides/_rels/slide#NUMBER#.xml.rels

Ersetze *#NUMBER#* durch die Nummer der Folie, mit der die Kommentardatei assoziiert werden soll. In dem Beispiel in Listing 8.25 verbinden wir die Kommentardatei mit der zweiten Folie.

```
1 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2 <Relationships ... >
3   <Relationship Id="rId1" Target="../slideLayouts/slideLayout12.xml"
      .../>
4 </Relationships>
```

Listing 8.25: Die Beispieldatei */ppt/slides/_rels/slide2.xml.rels*

In dieser Datei müssen wir ein Element erstellen, welches auf die Kommentardatei verweist. Aus dem Beispiel in Listing 8.20 wissen wir, dass unsere Kommentardatei */ppt/comments/comment4.xml* ist. Der Pfad von */ppt/slides/_rels/slide2.xml.rels* zu */ppt/comments/comment4.xml* kann relativ angegeben werden. Da *rId1* die höchste *Id* in Listing 8.25 ist und wir wissen dass unsere Kommentardatei */ppt/comments/comment4.xml* ist, müssen wir folgenden Knoten als Kindelement zu `<Relationships>` in der zweiten Zeile hinzufügen:

```
<Relationship Id="rId2" Type="http://schemas.openxmlformats.org/
   officeDocument/2006/relationships/comments" Target="../comments/
   comment4.xml"></Relationship>
```

Das Beispiel in Listing 8.25 sieht danach folgends aus:

```
1 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2 <Relationships ... >
3   <Relationship Id="rId1" Target="../slideLayouts/slideLayout12.xml"
      .../>
4   <Relationship Id="rId2" Target="../comments/comment4.xml" .../>
5 </Relationships>
```

Listing 8.26: Ausschnitt aus */ppt/slides/_rels/slide2.xml.rels*, nachdem nun auf die Kommentardatei verwiesen wird

8.5. Arbiträre Daten einbetten

Es ist möglich, eigene Daten in OOXML-Dateien persistent einzubetten [83]. Diese Daten sind zwar z.B. über PowerPoint einseh- und bearbeitbar, es ist allerdings durchaus schwierig dorthin zu gelangen, wenn man nicht weiß wonach man sucht. Hier [59] wird erklärt, wie diese Daten zu finden sind. Dazu muss der Punkt 3.V *Custom* geöffnet werden.
Diese Daten werden in der Datei */docProps/custom.xml* gespeichert. Falls diese Datei bereits existiert, so kann mit Kapitel 8.5.2 fortgefahren werden.

8.5.1. Vorbereitung

Zunächst muss die Datei mit einem gewissen Grundgerüst angelegt werden. Schreibe dazu

```
1 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2 <Properties xmlns="http://schemas.openxmlformats.org/officeDocument
    /2006/custom-properties" xmlns:vt="http://schemas.openxmlformats.
    org/officeDocument/2006/docPropsVTypes">
3 </Properties>
```

Listing 8.27: Das Grungerüst. `<Properties>` wird die Benutzerdefinierten Daten beherbergen

in die Datei */docProps/custom.xml*.
Hänge nun

```
<Relationship Id="rIdX" Type="http://schemas.openxmlformats.org/
    officeDocument/2006/relationships/custom-properties" Target="
    docProps/custom.xml"/>
```

als neues Kind an das `<Relationships>` Element in der Datei */_rels/.rels* an. Dabei entspricht das *X* aus *Id="rIdX"* der nächsten freien Ganzzahl. Es muss in der Reihenfolge der Kinder das letzte Kindelement sein.
Außerdem muss

```
<Override PartName="/docProps/custom.xml" ContentType="application/vnd.
    openxmlformats-officedocument.custom-properties+xml"/>
```

als neues Kind zu dem `<Types>` Element in der Datei */[Content_Types].xml* hinzugefügt werden. Auch dieses muss das letzte Kindelement sein.

8.5.2. Daten speichern

Es werden vier verschiedene Datentypen unterstützt [83]: *Text* (String), *Datum* (Date), *Zahl* (Integer/Float) und *Ja oder Nein* (Boolean). Alle Daten, die das Tool abspeichert, werden als zum String serialsiertes Objekt in der JSON [37] gespeichert. Daher wird hier nur erläutert, wie ein String (*Text*) hinzugefügt wird. Die übrigen Datentypen werden hier nicht behandelt.

Um nun das Objekt

```
{example: 'Hallo Welt'}
```

als *String* mit dem Namen *myString* hinzuzufügen, muss es serialisiert werden und als Typ *Text* abgespeichert werden. Füge dazu das Element

```
<property fmtid="{D5CDD505-2E9C-101B-9397-08002B2CF9AE}" pid="X" name="
    myString">
  <vt:lpwstr>{"example":"Hallo Welt"}</vt:lpwstr>
</property>
```

als neues Kind zu `<Properties>` in */docProps/custom.xml* hinzu. Die Reihenfolge der Kinder ist irrelevant. Das *X* ergibt sich aus der nächsten freien Zahl. **Die Zählung beginnt bei 2!**
Der Wert *{D5CDD505-2E9C-101B-9397-08002B2CF9AE}* für das Attribut *fmtid* bleibt immer gleich und verändert sich nicht [30].

Kapitel 9 Docker

Docker [22] ist ein Werkzeug, um sogenannte *Container* zu generieren und zu starten. In der Kurzfassung bilden diese Container eine in sich abgeschlossene Einheit, welche nach außen hin durch freigegebene Ports mit einem Hostsystem oder anderen Containern kommunizieren.

Das gesamte Tool an sich umfasst neben dem Frontend auch das Backend und die Datenbank und diese drei Komponenten werden jeweils durch eine Datei namens *Dockerfile* als Container gebaut und mithilfe einer *docker-compose.yml* Datei zu einer Komposition zusammengefasst. In den nachfolgenden Kapiteln wird im einzelnen erläutert, welche Schritte notwendig sind, um die jeweiligen Container zu erstellen und wie wir sie am Ende zu einer Komposition zusammenfassen.

9.1. Frontend Container

Zunächst muss das Frontend selbst gebaut werden, indem das Angular Projekt zu *.html*, *.css* und *.js* Dateien kompiliert wird. Wir werden insgesamt zwei Container bauen, von denen der erste dazu dient, das Frontend zu bauen und der zweite, um das gebaute Frontend zugänglich und benutzbar zu machen. Nach erfolgreichem Abschluss wird der erste Container gelöscht, da er nicht benötigt wird.

```
# build angular app
FROM node:12.13-alpine as node
WORKDIR /usr/src/app
# copy both, the package.json and the package.lock.json, into the image
COPY package.json ./
COPY package-lock.json ./
# then run npm install to install all dependencies of the project
RUN npm i
# we copy the rest of the project into the image
COPY . .
# run the serve script of the angular project
RUN npm run build

# serve built angular app
FROM nginx:1.17.1-alpine
# remove the nginx's defaultly served files
RUN rm -f /usr/share/nginx/html/*
# copy ng build's output to html folder to be served
COPY --from=node /usr/src/app/dist/frontend /usr/share/nginx/html
# copy nginx config file
```

```
COPY ./nginx.conf /etc/nginx/conf.d/default.conf
```

Listing 9.1: Der Inhalt der *Dockerfile* für das Frontend

Betrachte nun die *Dockerfile* in Listing 9.1. Da Angular [1] auf Node.js [47] aufbaut, wählen wir in Zeile vier ein Nodeabbild. Danach setzen wir in Zeile fünf den Arbeitsordner, also unsere Position innerhalb des Nodeabbildes, auf */usr/src/app* und im Anschluss kopieren wir die *package.json* und die *package-lock.json* an diese Position. Diese beiden Dateien geben an, welche NPM Pakete zum bauen des Frontends benötigt werden. In Zeile zehn werde diese Pakete installiert. Dann kopieren wir die restlichen Dateien des Frontends in den Arbeitsordner. Nun lassen wir in Zeile 14 das Frontend bauen.
Damit wir einen Container erzeugen, der nur die Software installiert hat, die benötigt wird, erstellen wir in Zeile 17 nun einen neuen Container, dieses mal auf Basis eines Abbildes der Webserver-Software NGINX [46]. Nachdem das Abbild heruntergeladen wurde, werden in Zeile 19 zunächst die Standardseiten gelöscht. Danach kopieren wir das gebaute Angular Frontend in den Ordner */usr/share/nginx/html*, damit diese von NGINX dargeboten werden können. In der letzten Zeile wird noch die Konfigurationsdatei für den Webserver an dem korrekten Ort abgelegt.
Nun haben wir einen fertigen Container, der das Frontend beinhaltet. Diesen pushen wir jetzt in die Container Registry [32] des RWTH GitLab, damit der Container von überall aus bezogen werden kann. Dieser wird später für die Komposition in Kapitel 9.4 benötigt.

9.2. Backend Container

Auch das Backend basiert auf Node.js, allerdings nicht auf Angular, da es keine grafische Oberfläche benötigt. Es stellt eine REST [79] API bereit, welche durch das Express Framework [23] ermöglicht wird. Dazu muss das Backend von außerhalb des Containers erreichbar sein.

```
1  # In this image, we transpile the typescript to javascript
   FROM node:12.13-alpine as node
   ENV DIR=/usr/src/app
   WORKDIR $DIR
5  # copy the package.json and ther ts.config into the image
   COPY package.json $DIR
   COPY tsconfig.json $DIR
   # then run npm install in productive mode to install all dependencies
      of the project
   RUN npm i  --production
10 # Add all the content to the WORKDIR
   ADD . /usr/src/app
   # finally transpile the source
   RUN npx tsc --outDir $DIR/dist
   # RUN the server
15 CMD ["node","./dist/server.js"]
   # Expose the port to make communication with the API from the outside
      world possible
   EXPOSE 4201
```

Listing 9.2: Der Inhalt der *Dockerfile* für das Backend

In Listing 9.2 ist der Bauplan für den Backend Container gezeigt. In der zweiten Zeile wird wieder zuerst Node als Abbild gewählt. In Zeile vier wird eine Hilfsvariable eingeführt, welche auf den Arbeitsordner verweist. Wir benötigen dieses mal die *package.json* und *tsconfig.json*. Diese wird zum transpilieren benötigt. Um einen Container zu erzeugen, muss das Backend nicht gebaut werden, da dies auf dem Server läuft und nicht wie das Angular Framework im Browser ausgeführt werden muss. Da das Backend aber in TypeScript [67] geschrieben wurde, muss es zu JavaScript [36] transpiliert werden. Bevor dies geschieht, werden in der neunten Zeile die NPM Abhängigkeiten installiert. Kopiere nun die restlichen Dateien in den Arbeitsordner. In Zeile 13 wird der Code transpiliert, danach wird der so erzeugte Code mittels Node.js ausgeführt. In der letzten Zeile wird der Port 4201 geöffnet, um eine Kommunikation mit der Außenwelt zu ermöglichen.
Auch dieser Container wird jetzt in die Container Registry des RWTH GitLab gepusht und später für die Komposition in Kapitel 9.4 benötigt.

9.3. Datenbank Container

Streng genommen wird kein eigener Container gebaut. Im Gegensatz zu den anderen beiden Fällen muss die Datenbank Daten speichern. Daten in einem Container existieren allerdings nur so lange, wie der Container ausgeführt wird [41]. Nach einem Neustart, z.B. wegen eines Updates oder einer Störung, wären die Daten verloren. Damit die Datenbank die in dem Kapitel 7.2.2 vorgestellten Tabellen und noch ein paar weitere Daten besitzt, müssen diese vorbereitet werden. Dazu wird der Ort in dem Container, an dem die Datenbank gespeichert wird, mit einem Ort außerhalb verknüpft. In unserem Fall liegt die Datenbank am Ort */var/lib/mysql* und soll mit dem Ort */database/mysql/db_state/* in unserem Git Repository verknüpft werden. Dabei ist anzumerken, dass die Datenbank nicht direkt in das Git Repository der GitLab Instanz schreibt, sondern in das lokal geklonte Repository.
Nachdem die zwei Orte verlinkt wurden, wird die Datenbank zum ersten mal ausgeführt. Nun müssen wir die Tabellen aus Kapitel 7.2.2 anlegen. Zusätzlich werden noch ein Mediatyp und die drei für OER infrage kommenden CC Lizenzen (CC0 [13], CC-BY [11], CC-BY-SA [12] in ihren möglichen Versionen angelegt. Jetzt kann die Datenbank abgeschaltet werden. Da der Ort */database/mysql/db_state/* im Git Repository liegt, können wir den Zustand mit einem Commit festhalten. Normalerweise würde dieser Ordner ignoriert werden, da wir nicht jede Änderung an unseren Daten ins Git pushen wollen würden. Dies kann aber erzwungen werden, somit können auf diese Weise auch nachträglich noch Änderungen an der grundlegenden Struktur der Datenbank vorgenommen werden.
Die vorbereitete Datenbank befindet sich nun an einem persistenten Ort und wir können im nächsten Kapitel eine Komposition aus unseren drei Containern definieren.

9.4. Komposition mit *docker-compose.yml*

Mithilfe der Datei */docker-compose.yml* können wir nicht nur mit einem Befehl alle Container gleichzeitig starten oder beenden, sondern wir können auch den Port, über den die Datenbank kommuniziert, nur dem Backend gegenüber freigeben. Dies erhöht die Sicherheit, da die Datenbank nun von niemand anderem als dem Backend erreicht werden kann.

```
1 # props to https://malcoded.com/posts/angular-docker/
  version: '2.1'
```

```
3
services:
5   frontend:
6     container_name: ba-webtool-oer-frontend-container
7     image: registry.git.rwth-aachen.de/lubna_ali81/ba-webtool-oer:front
        -latest
8     ports:
9       - 4050:80
10  database:
11    container_name: ba-webtool-oer-database-container
12    volumes:
13      - './database/mysql/db_state/:/var/lib/mysql'
14    environment:
15      - MYSQL_USER='root'
        - MYSQL_ROOT_PASSWORD='__THE_PASSWORD__'
        - MYSQL_DATABASE='default'
      image: mysql:latest
19  backend:
20    container_name: ba-webtool-oer-backend-container
21    image: registry.git.rwth-aachen.de/lubna_ali81/ba-webtool-oer:back-
        latest
22    ports:
23      - 4201:4201
24    depends_on:
25      - database
```

Listing 9.3: Quasi das Notenpapier unserer Komposition, die */docker-compose.yml*

In Listing 9.3 ist die Datei */docker-compose.yml* zu sehen. Sie ist in YAML [73] verfasst, einer Auszeichnungssprache. Mit dem Schlüsselwort *services* fassen wir unsere Container zusammen und nennen sie schlicht „services“. Der Schlüssel *container_name* lässt uns einen Namen für die erzeugten Container vergeben. Mit selbst vergebenen Namen sind sie leichter auseinander zu halten, da ansonsten zufällig welche erzeugt werden. Der Schlüssel *image* zeigt auf das Abbild, was als Basis genutzt werden soll. Das Frontend und das Backend holen sich ihre Abbilder aus der Container Registry, die Basis für die Datenbank bildet ein leeres MySQL Abbild. Für das Frontend und Backend werden die Ports 80 respektive 4201 geöffnet, damit das Frontend aufgerufen und mit der API kommunizieren kann. Das Backend besitzt außerdem den Schlüssel *depends_on*. Der Eintrag *database* verweist auf den Service *database* aus Zeile zehn. Auf diese Weise kann der Backend Container mit dem Datenbankcontainer sprechen und alle anderen Verbindungen von außen würden abegelehnt.
Der Service *database* führt noch zwei weitere Schlüssel, *environment* und *volumes*. Der Eintrag unter *volumes* verknüpft wieder die beiden Orte */database/mysql/db_state* und */var/lib/mysql* miteinander, damit die Daten weiterhin außerhalb des Containers gespeichert sind. In *environment* befinden sich Argumente für die MySQL-Ausführung. Dort ist definiert, dass der Benutzer „root“ mit dem Passwort „__THE_PASSWORD__“ angelegt werden soll, und dass das Datenbankschema „default“ als Standard ausgewählt werden soll.
Nachdem wir unsere Komposition fertig gestellt haben, können wir sie mit dem Befehl

```
docker-compose up
```

starten und sie lässt sich mit

```
docker-compose down
```

stoppen. Das Frontend lässt sich unter *http://localhost/* erreichen.

Teil IV

Konklusion

Kapitel 10 Konklusion

Das Tool bietet die bisher Möglichkeit, eine Powerpointdatei einzulesen und die in ihr enthaltenen Bilder zu extrahieren. Die extrahierten Bilder können, mithilfe externer Bildersuchmaschinen, gegen OER-konforme Bilder ausgetauscht werden. Im Zuge dessen werden auch weitere Informationen zu dem neuen Bild akquiriert, die für eine korrekte Quellenangabe vonnöten sind. Ein neues Bild kann direkt im Frontend des Tools bearbeitet werden, entweder durch Ausschneiden oder Drehen, um es den Vorgaben des zu ersetzenden Bildes anzupassen.
Statt ein Bild zu ersetzen kann dieses als OER deklariert werden, in dem die zur Quellenangabe benötigten Informationen manuell hinzugefügt werden.
Wenn alle gewünschten Operationen eingegeben wurden, setzt das Tool kleine Textboxen als Referenzen neben die Bilder. Die Referenzen verweisen jeweils auf eine Zeile auf den Quellenfolien, die das Tool einfügt. Auf diesen Quellenfolien werden die Quellen, basierend auf den Informationen, die das Tool zu den einzelnen Bildern erhalten hat, automatisch gemäß der TULLU-Regel [50] angegeben.
An das Ende der Präsentation fügt das Tool eine Haftungsausschlussfolie ein, dessen Text im Frontend des Tools spezifiziert wurde.
Zusätzlich wird jede Powerpointdatei mit einer GUID versehen und die gewählten Aktionen werden an das Backend übermittelt. Diese dienen dazu, die durchgeführten Aktionen nachzuvollziehen und schneller erneut auszuführen, falls das gleiche Bild in einer anderen Datei nocheinmal ersetzt oder als OER deklariert werden soll.

Dass Tool besitzt allerdings Einschränkungen.
Zum Beispiel wurde das Frontend zunächst ohne Rücksicht auf Sicherheit während der Kommunikation mit dem Backend implementiert. Jegliche Kommunikation mit dem Backend geschieht daher unverschlüsselt. Dies trifft auch auf die Kommunikation mit den externen Bildersuchmaschinen zu.
Außerdem unterstützt das Frontend des Tools bisher nur Bilder in den Formaten JPEG, PNG und GIF. Andere Formate wie z.B. BMP oder TIFF werden vom Tool schlicht ignoriert.
Ich habe es bisher auch nicht geschafft herauszufinden, warum Bilder, welche zum Präsentationsdesign gehören (z.B. das Hintergrundbild einer Inhaltsfolie oder der Titelfolie), in der resultierenden Powerpointdatei "aus Datenschutzrechtlichen Gründen" [sic] geblockt werden, auch wenn sie gar nicht vom Tool angefasst oder in irgendeiner anderen Weise manipuliert wurden.

Für die Zukunft könnten außerdem, abgesehen von den Einschränkungen, andere Punkte angegangen werden.

Zum Beispiel könnte ein Benutzersystem samt Anmeldung implementiert werden. Dann könnten Einstellungen wie die Toleranz von Seitenverhältnissen vom Nutzer direkt im Frontend angepasst und gespeichert werden. Aber auch der Name könnte gespeichert werden. Jeder Benutzer könnte darüber hinaus einer Institution angehören, welche jeweils ihre eigene Vorlage für einen Hauftungsausschluss angibt. Auf diese Weise würden Mitarbeiter einer Institution einfacher immer den gleichen Haftungsausschluss verwenden können, was allen Lehrmaterialien einer Institution ein einheitlicheres Aussehen bescheren würde.
Es könnte auch eine Möglichkeit geschaffen werden, vermerkte Operationen einer Sitzung einzusehen und rückgängig zu machen. Dies ist hilfreich, wenn einem ein Fehler unterlaufen ist oder man sich einfach nochmal der Korrektheit der vermerkten Aktionen vergewissern möchte. Nach momentanem Stand müsste das Tool und die Datei neu geladen werden, wenn ein Fehler unterlief. Dies bedeutet zwangsläufig, dass alle anderen Operationen, die korrekt waren, erneut angegeben werden müssen.
Ebenso hilfreich wäre eine Einstellung, ob der Beleg jedes mal mit zum Download angeboten werden soll und ob beiden Dateien, der Beleg und die Präsentation, erst zusammen gezippt und dann anschließend zum Download angeboten werden.
Außerdem könnte es in Zukunft möglich gemacht werden, Powerpointdateien automatisch in ein offenes Dateiformat zu konvertieren, bevor es zum Download angeboten wird.

Anhang A Abkürzungen

OER Open Educational Ressources i
OOXML Office Open Extensible Markup Language 10
XML Extensible Markup Language 57
ER Entity Relationship 39
NPM Node Package Manager 33
IDE Integrated Development Environment 39
MD5 Message-Digest Algorithm 5 41
PNG Portable Network Graphics 17
GIF Graphics Interchange Format 17
API Application Programming Interface 30
CC Creative Commons 11
CORS Cross-Origin Resource Sharing 33
MIT Massachusetts Institute of Technology 8
OCW OpenCourseWare 8
UNESCO United Nations Educational, Scientific and Cultural Organization 11
USU Utah State University 11
GUID Globally Unique Identifier 30
MIME Multipurpose Internet Mail Extension 30
CSV Comma-separated values 23
SDK Software Development Kit 29

Anhang B Literaturverzeichnis

[1] Angular. Zugriff am 02.12.2019. [Online]. Available: https://angular.io

[2] Angular CLI. Zugriff am 02.12.2019. [Online]. Available: https://cli.angular.io/

[3] Angular Material UI component library. Zugriff am 02.12.2019. [Online]. Available: https://material.angular.io

[4] BA OER Converter. Zugriff am 04.04.2020. [Online]. Available: https://documenter.getpostman.com/view/3974771/SzRw1Vzx?version=latest

[5] BA Webtool OER Backend Docs. Zugriff am 16.03.2020. [Online]. Available: https://lubna_ali81.pages.rwth-aachen.de/ba-webtool-oer/docs/backend/index.html

[6] BA Webtool OER Frontend Docs. Zugriff am 16.03.2020. [Online]. Available: https://lubna_ali81.pages.rwth-aachen.de/ba-webtool-oer/docs/frontend/index.html

[7] Basic Types · TypeScript. Zugriff am 13.03.2020. [Online]. Available: http://web.archive.org/web/20200313143712/https://www.typescriptlang.org/docs/handbook/basic-types.html

[8] BRIEF OF AMICI CURIAE THE INTERNET ARCHIVE, PRELINGER ARCHIVES, AND PROJECT GUTENBERG. Zugriff am 31.03.2020. [Online]. Available: http://web.archive.org/web/20200331135634/https://cyber.harvard.edu/openlaw/eldredvashcroft/supct/amici/internet-archive.html

[9] CC Search. Zugriff am 02.12.2019. [Online]. Available: https://search.creativecommons.org/

[10] CC Search. Zugriff am 05.04.2020. [Online]. Available: http://web.archive.org/web/20200403162345/https://search.creativecommons.org/about

[11] Creative Commons — Attribution 4.0 International — CC BY 4.0. Zugriff am 16.03.2020. [Online]. Available: http://web.archive.org/web/20200316022839/https://creativecommons.org/licenses/by/4.0/

[12] Creative Commons — Attribution-ShareAlike 4.0 International — CC BY-SA 4.0. Zugriff am 16.03.2020. [Online]. Available: http://web.archive.org/web/20200316022846/https://creativecommons.org/licenses/by-sa/4.0/

[13] Creative Commons — CC0 1.0 Universell. Zugriff am 15.03.2020. [Online]. Available: http://web.archive.org/web/20200219020647/https://creativecommons.org/publicdomain/zero/1.0/deed.de

[14] Creative Commons Catalog API. Zugriff am 15.03.2020. [Online]. Available: https://api.creativecommons.engineering/v1/

[15] Creative Commons Catalog API. Zugriff am 15.03.2020. [Online]. Available: https://www.flickr.com/search/

[16] Cross-Origin Resource Sharing (CORS) - HTTP | MDN. Zugriff am 16.03.2020. [Online]. Available: http://web.archive.org/web/20200316041519/https://developer.mozilla.org/de/docs/Web/HTTP/CORS

[17] Data Information Analysis Big - Free image on Pixabay. Zugriff am 05.05.2020. [Online]. Available: https://pixabay.com/illustrations/data-information-analysis-big-data-3329993/

[18] Data URLs - HTTP | MDN. Zugriff am 13.03.2020. [Online]. Available: http://web.archive.org/web/20200313175249/https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/Data_URIs

[19] data2type GmbH: XML-Technologien | PresentationML | Übersicht zu PresentationML. Zugriff am 27.02.2020. [Online]. Available: https://docs.microsoft.com/de-de/office/open-xml/open-xml-sdk

[20] Database tools and SQL - Help | PhpStorm. Zugriff am 13.03.2020. [Online]. Available: http://web.archive.org/web/20190615000237/https://www.jetbrains.com/help/phpstorm/relational-databases.html

[21] DeviantArt - Discover The Largest Online Art Gallery and Community. Zugriff am 05.04.2020. [Online]. Available: https://www.deviantart.com/

[22] Empowering App Development for Developers | Docker. Zugriff am 15.03.2020. [Online]. Available: https://www.docker.com/

[23] Express - Node.js web application framework. Zugriff am 16.03.2020. [Online]. Available: http://expressjs.com/

[24] Express cors middleware. Zugriff am 16.03.2020. [Online]. Available: http://web.archive.org/web/20200316041240/https://expressjs.com/en/resources/middleware/cors.html

[25] Git. Zugriff am 16.03.2020. [Online]. Available: https://git-scm.com/

[26] GitHub - conveyGmbH/OpenXmlSdkJs: Open XML SDK for JavaScript by Eric White. Zugriff am 08.05.2020. [Online]. Available: https://github.com/conveyGmbH/OpenXmlSdkJs

[27] Global Open Educational Resources Logo. Zugriff am 15.05.2020. [Online]. Available: https://commons.wikimedia.org/wiki/File:Global_Open_Educational_Resources_Logo.svg

[28] H5P – Create and Share Rich HTML5 Content and Applications RSS. Zugriff am 02.12.2019. [Online]. Available: https://h5p.org

[29] Home | The Metropolitan Museum of Art. Zugriff am 05.04.2020. [Online]. Available: https://www.metmuseum.org/

[30] How to: Set a custom property in a word processing document (Open XML SDK). Zugriff am 25.04.2020. [Online]. Available: http://web.archive.org/web/20200425072810/https://docs.microsoft.com/en-us/office/open-xml/how-to-set-a-custom-property-in-a-word-processing-document

[31] Hypertext Transfer Protocol (HTTP) Status Code Registry. Zugriff am 16.03.2020. [Online]. Available: http://web.archive.org/save/https://www.iana.org/assignments/http-status-codes/http-status-codes.xhtml

[32] Index · Container registry · Packages · User · Help · GitLab. Zugriff am 16.03.2020. [Online]. Available: https://gitlab.com/help/user/packages/container_registry/index/

[33] Internet Archive — Wikipedia. Zugriff am 31.03.2020. [Online]. Available: http://web.archive.org/web/20200331135710/https://de.wikipedia.org/wiki/Internet_Archive

[34] Internet Archive: Digital Library of Free & Borrowable Books, Movies, Music & Wayback Machine. Zugriff am 14.03.2020. [Online]. Available: https://archive.org/

[35] ISO 8601 - Wikipedia. Zugriff am 09.03.2020. [Online]. Available: https://en.wikipedia.org/wiki/ISO_8601

[36] Java | Oracle. Zugriff am 16.03.2020. [Online]. Available: https://www.java.com/de/

[37] JSON. Zugriff am 24.04.2020. [Online]. Available: http://web.archive.org/web/20200424154347/https://www.json.org/json-en.html

[38] JSZip. Zugriff am 05.05.2020. [Online]. Available: https://stuk.github.io/jszip/

[39] Link zur Testpräsentation. Zugriff am 12.05.2020. [Online]. Available: https://rwth-aachen.sciebo.de/s/PYbf4Hi9QFFqIjV

[40] Lizenzhinweisgenerator. Zugriff am 02.12.2019. [Online]. Available: https://lizenzhinweisgenerator.de

[41] Manage data in Docker | Docker Documentation. Zugriff am 16.03.2020. [Online]. Available: http://web.archive.org/web/20200316015553/https://docs.docker.com/storage/

[42] Martin-Notation – Wikipedia. Zugriff am 13.03.2020. [Online]. Available: https://web.archive.org/web/20200313135657/https://de.wikipedia.org/wiki/Martin-Notation

[43] MySQL. Zugriff am 02.12.2019. [Online]. Available: https://www.mysql.com/

[44] MySQL :: MySQL 5.7 Reference Manual :: 11.7 Data Type Storage Requirements. Zugriff am 14.03.2020. [Online]. Available: http://web.archive.org/web/20200314165241/https://dev.mysql.com/doc/refman/5.7/en/storage-requirements.html#data-types-storage-reqs-strings

[45] mysql2 - npm. Zugriff am 13.03.2020. [Online]. Available: http://web.archive.org/web/20150910003522im_/https://www.npmjs.com/package/mysql2

[46] NGINX | High Performance Load Balancer, Web Server, & Reverse Proxy. Zugriff am 16.03.2020. [Online]. Available: https://www.nginx.com/

[47] Node.js. Zugriff am 02.12.2019. [Online]. Available: https://nodejs.org/en/

[48] Nps,net promoter score,promoter,detractors,survey feedback - free image from needpix.com. Zugriff am 04.05.2020. [Online]. Available: http://web.archive.org/web/20200504184009/https://www.needpix.com/photo/1524931/nps-netpromoterscore-promoter-detractors-surveyfeedback-opinion-customer-satisfaction-review

[49] OER Converter 0.5.0. Zugriff am 15.03.2020. [Online]. Available: https://oertool.elearn.rwth-aachen.de/main

[50] OER leichtgemacht mit der TULLU-Regel. Zugriff am 03.12.2019. [Online]. Available: https://open-educational-resources.de/oer-tullu-regel/

[51] Online Book Catalog - Overview - Project Gutenberg. Zugriff am 31.03.2020. [Online]. Available: https://www.gutenberg.org/catalog/

[52] Open XML SDK 2.5 für Office. Zugriff am 27.02.2020. [Online]. Available: https://docs.microsoft.com/de-de/office/open-xml/open-xml-sdk

[53] Open XML SDK for JavaScript | Eric White. Zugriff am 08.05.2020. [Online]. Available: http://www.ericwhite.com/blog/open-xml-sdk-for-javascript/

[54] PhpStorm: The Lightning-Smart IDE for PHP Programming by JetBrains. Zugriff am 13.03.2020. [Online]. Available: http://web.archive.org/web/20200308141156/https://www.jetbrains.com/phpstorm/

[55] Pixabay License. Zugriff am 15.05.2020. [Online]. Available: https://pixabay.com/service/license/

[56] Polar Bear Mit Cubs Stock-Foto - Getty Images. Zugriff am 15.05.2020. [Online]. Available: https://www.gettyimages.de/detail/foto/polar-bear-with-cubs-lizenzfreies-bild/184961278

[57] Position Class (DocumentFormat.OpenXml.Presentation). Zugriff am 13.03.2020. [Online]. Available: http://web.archive.org/web/20200313135827/https://docs.microsoft.com/en-us/dotnet/api/documentformat.openxml.presentation.position?redirectedfrom=MSDN&view=openxml-2.8.1

[58] Projects · Dashboard · GitLab. Zugriff am 16.03.2020. [Online]. Available: https://git-scm.com/

[59] Properties of a Presentation in PowerPoint 2010 for Windows. Zugriff am 24.04.2020. [Online]. Available: http://web.archive.org/web/20170715222745/http://www.indezine.com/products/powerpoint/learn/share/2016/advanced-presentation-properties.html

[60] RFC 1321 - The MD5 Message-Digest Algorithm. Zugriff am 13.03.2020. [Online]. Available: http://web.archive.org/web/20200130123139/https://tools.ietf.org/html/rfc1321

[61] RFC 4180 - Common Format and MIME Type for Comma-Separated Values (CSV) Files. Zugriff am 05.05.2020. [Online]. Available: https://tools.ietf.org/html/rfc4180

[62] Sass: Syntactically Awesome Style Sheets. Zugriff am 05.05.2020. [Online]. Available: https://sass-lang.com/

[63] Suchen | Flickr. Zugriff am 15.03.2020. [Online]. Available: http://web.archive.org/web/20200303163844/https://www.flickr.com/services/api/

[64] The Access Compromise and the 5th R, by D. Wiley. Zugriff am 29.11.2019. [Online]. Available: https://opencontent.org/blog/archives/3221

[65] The first single application for the entire DevOps lifecycle - GitLab. Zugriff am 16.03.2020. [Online]. Available: https://about.gitlab.com/

[66] tutory.de - Arbeitsblätter online gestalten. Zugriff am 02.12.2019. [Online]. Available: https://www.tutory.de

[67] TypeScript - JavaScript that scales. Zugriff am 02.12.2019. [Online]. Available: https://www.typescriptlang.org/

[68] Wikimedia Commons. Zugriff am 05.04.2020. [Online]. Available: https://commons.wikimedia.org/wiki/Main_Page

[69] Wikipedia – Die freie Enzyklopädie. Zugriff am 31.03.2020. [Online]. Available: http://web.archive.org/web/20200331144516/https://de.wikipedia.org/wiki/Wikipedia:Hauptseite

[70] Willkommen | LibreOffice - Deutschsprachiges Projekt - Freie Office Suite . Zugriff am 01.04.2020. [Online]. Available: http://web.archive.org/web/20200401065440/https://de.libreoffice.org/

[71] Working with comments (Open XML SDK). Zugriff am 13.03.2020. [Online]. Available: http://web.archive.org/web/20200313140109/https://docs.microsoft.com/en-us/office/open-xml/working-with-comments

[72] XML Schema Date/Time Datatypes. Zugriff am 12.03.2020. [Online]. Available: https://www.w3schools.com/XML/schema_dtypes_date.asp

[73] YAML – Wikipedia. Zugriff am 16.03.2020. [Online]. Available: http://web.archive.org/web/20200316032445/https://de.wikipedia.org/wiki/YAML

[74] D. E. Atkins, J. S. Brown, and A. L. Hammond, "A review of the open educational resources (oer) movement: Achievements, challenges, and new opportunities," zugriff am 30.03.2020. [Online]. Available: https://hewlett.org/wp-content/uploads/2016/08/ReviewoftheOERMovement.pdf

[75] M. Braßler, *Videos als Open Educational Resources (OER) - Tipps und Tricks für die Erstellung mit Studierenden*. pedocs, 02 2017, pp. 207–237.

[76] T. Caswell, S. Henson, M. Jensen, and D. Wiley, "Open Educational Resources: Enabling universal education," *The International Review of Research in Open and Distance Learning*, vol. 9, 01 2008.

[77] P. P.-S. Chen, "The Entity-Relationship Model—toward a Unified View of Data," *ACM Trans. Database Syst.*, vol. 1, no. 1, p. 9–36, Mar. 1976. [Online]. Available: https://doi.org/10.1145/320434.320440

[78] M. Ebner, "OER-Certification in Higher Education," in *World Conference on Educational Media and Technology*, 06 2018.

[79] R. Fielding, "Architectural Styles and the Design of Network-based Software Architectures," Ph.D. dissertation, University of California, Irvine, 01 2000.

[80] N. Friesen, "Open Educational Resources: New Possibilities for Change and Sustainability," *The International Review of Research in Open and Distance Learning*, vol. 10, 11 2009.

[81] C. Goldberg, "Auditing Classes at M.I.T., on the Web and Free," *The New York Times*, April 2001, Zugriff am 15.05.2020. [Online]. Available: https://www.nytimes.com/2001/04/04/us/auditing-classes-at-mit-on-the-web-and-free.html

[82] J. Hilton III, D. Wiley, J. Stein, and A. Johnson, "The four 'R's of openness and ALMS analysis: Frameworks for open educational resources," *Open Learning*, vol. 25, pp. 37–44, 02 2010.

[83] International Electrotechnical Commission (IEC), *Information technology — Document description and processing languages — Office Open XML File Formats —Part 4:Transitional Migration Features*, ISO ISO/IEC 29 500-4:2016, 2016.

[84] S. Ovadia, "Addressing the Technical Challenges of Open Educational Resources," *portal: Libraries and the Academy*, vol. 19, pp. 79–93, 01 2019.

[85] D. Wiley, "Expert Meeting on Open Educational Resources," Zugriff am 30.03.2020. [Online]. Available: http://web.archive.org/web/20200211022712/http://www.oecd.org/education/ceri/36224377.pdf

[86] L. Yuan, S. MacNeill, and W. Kraan, "Open Educational Resources – Opportunities and Challenges for Higher Education."

Eidesstattliche Versicherung

Aufdermauer, Patrick	356057
Name, Vorname	Matrikelnummer (freiwillige Angabe)

Ich versichere hiermit an Eides Statt, dass ich die vorliegende Arbeit/Bachelorarbeit/ Masterarbeit* mit dem Titel

Webbasiertes Analysetool zum Ersetzen von Medienelementen

selbstständig und ohne unzulässige Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt. Für den Fall, dass die Arbeit zusätzlich auf einem Datenträger eingereicht wird, erkläre ich, dass die schriftliche und die elektronische Form vollständig übereinstimmen. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Aachen , 15. Mai 2020	
Ort, Datum	Unterschrift

*Nichtzutreffendes bitte streichen

Belehrung:

§ 156 StGB: Falsche Versicherung an Eides Statt
Wer vor einer zur Abnahme einer Versicherung an Eides Statt zuständigen Behörde eine solche Versicherung falsch abgibt oder unter Berufung auf eine solche Versicherung falsch aussagt, wird mit Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft.

§ 161 StGB: Fahrlässiger Falscheid; fahrlässige falsche Versicherung an Eides Statt
(1) Wenn eine der in den §§ 154 bis 156 bezeichneten Handlungen aus Fahrlässigkeit begangen worden ist, so tritt Freiheitsstrafe bis zu einem Jahr oder Geldstrafe ein.
(2) Straflosigkeit tritt ein, wenn der Täter die falsche Angabe rechtzeitig berichtigt. Die Vorschriften des § 158 Abs. 2 und 3 gelten entsprechend.

Die vorstehende Belehrung habe ich zur Kenntnis genommen:

Aachen , 15. Mai 2020	
Ort, Datum	Unterschrift