

Monitoring Akademischer Arbeiten im Feld Erdwissenschaften

**Projektarbeit im Rahmen der Grundausbildung des
Universitätslehrganges *Library and Information Studies* an der
Österreichischen Nationalbibliothek**

Konzept für ein automatisiertes Monitoring akademischer Arbeiten
im Bereich der Erdwissenschaften (Meteorologie, Geophysik und
Geologie) an österreichischen Universitäten

Eingereicht von:

Rabea Rudigier, BA BA

Christopher Vadeanu, BA MA

Wien, im September 2020

Inhaltsverzeichnis

1. Einleitung	3
2. Theoretische Einführung	5
2.1. Was ist Harvesting?	5
2.2 Wie läuft Harvesting ab?	5
2.3 Wie sollen/werden diese Daten gesichert?	7
2.4 Vorteile des Zusammenführens von Metadaten in Repositorien?	8
2.5 Harvesting in der Praxis. Ein Vergleich	8
2.5.1 Österreichische Nationalbibliothek	9
2.5.2 Schweizer Nationalbibliothek	9
2.5.3 Deutsche Nationalbibliothek	10
2.6 Wie sammeln andere Einrichtungen Hochschulschriften?	10
2.6.1 Universität Wien	11
2.6.2 ProQuest Dissertations & Theses Globa	12
2.7 Fazit	12
3. Praktische Umsetzung für die ZAMG	13
3.1 Crawler	13
3.2 Metadaten	17
3.3 Metadatenstandards	17
3.3.1 Dublin Core	18
3.3.2 MODS	18
3.3.3 Marc21	18
3.4 Austauschformate	19
3.4.1 XML	19
3.4.2 RDF	21
3.5 Softwaredokumentation	24
3.5.1 Eindeutiger Identifier	25
3.5.2 Python	28
3.5.3 SQL & MySQL	31
3.6 Virtualisierungssoftware	32
3.6.1 Oracle VirtualBox	34
3.6.2 Docker	37
3.7 Programm-Dokumentation	39
3.7.1 Der Crawler	40
3.7.2 Die Thesis-Klasse	40
3.7.3 Der Parser	43
3.7.4 Die Webapplikation	44
3.7.5 Die Docker Anwendung	47
4. Conclusio	50

1. Einleitung

Im Rahmen des Grundlehrgang „Library und Information Studies“ wurde im Auftrag von Hofrat Mag. Rainer Stowassers, Leiter Bibliothek und Archive der Zentralanstalt für Meteorologie und Geodynamik (ZAMG) sowie Hofrat Mag. Thomas Hofmann, Leiter der Bibliothek der Geologischen Bundesanstalt, ein Konzept für ein automatisches Monitoring akademischer Abschlussarbeiten im Bereich der Erdwissenschaften (Meteorologie, Geophysik und Geologie) an allen österreichischen Universitäten erstellt.

Die Ausgangslage präsentierte sich dabei wie folgend: Bis 2002 wurden alle Hochschulschriften von der Österreichischen Nationalbibliothek (ÖNB) gesammelt. Seitdem werden Bachelor- und Masterarbeiten von den jeweiligen Universitäten archiviert. Ab diesem Zeitpunkt erstreckt sich die Sammel- und Archivierungstätigkeit der ÖNB nur noch auf Dissertationen, welche an Österreichischen Universitäten beurteilt wurden.

Bachelor- und Masterarbeiten werden aufgrund der Änderung vorwiegend von den Institutionen veröffentlicht oder archiviert, an denen der Abschlussgrad erworben wurde. Die Sammlung der Metadaten zu den Arbeiten erfolgt momentan weitgehenden „händisch“ und uneinheitlich. Ziel des Projekts, ist es deshalb, eine Plattform zu schaffen, die einheitlich alle akademischen Abschlussarbeiten im Bereich der Erdwissenschaften und ihren Unterdisziplinen sammelt und präsentiert.

Die Herangehensweise an der Erstellung des Konzepts lag in einer zweigeteilten Ausarbeitung. Einerseits wurde ein Vergleich mit anderen Universitäten, beziehungsweise Bibliotheken, von Rabea Rudigier vorgenommen. Über eine Literaturrecherche soll herausgefunden werden wie andere Bibliotheken, die bereits Erfahrungen in dem Bereich gemacht haben, vorgehen.

In einem zweiten Teil wurde von Christopher Vadeanu, mit Unterstützung von Andreas Predikaka, ein Harvest-Prototyp erstellt. Dieser Prototyp soll aufzeigen wie einfach es sein könnte, wenn Universitäten ihre Hochschulschriften auf einem bestimmten Ort bereitstellen würden. Dort kann der Crawler anschließend zugreifen.

Eine ursprüngliche Herangehensweise mittels eines Crawlers, der vom österreichischen Bundesheer¹ bereitgestellt werden sollte, Echtdaten zu sammeln, war aufgrund der Corona-Krise 2020 leider nicht möglich. Aufgrund dieser Einschränkungen wurde das Projekt an die gegebene Situation angepasst.

¹ vgl. Klaus Mak/Joachim Klerx/Hans Christian Pilles/Johannes Göllner (Hrsg.), Wissensentwicklung mit „Crowd OSInfo“. Eine Innovation des Cyber Documentation & Research Center (CDRC) der Zentraldokumentation (ZentDok), Landesverteidigungsakademie (LVAk) , Wien 2015

2. Theoretische Einführung

Der erste Teil dieses Projektes soll zunächst darauf eingehen, was Harvesting von Metadaten überhaupt bedeutet und soll aufzeigen, welche möglichen Vor- und Nachteile bei diesem Vorgang gegeben sind. Im Anschluss sollen Beispiele genannt werden, um den zunächst theoretischen Teil mit Anwendungen aus der Praxis zu verdeutlichen.

2.1. Was ist Harvesting?

Immer mehr wissenschaftliche Abschlussarbeiten werden heutzutage online zur Verfügung gestellt. Durch die technische Entwicklung im Bereich von Computer und Internet, hat sich die Publikationspraxis in den letzten Jahren stark verändert. Dies bedeutet für viele Universitäten und Bibliotheken, dass sie sich den neuen technischen Entwicklungen anpassen und ihre Kompetenz in diesem Bereich ausbauen müssen. Für sie stellt sich die Frage, wie diese Abschlussarbeiten gesammelt und auf längere Zeit hin archiviert werden können.

Allgemein lässt sich festhalten:

„Als Harvesting bezeichnet man das automatische <Einsammeln> von Daten bzw. Metadaten aus Archiven und Repositorien über sogenannte Data-Provider. Für diesen Vorgang werden sogenannte Harvesting-Protokolle verwendet, die die Daten automatisiert abgreifen.“²

Sollen also verteilt liegende Informationen und Daten den NutzerInnen an einem einzigen Ort zur Verfügung gestellt werden, muss ein Metadateninformationssystem auf Katalogsdienste zugreifen können. Während diesem Zugriff, werden Metadatensätze aus dem Katalog eingesammelt und zusammengeführt.

2.2 Wie läuft Harvesting ab?

Zunächst wurde geklärt, was Harvesting überhaupt ist. Nun stellt sich die Frage, welche Voraussetzungen gegeben sein müssen und wie ein Harvesting abläuft.

Zunächst muss zwischen beiden Parteien (Client und Server) ein Protokoll vereinbart werden, nach welchen Kriterien und Abläufen die Daten eingesammelt werden. Dieses

² Harvesting, online unter: <https://www.forschungsdaten.info/praxis-kompakt/glossar/> (Letzter Zugriff: 16.08.2020).

Protokoll setzt Regeln für Format, Inhalt und Reihenfolge der Daten fest. Hierfür können bereits existierende Standards³ zum Einsatz kommen.

In diesem Fall ist das spezifische Ziel, neue Hochschulschriften zu finden und einzusammeln. Das bedeutet, dass Server und Client in einem Austausch stehen müssen, damit überprüft werden kann, welche Schriften bereits vorhanden und abrufbar sind, und welche noch gesammelt werden müssen. Wurden diese Informationen miteinander abgeglichen, wird dem Client anschließend nur jene Informationen geliefert, welche er noch nicht besitzt.

Wie häufig ein solcher Harvest vorgenommen wird, entscheidet der Client und kann sehr unterschiedliche Zeitintervalle umfassen (täglich bis mehrjährige Intervalle). Entscheidend ist hier vor allem die Häufigkeit, von der ausgegangen werden kann, dass neue Metadaten zur Verfügung gestellt werden. Zusammengefasst läuft der Vorgang wie folgt ab: Der Client schickt eine Anfrage an den Server. Dieser antwortet dem Client damit, dass er eine Liste mit Elementen schickt, welche Informationen zu den neuen Hochschulschriften beinhalten.⁴

Dabei können die Listen zwei unterschiedliche Formen an Informationen aufweisen:

1. Die Liste besteht aus Paketen, welche sowohl die Metadaten als auch die Publikation selbst beinhaltet. Hier wäre der Nachteil, dass es sich um eine sehr große Menge an Daten handelt.
2. Die Liste weist ausschließlich die Metadaten der neuen Hochschulschriften auf. Hier wäre der Vorteil, dass jene Datenmenge um ein vielfaches kleiner ausfallen würde. Hinzu kommt, dass die Aktualisierung der Metadaten leichter durchgeführt werden könnte, ohne dass die Publikation selbst nochmals heruntergeladen werden muss.

Welche der Form gewählt wird, hängt demnach mit der Verfügbarkeit der Publikation in diversen Formaten (u. a. PDF), aber vor allem auch - wie bereits erwähnt wurde - von jener Kapazität, welche dem Client zur Verfügung steht.

³ Hierzu zählen HTTP, XML, etc.

⁴ vgl. Jürgen Kett/Thomas Seidel, Automatisiertes Abliefern über Harvesting-Verfahren. Wege zur effizienten Ablieferung von Netzpublikationen, Leipzig/Frankfurt am Main 2020, online unter: <https://d-nb.info/1020283033/34> (letzter Zugriff 17.08.2020).

Weiter oben wurde bereits auf das Protokoll verwiesen, welches die Grundlage für den Harvest der Daten bildet. Eines der verbreitetsten Protokoll ist das OAI-Protokoll. OAI steht dabei für „Open Archives Initiative“. Diese entwickelt Standards für die Zusammenarbeit und den Datenaustausch von Daten. Das Open Archives Metadata Harvesting Protocol ist das Hauptwerk der OAI. Das darin entwickelte Framework kann von interessierten Institutionen genutzt werden, *„um über das OAI-Protokoll an einem Peer-to-Peer-Verbund mit anderen Institutionen teilzunehmen [...]“*⁵.

Folgende Vorteile beinhaltet dieses Protokoll:

- ist für das Harvesting von Metadaten ausgelegt
- Standard, der offen ist
- Viele Informationssystem und Dienstleister unterstützen diesen Standard
- Relativ einfaches Protokoll, welches mit Werkzeugen und Implementierungen ausgestattet

Neben der OAI zählt vor allem im Deutschen Sprachraum die „Deutsche Initiative für Netzwerkinformationen“ (DINI)⁶ zu den wichtigsten Initiativen.

2.3 Wie sollen/werden diese Daten gesichert?

Eine Möglichkeit hierfür könnte die Etablierung eines Dokumentenservers darstellen, der Hochschulschriften aufbewahrt. Denn schon seit geraumer Zeit stellt sich für viele Institutionen die Frage nach der Langzeitsicherung ihrer digitalen Daten. Es kann nicht mit absoluter Sicherheit davon ausgegangen werden, dass die heutigen Datenträger auch noch in Zukunft verwendet werden können bzw. kann nicht mit Sicherheit davon ausgegangen werden, dass die Daten ohne Verluste auf jenen Datenträgern gespeichert werden können. Hinzu kommt, dass eine Vielzahl von unterschiedlichen Datenformaten existiert, bei denen ebenfalls nicht davon auszugehen ist, dass sie über mehrere Jahrzehnte relevant und lesbar bleiben. Schon in jener kurzen Zeit, in der eine digitale Archivierung stattfindet, können heutzutage schon viele Datenformate nicht mehr gelesen werden, was zur Konsequenz hat, dass die darin enthaltenen Informationen zum Großteil

⁵ Sebastian Schulz, Ein Hochschulschriften-Server für die SLUB Dresden, Frankfurt (Oder), online unter: <https://tud.qucosa.de/api/qucosa%3A24847/attachment/ATT-0/> (letzter Zugriff: 16.08.2020), S. 38.

⁶ vgl. https://de.wikipedia.org/wiki/Deutsche_Initiative_für_Netzwerkinformation (letzter Zugriff: 16.08.2020).

verloren sind. Deshalb kann mit großer Wahrscheinlichkeit davon ausgehen, dass auch jene Datenformate, die in unserer Zeit relevant erscheinen, in Zukunft an Bedeutung abnehmen werden.

Es existiert eine Vielzahl von Plattformen und Datenbanken, auf denen die elektronischen Arbeiten zur Verfügung gestellt werden können. Für BenutzerInnen kann die Recherche nach geeignetem Material dadurch jedoch sehr mühsam und aufwendig werden, wenn alle Oberflächen für sich selbst nach Daten durchsucht werden müssen. Es wird vorausgesetzt, dass der Benutzer weiß, wo er seine Recherche ansetzen muss und so seine Informationen finden kann. Auch wird dabei vorausgesetzt, dass er zumindest ein Grundwissen über die verschiedenen Informationsspeicher verfügt. Um dem vorzubeugen, bzw. die Recherche zu vereinfachen, wäre die Schaffung eines einheitlichen Zugangs zu den heterogenen Datenquellen ratsam.

Die Voraussetzung dafür ist zunächst die Vereinigung der einzelnen Datenquellen. Das bisherige dezentrale System benötigt dafür gemeinsame Schnittstellen und Protokollen, damit NutzerInnen systemübergreifend arbeiten können. Grundlage einer solchen systemabgreifenden Lösung müssen dabei Standards sein, die von allen eingehalten werden.

2.4 Vorteile des Zusammenführens von Metadaten in Repositorien?

- Virtuell können thematisch zusammengehörende Arbeiten an einem Ort abgelegt werden, was den NutzerInnen die Recherche erleichtert
- Durch die zur Verfügungsstellung der Arbeiten in (globalen) Repositorien wird die Reichweite der Abschlussarbeit um eine Vielzahl gesteigert
- Viele Repositorien bieten darüber hinaus zusätzliche Werkzeuge, die u. a. das Erstellen von Publikationslisten und Bibliographien ermöglichen

2.5 Harvesting in der Praxis. Ein Vergleich

Um die verschiedenen Vorgehensweisen beim Harvesting verdeutlichen zu können, sollen in der Folge die Nationalbibliotheken aus Österreich, Deutschland und der Schweiz miteinander verglichen werden. Es ist hierbei anzumerken, dass sich die Beispiele auf das Sammeln von Webseiten und nicht, auf Hochschulschriften beziehen. Es soll hier vor allem

gezeigt werden, welche verschiedenen Vorgehensweisen gewählt werden können, sowie welche Informationen für die diversen Nationalbibliotheken als relevant aufscheinen und wie diese für die Endnutzer und -nutzerinnen zur Verfügung gestellt werden.

2.5.1 Österreichische Nationalbibliothek

Seit dem 1. März 2009 archiviert die Österreichische Nationalbibliothek Webseiten und Dokumente mit inhaltlichen Bezügen zu Österreich. Die Sammlung der Daten erfolgt dabei nach drei Schwerpunkten: regelmäßige Archivierung österreich-spezifischer Domains, thematische oder event-bezogene Crawls.⁷

2.5.2 Schweizer Nationalbibliothek

Die Schweizer Nationalbibliothek entschied sich alleinig für die selektive Strategie. Auf der Webseite wird der Crawl wie folgt beschrieben:

„Der Schwerpunkt liegt auf frei zugänglichen, landeskundlich relevanten Websites mit einem starken Bezug zur Schweiz: Websites über die Kantone und Gemeinden, spezifische Fachgebiete wie Sozialwissenschaften oder Schweizer Literatur. Die Auswahl treffen vor allem die Schweizer Kantonsbibliotheken und weitere Spezialbibliotheken.“⁸

Im Gegensatz zur Österreichischen Nationalbibliothek sieht die Schweizer Nationalbibliothek momentan von einem Domain-Harvesting der .ch-Domain ab. So wollen die Eidgenossen keine vollständige Sammlung von Webseiten aufbauen, sondern ehe eine Momentaufnahme kreieren, die den künftigen Generationen zur Verfügung gestellt werden soll. Die Auswahl findet hierbei durch die diversen Schweizer Kantonsbibliotheken und Spezialbibliotheken statt, diese sollen mit ihrer Kenntnis und Expertise eine repräsentative Auswahl an Webseiten auswählen. Der Crawl selbst erfolgt grundsätzlich einmal pro Jahr.

⁷ Anm. Zu den Österreich spezifischen Domains zählt die gesamte .at-Domain, welche gleichzeitig die .ac.at und .gv.at Domains beinhaltet. Darüber hinaus werden auch alle Webseiten mit den Endungen .wien und .tirol archiviert. Darüber werden weitere Webseiten gecrawlt, wenn diese thematisch relevant erscheinen. Im Zuge wichtiger Ereignisse können Webseiten nach diesen Schwerpunkten untersucht und archiviert werden.

⁸ Schweizer Nationalbibliothek, FAQ zur Webarchivierung, online unter: <https://www.nb.admin.ch/snl/de/home/fachinformationen/e-helvetica/webarchiv-schweiz/faq-zu-webarchivierung.html> (Letzter Zugriff: 20.08.2020).

Beim Harvesting kommt der „Heritrix“-Crawler zur Anwendung. Daneben wird die „PhantomJS“-Software eingesetzt. Der eigentliche Zugriff auf die gesammelten Seiten erfolgt anschließend mit der „Wayback Machine“

Nach dem Harvesting wird die Qualität des Sammelvorgangs überprüft. Entspricht die Webseite der gewünschten Qualität nicht, wird mit veränderter Einstellung nochmals geharvestet, um das gewünschte Ziel zu erreichen.

2.5.3 Deutsche Nationalbibliothek

Das Webarchiv der Deutschen Nationalbibliothek umfasst Webseiten der .de-Domain. Zusätzlich werden separat Webseiten zu ausgewählte Themen, Vorkommnissen oder Institutionen gesammelt und gespeichert. Bezüglich der Institutionen sind für die Deutsche Nationalbibliothek vor allem Webseiten von Bundesbehörden, Interessenverbänden sowie Kultureinrichtungen von besonderer Relevanz. Die gespeicherten Webseiten können mit Hilfe von Volltextsuche durchforstet werden, der Zugriff kann jedoch nur in den Lesesälen in Leipzig sowie Frankfurt am Main vor Ort erfolgen. Laut den Angaben der Nationalbibliothek wird für jede Seite individuell festgelegt, in welchen Abständen diese geharvestet wird, grundsätzlich werden die Webseiten jedoch einmal pro Halbjahr eingesammelt.⁹

Im Gegensatz zu Webseiten werden Socialmedia Seiten nicht grundsätzlich gesammelt. Ereignisse- oder Themenspezifisch kann jedoch eine Sicherung bestimmter Inhalte erstellt werden, wenn dies für relevant befunden wird.

2.6 Wie sammeln andere Einrichtungen Hochschulschriften?

Eines der erwähnenswertesten Beispiele für das Harvesten von Hochschulschriften ist das Projekte „**Theses Canada**“ der Library and Archives Canada. Sie betonen auf ihrer Webseite, dass Abschlussarbeiten eine wichtige Rolle auch für zukünftige wissenschaftliche Tätigkeiten haben und deshalb gesammelt und zur Verfügung gestellt werden soll. Neben Titeldatensätzen, welche von Abschlussarbeiten stammen, die seit 1965 aufgenommen wurden. Seit 1998 werden daneben auch teilweise PDFs angeboten, wenn diese zur

⁹ vgl. https://www.dnb.de/DE/Professionell/Sammeln/Sammlung_Websites/sammlung_websites_node.html#doc246604bodyText5 (letzter Zugriff:

Verfügung gestellt werden können. Etwa 70 kanadische Universitäten sind bereits Teil des Programmes.

Die AbsolventInnen reichen ihre Abschlussarbeit wie gewohnt bei der eigenen Universität ein. Bei der Einreichung wird eine Vereinbarung unterschrieben, welche der Universität erlauben, diese in ihr Repositorium¹⁰ zu übernehmen. Gleichzeitig stimmen die AbsolventInnen der Einbettung ihre Arbeit und deren Metadaten in der „Theses Canada“-Datenbank zu. Der Harvest findet dabei einmal pro Monat statt.¹¹

Gleichzeitig stellt sich hier jedoch auch die Frage, durch welche möglichen anderen Vorgehensweisen Institutionen und Hochschulen/Universitäten an Abschlussarbeiten kommen können, die potentiell für sie von Bedeutung sein könnten.

Es stellte sich heraus, dass viele Institutionen dabei auf einen Harvest verzichten und grundsätzlich auf die Eigeninitiative der AbsolventInnen bauen. Dies kann mehrere Gründe haben. Mögliche Gründe für einen Verzicht könnten sein:

- Finanzieller und personeller Aufwand muss einkalkuliert werden
- Bezüglich rechtlicher Fragen, kann nochmals auf die eigenen Vorgaben und Richtlinien verwiesen werden. Für die AbsolventInnen führt der Schritt der Einreichung vermutlich zu einer intensiveren Auseinandersetzungen mit der Plattform

Auch hier sollen kurz zwei Beispiele genannt werden:

2.6.1 Universität Wien

Die Universität Wien bietet seit dem Jahr 2008 das „**E-Theses**“-Archiv an, in welchem Diplomarbeiten und Dissertationen der Universität Wien in elektronischem Volltext zur Verfügung gestellt werden. Die zur Verfügungsstellung

¹⁰ Anm. Ein Repositorium ist eine Art von digitalem Archiv, welches digitale Objekte und Daten speichert und zugänglich macht.

¹¹ Theses Canada, online unter: <https://www.bac-lac.gc.ca/eng/services/theses/Pages/students.aspx> (letzter Zugriff: 21.08.2020).

erfolgt durch die freiwilligen Abgabe. Anschließend sind die Hochschulschriften einsehbar¹².

2.6.2 ProQuest Dissertations & Theses Global

Eines der größten Repositorien stellt „**ProQuest Dissertations & Theses Global**“ dar. Über fünf Millionen Arbeiten aus etwa 100 Ländern werden hier zur Verfügung gestellt. Etwa die Hälfte der Abschlussarbeiten sind dabei als Volltext in PDF-Format vorhanden.¹³

2.7 Fazit

Die zwei Herangehensweisen zeigen, dass Hochschulschriften auf unterschiedlichste Weise zusammengeführt werden können. Es ist von der Institution, deren finanzieller und personeller Ausstattung abhängig, wie das Vorgehen durchgeführt wird. Beide Optionen bieten Vor- und Nachteile, welche von den zuständigen Stellen bedacht werden müssen und anschließend ausschlaggebend für die jeweilige Entscheidung sind.

Im zweiten Teil dieser Arbeit wurde ausgearbeitet, wie die Sammlung von Hochschulschriften und deren Metadaten mit der Option des Harvest erfolgen kann.

¹² E-Theses, online unter: <http://othes.univie.ac.at>. (letzter Zugriff: 21.08.2020).

¹³ vgl. <https://about.proquest.com/products-services/dissertations/proquest-dissertations-faq.html#much>

3. Praktische Umsetzung für die ZAMG

3.1 Crawler

Crawling definiert den Prozess der Datenextraktion. Mit Hilfe dieser Technik werden Daten aus verschiedenen Webseiten und Repositorien extrahiert.¹⁴ Auf den Prozess des Crawlers bauen Scraper auf, die zusätzlich die gecrawlten Daten in eine strukturierte Form bringen. Besonders im Suchen von grauer Literatur zeigt sich durch Webscraping Software besonderer Nutzen, da die Suche danach erhebliche Ressourcen erfordern kann. In den Wissenschaften existieren dafür keine umfassenden Datenbankressourcen und so müssen Suchanfragen sowohl an webbasierte Suchmaschinen, Spezialdatenbanken wie Repositories für Abschlussarbeiten, Organisationswebsites, wie zum Beispiel Nichtregierungsorganisationen, als auch sonstige Organisationen, Regierungsdatenbanken und Universitätsregister gerichtet werden. Teilweise liefern Suchmaschinen Ergebnisse, welche sehr undurchsichtig ausfallen. Ein Gesamtergebnis muss allerdings immer manuell aufbereitet werden.¹⁵

Crawler sind Tools zum Speichern von Webinhalten. Andere Bezeichnungen für solche Programme sind "Robot", "Bot" und "Spider".¹⁶ Der Crawler speichert Webinhalte nach benutzerdefinierten Regeln, diese Inhalte werden anschließend von Computerprogrammen ausgewertet. Crawler ordnen die Informationen bestimmten Kategorien zu. Anschließend werden die Daten für die Auswertbarkeit indiziert und katalogisiert. Klassischerweise müssen Aufträge im Voraus definiert werden, mit den Ergebnissen wird ein Index angelegt und über eine Ausgabesoftware kann, anschließend, auf die Informationen zugegriffen werden. Zudem kann über ein Frontend dieser Index abgefragt werden. Eine wichtige Rolle ist die Unterstützung von Suchmaschinen, da so durchsuchbare Indices aufgebaut werden können. Ursprünglich stammt der Begriff von der ersten Suchmaschine, dem Webcrawler. Der bekannteste ist der Googlebot, der für

¹⁴Datahen, Data Harvesting War: Scraping vs using AP, online unter: <https://www.datahen.com/blog/data-harvesting-war-scraping-vs-using-api/> (letzter Zugriff: 08.07.2020).

¹⁵ Neal R. Haddaway, The Use of Web-scraping Software in Searching for Grey Literature. In: TGJ Volume 11, Number 3, Schweden 2015, S. 186.

¹⁶ Sotirios Batsakis/Euripides Petrakis/und Evangelos Milios, Improving the Performance of Focused Web Crawlers. In: Data & Knowledge Engineering, Volume 68, Issue 10, o. O. 2009, S. 1002.

Google einen Index für die Google-Suche zusammenstellt.¹⁷ Diese Suchmaschinen-Crawler rufen unabhängig von ihrem Thema eine große Anzahl von Webseiten ab. Im Gegensatz dazu suchen anwendungsbenutzende Crawler so viele relevante Seiten wie möglich nach einem bestimmten Thema ab.¹⁸ Weitere Einsatzmöglichkeiten von Crawler finden sich bei Preisvergleichsportalen und im Bereich des Data Mining, welche zum Beispiel öffentlich erreichbare E-Mailadressen von Unternehmen sammeln.¹⁹ Dieses Data Mining kann sowohl über den Crawler oder über ein externes Programm laufen.²⁰ Unter diesem Begriff werden alle Extraktionen größerer Datenmengen verstanden. So können Datenmuster erkannt oder die Zusammenhänge verstanden werden. In Verbindung mit dem Begriff „Big Data“²¹ ergeben sich zukünftig sicherlich zahlreiche Möglichkeiten.²² Fokussierte Crawler werden auch beim Harvesting von Publikationsdaten eingesetzt. Diese suchen, wie bereits erwähnt, gezielt nach Inhalten auf Webseiten und nur Seiten mit den bestimmten Eigenschaften werden gespeichert. Dieser Schritt beinhaltet allerdings noch nicht das Extrahieren von Metadaten, was sich wiederum als eigener Arbeitsschritt herausstellt. Metadaten können an unterschiedlichen Stellen und in unterschiedlichen Formaten aufgefunden werden.²³

Der Crawler greift bei der Ausführung meist auf öffentlich erreichbare Dateien eines Servers zu (meist Webseiten eines Webserver) und speichert diese dann. Zusätzlich gebe es noch die Möglichkeit jeden Datensatz einzeln in einem Datenbanksystem zu speichern.

¹⁷Ryte Wikie, Crawler, online unter: <https://de.ryte.com/wiki/Crawler> (Letzter Zugriff: 08.07.2020).

¹⁸ Sotirios Batsakis/Euripides Petrakis/und Evangelos Milios, Improving the Performance of Focused Web Crawlers. In: Data & Knowledge Engineering, Volume 68, Issue 10, o. O. 2009, S. 1002 f.

¹⁹ Ryte Wikie, Crawler, online unter: <https://de.ryte.com/wiki/Crawler> (Letzter Zugriff: 08.07.2020).

²⁰ Mike Thelwall, A web crawler design for data mining. In: Journal of Information Science 27(5), o. O. 2001) S. 319.

²¹ Anm.: Unter Big Data werden besonders große Datenmengen verstanden. Vgl. https://de.wikipedia.org/wiki/Big_Data (letzter Zugriff: 08.07.2020).

²² Patrick Rösing, Daten sammeln, Zusammenschaufeln und sie nutzbar machen: Scraping und Data Mining, In: Investigatives Recherchieren, Webseite zum Buch von Johannes Ludwig, online unter: <http://investigativ.org/kapitel-4/daten-sammeln-zusammenschaufeln-und-sie-nutzbar-machen-scraping-und-data-mining-hinweise-und-tipps-von-patrick-roesing/> (letzter Zugriff: 08.07.2020).

²³ Thomas Zinky,/ OliverHaasey/Marcel Waldvogelz, Webharvesting von Publikationsdaten. Distributed Systems Group Technical Report, Wien 2014, S. 1.

Probleme beim Ausführen eines Crawlers könnte eine Firewall machen. Sie wurden entworfen um den Zugriff auf bestimmte Informationen und Ressourcen, zu verhindern.²⁴

Ein Crawler ist daher in der Regel aber nur ein Datensammler. Mit Hilfe eines Scraper können die Daten bearbeitet werden. Der Crawler ruft die Daten ab und speichert sie identisch ab. Dafür muss ein Crawler nicht auf Seiten im Internet zugreifen. Ein Scraper hingegen, extrahiert die vom Crawler gespeicherten Daten, anschließend. Aber natürlich kann ein Webscraper auch den Crawler gleich beinhalten und alles in einem Vorgang machen (vergleiche den für das Projekt erstellten Prototypen). Scrapings kommen aber auch beim Gebrauch von erweitertem Browser zum Einsatz. Dabei können Anmeldeoptionen von Webseiten direkt über eine Browser-Systemleiste ausgeführt werden.²⁵ Der Begriff selbst kommt aus dem Englischen und bedeutet so viel wie „schaben“ oder „kratzen“. Auch ein Scraper kann mit dem notwendigen Know-how leicht programmiert werden. Zusätzlich stehen einem Nutzer diverse Werkzeuge zur Verfügung, darunter zahlreiche Scraper-Erweiterungsprogramme für den Internet-Browser und browserbasierende Online-Anwendungen.²⁶

Eine weitere Methode zur Datengewinnung ergibt sich über ein Application Programming Interface (API), mit dem eine Software mit einer anderen Software kommunizieren kann. Einige Webseiten bieten dadurch Programmschnittstellen, um so Daten und Funktionen für andere Entwickler und Unternehmen zu öffnen. Es ist die am häufigsten praktizierte Art des Daten- und Service-Austauschs zwischen Unternehmen, sowohl intern als auch extern. Diese Technik des Datenaustauschs über das Web wird immer beliebter.²⁷ Vereinfacht gesagt, ermöglicht die Schnittstelle eine Kommunikation und Interaktion zwischen zwei Systemen. Dadurch wird Dritten der Zugang zu verschlossenen Datenpools und Benutzerkreisen gewährt. Durch dieses gemeinsame Nutzen von Daten können ganz neue

²⁴ Mike Thelwall, A web crawler design for data mining. In: Journal of Information Science 27(5), o. O.2001, S. 320 f.

²⁵ Vgl. <https://de.ryte.com/wiki/Crawler> und <https://dzone.com/articles/web-scraping-vs-web-crawling-whats-the-difference> (letzter Zugriff: 08.07.2020).

²⁶ Patrick Rösing, Daten sammeln, Zusammenschaufeln und sie nutzbar machen: Scraping und Data Mining, In: Investigatives Recherchieren, Webseite zum Buch von Johannes Ludwig, online unter: <http://investigativ.org/kapitel-4/daten-sammeln-zusammenschaufeln-und-sie-nutzbar-machen-scraping-und-data-mining-hinweise-und-tipps-von-patrick-roesing/> (letzter Zugriff: 08.07.2020).

²⁷ Datahen, Data Harvesting War: Scraping vs using AP, online unter: <https://www.datahen.com/blog/data-harvesting-war-scraping-vs-using-api/> (letzter Zugriff: 08.07.2020).

Dienste sowie Mehrwerte entstehen. Wenn allgemein über APIs gesprochen wird, werden hauptsächlich externe APIs gemeint. Über zum Beispiel unzähligen frei verfügbaren Tools von Twitter, Flickr und YouTube können Inhalte automatisiert ausgelesen, hinzugefügt und verändert werden. Dabei kann eine Darbietung bestimmter Funktionen durch ein User Interface auch über eine externe API ausgeführt werden. Dadurch können Inhalte weiterverarbeitet und (re-) kombiniert werden.²⁸

Sowohl Webscraping als auch API-Scraping werden heutzutage häufig als Methoden zum Crawlen von Daten verwendet. Am wichtigsten ist jedoch die Auswahl der Methode mit geringfügigen Einschränkungen oder Problemen. Insbesondere wenn Sie regelmäßig Webcrawling durchführen, möchten Sie sicherstellen, dass Sie die besten Optionen verwenden. Nachteile der APIs zeigen sich bei der Verfügbarkeit und bei der mangelnden Anpassung. Nicht jede Webseite bietet APIs an und auch wenn sie die Möglichkeit anbieten bedeutet das jedoch nicht, dass Sie die API zum Extrahieren von Daten problemlos verwenden können. APIs bieten nicht den Zugriff auf alle verfügbaren Daten. Auch Änderungen werden oft erst Monate später in der API angezeigt. Wenn dieselben Daten aus derselben Quelle für dasselbe spezifische Ziel abgerufen werden müssen, passt eine API perfekt zu allen Datenanforderungen. Möglicherweise existiert auch ein Vertrag mit einer Website. Gemäß dem Vertrag können die API möglicherweise innerhalb bestimmter Grenzen verwendet werden.²⁹

Wenn die Datenanforderungen aber gleich sind und sich nicht innerhalb der Zeit ändern, werden keine Einschränkungen des API-Systems aufkommen. So zeigen sich APIs besonders im Zusammenhang des Crawlens bibliographischer Daten als die idealste Wahl. Universitäten können so leicht die Daten, die für die ZAMG wichtig wären, bereitstellen.³⁰

Zusammenfassend sind Webcrawler, sowie Scraper als auch APIs eine attraktive technologische Entwicklung auf dem Gebiet der Suche nach grauer Literatur. Besonders die Möglichkeit kostenlose und kostengünstige Softwares zu erwerben, oder diese gar

²⁸ Gründerszene , Web APIs. Ein nicht-technischer Erklärungsversuch, online unter: <https://www.gruenderszene.de/it/web-apis-ein-nicht-technischer-erklarungsversuch>, (letzter Zugriff: 08.07.2020).

²⁹ Daten, Data Harvesting War: Scraping vs using AP, online unter: <https://www.datahen.com/blog/data-harvesting-war-scraping-vs-using-api/> (letzter Zugriff: 08.07.2020).

³⁰ Daten, Data Harvesting War: Scraping vs using AP, online unter: <https://www.datahen.com/blog/data-harvesting-war-scraping-vs-using-api/> (letzter Zugriff: 08.07.2020).

selbst zu programmieren bietet eine große Chance und signifikante Vorteile für Personen und Institute mit begrenzten Ressourcen. So kann die Ressourceneffizienz erhöht und die Transparenz drastisch verbessert werden. Forscher könnten erheblich davon profitieren, wenn sie die Anwendbarkeit von Webscraping auf ihre eigene Arbeit untersuchen.

3.2 Metadaten

Metadaten sind überall um uns herum zu finden. Es sind Informationen, die sich auf Daten beziehen oder diesen übergeordnet sind, sprich es sind Daten über Daten. Wenn Metadaten richtig erzeugt wurden, fallen sie nicht wirklich auf und befinden sich meistens im Hintergrund. Die beschriebenen Daten sind meist größere Datensammlungen über Bücher, Datenbanken oder Dateien, um diese besser aufzufinden. Als typische Metadaten zu einem Buch können Autor, Titel, ISPN oder Verlag und Erscheinungsjahr genannt werden. Dabei findet oft keine bewusste Trennung zwischen Objekt und Metadaten statt. Wenn in einem Bibliothekskatalog eine Suche abgesetzt wird, so wird meist davon gesprochen, dass nach einem Buch gesucht wird. Tatsächlich werden aber die Metadaten zu jenem Buch gesucht.³¹

3.3 Metadatenstandards

Unter Metadatenstandards werden standardisierte Schemata verstanden. Sie beschreiben Ressourcen strukturiert und einheitlich. Zudem werden die Interpretierbarkeit und das Verständnis für die Daten durch die gemeinsame Sprache verbessert. Je nach Art der Verwendung gibt es unterschiedliche Standards. Dabei lassen sich Standards für die Beschreibung der Datenstruktur und Semantik, die Inhalte (zum Beispiel RDA³² oder RSWK³³), die Normdaten und Vokabulare (zum Beispiel die Gemeinsame Normdatei (GND), Geonames und VIAF³⁴) sowie für die Kommunikation unterscheiden.³⁵

³¹ Jeffrey Pomerantz, Metadata, Cambridge 2015S, 1-9

³² Anm.: Das internationale Regelwerk für die Katalogisierung in Bibliotheken.

³³ Anm.: Sind die Regeln für die Inhaltserschließung.

³⁴ vgl. <http://viaf.org/> (letzterZugriff: 13.07.2020).

Anm.: VIAF® (Virtual International Authority File) ist ein Normdatendienst, welcher mehrere Normdateien kombiniert.

³⁵ Programmfabrik: Metadatenstandards, online unter: <https://www.programmfabrik.de/wissen/metadaten/metadatenstandards/> (letzter Zugriff: 13.07.2020).

Strukturstandards dienen als formales Regelwerk zur Dokumentations- und Objektbeschreibung. Darunter fallen verschiedene Formate wie, um die bekanntesten zu nennen, Dublin Core, MARC 21 und MODS.

3.3.1 Dublin Core

Bei Dublin Core (DC) handelt es sich um eine Sammlung einfacher und standardisierter Konventionen zur Beschreibung von Metadaten, um diese einfacher auffindbar zu machen. Insgesamt baut sich DC aus nur 15 Kernelemente auf. Darunter fallen identifier, format, typ, language, title, subtitle, coverage, description, creator, publisher, contributor, rights, source, relation und date. Die Metadaten können dabei sowohl mit RDF als auch mit XML dargestellt werden.³⁶

3.3.2 MODS

Metadata Object Description Schema (MODS) ist ein Format für bibliographische Metadaten, welches auf XML basiert. Es wurde von der Library of Congress' Network Development and MARC Standards Office entwickelt und verwaltet. Dieses kann für eine Vielzahl von Zwecken und insbesondere für Bibliotheksanwendungen verwendet werden. Es enthält eine Teilmenge der MARC-Felder und verwendet sprachbasierte Tags anstelle numerischer Tags. In einigen Fällen werden aber Elemente aus dem bibliografischen Format MARC 21 neu gruppiert. Somit ist es umfangreicher als Dublin Core. Gleichzeitig ist es leichter verständlich (benutzerfreundlicher) und einfacher als MARC 21. Somit kann MODS als Kompromiss zwischen MARC 21 und Dublin Core angesehen werden. Aufgrund dieses Kompromisses wurde MODS auch für die XML-Beispieldatensätze, des Prototyps, gewählt.³⁷

3.3.3 Marc21

MARC (Machine-Readable Cataloging) ist ein bibliografisches Datenformat. Es zeichnet sich als ein allgemeines Format aus, das von verschiedensten Anwendungen gelesen und verarbeitet werden kann. Die Hauptaufgabe besteht in

³⁶ Dublincore, online unter: <https://www.dublincore.org> (letzter Zugriff: 13.07.2020).

³⁷ The Library of Congress, Metadata Object Description Schema (MODS), online unter: <https://www.loc.gov/standards/mods/mods-overview.html> (letzter Zugriff: 13.07.2020).

dem gemeinsamen Nutzen von Daten innerhalb verschiedener Bibliotheken. MARC 21 bildet die Grundlage der meisten heute genutzten Bibliothekskataloge.³⁸

3.4 Austauschformate

Unter einem Austauschformat wird meistens ein Datenformat verstanden, welches mit vielen verschiedenen Anwendungen kompatibel ist. Für die Weitergabe von Daten stehen zahlreiche Exportformate zur Auswahl. Als Beispiel werden XML und RDF näher erläutert. Da MODS auf XML basiert, wurden auch für den Prototyp die bibliographischen Daten in XML geschrieben.³⁹

3.4.1 XML

„Um digital gespeicherte Information lesen zu können, muss man das Format kennen, in dem sie dargestellt sind.“⁴⁰

Um dauerhaft darauf zuzugreifen, ist es notwendig die Daten durch dauerhafte internationale Standards zu repräsentieren. Datenformate sind als eine formale Sprache, mit der Daten beschrieben werden können, zu verstehen. Für die unterschiedlichen Arten von Informationen gibt es auch eine Unzahl an Datenformaten. Um eine formale Sprache zu definieren, benötigt man standardisierte Sprachen. Eine solche Sprache ist XML (Extensible Markup Language). Darunter wird ein Verfahren verstanden, das Texte auszeichnet und Informationen kodieren kann. Dadurch können die Daten gespeichert und transportiert werden. Über diese Sprache werden Dokumente in Form einer Baumstruktur dargestellt. Wichtig ist dabei die Schachtelung zu beachten. Ein XML-Element kann andere Elemente enthalten. Umgekehrt müssen alle Elemente vollständig in einem anderen Element enthalten sein, wodurch die Struktur strikt hierarchisch ist. Die Dokumente können sowohl vom Menschen als auch von der

³⁸ Deutsche Nationalbibliothek, Fedlbeschreibung der Titeldaten der Deutschen Nationalbibliothek und der Zeitschriftendatenbank im Format MARC 21, online unter: https://www.dnb.de/DE/Professionell/Metadatendienste/Exportformate/MARC21/marc21_node.html (letzter Zugriff: 13.07.2020).

³⁹ Programmfabrik, Austauschformate und Schnittstellen, online unter: <https://www.programmfabrik.de/wissen/metadaten/metadatenstandards/austauschformate-und-schnittstellen/> (letzter Zugriff: 13.07.2020).

⁴⁰ Michael Weigend, Python 3. Lernen und professionell anwenden. Das umfassende Praxisbuch, 8. Auflage, Frenchen 2019, S. 729.

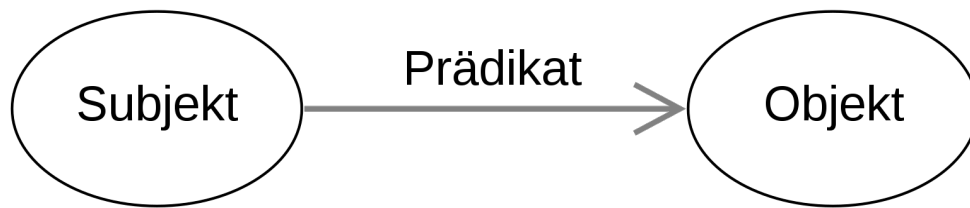
Maschine gelesen werden. Als Kodierung werden die gleichen Zeichen wie für den Text oder die zu den beschriebenen Daten verwendet, welche aus einem bestimmten Zeichenvorrat stammen und meistens in der Form UTF-8 kodiert sind. Dadurch kann XML mit jedem gängigen Editor beschrieben werden und auf jeder Plattform und jedem Betriebssystem gelesen werden.⁴¹ Die Baumstruktur kann als ein umgekehrter Wurzelbaum bildlich vorgestellt werden. Dieser Baum ist ein spezieller gerichteter Graph ohne Zyklen und mit genau einer Wurzel (root) von dem weitere Felder ausgehen. Allgemein besteht ein gerichteter Graph aus Knoten und Pfeilen (gerichteten Kanten), die von einem Knoten zum anderen führen. Vom Root gehen die Kinder (child nodes) aus. Eltern mit demselben Elternknoten werden Geschwister genannt und befinden sich im Baum auf derselben Hierarchieebene. Knoten, die sich zwar auf derselben Ebene befinden, aber von einem anderen Knoten ausgehen, sind keine Geschwister. Textknoten, die Textpassagen aufweisen, haben keine weitere Kinder und werden Blätter genannt.⁴²

Das XML-Dokument wird durch ein komplexes Aggregat verschiedener Objekte dargestellt. Dabei ist es aus Elementen aufgebaut, diese können ihrerseits wieder Elemente oder auch Texte enthalten. Jedes Element wird von einem Start-Tag gebildet, dieses mit einer spitzen Klammer beginnt < gefolgt von dem Namen des Elements und mit einer weiteren spitzen Klammer > endet, und einem End-Tag, welches ähnlich aufgebaut ist aber mit </, zum Beispiel </Autor>, beginnt. Ein Element muss dabei das gesamte Dokument repräsentieren. Im Beispiel der Projektarbeit <mods> und </mods>. Dazwischen sind alle Daten zusammengefasst.⁴³ Wenn ein Element keine Textdaten oder andere Elemente enthält, wird es mit <.../>, zum Beispiel <subTitle/>, geschrieben und als leeres Element bezeichnet. Auch können Elemente mit Attributen, die Teil des Anfangstags sind, ergänzt werden. Dadurch können wichtige Zusatzinformationen

⁴¹ Michael Weigend, Python 3. Lernen und professionell anwenden. Das umfassende Praxisbuch, 8. Auflage, Frenchen 2019, S. 729 und Fotis Jannidis, Kohle, Hubertus und Rehbein, Malte, Digital Humanities, Stuttgart 2017, S. 128.

⁴² Michael Weigend, Python 3. Lernen und professionell anwenden. Das umfassende Praxisbuch, 8. Auflage, Frenchen 2019, S. 730.

⁴³ Weigend, Michael, Python 3. Lernen und professionell anwenden. Das umfassende Praxisbuch, 8. Auflage, Frenchen 2019, S. 729 f.



angegeben werden. Ein Tag kann auch unterschiedliche Attribute gleichzeitig aufweisen. Ein Attribut zeigt sich in den vorliegenden XML-Daten zum Beispiel als `<name type=" Betreuer">`. Die Anführungszeichen sind wichtig, damit es auch als Attribut gelesen werden kann.⁴⁴

XML läßt nicht fest wie ein Element zu benennen ist, wodurch sich freigewählte Strukturen von Texten und Daten codieren lassen. Es gibt allerdings einige praktische Hinweise. Durch dessen Regel-Set können Daten und Dokumente beschrieben werden. Hauptsächlich wird es als Datenaustauschformat verwendet und dient als syntaktische Grundlage für Metadatenstandards. Für die unterschiedlich aufgebauten Syntaxen gibt es unterschiedliche Schemata. Dabei beschreibt die Schemasprache Vokabular und Regeln. Durch die Schemata können Elemente und Attribute definiert werden und die Verschachtelung so wie ihre Inhaltstypen festgelegt werden.⁴⁵

3.4.2 RDF

„Resource Description Framework“ (RDF) ist eine Rahmenstruktur zur Beschreibung von Ressourcen. RDF ist also eine logische Struktur, nach der Daten organisiert sind. Man versteht darunter ein generisches Datenmodell für beschreibende Aussagen über Entitäten.

Das Datenmodell beruht zudem auf einem gerichteten Graphen. Die Aussagen werden als Tripel definiert, bestehend aus einem Prädikat, einem Subjekt und einem Objekt. Die Beziehungen sind über dem Prädikat von einem Subjekt auf das Objekt gerichtet. So kann man einfache Vokabularien bzw. „Ontologien“ im Web zur Verfügung stellen, die von anderen verwendet werden können.

⁴⁴Fotis Jannidis, Kohle, Hubertus und Rehbein, Malte, Digital Humanities, Stuttgart 2017, S. 130.

⁴⁵ Fotos Jannidis, Kohle, Hubertus und Rehbein, Malte, Digital Humanities, Stuttgart 2017, S. 135.

Der Vorteil von RDF ist, dass das Subjekt eines Tripels mit einer „uniform resource identifier“ (URI) identifiziert werden muss. Über dem Prädikat hingegen lassen sich Aussagen über die Ressource treffen und können als Metadatenformat abgelegt werden. Durch Referenzierung können andere Autoren das Vokabular weiterverwenden. Ein bekanntes Beispiel wäre die Repräsentation von Dublin Core in RDF. RDF-Deklarationen bilden zudem selbst Ressourcen, die mit weiteren Aussagen ergänzt werden können. Mit RDF können beliebige Strukturen definiert werden, wobei es viele Standards, Vokabularien und Ontologien gibt.⁴⁶

⁴⁶ Jeffrey Pomerantz, Metadata, Cambridge 2015, S. 140-142

Beispiel einer XML-Datei. Als Format wurde MODS verwendet. Es wurde aus Datenschutzgründen als Inhalt die Masterarbeit des Verfassers verwendet.

```
<?xml version="1.0" encoding="UTF-8"?>
<mods version="3.6">
  <identifier type='urn'>urn:nbn:at:at-ubg:0-41188</identifier>
  <titleInfo>
    <title>Topographisch-archäologische Untersuchung zur Belagerung der Burg
Falkenberg, MG Straß im Straßertale (Niederösterreich), im Winter 1299/1300</title>
    <subTitel/>
  </titleInfo>
  <name type="personal">
    <namePart type="family">Vadeanu</namePart>
    <namePart type="given">Christopher</namePart>
    <namePart type="date"></namePart>
    <role>
      <roleTerm type="code" authority="marcrelator">aut</roleTerm>
      <roleTerm type="text" authority="marcrelator">Verfasser</roleTerm>
    </role>
  </name>
  <name type="Betreuer">Doneus, Michale</name>
  <name type="Betreuer2">Kühtreiber, Thomas</name>
  <extent>210 Seiten</extent>
  <genre>Hochschulschrift</genre>
  <note type="thesis">Masterarbeit -- Universität Wien, 2019</note>
  <language type="code">ger</language>
  <place type="code">XA-AT</place>
  <originInfo>
    <place>
      <placeTerm type="text">Wien</placeTerm>
    </place>
    <publisher></publisher>
    <dateIssued type="production">2019</dateIssued>
  </originInfo>
  <abstract>
    Ziel dieser Masterarbeit war es, die Belagerung der Burg Falkenberg im Winter 1299/1300, die durch zwei
    schriftliche Quellen historisch belegt ist, mit archäologischen und topo-graphischen Mitteln in Form einer Untersuchung
    des Umfeldes modellhaft nachzuzeichnen. Bei den Quellen handelt es sich um die Reimchronik des steirischen
    Chronisten Ottokar und um die Annalen Bernhard Links. Beide Texte geben Auskunft über die beteiligten Parteien und
    den Verlauf der Belagerung. Die Basis der topographischen Untersuchung bildete ein digitales Geländemodell für das
    Umfeld der Burg Falkenberg. Mit Hilfe von Hangneigungskarten, schattierten Bildern und „Raster surfaces“ konnten
    mögliche Belagerungsstellungen herausgearbeitet werden. Unter anderem konnte dabei im Westen der Burg eine
    Belagerungsschanze ausgemacht werden. Der Fokus der Arbeit lag allerdings auf der Erstellung von Sichtbarkeitskarten
    und dem Einbezug möglicher Reichweiten. Sichtbarkeitskarten geben Auskunft über die Standortwahl und zeigen
    raumtaktische Vorteile auf. Besonders in Kombination von Sichtfeld und möglicher Reichweite mit Hilfe der „viewshed2-
    Funktion“ konnte ein mögliches Schussfeld dargestellt werden. Um die historischen Quellen mit archäologischen
    Quellen in Beziehung zu setzen, war es zunächst wichtig, die im 13. und 14. Jahrhundert zum Einsatz gekommenen
    Belagerungswaffen näher zu beschreiben und ihre ballistische Leistung herauszuarbeiten. Die so gewonnenen
    Erkenntnisse bildeten zusammen mit den archäologischen Streufunden, welche von einem Privatsammler mittels
    Metalldetektor im Umfeld der Burg aufgefunden wurden, die Grundlage der anschließenden Forschung im ArcGIS.
    Besonders mit Hilfe der Geschosspitzen, welchen zusammen mit den anderen Funden ein eigenes Kapitel gewidmet
    ist, konnte die Echtheit der historischen Quelle archäologisch verifiziert werden.
  </abstract>
  <note></note>
</mods>
```

3.5 Softwaredokumentation

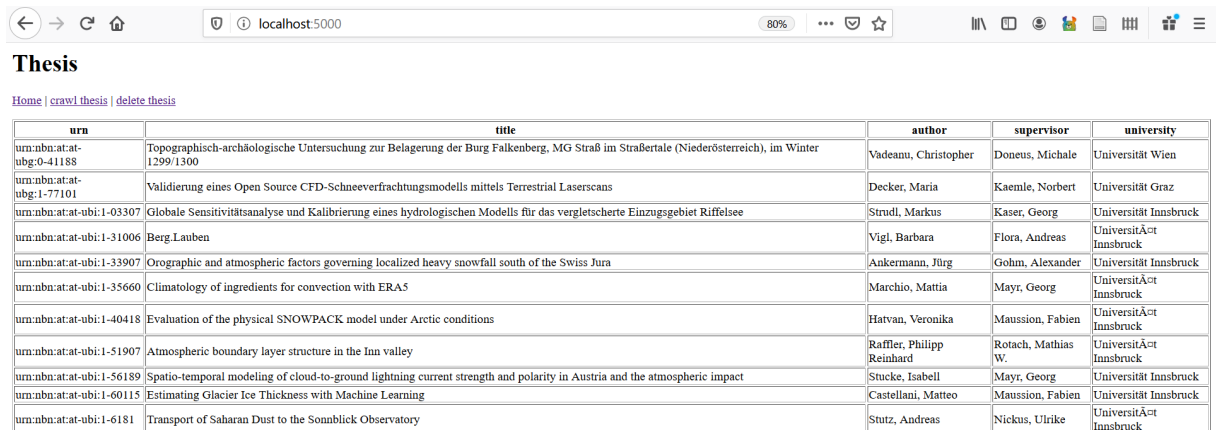
Im folgenden Kapitel wird näher auf die Applikation des Prototyps eingegangen und auf die Funktionalität erläutert. Eine Softwaredokumentation beschreibt wie die Software funktioniert, was sie erzeugt bzw. verarbeitet und was zum Betrieb erforderlich ist.

Das vorliegende Programm dient als Prototyp für einen möglichen Vorschlag einer Umsetzung eines Monitorings von Hochschulschriften aus dem Fachbereich der Erdwissenschaften. Das Programm soll aufzeigen, dass es von Vorteil wäre, wenn die Universitäten ihre Daten strukturiert bereitstellen würden. Diese können dann über APIs oder mittels Scraper/Harvester erreicht und eingesammelt werden. Über einen Parser wäre es dann möglich die Daten zu lesen und diese schließlich in ein Datenbanksystem einzuspeisen. Hauptaufgabe des, für das Projekt programmierten, Python-Programmes ist es, eine Liste aller Metadaten-Dateien einer Institution über eine Textdatei abzurufen und der Reihe nach den einzelnen Dateien im XML-Format einzulesen. Aus diesen Dateien werden anschließend mittels eines Parsers (siehe Python und XML) die relevanten Metadaten extrahiert und ein Thesis-Objekt erzeugt. Im Anschluss daran werden die Daten einem MySQL Datenbanksystem (siehe SQL und MySQL) übergeben. Für die Tests wurden für drei Universitäten, je zehn Testdateien im XML-Format angereichert und auf einem lokalen Server (VBox) beziehungsweise auf drei Server (Docker) kopiert. Abrufbar sind diese Dateien via http. Die XML-Daten wurden im Metadaten-Schema MODS geschrieben.

Eine programmierte Webapplikation macht diesen Prozess deutlich erkenntlich. Über einen Button „crawl thesis“ wird der Vorgang gestartet und wenn alles funktioniert, sollte eine Datenbank aufgebaut und die Datensätze als HTML-Datei ausgegeben angezeigt werden. Diese Datenbank kann anschließend über „delete thesis“ wieder gelöscht werden. Dieser Vorgang eignet sich sehr gut, um zu testen.

Für Testzwecke wurde mittels Virtualisierungssoftware (genauere Informationen siehe gleichnamiges Kapitel „Virtualisierungssoftware“) gearbeitet. Der Vorteil dabei ist die Mobilität des Prototyps. Auch können einzelne Programmabschnitte einzeln getestet werden. Zunächst wurde in einer Virtuellen Maschine gearbeitet. Allerdings wurde der Nachteil der Portabilität rasch ersichtlich und so wurde zusätzlich mit Docker gearbeitet. Da beide Virtualisierungssoftwares auf dem Betriebssystem Linux basieren, musste in

einem ersten Schritt erstmal der Umgang und die wichtigsten Befehle zum Bedienen erlernt werden.



urn	title	author	supervisor	university
urn:nbn:at-at-ubg:0-41188	Topographisch-archäologische Untersuchung zur Belagerung der Burg Falkenberg, MG Straß im Straßertale (Niederösterreich), im Winter 1299/1300	Vadeanu, Christopher	Doneus, Michale	Universität Wien
urn:nbn:at-at-ubg:1-77101	Validierung eines Open Source CFD-Schneeverfrachtungmodells mittels Terrestrial Laserscans	Decker, Maria	Kaemle, Norbert	Universität Graz
urn:nbn:at-at-ubi:1-03307	Globale Sensitivitätsanalyse und Kalibrierung eines hydrologischen Modells für das vergletscherte Einzugsgebiet Riffelsee	Strudl, Markus	Kaser, Georg	Universität Innsbruck
urn:nbn:at-at-ubi:1-31006	Berg Lauben	Vigl, Barbara	Flora, Andreas	Universität Innsbruck
urn:nbn:at-at-ubi:1-33907	Orographic and atmospheric factors governing localized heavy snowfall south of the Swiss Jura	Ankermann, Jürg	Gohm, Alexander	Universität Innsbruck
urn:nbn:at-at-ubi:1-35660	Climatology of ingredients for convection with ERA5	Marchio, Mattia	Mayr, Georg	Universität Innsbruck
urn:nbn:at-at-ubi:1-40418	Evaluation of the physical SNOWPACK model under Arctic conditions	Hatvan, Veronika	Mausson, Fabien	Universität Innsbruck
urn:nbn:at-at-ubi:1-51907	Atmospheric boundary layer structure in the Inn valley	Raffler, Philipp Reinhard	Rotach, Mathias W.	Universität Innsbruck
urn:nbn:at-at-ubi:1-56189	Spatio-temporal modeling of cloud-to-ground lightning current strength and polarity in Austria and the atmospheric impact	Stucke, Isabell	Mayr, Georg	Universität Innsbruck
urn:nbn:at-at-ubi:1-60115	Estimating Glacier Ice Thickness with Machine Learning	Castellani, Matteo	Mausson, Fabien	Universität Innsbruck
urn:nbn:at-at-ubi:1-6181	Transport of Saharan Dust to the Sonnblick Observatory	Stutz, Andreas	Nickus, Ulrike	Universität Innsbruck

Abbildung: Webapplikation mit abgebildeten Beispieldaten aus der Datenbank.

Natürlich wäre es am Besten, wenn die unterschiedlichen Universitäten ihre Hochschulschriften in das gemeinsam genutzte Verbundsystem eingeben würden. Da dies leider nicht immer gegeben ist, wäre zumindest eine Korporation zu empfehlen, in dem die Universitäten die Metadaten auf einem bestimmten Ort, auf dem ein Crawler zugreifen kann, in maschinenlesbarer Form zu Verfügung stellen. Das Metadatenchema sollte einheitlich sein, muss aber nicht MODS sein. Ein Parser kann relativ leicht umgeschrieben werden und so wäre es auch möglich gegebenenfalls RDF zu lesen.

3.5.1 Eindeutiger Identifier

Eine ID wird in der Informatik, grundlegend, für das Identifizieren beliebiger Objekte eingesetzt. Produkte weisen Seriennummern, Katalognummern auf. Abstrakte Ideen werden mit einer Notation identifiziert. Sogar der Mensch weist eine ID in Form einer Ausweisnummer oder durch seine Telefonnummer, welche einem bestimmten Teilnehmer, bzw. Anschluss, zugeordnet wird, auf. Wichtig beim Betrachten einer ID ist das kontrollierte Vokabular zu berücksichtigen. Viele Identifier werden auch als Abkürzung betrachtet.⁴⁷

⁴⁷ LinkFang, Identifier, online unter: https://de.linkfang.org/wiki/Identifier#cite_ref-1 (Letzter Zugriff: 06.07.2020).

In den meisten im Buchhandel erschienenen Monografien wird als eindeutiger Identifikator eine Internationale Standard-Buchnummer (ISBN) angeführt. Seit 2007 besteht sie aus fünf Teilen mit 13 Ziffern. Für Zeitschriften, Zeitungen und für andere fortlaufende Sammelwerke gibt es die Internationale Standard Serial Number (ISSN) für die eindeutige Identifizierung. Sie besteht aus acht Ziffern, die in zwei Gruppen zu je vier Ziffern geteilt wird. Daneben gibt es noch eine Vielzahl von anderen Identifikatoren. Musikdrucke werden so mit der International Standard Music Number (ISMN) identifiziert. Audiovisuelle Objekte verwenden eine Internationale Standard Audiovisual Number (ISAN). Texte verwenden einen International Standard Text Code (ISTC) und digitale Objekte werden in der Regel mit dem Digital Object Identifier (DOI) gekennzeichnet.⁴⁸

Graue Literatur⁴⁹ wird nicht von Verlagen veröffentlicht und die Vergabe eines Identifier hängt stark vom Repository ab. Einige vergeben nur einen Uniform Resource Name (URN). URN ist eine Uniform Resource Identifier (URI), und dient genau wie eine DOI als dauerhafte, ortsunabhängige Bezeichnung. Aufgrund ihrer Dauerhaftigkeit werden sie auch als „Persistent Identifier“ bezeichnet. Der Aufbau besteht aus Ziffern und/oder alphanumerischen Zeichen, welche einem digitalen Objekt zugeordnet werden. Zum Unterschied verweist eine URL nicht auf einen bestimmten Inhalt, sondern auf einen „Ort“ im Internet. Wird der Datensatz von einem bestimmten Ort verschoben, wird die URL zur Auffindung nutzlos. Vergleichbar ist dieses Phänomen mit einer Wohnadresse. Zieht eine Person um, ist diese unter dem alten Ort auch nicht mehr zu finden. Möglich wäre es aber auch dass ein Datensatz über mehreren URL abgerufen werden kann, wodurch die wissenschaftliche Zitierbarkeit leidet.⁵⁰

⁴⁸ Klaus Gantert, Bibliothekarisches Grundwissen. 9. Auflage, Berlin/Boston 2016), S. 91 f.
Vgl. Jeffrey Pomerantz, Metadata, Cambridge 2015, S. 62-64.

⁴⁹ Anm.: Bücher werden in der Regel von Verlagen veröffentlicht. Bücher, die außerhalb des Bücherhandels erscheinen, wie zum Beispiel Hochschulschriften, werden als „graue Literatur“ bezeichnet. Diese Bücher richten sich meist nur an einen begrenzten Personenkreis, enthalten aber wichtige Informationen für die Forschung. Meist sind sie auch nur in elektronischer Form veröffentlicht.
Vgl. Klaus Ganter, Bibliothekarisches Grundwissen. 9. Auflage, Berlin/ Boston 2016, S. 78 f.

⁵⁰ [forschungsdaten.info](https://www.forschungsdaten.info/themen/veroeffentlichen-und-archivieren/persistente-identifikatoren/) Was ist ein persistenter Identifikator?, online unter: <https://www.forschungsdaten.info/themen/veroeffentlichen-und-archivieren/persistente-identifikatoren/> (letzter Zugriff: 06.07.2020)
Vgl. Jeffrey Pomerantz, Metadata, Cambridge 2015, S. 62-64.

Für die vorliegenden Datensätze wurde als ID eine URN verwendet. Eine DOI ist mit einer kostenpflichtigen Lizenz verbunden. Deshalb werden in Österreich oft nur URN vergeben. So zum Beispiel vergibt die Universität Innsbruck bei ihren Hochschulschriften nur eine URN als Identifier.

Wichtig sind eindeutige Identifier auch für die Zuweisung in einer Datenbank. Um einen Datensatz nicht doppelt hinzuzufügen, sollte der Tabelle eine Spalte mit einem Primärschlüssel (ID) hinzugefügt werden. Ein Primärschlüssel ist eine Spalte oder eine Reihe von Spalten, die jede Zeile in der Tabelle eindeutig identifiziert. Der Primärschlüssel kann auch aus mehreren Spalten bestehen, diese Wertekombinationen müssen aber in diesen Spalten eindeutig sein.⁵¹ Ein Schlüssel könnte zwar auch eine Kombination von Titel und Autorennamen sein, aber viel leichter ist es jedoch, einen schon vorgegebenen Identifier, wie eine DOI oder eine URN zu verwenden.

In einem XML-File können IDs, Attribute oder auch Elemente definiert werden. Hierbei wäre generell zu empfehlen, dass die ID, das Wurzelement des Dokuments darstellt, da diese für das Dokument global sind. Im Beispiel XML-Schema, das für die Arbeit verwendet wurde, wobei ein Schlüsselwert hauptsächlich für die Datenbank zur genauen Erkennung wichtig war, wurde die URN, einfachheitshalber, da erst später ergänzt, nicht als Root-Element verwendet. Das Root-Element beschreibt rein das verwendete Schema. Auch wurde nicht das XML-Schema nach W3C verwendet. Hierbei könnte zum Beispiel eine ID nicht mit einer Zahl beginnen, wodurch es nicht möglich ist, eine ISBN direkt als ID zu verwenden. Sie müsste als eine Kombination mit einem Präfix geschrieben werden, zum Beispiel ID isbn-3810518883.⁵²

⁵¹ MySQLTutorial, MySQL Primary Key, online unter: <https://www.mysqltutorial.org/mysql-primary-key/> (letzter Zugriff: 05.07.2020)

⁵² Data2type, xs:ID und xs:IDREF, online unter: <https://www.data2type.de/xml-xslt-xslfo/xml-schema/definition-von-eindeutigkeit-s/xs-id-und-xs-idref/> (letzter Zugriff: 06.07.2020).

3.5.2 Python

Python ist eine universelle höhere Programmiersprache, mit effizienten abstrakten Datenstrukturen. Die Sprache ist ein effektiver Ansatz zur objektorientierten Programmierung. Durch ihren Aufbau, der elegante Syntaxen und dynamische Typisierungen aufweist, ist sie sowohl für Skripten als auch für schnelle Anwendungsentwicklung besonders gut geeignet. Die Sprache ist einfach, wenn nicht sogar minimalistisch. Auf unnötige Sprachelemente wurde verzichtet. Oft weist die Sprache ein offenes, gemeinschaftlich genutztes Entwicklungsmodell auf. Eine umfangreiche Anzahl an Standardbibliotheken (über 125000) und der Python-Interpreter sind als Quelltexte und in binärer Form frei verfügbar. In Python können Befehle direkt eingegeben und auf ihre Wirkung hin überprüft werden, wodurch das Erlernen der Sprache erleichtert wird. Durch den interaktiven Interpreter ist es einfach Codefragmente auszuprobieren. Java zum Beispiel weist keinen solchen Interpreter auf, weil Java keine Interpretersprache ist, sondern den Code vor der Ausführung compiliert.⁵³

Doch ist sie mit all ihren Zusatzelementen eine sehr mächtige Programmiersprache⁵⁴. Zudem ist Python eine der interessantesten Sprachen für Datenanalyseprojekte, da sie sich wegen ihrer klaren Syntax und leicht verständlichen Struktur perfekt dafür eignet. Wird Python richtig angewendet, wird es relativ einfach große Datenmengen zu analysieren. Dadurch können Betriebe rasch aus ihren Datenbeständen neue Erkenntnisse erhalten.

Python ist zudem einfacher zu lesen. Durch die unterschiedliche Syntax, gegenüber anderen Sprachen, sehen Quellcodes in Python sauberer aus. Es müssen keine speziellen Zeichen eingeführt werden, allerdings bekommt die Bedeutung von Leerraum in Python eine wichtigere Relevanz zugesprochen. Ein richtiges Einrücken kann über die Funktionalität entscheiden. Der Leerraum hilft aber auch, dass zum

⁵³ DEVInsider, Der Unterschied von Compiler und Interpreter, online unter: <https://www.dev-insider.de/der-unterschied-von-compiler-und-interpreter-a-742282/> (letzter Zugriff: 03.08.2020).

⁵⁴ Python Tutorial, Das Python-Tutorial, online unter: <https://py-tutorial-de.readthedocs.io/de/python-3.3/> (letzter Zugriff: 01.07.2020).

Vgl. Michel Weigend, Python 3. Lernen und professionell anwenden. Das umfassende Praxisbuch, 8. Auflage, Frenchen 2019, S. 2.

Beispiel das Setzen von strukturwichtigen geschweiften Klammern, vermieden werden kann. So fällt es einem leichter, die tatsächliche Struktur und die Algorithmen im Code zu beachten. Dadurch lässt sich ein Code auch leichter warten.⁵⁵

Auch die objektorientierenden Eigenschaften sind leicht verständlich, aber dennoch mächtig. Die Sprache unterstützt diese Eigenschaften ohne einen Großteil des syntaktischen Overheads, der bei vielen anderen objektorientierten Sprachen zu finden ist. Die Schnittstellen werden weniger streng definiert als in Java. Dadurch wird es einfacher Objekte zu definieren, ohne einen Code zu schreiben, der nur für die Beschreibung existiert. Zusammen mit dem Vorteil, dass Standardbibliotheken eine große Zahl an Schnittstellen besitzen, zeigt sich die Besonderheit in der Wiederverwendbarkeit der Objekte.⁵⁶

Seit dem Jahre 2008 wurde eine neue Version, Python 3, veröffentlicht, die mit dem Vorgänger nicht mehr kompatibel ist. Zum Unterschied zu seinen Vorgängerversionen wurden Redundanzen bei Befehlssätzen entfernt. Dadurch sind schönere Programmtexte möglich.⁵⁷

Python verfügt über eine beeindruckende Vielfalt an Codierungen und einen hervorragenden XML-Parser, der Zeichendaten aus XML als Unicode-Strings zur Verfügung stellt. Die Standardbibliothek weist auch Implementierungen der Standardschnittstellen DOM und SAX auf. Darüber hinaus wird Python auch durch alternativ verfügbare Parser und Schnittstellen unterstützt. Natürlich weisen aber auch andere Sprachen, wie Java und Perl, beeindruckende Bibliotheken von nützlichen Datenstrukturen auf. Aber wie bereits erwähnt, bringt Python beim Programmieren deutlich seine Vorteile mit sich.⁵⁸

⁵⁵ data <2>type, Die Stärke von Python und XML, online unter: <https://www.data2type.de/xml-xslt-xslfo/xml/python-und-xml/einstieg-in-python-und-xml/die-staerke-von-python-und-xml/> (letzter Zugriff: 01.07.2020).

⁵⁶ data <2>type, Die Stärke von Python und XML, online unter: <https://www.data2type.de/xml-xslt-xslfo/xml/python-und-xml/einstieg-in-python-und-xml/die-staerke-von-python-und-xml/> (letzter Zugriff: 01.07.2020).

⁵⁷ Michael Weigend, Python 3. Lernen und professionell anwenden. Das umfassende Praxisbuch, 8. Auflage, Frenchen 2019, S. 21.

⁵⁸ data <2>type, Die Stärke von Python und XML, online unter: <https://www.data2type.de/xml-xslt-xslfo/xml/python-und-xml/einstieg-in-python-und-xml/die-staerke-von-python-und-xml/> (letzter Zugriff: 01.07.2020).

Was ist nun ein Parser, der mitunter eines der wichtigste Teil im vorliegenden Programm darstellt? Ein Parser ist ein Computerprogramm, dass Zeichenketten analysiert, zerlegt und umwandelt. Dadurch wird der Quellcode aufbereitet. Folglich kann das Dokument in ein geeigneteres Format übersetzt und weiterverarbeitet werden. Ein XML-Parser zum Beispiel analysiert das XML-Dokument, wodurch die darin erhaltenen Informationen, wie Elemente und Attribute, für die Weiterverarbeitung zur Verfügung gestellt werden.⁵⁹

Damit man auf die einzelnen Elemente gezielt zugreifen kann, könnte man Textverarbeitungstechniken anwenden. Praktischer ist es aber die textuellen XML-Dokumente zunächst in logische Bestandteile zu zerlegen umso eine Datenstruktur aufzubauen, die die Baumstruktur wiedergibt. Dieser Vorgang wird als Parsen bezeichnet. Python Standardbibliotheken bieten dafür eine nützliche Anzahl von Schnittstellen für die Arbeit mit XML. Sie abstrahieren von der Implementierung in einer bestimmten Programmiersprache. Dabei werden die Namen der Klassen, die Attribute und Methoden festgelegt.⁶⁰ Die beiden bekanntesten Zugriffsmechanismen sind das *Simple API XML* (SAX) und das *Document Objekt Model* (DOM). Beide unterscheiden sich stark voneinander. SAX ist eine relativ primitive Schnittstelle, die keine großen Datenstrukturen aufbaut. Dadurch ist es schneller und leichter größere Dokumente effizient zu verarbeiten. DOM aber übergibt dem Anwender das gesamte Dokument, aus dem die Datei erst herausgefunden werden muss. Die Anwendung erzeugt aus dem XML-Dokument ein Objekt, auch Knoten genannt, die Teile eines Dokuments darstellen. Dieses Dokument wird als hierarchische Struktur dargestellt.⁶¹ In einem dritten Schritt wird aus dem geänderten Objekt wieder ein textuelles XML-Dokument erstellt. Außerdem ist DOM sprachübergreifend, so zum Beispiel auch in Java implementiert.⁶² Für das

⁵⁹ vgl. <https://de.wikipedia.org/wiki/Parser> (letzter Zugriff: 02.07.2020).

⁶⁰ Michael Weigend, Python 3. Lernen und professionell anwenden. Das umfassende Praxisbuch, 8. Auflage, Frenchen 2019, S. 736.

⁶¹ data <2>type, Die Stärke von Python und XML, online unter: <https://www.data2type.de/xml-xslt-xslfo/xml/python-und-xml/einstieg-in-python-und-xml/die-staerke-von-python-und-xml/> (letzter Zugriff: 02.07.2020).

⁶² Michael Weigend, Python 3. Lernen und professionell anwenden. Das umfassende Praxisbuch, 8. Auflage, Frenchen 2019, S. 736.

vorliegende Programm wurde aber The *ElementTree XML API* in Python verwendet. Es ist eine der einfachsten und leichtesten XML-Prozessor Schnittstellen. Der 'ElementTree' in diesem Modul behandelt das gesamte XML-Dokument als Baum. Die Klasse 'Element' repräsentiert einen einzelnen Knoten in diesem Baum. Lese- und Schreibvorgänge für XML-Dateien werden auf ElementTree-Ebene ausgeführt. Interaktionen mit einem einzelnen XML-Element und seinen Unterelementen werden auf eben dieser Elementebene durchgeführt.⁶³

3.5.3 SQL & MySQL

Structured Query Language (SQL) ist eine strukturierte Datenbankabfragesprache, die genutzt wird, um Daten zu strukturieren und zu verwalten. Im Zentrum der Sprache lassen sich Daten in Tabellen abfragen, einfügen, löschen und aktualisieren. Demgegenüber haben unterschiedliche Anbieter zahlreiche Datenbanksysteme veröffentlicht. Diese Datenbanksysteme, die Informationen in Tabellen speichern, werden als relationale Datenbanksysteme bezeichnet. Zu den führenden Vertretern gehört MySQL. Sie sind sehr weit verbreitet und werden auch in der vorliegenden Ausarbeitung verwendet. Der Vorteil an MySQL ist, dass sie kostenlos angeboten wird. Oft wird sie mit der Web-Programmiersprache PHP zusammen verwendet. Zudem steht das System auf den wichtigsten Betriebssystemen zur Verfügung. Die Abfragesprache in den einzelnen Datenbanksystemen unterscheiden sich zwar etwas, das heißt sie bilden Dialekte aus, doch grundlegend beruhen sie auf den SQL-Standards. Dieser Standard ist durch die ISO vorgegeben.⁶⁴

⁶³ Tutorialspoint, The ElementTree XML API in Python, online unter: <https://www.tutorialspoint.com/the-elementtree-xml-api-in-python> (letzter Zugriff: 02.07.2020).

⁶⁴ Michael Laube, Einstieg in SQL. 2. Auflage, Bonn 2019, S. 19-24.

mysql> DESCRIBE thesis;

Field	Type	Null	Key	Default	Extra
URN	varchar(250)	NO	PRI	NULL	
title	varchar(250)	NO		NULL	
subTitle	varchar(250)	YES		NULL	
author	varchar(60)	NO		NULL	
language	varchar(25)	NO		NULL	
supervisor	varchar(60)	NO		NULL	
sec_supervisor	varchar(60)	YES		NULL	
genre	varchar(80)	NO		NULL	
university	varchar(150)	NO		NULL	
production	int	NO		NULL	
abstract	longtext	YES		NULL	

11 rows in set (0.06 sec)|

Abbildung: In MySQL erstellter TABLE mit den relevanten Feldern für den vorliegenden Prototyp (Angezeigt im Editor).

3.6 Virtualisierungssoftware

Virtualisierung wurde entwickelt um eine virtuelle Version eines kompletten Systems, inklusive Betriebssystem und emulierter Hardware, zu schaffen. So können unterschiedliche Betriebssysteme auf einer Hardware zum Laufen gebracht werden. Diese Technik ist besonders beliebt bei der Softwareentwicklung, um so eine sichere Umgebung zum Testen zu erzeugen. Selbst wenn eine Applikation ein komplettes Betriebssystem zum Abstürzen bringt, laufen die anderen virtuellen Maschinen weiter. Um Applikationen zum Beispiel aus Gründen der Sicherheit oder des Datenschutzes getrennt zu halten, dafür aber nicht jeweils eine eigene Hardware bereitgestellt werden kann, werden virtuelle Maschinen erzeugt. Auf diesen können die Anwendungen getrennt laufen.

Virtuell ist dabei die Hardware die dem Betriebssystem zur Verfügung steht. Sie stehen den parallel arbeitenden Betriebssystemen nicht direkt zur Verfügung und werden über eine im Hintergrund laufende Virtualisierungssoftware überwacht und verwaltet.

Um die Hardware eines Hauptrechners zu simulieren muss die virtuelle Maschine auf einem tatsächlichen System aufgesetzt werden. Als Instanz, um eine Verbindung zwischen den beiden Ebenen zu erstellen, dient eine sogenannte Hyperversion. Diese dient als

Vermittler, ohne sich zu sehr von den unterschiedlichen Ebenen beeinflussen zu lassen. Zudem wird eine Abstraktionsschicht zur Verfügung gestellt, die verhindert, dass Treiber auf die Hardware zu greifen. So kann sichergestellt werden, dass sich die unterschiedlichen Betriebssysteme nicht in die Quere kommen.

Dabei lassen sich zwei Arten von Hyperversionen unterscheiden:

- Typ 1 setzt dabei, als Betriebssystem, direkt auf der Hardware auf und muss über notwendige Treiber unterstützt werden. Der Vorteil ist allerdings, dass das Gesamtsystem weniger Ressourcen verbraucht. Dadurch zeichnet der Hypervisor sich als ein vollwertiges Betriebssystem aus. Es wird eine Virtualisierungsschicht gebildet, bei der mehrere Betriebssysteme gleichzeitig auf einem System betrieben werden können. Ein Konflikt wird allerdings durch den Hypervisor verhindert, wodurch dadurch ergeben sich aber sehr hohe Sicherheitsanforderungen an ihn gestellt werden.
- Unter einem Typ-2-Hypervisor versteht man eine Anwendung, die auf ein vollwertiges Betriebssystem setzt und alle Ressourcen nützt, die ihr zur Verfügung stehen. Diese ist dadurch auf allen Systemen lauffähig (vergleiche VirtualBox).

Neben dem Hypervisor gibt es auch Betriebssystem-Virtualisierungen mit Container (vergleiche Docker). Sie wirken auf den ersten Blick dem des Hypervisor sehr ähnlich. Dabei läuft nur ein Betriebssystem worauf mehrere virtuelle Laufzeitumgebungen erzeugt werden. Diese sind vom Rest des Systems isoliert. Diese Container wirken auf die laufenden Programme wie normale Betriebssysteme. Da sie dadurch nur Abbilder des Wirtssystems sind, sind die Laufzeitumgebungen schnell erzeugt. Es können aber keine einzelnen Abbildungen verändert werden. Die Abbilder werden nur über das Grundsystem verändert. Im Gegensatz zu den virtuellen Maschinen, die ein eigenes Betriebssystem aufweisen um dadurch mehrere ressourcenintensive Funktionen auf einmal auszuführen, sind Container weniger aufwendig.⁶⁵

⁶⁵ Vgl.

- <https://www.elektronik-kompodium.de/sites/com/1101011.htm> (letzter Zugriff: 01.07.2020).
- <https://www.redhat.com/de/topics/virtualization/what-is-a-hypervisor> (letzter Zugriff: 01.07.2020).
- <https://www.redhat.com/de/topics/containers/whats-a-linux-container> (letzter Zugriff: 01.07.2020).
- <https://www.ionos.at/digitalguide/server/knowhow/was-ist-ein-hypervisor/> (letzter Zugriff: 01.07.2020).
- Karl Matthias/Sean P. Kane, Docker. Deployment, Testen und Debugging von Containern in Produktivumgebungen. Praxiseinstieg, 2. Auflage, Frechen 2020, S. 125.

3.6.1 Oracle VirtualBox

Die VirtualBox (VBox) ist eine Virtualisierungssoftware des US-Amerikanischen Unternehmens Oracle. Sie wurde ursprünglich von Innotek entwickelt, einem Unternehmen, das später von SunMicrosystems im Jahr 2008 übernommen wurde. Oracle VM VirtualBox wird als Anwendung auf einem vorhandenen Host-Betriebssystem installiert. Mit dieser Hostanwendung können zusätzliche Gastbetriebssysteme, die jeweils als Gastbetriebssystem bezeichnet werden, geladen und ausgeführt werden, wobei jedes über eine eigene virtuelle Umgebung verfügt.⁶⁶ Es ist sowohl für den professionellen als auch für den privaten Gebrauch zu nützen, da es als Open-Source-Software frei verfügbar ist. VirtualBox ist ein universeller Vollvirtualisierer für x86-Hardware, der für Server-, Desktop- und Embedded-Anwendungen ausgerichtet ist. Die VBox verfügt über verschiedene Erstellungsprozesse für Windows-, Mac OS X-, Linux- und Solaris-Hosts. Dabei werden Festplatten in Containerdateien emuliert.

„Eine Containerdatei ist in der digitalen Datenverarbeitung eine Datei, die ihrerseits wiederum unterschiedliche Dateien und Dateitypen enthalten kann und eine einheitliche Dateierweiterung besitzt. Die Art und Weise der Anordnung dieser Dateien wird als Containerformat bezeichnet. Der einfachste Fall einer Containerdatei ist das Zusammenfassen mehrerer Dateien in einer (oft komprimierten) Archivdatei, wie beispielsweise bei den Dateiformaten TAR oder ZIP.“⁶⁷

Daneben kann die VBox auch mit unterschiedlichen Festplattendateien umgehen. Es wird eine große Datei auf der Hostfestplatte als virtuelle Festplatte für den Gast angelegt.⁶⁸

Über eine Gasterweiterung kann eine Integration zwischen Wirt- und Gastsystem durchgeführt werden und so sind zum Beispiel „Gemeinsame Ordner“ und eine

⁶⁶ Viser, Predavanje, online unter: <https://viser.edu.rs/uploads/2018/03/7.%20Predavanje%20-%202.pdf> (letzter Zugriff: 01.07.2020).

⁶⁷ mikiwiki, Containerdatei, online unter: <http://mikiwiki.org/wiki/Containerdatei> [(letzter Zugriff: 01.07.2020)].

⁶⁸ vgl. Linuxforen, online unter: <https://www.linuxforen.de/forums/showthread.php?236444-VirtualBox&p=1534894#post1534894> (letzter Zugriff: 01.07.2020).

Drag and Drop Funktion möglich. Auch die Grafikauflösung kann so an die Fenstergröße im Wirt angepasst werden.⁶⁹

Vereinfacht gesagt bildet die VBox einen eigenen PC ab, auf dem jedes Betriebssystem installiert werden kann. Dadurch kann ohne Rebooten sowohl Windows als auch Linux parallel betrieben werden. In der vorliegenden Arbeit wurde ein Linux Betriebssystem simuliert. Da allerdings die Steuerung hauptsächlich über die Kommandozeile erfolgt, ist der Einstieg ins Linux-System etwas schwieriger als in das benutzerfreundlichere Windows.

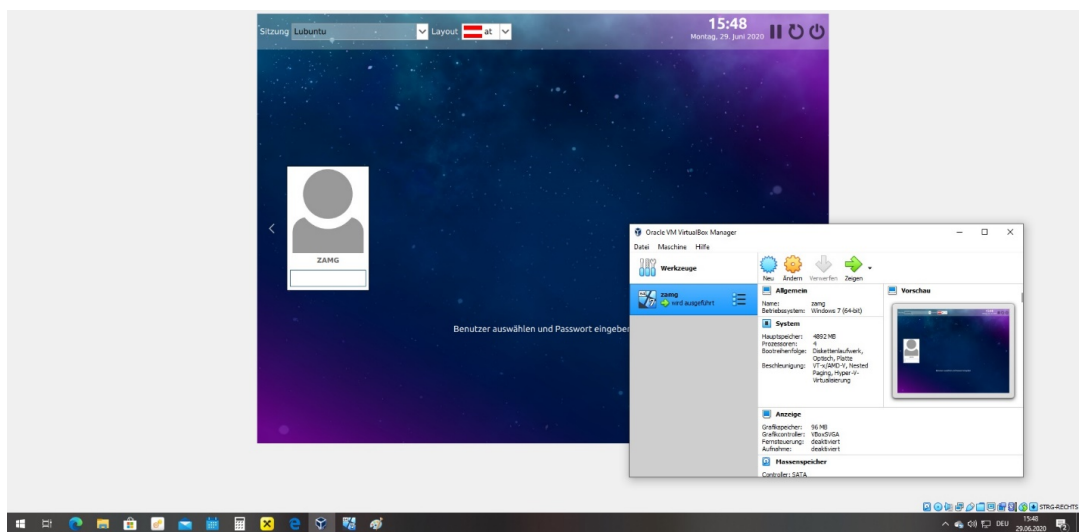


Abbildung: Die Abbildung zeigt den Oracle VM VirtualBox Manager und die Linux-Startseite mit Passwortaufforderung.

⁶⁹ vgl. <https://de.wikipedia.org/wiki/VirtualBox> (letzter Zugriff: 01.07.2020).

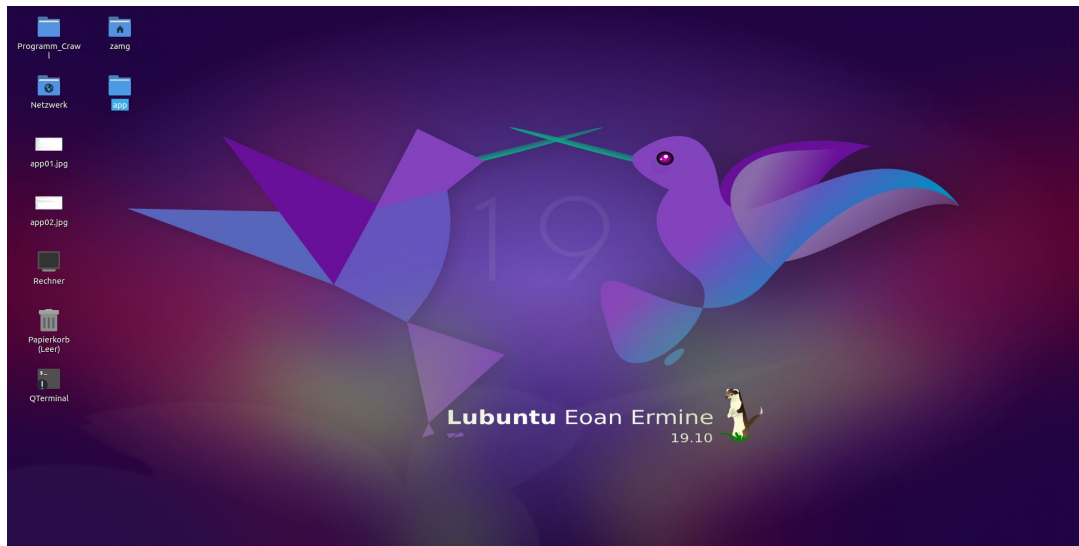


Abbildung: Desktop der VBox-Anwendung

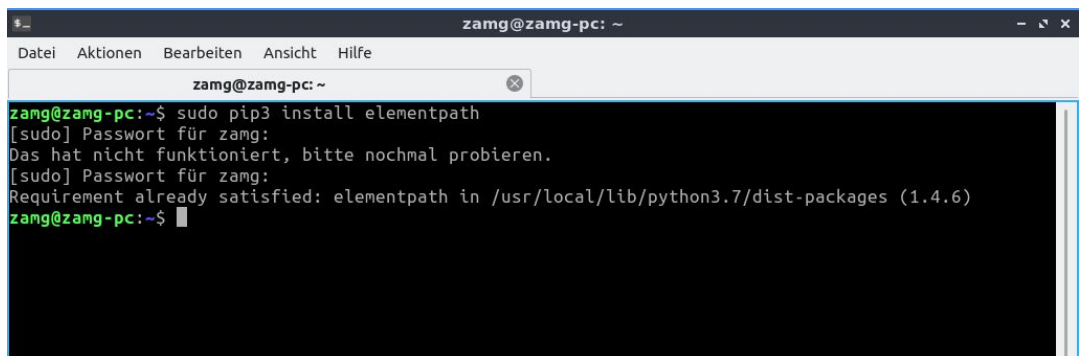


Abbildung: Terminal der Linux-Anwendung. Hauptsächlich wird in Linux über Befehle gearbeitet.

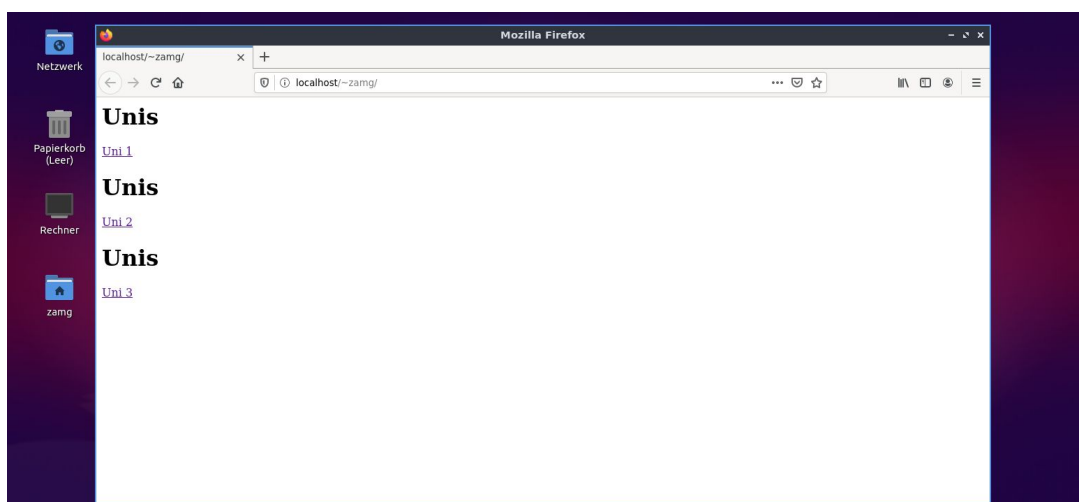


Abbildung: Lokal erstellte Webseite zum Abrufen der Unis. Auf diese Seite greift auch der Crawler zu, damit die XML-Daten geharvestet werden können.

3.6.2 Docker

„Docker ist ein Tool, das eine einfache Kapselung des Erstellungsprozesses von Artefakten zur Verteilung beliebiger Anwendungen verspricht. Dabei ist das Deployment für beliebige Umgebungen skalierbar, und der Workflow sowie die Reaktionsfähigkeit der betreffenden agilen Unternehmen werden optimiert.“⁷⁰

Docker wurde zum ersten Mal auf der Python Developers Conference in Santa Clara, Kalifornien, am 15.03.2013, als Revolution der Softwareentwicklung, -verteilung und -nutzung, vorgestellt. Das Produkt wurde wenige Wochen später bereits auf GitHub⁷¹ als Open Source zur Verfügung gestellt. Docker ist grundlegend auf die Virtualisierung mit Linux ausgerichtet, kann aber auch mittels Hyper-V oder VirtualBox auf Windows verwendet werden.

Docker überspannt eine Reihe von Branchensegmente, wenngleich es vordergründig als Virtualisierungsplattform verstanden wird. Es ist ein guter Ausgangspunkt, um in der Praxis auftretende organisatorische Probleme zu lösen und ermöglicht zudem ein schnelleres Entwickeln von Software. Docker ermöglicht die zu entwickelte Software zu isolieren, wodurch einerseits eine geringere Kommunikation zwischen Teams erforderlich ist. Ein Entwicklungsteam muss daher nicht mehr darauf warten, dass ein Team aus Administratoren eine Library auf einer höheren Version bereitstellt. Andererseits wirkt sich der Gebrauch von Container auch hinsichtlich der Softwarearchitektur recht bestimmend aus, wodurch die Anwendung begünstigt wird und besonders stabil angelegt wird. Docker arbeitet auf einem Image-basierten Bereitstellungsmodell, wodurch es leichter wird eine Anwendung von Services mit ihren Abhängigkeiten in mehreren Umgebungen zusammen zu nutzen. Bemerkenswert ist auch, dass Docker nicht nur ein init-System aufweisen, wodurch mehrere Prozesse gemeinsam verwaltet werden können, sondern auch eine Aufschlüsselung von Anwendungen in ihren einzelnen Prozessen. Durch Wegwerf-Container überlebt in der neuen Anwendung nur die Anwendung selbst. So kann es verhindert werden, dass ein Programm auf

⁷⁰ Karl Matthias/Sean P. Kane, Docker. Deployment, Testen und Debugging von Containern in Produktivumgebungen. Praxiseinstieg, 2. Auflage, Frechen 2020, S. 25.

⁷¹vgl. <https://github.com/> (letzter Zugriff: 01.07.2020).

Überbleibsel einer vorhergehenden Version zugreift. Durch den Linux Kernel können einzelne Prozesse auch isoliert und so unabhängig voneinander ausgeführt werden. So kann die Fähigkeit, mehrere Prozesse und Apps getrennt voneinander zu betreiben, genutzt werden. Ein weiterer Vorteil liegt in der hohen Portabilität, wodurch Anwendungen sich leicht auf einen anderen Server verschieben lassen. So sind Anwendungen besser skalierbar und zuverlässiger.

Docker-Container bestehen aus Prozessen, die auf dem Host selbst laufen. Wurde anfänglich nur ein einzelnes Netzwerk-Layer unterstützt, werden mittlerweile eine ganze Reihe von Konfigurationen für die Anwendungsanforderungen erfüllt. Als Standardkonfiguration wird meistens die *Bridge Mode* benutzt. Der Docker-Server fungiert als Brücke und die Container sind die daran angeschlossenen Clients. Er leitet über zwei Anschlüsse einfach den Daten-Traffic von einem zum anderen über. Bildlich gesehen kann er sich als virtuelles Netzwerk vorgestellt werden, an dem der Host angeschlossen ist.⁷²

Ein Nachteil bei der Nutzung von Docker könnte sich ergeben, sobald immer mehr Container erstellt werden, um eine App in ihre Bestandteile zu zerlegen. Dadurch könnte die Verwaltung und Orchestrierung sehr schwierig werden. Umgangen kann das Problem durch Gruppierung der einzelnen Container werden. Auch kann es zu Einschränkungen beim Säubern von untergeordneten Prozessen kommen. Abhilfe schafft eine Änderung der Konfigurationsdatei.⁷³

Im Projekt wurde Docker hauptsächlich aufgrund seiner Portabilität genutzt, da die VBox aufgrund ihrer Datenmenge nur schwer versendbar wurde. Dadurch war es leichter mit dem Betreuer Andreas Predikaka zusammenzuarbeiten.

⁷² Karl Matthias/Sean P. Kane., Docker. Deployment, Testen und Debugging von Containern in Produktivumgebungen. Praxiseinstieg, 2. Auflage, Frechen 2020, S. 25-44.

⁷³ redhat, Docker. Funktionsweise, Vorteile, Einschränkungen, online unter: <https://www.redhat.com/de/topics/containers/what-is-docker> (letzter Zugriff: 01.07.2020).

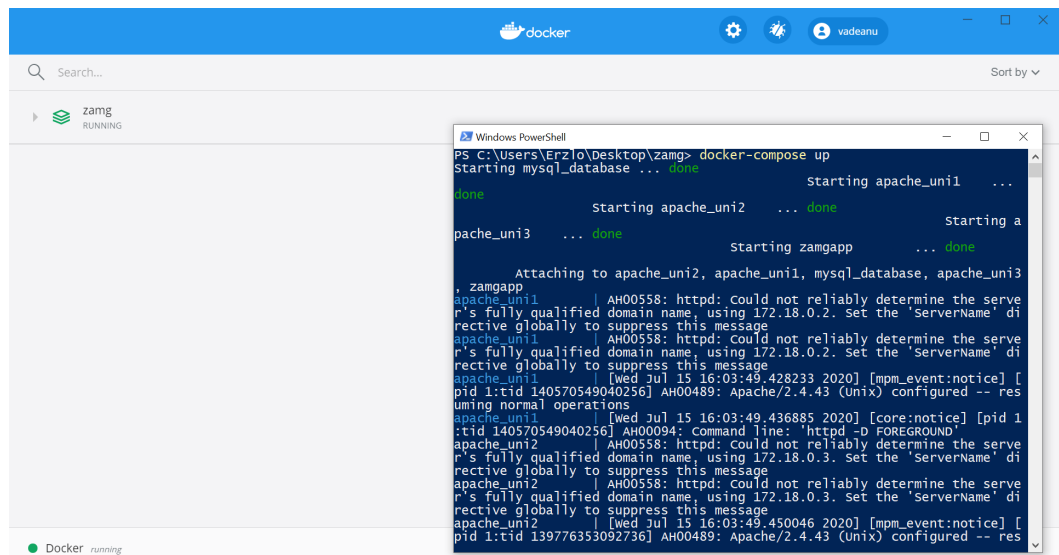


Abbildung: Docker-Desktop Anwendung für Windows. Hauptsächlich wird bei Docker über die Eingabeaufforderung oder über eine PowerShell gearbeitet.

3.7 Programm-Dokumentation

Auch wenn der Quellcode in der Regel selbsterklärend sein sollte, beziehungsweise bereits im Quellcode durch Kommentare mit eingebunden wurde, wird dennoch hierbei versucht auch für nicht Informatiker den Aufbau des Programmes zu beschreiben. Der Code wurde hauptsächlich in PyCharm geschrieben. Es ist eine integrierte Entwicklungsumgebung des Unternehmens JetBrains für Python. Die Software ist sowohl kostenlos als auch als Professionelle-Version verfügbar.⁷⁴

Im Weiteren werden die einzelnen Programmteile vorgestellt, die zusammen mit Docker-Container (bzw. VBox) den Prototyp ergeben. Das Programm läuft dabei überall wo auch Docker läuft. Für Dokumentationszwecken, bzw. um anderen die Möglichkeit zu geben an dem Programm weiterzuarbeiten, wurde der Code auch auf github.com⁷⁵ gestellt. GitHub selber ist ein netzbasierter Dienst zur Versionsverwaltung für Software-Entwicklungsprojekte.⁷⁶

⁷⁴jetbrains, PyCharm, The Python IDE for Professional Developers, online unter: https://www.jetbrains.com/pycharm/promo/?gclid=EALaIQobChMIhVpMht_K6glVa4QBh3Mjg-OEAAAYASAAEgJhRPD_BwE (letzter Zugriff: 13.07.2020).

⁷⁵ vgl. <https://github.com/vadeanu/ULG-PROJEKT-ZAMG> (letzter Zugriff: 13.07.2020).

⁷⁶vgl. <https://de.wikipedia.org/wiki/GitHub> (letzter Zugriff: 13.07.2020).

3.7.1 Der Crawler

Der Crawler ist einfach aufgebaut und crawlt in erster Linie einfach eine URL ohne Authentifizierung. Die Authentifizierung wurde bei dem Prototyp der Einfachheit halber weggelassen. Der Inhalt des Crawlers wird in einem Textfeld gespeichert. Die Daten werden dabei nur in Memory gehalten, damit der Code schlanker wird.

```
import requests

class Crawler:
    successful = False
    text = ""

    def crawl(self, url):
        self.successful = False
        self.text = ""
        result = requests.get(url)
        if result.status_code == 200:
            self.successful = True
            self.text = result.text

    def wasSuccessful(self):
        return self.successful

    def getText(self):
        return self.text
```

Da ein Crawler auch hin und wieder nicht funktionieren kann, muss der Erfolg überprüft werden („wasSuccessful“). Mit der Definition „getText“ wird der gespeicherte Text für die Weiterverarbeitung geholt.

3.7.2 Die Thesis-Klasse

In dieser Klasse werden die Metadaten gehalten, die geparkt und in die Datenbank eingespielt werden sollen. Hauptbestandteil des Codes sind Getter- und Setter-Funktionen, um die Metadaten zu setzen und auszulesen. Auch ist eine „isValid Funktion“ vorhanden, um zu überprüfen ob die Daten in den Feldern valide sind. Mit der Funktion „inspect“ kann der Inhalt aller Metadatenfelder ausgegeben werden.

```
class Thesis:
    urn = ""
    title = ""
```



```

subtitle = ""
author = ""
language = ""
supervisor = ""
sec_supervisor = ""
genre = ""
university = ""
production = -1
abstract = ""

def getUrn(self):
    return self.urn

def setUrn(self, urn):
    self.urn = urn

def getTitle(self):
    return self.title

def setTitle(self, title):
    self.title = title

def getSubtitle(self):
    return self.subtitle

def setSubtitle(self, subtitle):
    if subtitle is None:
        subtitle = ""
    self.subtitle = subtitle

def getAuthor(self):
    return self.author

def setAuthor(self, author):
    self.author = author

def getLanguage(self):
    return self.language

def setLanguage(self, language):
    self.language = language

def getSupervisor(self):
    return self.supervisor

def setSupervisor(self, supervisor):
    self.supervisor = supervisor

def getSec_supervisor(self):
    return self.sec_supervisor

def setSec_supervisor(self, sec_supervisor):
    if sec_supervisor is None:
        sec_supervisor = ""
    self.sec_supervisor = sec_supervisor

def getGenre(self):
    return self.genre

def setGenre(self, genre):
    self.genre = genre

def getUniversity(self):
    return self.university

def setUniversity(self, university):
    self.university = university

def getProduction(self):
    return self.production

def setProduction(self, production):
    self.production = production

def getAbstract(self):
    return self.abstract

def setAbstract(self, abstract):
    if abstract is None:

```

```

        abstract = ""
        self.abstract = abstract

def isValid(self):
    if self.urn == "" or \
        self.title == "" or \
        self.author == "" or \
        self.language == "" or \
        self.supervisor == "" or \
        self.genre == "" or \
        self.university == "" or \
        self.production == -1:
        return False
    else:
        return True

def inspect(self):
    print("urn: " + self.urn)
    print("title: " + self.title)
    print("subtitle: " + self.subtitle)
    print("author: " + self.author)
    print("supervisor: " + self.supervisor)
    print("sec_supervisor: " + self.sec_supervisor)
    print("language: " + self.language)
    print("production: " + repr(self.production))
    print("abstract: " + self.abstract)
    print("genre: " + self.genre)
    print("university: " + self.university)
    print("is valid: " + str(self.isValid()))
    return „Test“

```

3.7.3 Der Parser

Dieser Code ist das Herzstück der Applikation.

```
import xml.etree.ElementTree as ET
from thesis import Thesis
from crawler import Crawler

class XmlParser:

    def readXML(self, xmldata):

        #fähgt korrupte Dateien auf, falls Fehler in XML-files:
        obj = Thesis()
        try:
            root = ET.fromstring(xmldata)
        except:
            print('Datensatz fehlerhaft')
            print(xmldata)
            return obj

        family = ""
        given = ""

        for element in root:

            if element.tag == "identifizier":
                URN = element.text
                obj.setUrn(URN)
            if str(element.attrib) == '{"type": "thesis"}':
                #print('yessssssss')
                thesis = element.text
                #print (thesis)
                genre = thesis.split(' -- ')[0]
                obj.setGenre(genre)
                year = thesis.split(' -- ', ) [1]
                university = year.split(', ')[0]
                obj.setUniversity(university)
            for subelement in element:
                if subelement.tag == "title":
                    title = subelement.text
                    obj.setTitle(title)
                if subelement.tag == "subTitle":
                    subTitle = subelement.text
                    obj.setSubtitle(subTitle)
                if str(subelement.attrib) == '{"type": "family"}':
                    #print("treffer")
                    family = subelement.text
                if str(subelement.attrib) == '{"type": "given"}':
                    given = subelement.text
                if subelement.tag == 'dateIssued':
                    production = subelement.text
                    obj.setProduction(production)
            # for subelement in element:
            if str(element.attrib) == '{"type": "Betreuer"}':
                Betreuer = element.text
                obj.setSupervisor(Betreuer)
            if str(element.attrib) == '{"type": "Betreuer2"}':
                Betreuer2 = element.text
                obj.setSec_supervisor(Betreuer2)
            if element.tag == 'language':
                language = element.text
                obj.setLanguage(language)
            if element.tag == 'abstract':
                abstract = element.text
                obj.setAbstract(abstract)

            autor = family + ", " + given
            obj.setAuthor(autor)
        #Einrückung beachten!
        return obj
```

Der Parser ist so geschrieben, dass er jedes einzelne XML-Element prüft und die vordefinierten Stellen ausliest. Die Daten bekommt er über den Crawler in der Klasse „Thesis“ (thesis.py) geliefert. Am Codeanfang wird auf eventuell korrupt eingelesene Daten geprüft, damit es später zu keinen Fehlermeldungen kommt. Um Subelemente (Kinderknoten von Kinderknoten) richtig lesen zu können, werden die Attribute abgefragt. Wichtig ist beim Codieren in Python die richtige Einrückung zu beachten. Ein Leerzeichen mehr oder weniger entscheidet über die Funktionalität.

3.7.4 Die Webapplikation

Diese Applikation wurde zusammen mit Andreas Predikaka, dem Betreuer der vorliegenden Arbeit, geschrieben. Grundsätzlich wird mit crawlthesis über die drei Serverurls geloopt⁷⁷ und für jeden Server das thesis.txt File eingelesen, um in einem weitergehenden Schritt für jede Zeile im Text das XML einzulesen. Vom Parser wird dann ein Thesis-Objekt erstellt. Durch einen URN Check wird überprüft ob das Objekt bereits in der Datenbank vorhanden ist, wodurch doppelte Einträge verhindert werden.

```
from typing import List, Dict
from flask import Flask
import logging
from flask_mysql import MySQL
from crawler import Crawler
from xmlparser import XmlParser

app = Flask(__name__)
logging.basicConfig(level=logging.INFO)

#Datenbank login:
app.config['MYSQL_HOST'] = 'mysql_database'
app.config['MYSQL_USER'] = 'root'
app.config['MYSQL_PASSWORD'] = 'zamg'
app.config['MYSQL_DB'] = 'zamg'

mysql = MySQL(app)

def getThesis():
    cursor = mysql.connection.cursor()
    cursor.execute('SELECT urn, title, author, supervisor,
university FROM thesis')
```

⁷⁷ Anm.: Ein Loop ist eine Programmschleife, beziehungsweise eine Folge von Programmteilen, die mehrfach durchlaufen werden können.

Vgl. Duden, online unter: <https://www.duden.de/rechtschreibung/Loop> (letzter Zugriff: 06.07.2020).

```

        results = cursor.fetchall()
        cursor.close()

    return results

def deletethesis():
    cursor = mysql.connection.cursor()
    cursor.execute('delete from thesis')
    mysql.connection.commit()

def crawlthesis():
    cursor = mysql.connection.cursor()

    serverurls = { 'http://apache_uni1', 'http://apache_uni2',
'http://apache_uni3' }

    c = Crawler()
    parser = XmlParser()

    for url in serverurls:
        app.logger.info('crawling ' + url)
        c.crawl(url + "/thesis.txt")
        if c.wasSuccessful():
            for filename in c.getText().splitlines():
                thesisurl = url + "/" + filename
                app.logger.info('crawling ' + thesisurl)
                c.crawl(thesisurl)
                if c.wasSuccessful():
                    thesis = parser.readXML(c.getText())
                    print(thesis.inspect())

                    if thesis.isValid():
                        # check ob thesis schon eingefuegt
                        sql = "select count(*) from thesis where urn
= '" + thesis.getUrn() + "'";
                        cursor.execute(sql)
                        result = cursor.fetchone()
                        numberOfRows = result[0]
                        if numberOfRows > 0:
                            app.logger.info("Thesis " +
thesis.getUrn() + " befindet sich schon in der Datenbank")
                            continue
                        #Einfuegung der geparsten Metadaten in die Datenbank:
                        sql = "insert into thesis (urn, title,
subtitle, author, language, supervisor, sec_supervisor, genre,
university, production, abstract) values (" \
                            '"' + thesis.getUrn() + "', " \
                            '"' + thesis.getTitle() + "', " \
                            '"' + thesis.getSubtitle() + "', " \
                            '"' + thesis.getAuthor() + "', " \
                            '"' + thesis.getLanguage() + "', " \
                            '"' + thesis.getSupervisor() + "', " \
                            '"' + thesis.getSec_supervisor() + "',
                            " \
                            '"' + thesis.getGenre() + "', " \
                            '"' + thesis.getUniversity() + "', " \
                            '"' + thesis.getProduction() + "', " \
                            '"' + thesis.getAbstract() + "'" \
                            ")"

                        cursor.execute(sql)
                        mysql.connection.commit()
                        app.logger.info("Thesis " + thesis.getUrn()
+ " in der Datenbank gespeichert.")

```

```

        else:
            app.logger.info("Thesis kann nicht
importiert werden. Nicht valide.")
        else:
            app.logger.info("crawl von " + thesisurl + " war
nicht erfolgreich.")

def generateHtml():
    list = getThesis()
    html = '<h1>Thesis</h1>'
    html = html + '<a href="/">Home</a> | <a href="/
crawlthesis">crawl thesis</a> | <a href="/deletethesis">delete
thesis</a><p/>'
    if len(list) > 0:
        html = html + '<table border="1">'
        html = html + '<tr><th>urn</th><th>title</th><th>author</
th><th>supervisor</th><th>university</th></tr>'
        for item in list:
            html = html + '<tr><td>' + item[0] + '</td><td>' +
item[1] + '</td><td>' + item[2] + '</td><td>' + item[3] + '</td><td>'
+ item[4] + '</td></tr>'
            html = html + '</table>'
        else:
            html = html + '<p>keine Daten</p>'

    return html

@app.route('/')
def index():
    return generateHtml()

@app.route('/deletethesis')
def delete():
    deletethesis()
    return generateHtml()

@app.route('/crawlthesis')
def crawl():
    crawlthesis()
    return generateHtml()

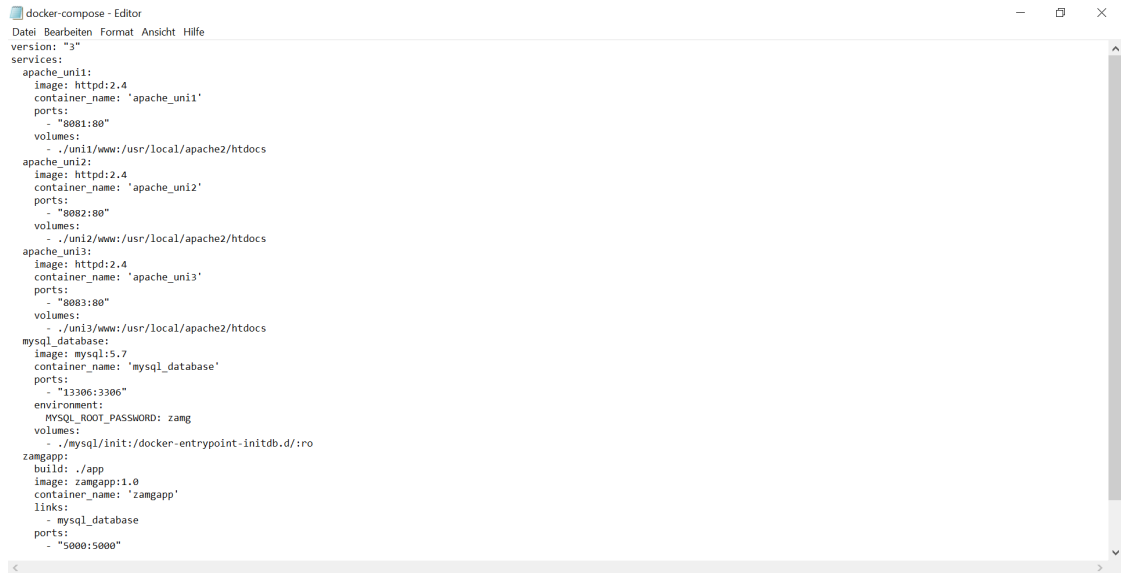
if __name__ == '__main__':
    app.run(host='0.0.0.0')

```

Weiteres wird über das Skript bestimmt, welche Metadaten über die Webapplikation angezeigt werden sollen. Auch werden am Ende des Codes die Funktionen der Applikation bestimmt. So zum Beispiel „deletethesis“ und „crawlthesis“, welche die Menüpunkte der Applikation sind. Die Webapplikation zeigt beim Aufrufen über <http://localhost:5000> alle Thesen aus der Datenbank an.

3.7.5 Die Docker Anwendung

Es wurden insgesamt fünf Container erstellt, die im Docker-compose.yml beschrieben werden.



```
docker-compose - Editor
Datei Bearbeiten Format Ansicht Hilfe
version: "3"
services:
  apache_uni1:
    image: httpd:2.4
    container_name: 'apache_uni1'
    ports:
      - "8081:80"
    volumes:
      - ./uni1/www:/usr/local/apache2/htdocs
  apache_uni2:
    image: httpd:2.4
    container_name: 'apache_uni2'
    ports:
      - "8082:80"
    volumes:
      - ./uni2/www:/usr/local/apache2/htdocs
  apache_uni3:
    image: httpd:2.4
    container_name: 'apache_uni3'
    ports:
      - "8083:80"
    volumes:
      - ./uni3/www:/usr/local/apache2/htdocs
  mysql_database:
    image: mysql:5.7
    container_name: 'mysql_database'
    ports:
      - "13306:3306"
    environment:
      MYSQL_ROOT_PASSWORD: zamg
    volumes:
      - ./mysql/init:/docker-entrypoint-initdb.d/:ro
  zamgapp:
    build: ./app
    image: zamgapp:1.0
    container_name: 'zamgapp'
    links:
      - mysql_database
    ports:
      - "5000:5000"
```

Abbildung: Docker-compose.yml File, angezeigt über den Editor.

Es laufen drei Apache Webserver Container, mit denen drei Universitäten simuliert werden. Im Univerzeichnis befinden sich jeweils die WWW-Verzeichnisse, welche die Root-Verzeichnisse für die Webseiten darstellen. In diesen Verzeichnissen befinden sich das thesis.txt File und die XML-Dateien. Über den Browser sind die einzelnen Container nach dem Start der Container (zum Beispiel <http://localhost:8081> für die UNI1) abrufbar. Außerdem gibt es noch einen Container mysql_database, in dem die Datenbank läuft. Alle vier bis jetzt genannten Container sind Standard- Container, die bereits vorgefertigt genau das machen, was für den Prototyp benötigt wird. Darüber hinaus gibt es als letzten Container, die Python Applikation in dem Crawler, Scraper und die Webapplikation läuft. Für diesen speziellen Container wird ein Dockerfile erzeugt, das von einem Python Container erbt und die notwendigen Libraries für die Pythonapplikation installiert und die Webapplikation startet.



Abbildung: Screenshot von <http://localhost:8081>

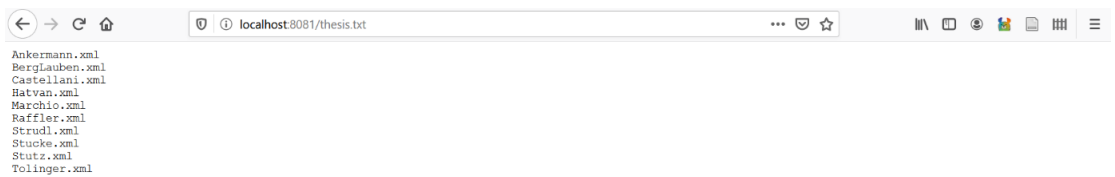


Abbildung: Screenshot von <http://localhost:8081/thesis.txt>

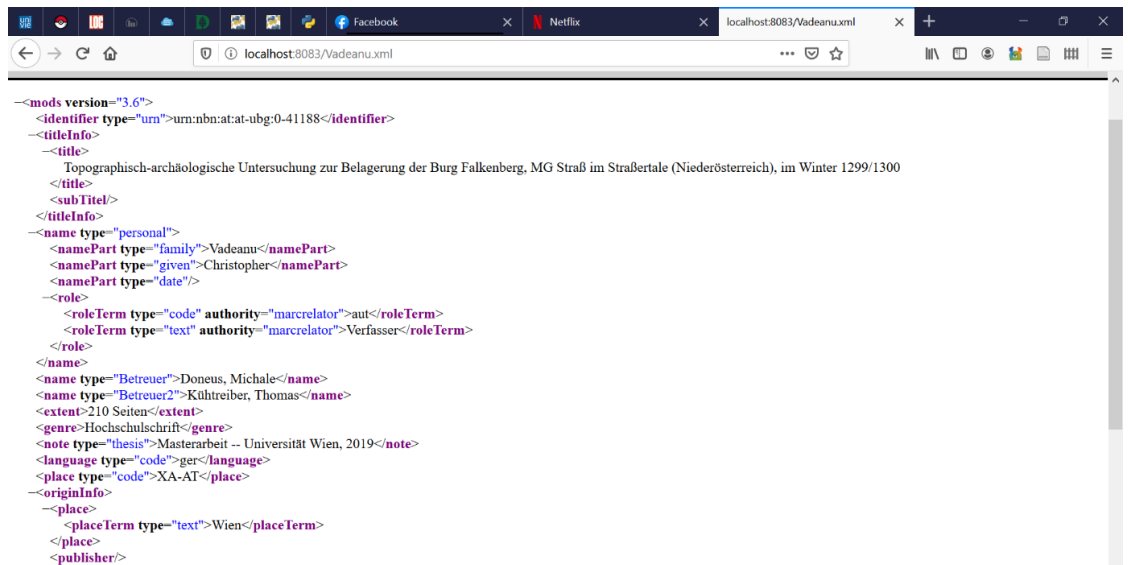


Abbildung: Screenshot von <http://localhost:8083/Vadeanu.xml>. Es können auch die einzelnen XML-Daten über den Localhost abgerufen werden.

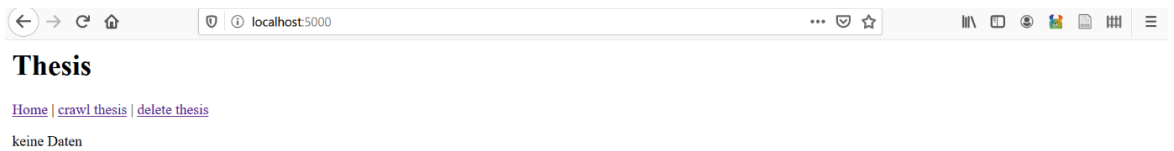


Abbildung: Screenshot von <http://localhost:5000> (es wurden noch keine Daten gecrawlt).

urn	title	author	supervisor	university
urn:nbn:at:at-ubg:0-41188	Topographisch-archäologische Untersuchung zur Belagerung der Burg Falkenberg, MG Straß im Straßertale (Niederösterreich), im Winter 1299/1300	Vadeanu, Christopher	Doneus, Michale	Universität Wien
urn:nbn:at:at-ubg:1-77101	Validierung eines Open Source CFD-Schneeverfrachtungsmodells mittels Terrestrial Laserscans	Decker, Maria	Kaemle, Norbert	Universität Graz
urn:nbn:at:at-ubi:1-03307	Globale Sensitivitätsanalyse und Kalibrierung eines hydrologischen Modells für das vergletscherte Einzugsgebiet Riffelsee	Strudl, Markus	Kaser, Georg	Universität Innsbruck
urn:nbn:at:at-ubi:1-31006	Berg.Lauben	Vigl, Barbara	Flora, Andreas	Universität Innsbruck
urn:nbn:at:at-ubi:1-33907	Orographic and atmospheric factors governing localized heavy snowfall south of the Swiss Jura	Ankermann, Jörg	Gohm, Alexander	Universität Innsbruck
urn:nbn:at:at-ubi:1-35660	Climatology of ingredients for convection with ERA5	Marchio, Mattia	Mayr, Georg	Universität Innsbruck
urn:nbn:at:at-ubi:1-40418	Evaluation of the physical SNOWPACK model under Arctic conditions	Hatvan, Veronika	Maussion, Fabien	Universität Innsbruck
urn:nbn:at:at-ubi:1-51907	Atmospheric boundary layer structure in the Inn valley	Raffler, Philipp Reinhard	Rotach, Mathias W.	Universität Innsbruck
urn:nbn:at:at-ubi:1-56189	Spatio-temporal modeling of cloud-to-ground lightning current strength and polarity in Austria and the atmospheric impact	Stucke, Isabell	Mayr, Georg	Universität Innsbruck
urn:nbn:at:at-ubi:1-60115	Estimating Glacier Ice Thickness with Machine Learning	Castellani, Matteo	Maussion, Fabien	Universität Innsbruck
urn:nbn:at:at-ubi:1-6181	Transport of Saharan Dust to the Sonnblick Observatory	Stutz, Andreas	Nickus, Ulrike	Universität Innsbruck
urn:nbn:at:at-ubw:1-29463.21369.165762-0	Dual-polarisiertes Wetterradar versus Mikrowellenradar	Plank,	Steinacker, Reinhold	Universität Wien/Fakultät für Geowissenschaften Geographie und

Abbildung: Screenshot von <http://localhost:5000> (mit gecrawlten Daten).

4. Conclusio

Dieses Projekt verfolgte das Ziel, ein automatisches Monitoring für Hochschulschriften aus dem Fachbereich Erdwissenschaft zu konzipieren.

Zunächst wurde recherchiert, wie vergleichbare Institutionen die Problematik der sehr verstreut aufbewahrten Hochschulschriften angehen. Es zeigte sich, dass ein Monitoring bisher nur sehr begrenzt zur Anwendung kommt. Vorwiegend verlassen sich die Institutionen und wissenschaftlichen Einrichtungen darauf, dass interessierte AbsolventInnen auf die Institution bzw. Einrichtung selbst zukommen.

Mit Hilfe von Andreas Predikaka wurde deshalb ein Prototyp entwickelt, welcher der ZAMG als Möglichkeit präsentiert wird, um in Zukunft ein automatisiertes Monitoring durchführen zu können. Es muss jedoch darauf hingewiesen werden, dass der Erfolg des Monitorings vor allem mit der Kooperation der Universitäten einher geht.

Mit dieser Ausarbeitung soll verdeutlicht werden, dass ein Harvesting und eine anschließende Zusammenführung der Hochschulschriften an einen Ort technisch umsetzbar ist und zu einer Verbesserung der wissenschaftlichen Tätigkeiten führen kann.

Der Vorteil dieser Vorgehensweise ist neben der einfacheren Auffindbarkeit von Hochschulschriften vor allem auch die Vernetzung der Wissenschaftlerinnen innerhalb des Fachgebiets sowie die Wissensdokumentation. Hinzu kommt, dass nicht nur die Recherche vereinfacht wird, sondern auch die momentan und vergangene wissenschaftliche Tätigkeit des Fachgebiets der Erdwissenschaften an einem Ort abgebildet werden kann.

Das Projekt konnte demnach aufzeigen, wie ein solcher Harvesting-Vorgang aussehen könnte und wie dieser im Alltag für die ZAMG und das Fachgebiet der Erdwissenschaften umgesetzt werden könnte.