File Synchronization as a Way to Add Quality Metadata to Research Data

Master Thesis - Master in Library and Information Science (MALIS)

Faculty of Information Science and Communication Studies -
Technische Hochschule Köln

Presented by: Ubbo Veentjer
on: September 27, 2016
to: Dr. Peter Kostädt (First Referee)
Prof. Dr. Andreas Henrich (Second Referee)

Abstract

Research data which is put into long term storage needs to have quality metadata attached so it may be found in the future. Metadata facilitates the reuse of data by third parties and makes it citable in new research contexts and for new research questions. However, better tools are needed to help the researchers add metadata and prepare their data for publication. These tools should integrate well in the existing research workflow of the scientists, to allow metadata enrichment even while they are creating, gathering or collecting the data. In this thesis an existing data publication tool from the project DARIAH-DE was connected to a proven file synchronization software to allow the researchers prepare the data from their personal computers and mobile devices and make it ready for publication. The goal of this thesis was to find out whether the use of file synchronization software eases the data publication process for the researchers.

Forschungsadaten, die langfristig gespeichert werden sollen, benötigen qualitativ hochwertige Metadaten um wiederauffindbar zu sein. Metadaten ermöglichen sowohl die Nachnutzung der Daten durch Dritte als auch die Zitation in neuen Forschungskontexten und unter neuen Forschungsfragen. Daher werden bessere Werkzeuge benötigt um den Forschenden bei der Metadatenvergabe und der Vorbereitung der Publikation zu unterstützen. Diese Werkzeuge sollten sich gut in den bestehenden Forschungsprozess der WissenschaftlerInnen integrieren, um die Metadatenvergabe schon während der Erstellung, Erhebung und Sammlung der Daten zu ermöglichen. In der vorliegenden Arbeit wurde ein existierendes Datenpublikationstool aus dem Projekt DARIAH-DE mit einer bewährten Dateisynchronisationssoftware verknüpft, um Daten von den Arbeitsplatzrechnern und Mobilgeräten der Forschenden früher auf eine Publikation vorbereiten zu können. Das Ziel der vorliegenden Arbeit ist es herauszufinden, ob der Einsatz von Dateisynchronisationssoftware die Datenpublikation der Forschenden vereinfachen kann.

Keywords

Research Data, Research Data Lifecycle, Metadata, File Synchronization, Data Publication

Schlagwörter

Forschungsdaten, Forschungsdatenkreislauf, Metadaten, Dateisynchronisation, Datenpublikation

Contents

1 Introduction

The project DARIAH-DE is the German part of the European DARIAH network.[1] DARIAH, the "Digital Research Infrastructure for the Arts and Humanities", focuses on building and connecting digital research data, tools and methodologies for the Arts and Humanities, with its roots in the Digital Humanities.[2] One main area of interest for DARIAH-DE is research data management.[3] Research data management needs quality metadata[4] for the research data, to preserve the context of the data far beyond a research project life cycle. DARIAH-DE offers the DARIAH-DE repository to store research data together with its metadata, adding persistent identifiers to every data object.[5]

For publishing research data to the DARIAH-DE repository, a new tool, the DARIAH-DE Publikator, is developed by the DARIAH-DE project. This is a web based user interface, which allows uploading data and describing it with metadata utilizing standard metadata vocabularies like Dublin Core (DC) metadata terms[6]. The data objects (files with metadata) are organized in collections. The Publikator is a workspace, where the progressing work on the collection is saved, before the researcher publishes it to the DARIAH-DE repository.

The browser-based approach for uploading data still has its drawbacks. Using the browser for file upload does not adopt well for data which is still changing, as the user needs to manually update changed files in the Publikator. So this does not work well for a scenario where the researcher would like to use the Publikator early in the research project for data management. The value of the Publikator would be greater if it could accompany the researcher from the beginning of the research process, as the management of metadata and the organization of data would become a continuous process, in contrary to a final step before the research project finalizes. Additionally, it may be frustrating if larger files or folders are not fully uploaded to the storage because of network instability or other reasons, so the web upload may need to be repeated manually more than once. Especially in situations where no internet connection is available, like during a visit in an archive or while traveling, the browser based approach may interfere with the researcher's work.

On the other hand there are already existing practically approved tools to manage data sharing with web locations, so-called file synchronization services.

File synchronization services like Dropbox[7] have started to play an important role in the data management of private persons, but also for researchers, as they offer a convenient way to work with data on different devices and with groups of people. Working on files stored on the own hard disk and synchronized to cloud systems in the background offer the flexibility of not caring about off- or on-line status. Also it offers the comfort to just not think about the details, the files will just be synchronized without further user interaction.

[1] DARIAH-DE (2016a)
[2] DARIAH-EU (2016)
[3] DARIAH-DE (2016g)
[4] The quality of metadata is hard to measure, but there is an agreement that in this context "quality is about fitness for purpose" (Guy, Powell, & Day (2004))
[5] Funk & Schmunk (2015)
[6] DCMI Usage Board (2012)
[7] Dropbox Inc. (2016g)

While well done cloud services offer usability and comfort, data security and privacy aspects have to be taken into account, especially for research projects. So it is important to have the data hosted in trusted environments, like the own computing center or within a research infrastructure like DARIAH-DE[8].

Another missing feature in existing file synchronization services useful for managing research data is a possibility to attach standardized metadata to files. Normally there are only options to add comments to files or folders.

The subject of this master thesis is to combine these two technologies, the DARIAH-DE Publikator for the metadata management for the research data and a file synchronization service. This allows to have data residing on the researchers devices and synchronized to a server known to the Publikator.

The software developed for this thesis is installed on an own server and is available for testing.[9] User documentation explaining how to use the software has also been written.[10]

1.1 Use Cases

The following three use cases shall illustrate the benefits of the software developed within this thesis. Actors are Alice and Bob, two scholarly researchers in the humanities.

1. Bob has collected a large amount of material on his hard disk for his research project, which is nicely organized in folders. Now in the preparation of the final publication of his work, he seeks a way to also publish the research data he bases his publication upon. Using the DARIAH-DE file synchronization service allows him to take the folder structure on his hard disk, view the contained files in the DARIAH-DE Publikator and add some Dublin Core metadata to his data objects describing the content and the context of these files. Finally he hits the publish button to have his data together with the metadata published within the DARIAH-DE repository. The folder structure on his hard disk is reflected as subcollections. Persistent identifiers (PIDs) are attached to each collection, subcollection and file. So he is able to reference the data he based his publication on in a stable manner from within his paper.
2. Alice starts a research project, for which she needs to travel to Vietnam to collect material from archives and to take photos or videos from interviews with contemporary witnesses. Using the DARIAH-DE file synchronization service allows her to synchronize her collected data automatically every time she is able to connect to the Internet. She adds metadata to her files early in the research process, during or shortly after the data is created, so building a solid foundation for a later publication. The DARIAH-DE Publikator supports her by extracting embedded metadata, as GPS coordinates and timestamps, from photos and videos. So even without deeper annotation there is already some context added to the data.

   After longer periods of being off-line she just connects her devices (a laptop, a tablet PC and a mobile phone) to the sometimes unstable network connection. While she sleeps her devices

[8] DARIAH-DE (2016d)
[9] Veentjer (2016b). https://portal.sftest.de.dariah.eu/
[10] Veentjer (2016c). https://sftest.de.dariah.eu/docs/

synchronize the data with the synchronization server in the background. The local file synchronization client knows how to continue uploading the data whenever network access is available. So before Alice continues her travel the next day her data is safe on the server.

3. Alice and Bob work together on a paper, which incorporates some of the data Alice collects in Vietnam. Being invited to Alice's file synchronization group Bob is able to already work with the data while Alice is still traveling. The DC-metadata annotations Alice leaves on the files help Bob to put the data in the right context. He copies some of the data items in new subcollections which they publish to the DARIAH-DE repository, to reference these items in their paper.

While use case 1. focuses on the publication and data publication at the end of the research process, use case 2. shows the advantages of preparing the data early in the research process. It also shows how data synchronization is used as a backup and collection management solution. Also it highlights the benefit of embedded metadata extraction. Use case 3. deals with the benefits of collaborative work with the help of data synchronization and the DARIAH-DE Publikator.

Note: Not the whole toolchain to fulfill all the requirements for the use cases is developed within this thesis. So the connection of a file synchronization tool and the Publikator with the group management of the DARIAH-AAI, which would be needed for use case three to work, is out of scope. But the foundation to offer such services is laid within the development of this thesis.

1.2 Thesis Organization

After giving some background information about the context of the thesis, some file synchronization services are evaluated for being used within this thesis. Afterwards the application design is discussed, and later on some implementation details are highlighted. The thesis concludes with an evaluation of the use cases and describes some possible future works.

Code blocks inlined with the text are accentuated like `console.log('hello world');`.

2 Background Information

2.1 DARIAH-DE

DARIAH-DE started in 2011 and is the german partner in the european DARIAH project network.[11] Its goal is to support "digitally enabled research and teaching in the arts and humanities"[12]. To reach this goal, different services are developed or integrated in the DARIAH-DE infrastructure. Integration in this

[11] DARIAH-DE (2016a)
[12] DARIAH-DE (2016h), p. 2

case basically means to connect the services to the DARIAH AAI[13]. Integration could also mean to make use of more DARIAH services, like the PID service[14], the generic search[15] or the collection registry[16].

> DARIAH-DE also offers the possibility to integrate own tools or services within the infrastructure. Tools for integration have to meet three core criteria:[17]
>
> 1. The service or tool has to be relevant for research in the arts and humanities and connected research disciplines.
> 2. The service or tool should already use resources from the DARIAH core infrastructure (like collection- and schema registry, metadata standards, AAI, bit preservation, PID service etc.), or the connection should be planned. There should be a sustainable strategy for implementing data collections within the DARIAH-DE infrastructure.
> 3. All existing research data must be licensed as open access, the source code of the tools and services must be under an open source license. This should allow interoperability of research data and enable interdisciplinary reuse.

2.2 Target Group

> "DARIAH-DE supports digitally-enabled research and teaching in the arts and humanities."[18]

DARIAH-DE and TextGrid published a report which contains an analysis of the target group for these Digital Humanities infrastructure projects.[19] Also, DARIAH-DE published a report on the requirements of the target group.[20]

> "The users that DARIAH-DE targets are scholars from Arts and Humanities disciplines, those working in research projects (including joint collaborative research projects) and/or in research institutions."[21]

A survey about the use of tools in the Digital Humanities was done for report 1.2.1. In chapter 5.3.13 it summarizes the following requirements of the target group, what the tools should offer:[22]

- **Integration:** Programs should integrate in the workflow the researcher is used to.
- **Transferability of Data:** Data should not be bound to one system, but easily transferable into other programs to continue work there.

[13] Authorization and Authentication Infrastructure - see chapter 2.3
[14] DARIAH-DE (2016o)
[15] DARIAH-DE (2016l)
[16] DARIAH-DE (2016b)
[17] translation of DARIAH-DE (2016n)
[18] DARIAH-DE (2016a)
[19] Göbel et al. (2015)
[20] Stiller et al. (2015)
[21] Romanello, Stiller, & Thoden (2015), p.8 citing Stiller et al. (2015)
[22] Stiller et al. (2015), p.37f.

- **Availability:** The data should be available for all participants of the project any time in any place.
- **Platform Independence:** Software not available on some operating systems could complicate the workflow.

2.3 DARIAH AAI

The Authorization and Authentication Infrastructure (AAI) for DARIAH is responsible for authorizing users to the service providers. It allows central user management, single sign-on and rights delegation.[23]

The DARIAH AAI is based on the SAML[24] standard by OASIS.[25] The SAML implementation in use is Shibboleth[26] to implement single sign on (SSO). Shibboleth allows a federation where users belonging to participating institutions can get access to Shibboleth protected services, like most DARIAH-DE services. Institutions participating in the Shibboleth Federation run an identity provider (IDP), a server which is connected to the user management of the institution, and authenticates users as institution members towards service providers. A service provider (SP) can get information about users from the IDP in a secure way. The service provider is a Shibboleth secured service, like the DARIAH-DE storage.

DARIAH-DE is a member of the DFN-AAI[27], so all members of participating institutions are able to authenticate on DARIAH-DE services. DARIAH-DE also runs an own IDP, so DARIAH-DE services can be accessed with an institutional login, or with a DARIAH-DE account.[28]

2.4 DARIAH-DE Repository and the Publikator

The DARIAH-DE Repository is a composition of different services, fig. 1 shows how these interact.

DARIAH-DE offers two facilities for research data storage, the "own storage" and the DARIAH-DE repository[29] (public storage). While both utilize a REST interface based on the DARIAH-DE storage API[30] the own storage is meant for temporary storage of files, while the repository is for long term preservation of research data, adding PIDs and valuable metadata to stored items. To deposit data into the DARIAH-DE repository there is the DARIAH-DE Publikator, which is available in the DARIAH-DE portal[31].

[23] DARIAH-DE (2016e)
[24] OASIS (2016). http://www.oasis-open.org/committees/security
[25] DARIAH-DE (2016f). https://de.dariah.eu/aai
[26] Shibboleth (2016). https://shibboleth.net/
[27] Deutsches Forschungsnetz (DFN) (2016). https://www.aai.dfn.de/
[28] all information from DARIAH-DE (2016f) and DARIAH-DE (2016e)
[29] Funk & Schmunk (2015)
[30] Funk et al. (2012)
[31] DARIAH-DE (2016c). https://de.dariah.eu/publish

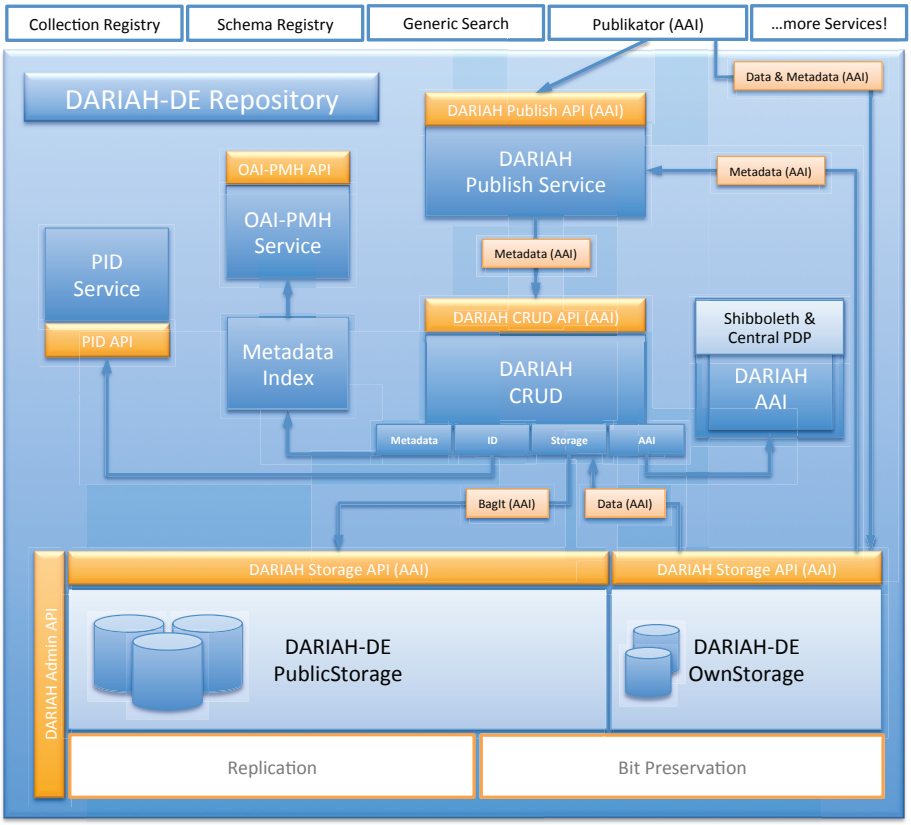

Figure 1: DARIAH-DE Repository

The DARIAH-DE portal is running with Liferay[32]. "Liferay Portal" is a content management system (CMS) implemented in Java. Own extensions to the CMS can be developed as portlets, whose structure is defined in a Java specification[33]. The Publikator is implemented as a portlet, which is integrated in the DARIAH-DE Liferay Portal.

The Publikator offers the possibility to add Dublin Core (DC) metadata terms[34] to data files to be deposited into the DARIAH-DE Repository.[35] The metadata is attached to collections and single files, which are uploaded with the web browser and stored in the DARIAH-DE own storage. A collection is a RDF[36] file in turtle[37] format, which contains metadata for the collection itself, links from the collection to each included file (its storage ID / storage location), and metadata for each of these files. The collection itself is stored in the own storage, referenced by its DARIAH-DE storage ID. This storage ID is connected to the Liferay user data, so an authenticated user can continue working on collections any time.

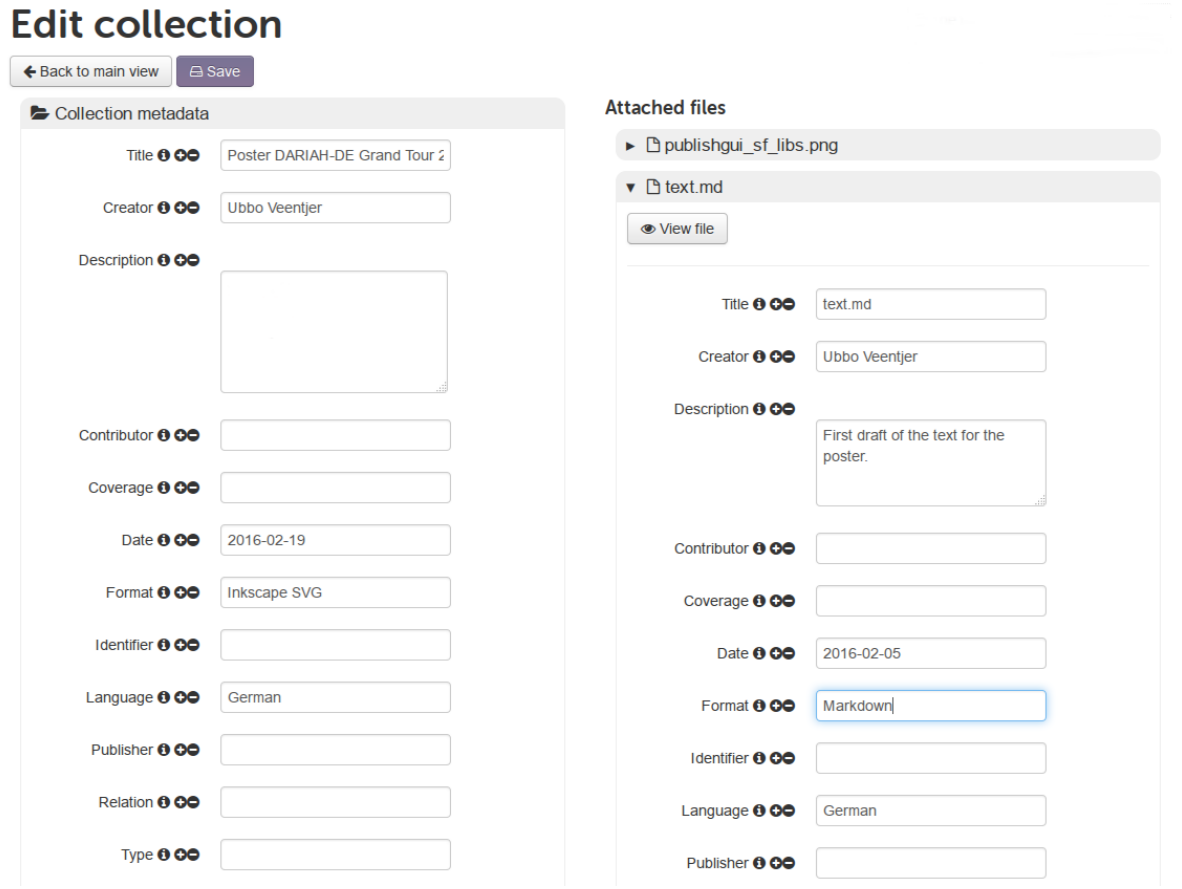

Figure 2: Screenshot of the DARIAH-DE Publikator in February 2016

[32] Liferay Inc. (2016). http://liferay.com
[33] JCP (2003). https://jcp.org/en/jsr/detail?id=168
[34] DCMI Usage Board (2012). http://dublincore.org/documents/2012/06/14/dcmi-terms/
[35] Funk & Schmunk (2015), p.12
[36] Schreiber & Raimond (2014)
[37] Becket, Berners-Lee, Prud'hommeaux, & Carothers (2014)

Collections with their data and metadata can be published to the DARIAH-DE repository any time when the researcher is ready. At publishing, a check for required metadata is done before finally depositing to the repository. This is done by the *DARIAH Publish Service* (DH-publish). Persistent identifiers utilizing the EPIC PID System[38] are attached to the collection and each published item for stable citability of the data.[39]

2.4.1 Data Model

The data model for the Publikator and the DARIAH-DE repository is written in RDF-Schema[40]. It defines a dariah:DataObject, which is described by Dublin Core metadata terms[41]. A dariah:Collection is a subclass of an dariah:DatabObject, thus sharing its properties. Furthermore a dariah:Collection is able to aggregate dariah:DataObjects.[42]

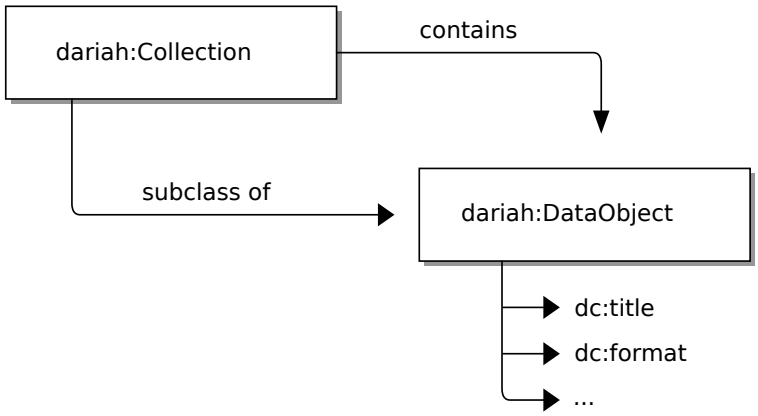

Figure 3: Data model of the Publikator

[38] DARIAH-DE (2016p)
[39] Funk & Schmunk (2015), p.8
[40] Brickley & Guha (2014)
[41] DCMI Usage Board (2012)
[42] DARIAH-DE (2016k). http://de.dariah.eu/rdf/dataobjects/terms/

The following source listing shows how the RDF metadata (serialized in turtle) looks for the collection from fig. 2:

```
@prefix dcterms:       <http://purl.org/dc/terms/> .
@prefix dc:            <http://purl.org/dc/elements/1.1/> .
@prefix rdf:           <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs:          <http://www.w3.org/2000/01/rdf-schema#> .
@prefix dariah:        <http://de.dariah.eu/rdf/dataobjects/terms/> .
@prefix dariahstorage: <https://de.dariah.eu/storage/> .

dariahstorage:400751  a  dariah:Collection;
        dc:creator       "Ubbo Veentjer";
        dc:date          "2016-02-19";
        dc:format        "Inkscape SVG";
        dc:language      "German";
        dc:title         "Poster DARIAH-DE Grand Tour 2016";
        dcterms:hasPart  dariahstorage:400752, dariahstorage:400753.

dariahstorage:400752
        a          dariah:DataObject;
        dc:format  "image/png";
        dc:title   "publishgui_sf_libs.png".

dariahstorage:400753
        a               dariah:DataObject;
        dc:creator      "Ubbo Veentjer";
        dc:date         "2016-02-05";
        dc:description  "First draft of the text for the poster";
        dc:format       "Markdown";
        dc:language     "German";
        dc:title        "text.md".
```

2.5 File Synchronization

The idea of synchronizing files on different computers is rather old, as the implementation of rsync shows, which had its initial release in 1996.[43] The problem was rethought, as mobile devices came up, and end-users started to own more than one device.[44] At least since the release of Dropbox in 2008[45] the idea of using the cloud for file storage and to synchronize with different devices and share with different people became popular and reached a larger audience. The usability and simplicity of

[43] Tridgell (1996)

[44] Balasubramaniam & Pierce (1998), p. 1

[45] Wikipedia (2016b). https://en.wikipedia.org/wiki/Dropbox_%28service%29

Dropbox may have played an important role in its success.[46] As of now there are many commercial and non-commercial solutions and offerings for file synchronization available, often in combination with cloud storage.[47]

2.5.1 Technical Background

File synchronization tools try to provide users with the same files on different devices, which is not only simple replication. Files change on the different devices, and the real problem lies in conflicts, when files change on two devices, and there is no true "latest revision", therefore Balasubramaniam and Pierce noted in 1998:

> The overall goal of a file synchronizer is very easy to state: it must *detect conflicting updates* and *propagate non-conflicting updates*.[48]

How a file synchronizer behaves on conflicts, and how it propagates non-solvable problems to the user, is still a key distinction feature of existing implementations, and something which needs to be investigated comparing different file synchronization products.

After this very basic (and most problematic) requirement, there are more distinction features, which can offer better usability and performance of the implementation, like deduplication and delta synchronization support.

Basic **deduplication** means that the software tries to check if a file is already existing on the server before uploading it. This can be done with checksums utilizing hash function, so-called hash sums. With a hash sum it is possible to create a unique fingerprint of a file based on its content. So if the hash sum of a file on the client matches the hash sum of the file on a server, the file is not uploaded again. This can feel like a very fast synchronization of big files, as for example an mp3 file or a whole movie can be synchronized in seconds, if already existing on the server, e.g. from other users.

Deduplication can also happen on **block level**, which means that bigger files are split into so-called blocks of a smaller size. These blocks also get a unique hash sum representing the content of the data. This adds the possibility for **delta synchronization**. Delta synchronization tries to find only the changed blocks of a file, and only synchronize changed blocks, which can lead to better performance and less bandwidth usage.[49]

However, delta- or block level synchronization can only play a beneficial role in a limited bunch of use cases. A use case where this technique can save bandwidth would be large media files like videos. But these are normally compressed, and in the moment the data is compressed, not only some blocks but the whole file changes on updates.[50]

[46] Dunn (2008)
[47] Wikipedia (2016a). https://en.wikipedia.org/wiki/Comparison_of_file_synchronization_software
[48] Balasubramaniam & Pierce (1998), p.1
[49] see Heckel (2012), p.11ff
[50] see Syncplicity Blog (2009)

Also, deduplication has some drawbacks, Dropbox for example disabled it, as users found out how to use Dropbox's deduplication feature for file sharing. Basically they exchanged hash sums for e.g. movies, and tricked the Dropbox client into believing that a file with a specific hash has been added to the users account, so the file of a movie appeared in the user's Dropbox account.[51]

For the use cases of this thesis the disadvantages of deduplication can be ignored, as the chance of possible illegal use or data theft with the help of deduplication are ignorable in a controlled environment like DARIAH-DE. How beneficial delta synchronization will be depends on the data stored by the target group.

To sum it up, there is no real need for the synchronization tool to be used within this thesis to provide delta synchronization and block level deduplication, as the performance benefits should be ignorable. However, it could be beneficial to know which tools allow using these techniques for future research projects. There could be use cases in the digital humanities where working with large files is part of a project. TEI files as an example could become large, if they are deeply annotated. Also, some RDF data files are very space demanding, as e.g. the GND data dump of the DNB has an uncompressed size of about 21GB.[52]. If a project deals with editing this kind of files, either with a tool or with a XML editor, and it is known that with every saving only a smart portion of the large file changes, the synchronization speed difference between a synchronization tool with delta deduplication and one without may be in the magnitude of hours vs. seconds, also depending on the speed of the internet connection.

2.6 Research Data Lifecycle

There are numerous definitions of "research data" and the "research data lifecycle" available. As this thesis is developed in the context of the DARIAH-DE project, a closer look at the ground work done in this project will be taken. The DARIAH-DE working paper 11 - "Diskussion und Definition eines Research Data LifeCycle für die digitalen Geisteswissenschaften"[53] discusses the definition of a research data lifecycle for the digital enhanced Arts and Humanities, where in this case the digitally working research disciplines in the Humanities, not only the so called "Digital Humanities" are targeted. In this working paper some definitions are proposed, which are adopted for this thesis. Some definitions which are important for this thesis will be summarized and translated in this chapter.

First the term "data" will be examined. A single digital object or a collection of many of them could be understood as data. Puhl et al. (2015) reference the definition of Funk (2010), who differentiates between physical, logical and conceptional objects. The physical object is the information in its "raw" form, how it is written to its storage medium. Depending on the type of the medium this could be magnetically (hard disk, floppy disk) or optically readable information (CD-ROM, DVD). The physical object consists of this raw sequence of "0" and "1", the bits which form the bitstream.[54] The logical object is the information which could be read from this bitstream, which is presented as a file by the operating system. As a file has a file format, a program can be chosen which can present this file to the

[51] see Wikipedia (2016c)
[52] Deutsche Nationalbibliothek (DNB) (2016)
[53] Puhl et al. (2015)
[54] see Funk (2010), p. 140 f.

user, for example an image viewer for viewing an image file. The conceptual object describes the idea of how the logical object was intended to be used on creation. This could for example include macros in Excel sheets which offer functionality beyond the raw tabular data. For the conceptional object to be preserved additional metadata has to be added to the logical object such as the software, its version and the operating system used.[55]

The information to describe the conceptional object should be ingested into the repository, which are the logical object and metadata to preserve its context. The file synchronization software will only work with the logical object. The Publikator and the DH-publish service will deal with the steps necessary to preserve the conceptional object, such as extracting technical metadata.[56]

To define the term "Research data" Puhl et al. (2015) state:

> Unter digitalen geistes- und kulturwissenschaftlichen Forschungsdaten werden innerhalb von DARIAH-DE all jene Quellen/Materialien und Ergebnisse verstanden, die im Kontext einer geistes und kulturwissenschaftlichen Forschungsfrage gesammelt, erzeugt, beschrieben und/oder ausgewertet werden und in digitaler Form zum Zwecke der Archivierung, Zitierbarkeit und zur weiteren Verarbeitung aufbewahrt werden.[57]

This sentence can be translated to

> In DARIAH-DE digital research data for the Arts and Humanities is understood as the sources, materials and results that are based on a research question, which are collected, generated, described and/or evaluated and stored in machine-readable form for analyzing, archiving purposes, citability, and for further processing.

Within this thesis the term "research data" should reference the data collected in the research process. For this data to be useful for further reuse, it needs to have metadata to be findable by other researchers in a later state.

Chapter 4.1.3 of the working paper discusses the difference between data and metadata. It is stated that

> "Metadata is often called data about data or information about information."[58]

There are formats where the data object can be described as metadata, as with TEI[59], where the file often contains the structural information (the metadata) about an original text.

Puhl et al. (2015) also mention the existence of descriptive metadata embedded in files like MPEG or Open Office files. For this thesis the approach is taken to make use of embedded metadata to automatically extract some metadata from the synchronized files for the Publikator. Nonetheless this will only be an addition to the metadata which is managed by the Publikator. For this thesis metadata

[55] see Funk (2010), p. 142 ff.
[56] see DARIAH-DE (2016j). https://wiki.de.dariah.eu/display/publicde/Das+DARIAH-DE+Repositorium
[57] Puhl et al. (2015), p. 14
[58] NISO (2004)
[59] Text Encoding Initiative (TEI) (2016)

should be understood as the data describing a digital object provided by the researcher, which is stored separately from the object.

To put the term "research data lifecycle" into context the graph from the Data Documentation Alliance is used here[60] (fig. 4), as also the DARIAH-DE graph explaining the research data lifecycle is based on this one.[61]

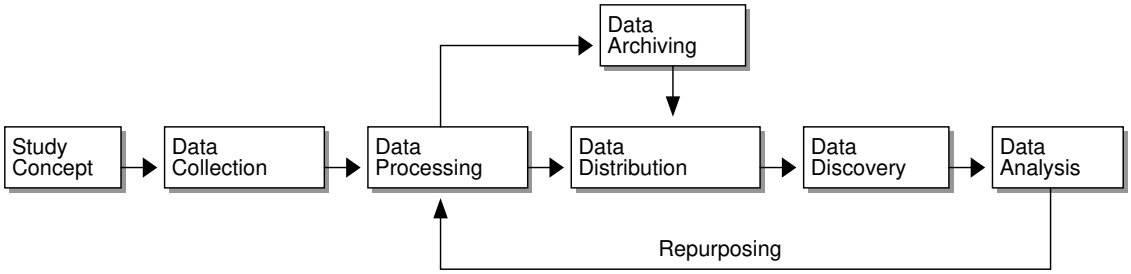

Figure 4: DDI Research Data Lifecycle

As the Publikator is a tool which is used before data archiving, it is located on the arrow between "Data Processing" and "Data Archiving". If a file synchronization tool is integrated into this workflow, it is located at the box "Data Collection", which is earlier in the process. The connection of a file synchronization service with the Publikator can be seen as a shortcut between "Data Collection" and "Data Archiving" (see fig. 5). This reflects a requirement of the researchers, which would like to have technical support to add metadata for publication to their research data already in the data collection phase. The benefit of the connection of a file synchronization software to the Publikator is the ability to cover a wider part of the research process.

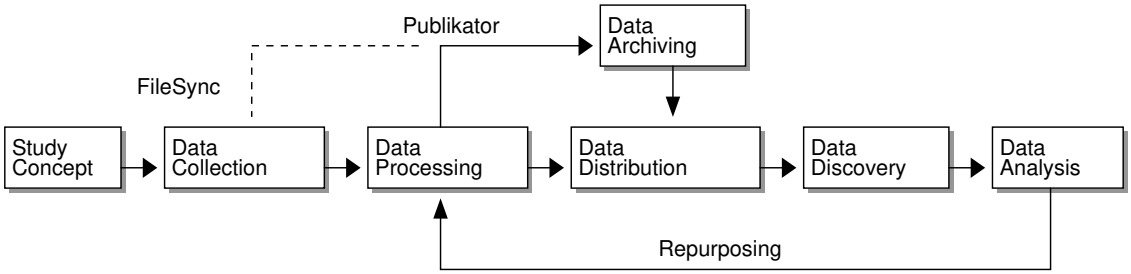

Figure 5: Research Data Lifecycle with the Publikator and file synchronization (FileSync)

[60] DDI Structural Reform Group (2004), p.8

[61] DARIAH-DE (2016i). https://de.dariah.eu/research-data-lifecycle

The tools developed within DARIAH-DE are covered by these theoretic ground works, such as done by Puhl et al. (2015). As the software built within this thesis is integrating with these tools, it is important to have a common understanding of these basics. In this thesis the terms *data*, *metadata*, *research data* and *research data lifecycle* should be understood as defined here, if not stated otherwise.

2.7 Legal Aspects of Cloud Storage Solutions in Research Projects

Scholarly researchers like to use cloud based storage solutions for backup and synchronization of data and also for collaboration in projects. While cloud based solutions offer comfort and flexibility, the juridical requirements with regard to privacy protection and data safety have to be taken into account. The officials responsible for data protection at the University of Wuppertal state for example that the usage of external cloud storage solutions like Dropbox is prohibited if the data to be stored includes private data about persons, job-related or confidential material.[62]

The DFG published a paper with recommendations on the usage of cloud services. This paper recommends the usage of cloud services from institutions which are in the same judicial area as the data owners and have no commercial interests, as the requirements for data security and privacy protection are easier to fulfill this way.[63] The paper concludes with a suggestion for project proposals, which include cloud services, to first check if there are offerings from local partners like university computing centers.[64]

DARIAH-DE offers storage solutions where the data is stored in national research computing centers like the GWDG[65].

3 Evaluation of Software Solutions for File Synchronization Services

This chapter deals with the evaluation and comparison of file synchronization tools. First the requirements for this software are explained, afterwards the tools in this evaluation are introduced, before they are compared. Finally, a synchronization tool is chosen to build the software developed within this thesis.

3.1 Requirements

One basic requirement for software within the DARIAH project is its availability as open source[66]. It would be beneficial if the software brings the possibility to be easily integrated within the DARIAH AAI, namely this would be Shibboleth[67] and/or OAuth[68] connectability.

[62] University of Wuppertal (2016)
[63] Deutsche Forschungsgemeinschaft (DFG) (2014), chapter 5
[64] Deutsche Forschungsgemeinschaft (DFG) (2014), chapter 7
[65] Gesellschaft für wissenschaftliche Datenverarbeitung Göttingen (GWDG) (2016). https://www.gwdg.de/
[66] see chapter 2.1
[67] Shibboleth (2016). https://shibboleth.net/
[68] OAuth (2016b). https://oauth.net/

It has to be differentiated between client and server side software, as the server side software (the service) is mainly offering the low level features. Some functions like versioning are often only available in the web interface, so the usability of the user visible web interface has to be explored. The client side software is the tool the user installs on the desktop system or mobile device. Ease of installation and usability of features have to be evaluated.

From the use cases and the DARIAH context the requirement evolves that the client software should be available for all major desktop operating systems (Windows, Mac OS, Linux), and also for the major mobile platforms. According to Gartner the major mobile operating systems, as of the first quarter 2016, are Android with 84.1% market share and iOS with 14.8% market share, all other mobile operations being below 1%.[69]

Technically pleasing would be a file synchronization tool which supports delta synchronization and deduplication, but this is a rather soft requirement. More important is the usability of the software, a requirement which evolves from the target group. So the client side software should be easy to use and understandable. A very important point is the handling of synchronization conflicts, especially how the conflicts are communicated to the user. The user has to be informed of possible data loss and options for resolving the conflict have to be presented.

Information security is another important point, it should be possible to encrypt the data transfer between client and server. Features like client side encryption[70] would be a nice add-on (and should actually be possible with every synchronization software in our days), but is unfortunately interfering with the way the data access from the Publikator has to be implemented within this thesis. This feature would be of interest, if the user plans to also synchronize private data, which is not managed with the Publikator. Possibly research could be done in a separate project if there are possibilities to implement a metadata management solution in the Publikator for client side encrypted data.[71]

Also relevant is the overall performance of the tool, which includes the synchronization speed and the CPU/memory demand of the software in idle state and while synchronizing.

Of interest for developing the integration with the DARIAH-DE Publikator is the API for accessing information from the synchronization service, and how well the API is documented.

3.2 Tools in the Evaluation

Three tools were chosen for this evaluation. These are Dropbox, ownCloud and Seafile. Dropbox does not meet the basic requirement of being open source, but joins the evaluation as an example of a professional commercial offering. ownCloud and Seafile are taken into consideration because they seem to be the most advanced open source solutions for file synchronization services.

[69] Gartner, Inc. (2016). http://www.gartner.com/newsroom/id/3323017
[70] Zafer (2015). http://www.infosectoday.com/Articles/Client-Side_Encryption.htm
[71] see chapter 6.4.3

The evaluation was done in April 2016 with the following versions of the desktop clients:

- Dropbox: 3.16.1
- ownCloud: 2.1.1
- Seafile: 5.0.7

3.2.1 Dropbox

Dropbox[72] is a commercial file synchronization cloud based service operated by Dropbox Inc. It offers client software for all major desktop (Windows, Mac OS, Linux) and mobile (Android, iOS, Windows Phone) operating systems. Dropbox started its services in 2007, and got very good reviews at that time because of its good usability and simpleness:[73]

> [...] but unlike the others, it has almost no user interface. All it has is a tray icon [...][74]

Dropbox quickly gained a large user base, counting around 500 million registered accounts in 2016[75].

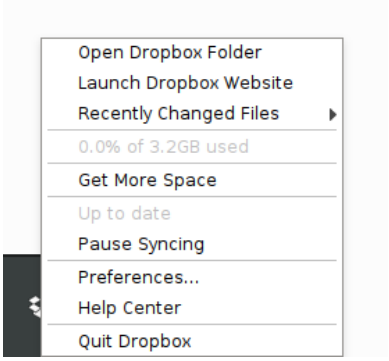

Figure 6: Dropbox desktop client - popup dialog for tray icon

3.2.1.1 Conflict Handling Synchronization conflicts in Dropbox are handled by creating conflict files on the client. These conflict files are also synchronized with the server, and visible on all other connected clients[76]. Example:

```
test.txt
test (wintermutes conflicted copy 2016-04-02).txt
```

3.2.1.2 Privacy and Security Dropbox uses SSL encryption for transfering data from the client to the server, and stores data in the Amazon cloud encrypted with AES[77]. The data is still visible for Dropbox administrators, as Dropbox itself does not provide client side encryption, but Dropbox help states that external tools may be used for client side encryption.[78] Documents leaked by Edward Snowden indicate that Dropbox was meant to be a future collaborator in the NSA PRISM program.[79]

[72] Dropbox Inc. (2016g). https://www.dropbox.com/
[73] Wikipedia (2016b). https://en.wikipedia.org/wiki/Dropbox_%28service%29#Reception
[74] Dunn (2008)
[75] Dropbox Inc. (2016b). https://blogs.dropbox.com/dropbox/2016/03/500-million/
[76] Dropbox Inc. (2016f). https://www.dropbox.com/help/36
[77] Dropbox Inc. (2016e). https://www.dropbox.com/help/27
[78] Dropbox Inc. (2016d). https://www.dropbox.com/help/28
[79] Greenwald & MacAskill (2013)

3.2.1.3 Usability

After installation Dropbox creates a folder ~/Dropbox which is synchronized with the Dropbox server. Adding files to the synchronization is a simple operation on the file system, like copying files into the Dropbox folder. Management of the files (like revision history) mainly happens in the Dropbox web interface. Synchronization of folders outside the Dropbox folder is not possible, as a workaround it is suggested to create symbolic links.[80]

3.2.1.4 API Documentation

Dropbox offers a REST API and comprehensive documentation about it.[81]

3.2.2 ownCloud

ownCloud[82] is an open source solution for file synchronization and cloud file storage. It offers some more features, like calendar and contact synchronization and collaborative document edition.[83] Native file synchronization clients exist for Windows, Linux, Mac OS and also for Android, iOS and Windows Phone. The server side software is written in PHP and installable on own servers. Available are the open source edition and a paid enterprise edition, which offers commercial support and some more advanced features.[84]

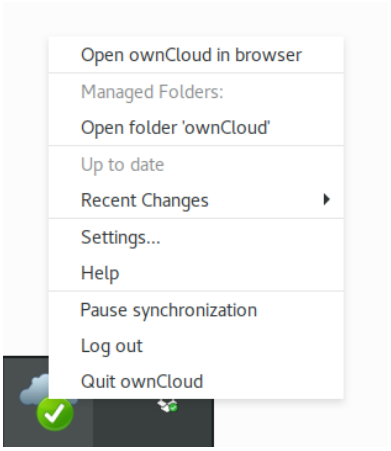

Figure 7: ownCloud desktop client - popup dialog for tray icon

The ownCloud server already offers Shibboleth support.

On the 2nd of June 2016 the former ownCloud developer Frank Karlitschek announced the fork of ownCloud into a project named Nextcloud[85]. This project is supported by a company named Nextcloud GmbH[86], which offers commercial support for Nextcloud. Reasons for the fork are a more community friendly and open source centric approach, where there is no dual licensing necessary for contributors. There will be more features, which are now only available in the paid enterprise version and not in the open source version. Karlitschek states that there still will be enterprise features. As of writing, the Nextcloud fork does not differ much from ownCloud, so the results of the evaluation should be comparable. If Nextcloud takes off it could be an interesting alternative for ownCloud and Seafile.

[80] Dropbox Inc. (2016c). https://www.dropbox.com/help/12
[81] Dropbox Inc. (2016h). https://www.dropbox.com/developers-v1/core/docs
[82] ownCloud (2016d). https://owncloud.org/
[83] ownCloud (2016c). https://owncloud.org/features/
[84] ownCloud GmbH (2016). https://owncloud.com/features/
[85] Karlitschek (2016). http://karlitschek.de/2016/06/nextcloud/
[86] Nextcloud GmbH (2016). https://nextcloud.com/

3.2.2.1 Conflict Handling ownCloud creates conflict files, which are only stored on the client, from which the later change is synced. Conflict files are not synchronized with the cloud. Example for a conflict file:

```
testsync.txt
testsync_conflict-20160402-141843.txt
```

3.2.2.2 Privacy and Security As the ownCloud server software is freely available, it is up to the user to decide which hosts to choose or to host ownCloud by theirself. This means the data can be stored by the trusted local computing center. ownCloud allows server side encryption, which means that the administrator can allow encryption of the user's file on the server.[87]. Data transfer between client and server can be secured with SSL with activating HTTPS on the web server, as communication with the server is done via HTTP.

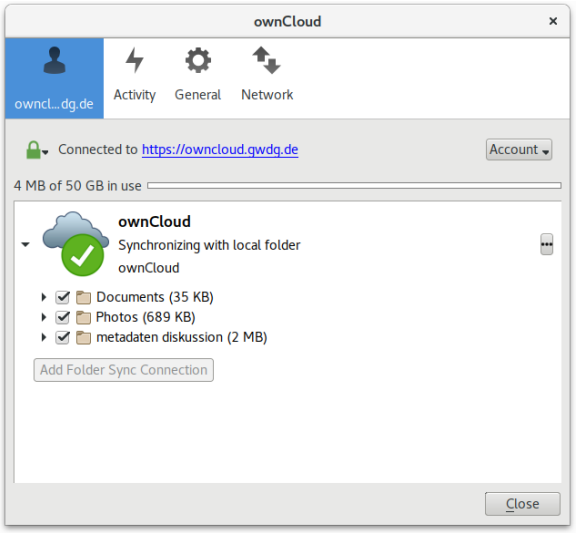

Figure 8: ownCloud desktop client - settings screen

3.2.2.3 Usability ownCloud asks for a server address to connect with after installation. Then it allows to create or choose a folder to be synchronized, with the folder ~/ownCloud being the default. It is not easily possible to add arbitrary directories to the synchronization. As with Dropbox, only folders inside the ownCloud data folder are synchronized. The usage of different ownCloud servers from the GUI is possible.

3.2.2.4 API Documentation The ownCloud REST APIs are based on the Open Collaboration Services API specification (OCS).[88]. OCS describes API calls for web communities, allowing for finding users, sharing data and collaboration.[89]

3.2.3 Seafile

The open source file synchronization server Seafile[90] offers data synchronization features with native clients for multiple platforms (Linux, Mac OS, Windows, Android, iOS). Available server side software are the Seafile community edition, and a paid professional edition. The professional edition offers

[87] ownCloud (2016a) / Schießle (2015)
[88] ownCloud (2016b)
[89] Karlitschek (2013)
[90] Seafile (2016d). https://www.seafile.com/en/home/

advanced features like high availability, an ElasticSearch backend and extended LDAP integration, while the community edition is sufficient for smaller projects. Community edition and sources are available from GitHub[91].

3.2.3.1 Conflict Handling Seafile creates conflict files on the client. These also get transferred to the server, and so they are visible to all participating parties/devices.[92] Example for such a file:

```
test2.md
test2 (SFConflict ubbo@sftest.de.dariah.eu 2016-04-02-15-47-07).md
```

3.2.3.2 Privacy and Security Other than ownCloud or Dropbox, Seafile has a client side encryption feature integrated in the desktop client, so the user can synchronize files not viewable by the server administrator.[93] This privacy feature is not yet supported by the Android client, and may get insecure if the user views encrypted libraries in the web browser, as in this case the password is stored temporarily on the server.[94] Not encrypted on server side is the metadata for the files like filename and size, which may be seen by the server administrator.[95] Communication between server and client can be secured utilizing SSL via HTTPS.

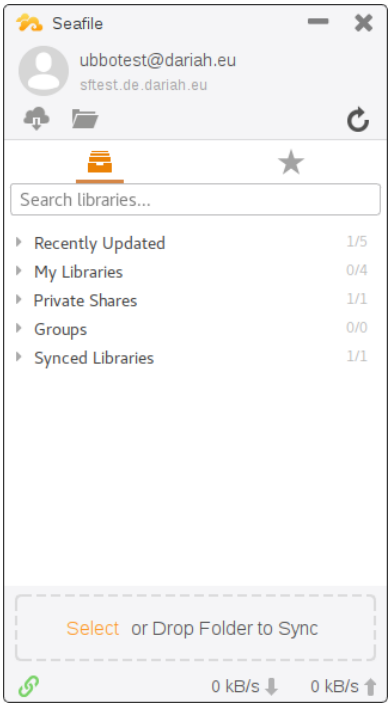

Figure 9: Seafile desktop client - user interface with drop zone for adding new libraries

3.2.3.3 Usability Seafile is based on the concept of synchronization libraries. A library is a sub folder from the ~/Seafile folder, but may also be a folder from any other place in the file system. The GUI client shows the status of these libraries. Other than Dropbox and ownCloud, in Seafile it is easily possible to add arbitrary folders to the synchronization anytime. It is just a matter of dropping a folder from the file manager onto a drop area in the GUI to have a new folder synchronized. While it is possible to connect to different Seafile servers from the client, synchronization of the same folder with different Seafile servers is not working.[96]

3.2.3.4 API Documentation Seafile provides full developers documentation of their RESTful API.[97]

[91] Seafile (2016c). https://github.com/haiwen/seafile
[92] more insight about Seafile's conflict handling offers the document Seafile (2016i)
[93] Seafile (2016g). http://manual.seafile.com/security/security_features.html
[94] Seafile (2016g). http://manual.seafile.com/security/security_features.html
[95] see ef4 (2016) and Seafile (2016g)
[96] Seafile-Forum (2015)
[97] Seafile (2016e). https://manual.seafile.com/develop/web_api.html

3.3 Feature Comparison

Product	OS	WLM	Mob	ShS	ShC	OAuth	DD	DS	VS	SSL
Dropbox	-	X	X	-	-	X	X	X	X	X
Seafile	X	X	X	X	X	-	X	X	X	X
OwnCloud	X	X	X	X	X	X	-	-	X	X

- OS: OpenSource
- WLM: Clients for Windows, Linux and Mac Os
- Mob: Clients for Mobile Platforms (Android and iOS)
- ShS: Shibboleth Support (Server)
- ShC: Shibboleth Support (Client)
- OAuth: OAuth Support
- DS: Delta Synchronization
- DD: Deduplication
- VS: Versioning Support
- SSL: Secure communication

3.4 Performance Comparison

For this thesis, no own performance comparison of Seafile and ownCloud has been done. However, there are blog posts describing performance comparisons of these tools. Christoph Dyllick-Brenzinger of ionas.com tested how ownCloud 7.0.4 compares to Seafile 4.0.1 running as a Server on a raspberry pi. In his test Seafile takes a clear lead in synchronization performance.[98] This may also be caused by the low CPU resources of the server, which would not be relevant for the use case of this thesis (server side computing power should not be a limiting factor for a setup in the DARIAH-DE environment). Jürgen Donauer from bitblokes.de did another test with Seafile 4.0 and ownCloud 7.0, finding that Seafile performs a lot better with lots of small files, while both offer comparable performance with medium and large sized files.[99]

[98] Dyllick-Brenzinger (2015)
[99] Donauer (2015)

3.5 Conclusion

As Dropbox is only evaluated for reference, the decision is whether to use ownCloud or Seafile for integration with the DARIAH-DE Publikator. Both tools do their job very well and offer a comparable feature set regarding to file synchronization.

The library concept of Seafile matches the libraries in the DARIAH-DE Publikator very well. It is beneficial if the users can just pick any folder on their hard disk to make it a Publikator library. This provides a better user experience, than having to copy folders into the ownCloud folder, or to create symbolic links.

The REST API in Seafile is very well documented, so the benefit for this thesis is a faster development process.

The possibility of easily available client side encryption did not play any role for the decision, as the Publikator will not be able to deal with encrypted libraries. Of minor importance are the performance tests or delta sync / block level deduplication, as the differences or benefits are negligible. The groupware features beyond simple file synchronization ownCloud has to offer do not matter for the use cases of this thesis.

Still the integration of Seafile in the Publikator will just be a proof of concept. The outcome should be transferable to an integration of ownCloud, if necessary.

4 Application Design

After the background information is laid out, this chapter sums up the benefits of the implementation and discusses the theoretical design of the application. Not all requirements from the use cases could be met: on the one hand the DARIAH AAI, as of writing, does not yet provide group permissions for the DARIAH-DE storage. On the other hand developing some features would exceed the time budget for the implementations within this thesis. So the focus lies on implementing the connection of the Publikator with Seafile, loosely connecting Seafile to the DARIAH AAI with a Shibboleth login, and implementing subcollections and metadata extraction within the DARIAH-DE Publikator.

The source code of the Publikator is publicly available in a git repository.[100] It is licensed under the "Apache License Version 2.0"[101] open source license. All the source code written within this thesis for the Publikator portlet can also be found in this repository.

[100] DARIAH-DE (2016m). https://projects.gwdg.de/projects/publish-gui-portlet/repository
[101] Apache Software Foundation (2004). https://www.apache.org/licenses/LICENSE-2.0

4.1 Extending the Data Model of the Publikator for Seafile Integration

One fundamental goal in the implementation of the Seafile connection with the Publikator was to be as less invasive in the existing implementations as possible. As the connection between a dariah:Collection and the contained data files is stated in the metadata of the collection (with the dcterms:hasPart relation), the data model has to be extended in a first step. The final goal is to allow publishing a Seafile managed collection with the DH-publish service.

To publish a collection with DH-publish, the storage ID of the collection metadata is handed over to DH-publish. This could for example be the ID 1234, which is located at https://de.dariah.eu/storage/1234. In this case the collection metadata DH-publish loads from this location would look like:

```
@prefix dariah: <http://de.dariah.eu/rdf/dataobjects/terms/> .
@prefix dariahstorage: <https://de.dariah.eu/storage/> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix dcterms: <http://purl.org/dc/terms/> .

dariahstorage:1234  a  dariah:Collection ;
  dc:title          "Test1";
  dcterms:hasPart   dariahstorage:2345.

dariahstorage:2345  a  dariah:DataObject ;
  dc:title          "image1.png".
```

If the abbreviated URI dariahstorage:1234 is expanded with the prefix definition in the header, this leads to the long form https://de.dariah.eu/storage/1234 which in this case is a self reference, the metadata attached here is describing this collection. Also it is said that a data file with name "image1.png" is attached to this collection, and can be retrieved from https://de.dariah.eu/storage/2345. DH-publish reads the metadata file including all the objects defined by the dcterms:hasPart relation and transfers them to the repository.

As the storage location for an object is stored in its URI, we can add objects to be retrieved from Seafile by DH-publish in the same way:

```
@prefix dariah: <http://de.dariah.eu/rdf/dataobjects/terms/> .
@prefix dariahstorage: <https://de.dariah.eu/storage/> .
@prefix seafile: <https://sftest.de.dariah.eu/files/> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix dcterms: <http://purl.org/dc/terms/> .

dariahstorage:1234  a  dariah:Collection ;
  dc:title          "Test1";
  dcterms:hasPart  seafile:2345.
```

```
seafile:2345  a  dariah:DataObject ;
  dc:title    "image1.png".
```

With a metadata file like this DH-publish would expect to retrieve the data file "image1.png" contained in the collection from https://sftest.de.dariah.eu/files/2345. This leads to the next crucial point, how to construct the URI to allow a service residing at https://sftest.de.dariah.eu/files/ handing out the referenced file from Seafile.

4.1.1 Creating Unique Identifiers for Data Objects in Seafile

To read a file with the Seafile web API, the method `download-file` is used[102]. Its usage is:

```
GET https://[server-url]/api2/repos/[repo-id]/file/?p=[/path/to/file]
```

In consequence a service handing out a file from Seafile would need to have the Seafile library ID and the complete path to the file inside this library. As the basic goal is to have a download location like https://sftest.de.dariah.eu/files/ID, and a fixed prefix for the first part of this URI, all this information has to go into the identifier part of the URI.

So the ID for a file with the path "/1.png" residing in a library with the ID "18c544c9-b1ad-474a-a9db-a283f7c63d42" is constructed like this:

```
18c544c9-b1ad-474a-a9db-a283f7c63d42:/1.png
```

As the path may contain characters which are not allowed in URLs, this is encoded in Base64[103]:

```
MThjNTQ0YzktYjFhZC00NzRhLWE5ZGItYTI4M2Y3YzYzZDQyOi8xLnBuZw
```

This is a really long identifier already, and it may be much longer for files in deeper nested path structures, hopefully compressing with deflate[104] helps:

```
M7RINjUxSbbUTTJMTNE1MTdJ1E20TEnSTTSyME4zTzYzTjExstI31CvISwcA
```

[102] Seafile (2016e). http://manual.seafile.com/develop/web_api.html#download-file
[103] Josefsson (2006)
[104] Oracle (2016)

The deflate algorithm adds two characters in this case. But some quick tests have shown that using deflate the ID gets shorter if the path is longer. If deflate really has an effect on this kind of identifiers is another question, but not further researched within this thesis.

This identifier may not look nice because of its pure length, but it is a self contained unique reference to a file within Seafile. So using this kind of ID allows the publication of files stored in Seafile with DH-publish.

NOTE: The internal ID for a file in Seafile is a hash sum for the content of the file[105] and so it changes when the file changes. This means it is not usable as a unique reference to a file in the Publikator data model.

4.1.2 Additions to the Data Model to Manage a Seafile Collection with the Publikator

To have the possibility to deal with a Seafile library inside the Publikator, some more information need to be included in the RDF metadata files. The Publikator needs to know which Seafile library is tracked, and also which path.

```
dariah:describesSeafileLibrary
          seafile:a16ea413-e7ff-4337-92d7-ecee62a24dd7;
dariah:describesSeafileLibraryPath "/".
```

Here, the identifier (a16ea413-e7ff-4337-92d7-ecee62a24dd7) is a reference to the identifier of the library in Seafile. The SeafileLibraryPath references the path in the folder structure, in this case the root, which is required for handling subdirectories.[106]

The complete metadata for a Seafile managed library containing one file may look like this:

```
@prefix dariah: <http://de.dariah.eu/rdf/dataobjects/terms/> .
@prefix dariahstorage: <https://de.dariah.eu/storage/> .
@prefix seafile: <https://sftest.de.dariah.eu/files/> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix dcterms: <http://purl.org/dc/terms/> .

dariahstorage:1234  a  dariah:Collection ;
  dariah:describesSeafileLibrary
            seafile:a16ea413-e7ff-4337-92d7-ecee62a24dd7;
  dariah:describesSeafileLibraryPath "/".
  dc:title          "Test1";
  dcterms:hasPart   seafile:rEALLYlONGiD.
```

[105] to be more precise, it seems to be a SHA1 hash of the JSON object which is transferred to Seafile, see Seafile (2016h)
[106] see chapter 5.4.1

```
seafile:rEALLYlONGiD  a  dariah:DataObject ;
  dc:title    "image1.png";
  dc:format   "image/png".
```

4.2 Architecture of Seafile and Seahub

The server side software of Seafile is split into two components, Seafile and Seahub.[107] Seafile is the service daemon, which manages all services necessary for file synchronization, while Seahub is the web interface, which also offers the web API for Seafile. Seafile is written in C, while Seahub is written in the Python web framework Django[108].

5 Implementation

This chapter describes the concrete work done on implementing the service.

5.1 Authorization Flow

The user login for the DARIAH-DE Publikator is done using Shibboleth, relying on the Liferay Shibboleth integration. Seafile also offers a Shibboleth Login. It is possible to login with Shibboleth from the Seafile desktop client, which allows any DARIAH-DE user to use the Seafile service from their desktop with their DARIAH-DE login credentials.

To access the user's libraries in Seafile from within the Publikator, the Publikator needs an authentication token from Seafile. This requires a login delegation. The Publikator backend needs to retrieve a Seafile authentication token with the Shibboleth login of the user, but the user's login credentials are not known to the Publikator. The login credentials should also not be made available to, or requested by, the Publikator.

The approach taken in this thesis incorporates some of the ideas of OAuth2[109], but is still rather basic. As of the time of writing the thesis, there is only a beta version of an OAuth2 provider in DARIAH-DE available, and it would have taken too much time to implement a real OAuth solution in Seafile. OAuth has been on the Seafile roadmap in the past, but this entry has been removed.[110] DARIAH-DE has a not yet public draft, which describes how OAuth2 should be implemented for DARIAH-DE.[111]

For this thesis it was decided to utilize HTTP redirects to transfer a Seafile authentication token from Seafile to the Publikator. This is kind of similar to Shibboleth, as it also makes heavy use of HTTP redirects for login to a service provider. The Seafile authentication token is encrypted with a secret only known

[107] Seafile (2016b)
[108] Django Software Foundation (2016). https://www.djangoproject.com/
[109] OAuth (2016a). https://oauth.net/2/
[110] Seafile (2016f)
[111] Haase, Gietz, Widmer, Funk, & Veentjer (2016)

to Seafile and the Publikator. This ensures safety of the user's Seafile data in case the token gets known to a third party.

If a user opens the Seafile view in the Publikator, it is checked whether a Seafile authentication token is already available in the user's session. If not, the Publikator redirects to the Shibboleth login location on the Seafile server[112]. As the location /shib-login/ requires Shibboleth authentication, this in turn redirects to the identity provider (IDP). As the user is already logged into the Publikator, a Shibboleth session should be available in the browser, so no further user interaction should be required. The IDP redirects back to the location for creating a Seafile authentication token for the Publikator, which was requested with the `next` parameter[113].

For generating a Seafile authentication token for the Publikator a new view was added to Seahub (the Seafile web frontend). Seahub is implemented with the web framework Django[114]. A Python class named `DariahPublishToken` was implemented in the file *views_dariah.py*. In the file *seahub/api2/urls.py* the class `DariahPublishToken` is declared to be responsible for the URL */api2/dariah-publish-token*. When this URL is accessed, Shibboleth login has already been done, and the class `DariahPublishToken` has access to a Seafile authentication token for the logged in user. This token is encrypted as a JSON Web Token (JWT)[115] with a secret password only known to Seahub and the Publikator. Now a redirect back to the Publikator happens, attaching the encrypted token as query parameter named `authToken`[116].

The Publikator decrypts this token with its secret password and gets access to the Seafile authentication token. Now the Publikator is able to do Seafile API requests with the user's credentials, such as listing the user's Seafile collections within the Publikator. As the redirects from the Publikator to the Seafile server to the IDP and back are rather fast and require no user interaction, the process of authentication token exchange is almost unnoticeable to the user.

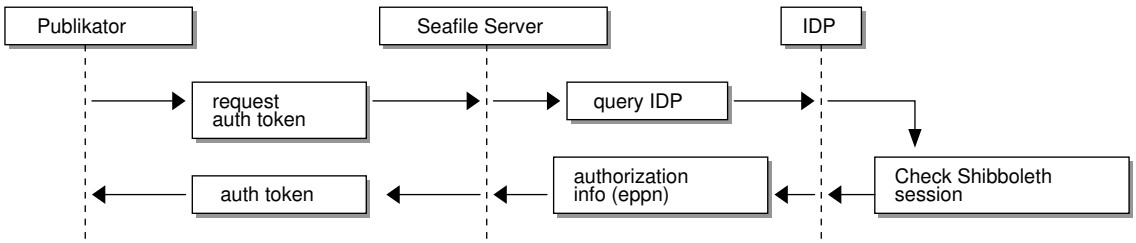

Figure 10: Auth-flow - transfering a Seafile auth token to the Publikator with HTTP redirects

[112] https://sftest.de.dariah.eu/shib-login/?next=/api2/dariah-publish-token
[113] https://sftest.de.dariah.eu/api2/dariah-publish-token
[114] Django Software Foundation (2016). https://www.djangoproject.com/
[115] Jones, Bradley, & Sakimura (2015)
[116] e.g. https://de.dariah.eu/publish/-/publishgui/seafile?authToken=eyJhbGciOiJICNg[...]

5.2 Seafile API Connection

Liferay is a Java based content management system (CMS), which allows writing portlets in Java and Java Server Pages (JSP). The DARIAH-DE Publikator is a Liferay portlet, which uses Spring[117] and JSP for the parts which are rendered server-side, also providing the REST endpoints for the client application which is written in JavaScript. The JavaScript parts of the Publikator use Ractive.js[118] for databinding, view rendering and update.

Below some lines of source code from *EditCollectionView.jsp* to illustrate how the JSP and the JavaScript parts interact:

```
1  <portlet:resourceURL
2    id="dariahStorageLoad" var="dariahStorageLoadUrl" >
3  </portlet:resourceURL>
4  <portlet:resourceURL
5    id="dariahStorageUpdate" var="dariahStorageUpdateUrl" >
6  </portlet:resourceURL>
7
8  [...]
9    <did id="dariah-collection-edit"></div>
10 [...]
11
12 <script>
13   const i18nMap = {
14     <c:forEach items="${i18n}" var="entry">
15       '${entry.key}' : '${entry.value}',
16     </c:forEach>
17   };
18
19   const edit = new DariahCollectionEdit.default({
20     i18nMap,
21     elementId: 'dariah-collection-edit',
22     schemaUrls: [
23       '/publish-gui-portlet/schema/dariahDataobjects.ttl',
24       '/publish-gui-portlet/schema/dcelements_${language}.ttl'
25     ],
26     loadFileUrl: '${dariahStorageLoadUrl}&storageId=',
27     updateFileUrl: '${dariahStorageUpdateUrl}',
28   });
29 </script>
```

Visible in this example are JSP tags, HTML tags and JavaScript code. The JSP tags (lines 1 to 6) bind the AJAX-request URLs from the Java backend to variables in the JSP. As the internationalization framework

[117] Pivotal Software, Inc. (2016). https://spring.io/
[118] Ractive.js (2016c). http://www.ractivejs.org/

from Liferay is used, all the i18n definitions are filled into an JavaScript object (lines 13 to 17), to have them usable from JavaScript. The decision which language is used is done by Liferay based on browser settings or user preferences. The JavaScript class `DariahCollectionEdit` is responsible for showing the metadada editor to the user. It is initialized with variables from the JSP (lines 19 to 28), like the internationalization object `i18nMap`, the URLs for AJAX requests, such as `${dariahStorageUpdateUrl}` and the ID of the HTML element where to render the GUI (`daria-colletion-edit`). What makes this kind of JavaScript and JSP glue particularly ugly is the ambiguity of this code, as it has to be understood well which variables are replaced by the JSP and how the resulting HTML will look like to grasp what JavaScript will be executed. For further confusion the ECMAScript 2015[119] syntax for templating also allows variables replacement with the same notation as JSP, like `${varname}`. For this reasons the place where JavaScript and JSP meet is restricted to the JSP files, and kept as small as possible.

The JavaScript parts of the metadata editor are based on the tgForms JavaScript module[120] for displaying metadata input forms generated from an RDF-schema. In the Publikator, tgForms was modified to let all rendering being done by Ractive.js, used for (turtle) data loading to provide the data model for Ractive.js. The RDF schema files for the metadata editor and the metadata which is edited by the user are serialized, transferred and stored in the turtle RDF format. tgForms uses n3.js[121] for parsing and writing the turtle data to translate between the JavaScript objects used by Ractive.js and the RDF serialization of this data.

Seafile offers a RESTful API for accessing its data structures.[122] The data is transferred in JSON[123].

A HTTP client was written to access the Seafile web API from the Publikator portlet. As high level REST client the JAX-RS client was chosen. The JAX-RS client allows to register a Jackson[124] feature, so JSON responses from Seafile can be mapped easily to POJOS (Plain Old Java Objects). Jackson allows a Java annotation to ignore unknown JSON Properties and only map the ones defined in the POJO:

```
@JsonIgnoreProperties(ignoreUnknown = true)
```

This allows to write Java classes for JSON mapping with only the JSON fields defined, which are needed by the Publikator portlet.

This kind of JSON POJO mapping also has the benefit that this POJOs could be easily addressed by the JSP. So for example a Seafile library POJO which maps the JSON field `name` and `desc` can be accessed from JSP:

```
<c:forEach items="${seafileLibraries}" var="lib">
    <li>${lib.name} - ${lib.desc}</li>
</c:forEach>
```

[119] ECMA International (2016). http://www.ecma-international.org/ecma-262/6.0/
[120] Riebl (2016). https://github.com/hriebl/tgForms/
[121] Verborgh (2016). https://github.com/RubenVerborgh/N3.js/
[122] Seafile (2016e)
[123] Bray (2014)
[124] Jackson Project (2016). https://github.com/FasterXML/jackson

A new Spring controller was implemented, namely the SeafileViewController. The Controller cares for providing the data, which is shown in the JSP file *SeafileView.jsp*, which acts as the view. If the user chooses to add a Seafile library to the Publikator, the turtle file containing the collection description has a predicate *desribesSeafileLibrary* added, with a reference to the unique ID of the Seafile library.

5.3 Views and Components Needed in Publikator Portlet

Figure 11: DARIAH-DE Publikator - Seafile disconnected

To give the user access to his/her Seafile libraries from the Publikator, some new GUI elements were implemented. A small box was added to the main Publikator view, which informs the user about the possibility to manage the metadata of Seafile libraries within the Publikator. This box shows the connection state with Seafile, if there is no Seafile authentication token stored in the user's session it shows "disconnected" (fig. 11), "connected" (fig. 12) otherwise.

Figure 12: DARIAH-DE Publikator - Seafile connected

If the user clicks on "Connect to Seafile" the Publikator retrieves an Seafile authentication token as described in chapter 5.1.

Clicking on "Manage Seafile Libraries" opens a view listing all the user's libraries in Seafile with their description, the contained files and a hint if this library is already added to the DARIAH-DE Publikator. The user may select libraries to be added to the Publikator in this view. This is done in the SeafileView, where the implementation was described before. Libraries already added to the Publikator show the note "managed with the Publikator" in the title line (fig. 13).

If a Seafile library is opened with the metadata editor, a box is shown in the top right corner which also shows the Seafile connection state, and shows a button to reload the file list from Seafile (fig. 14).

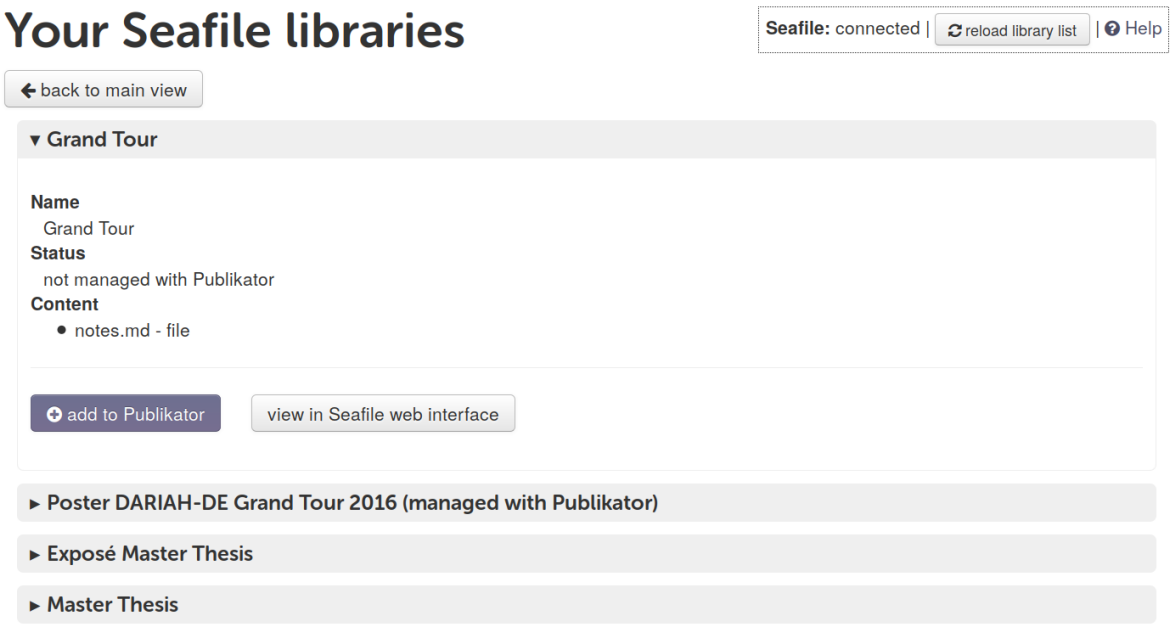

Figure 13: DARIAH-DE Publikator - Seafile library listing

All these Seafile related boxes in the Publikator contain a link to a help section, which explains how to install the Seafile desktop client, and which benefits the usage of Seafile may have for the user.[125]

To have the file list in the edit-metadata-view synchronized with a Seafile library, the Turtle-RDF needs to be modified by a component being aware of the state of a library in Seafile and the RDF-metadata from the DARIAH-DE storage. This is different from the existing Publikator collections, as the RDF-metadata for them is written and managed entirely by the JavaScript. If the user opens a DARIAH collection in the metadata editor, the collection metadata is loaded from the REST service in RDF-turtle format. The REST service analyzes the RDF loaded from the DARIAH storage. If the property dariah:describesSeafileLibrary is found, it connects to the Seafile API and adds all files from the folder which is stated in the property dariah:describesSeafileLibraryPath to the RDF. This assures that newly added files in Seafile are also visible in the Publikator. If there are removed items in the Seafile library, which still show up in the metadata, these items are automatically removed. This behavior also applies if an item is renamed or moved to a subfolder, in which case the entry for the old name will be removed, and a new item without metadata will show up. This solution is not optimal, a concept for better handling of deletion, renaming and moving is presented in chapter 6.4.1.

To access files in Seafile, the Seafile API only offers the possibility to create download links. These download links make use of one time access tokens. It is necessary to get hold of files stored in Seafile from the Publikator to create the preview link, to publish files from Seafile with DH-publish and to extract metadata embedded in files. To achieve this, a component was developed for the Publikator, which internally requests the download link for a Seafile URI from Seafile, reads the data from the download link and offers it to the Publish-GUI application for further use.

[125] Veentjer (2016c). https://sftest.de.dariah.eu/docs/

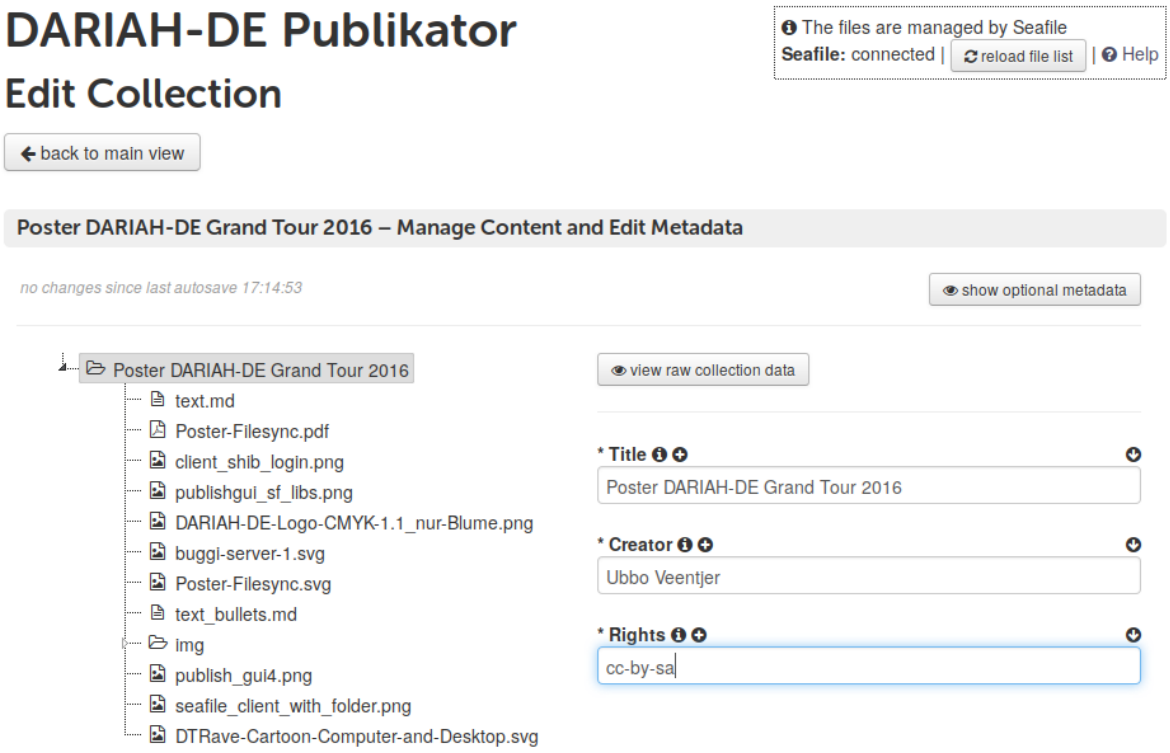

Figure 14: DARIAH-DE Publikator - Metadata editor for a library managed by Seafile

5.4 Subcollections

An essential piece to show the benefits of file synchronization connected to the Publikator is the ability of the Publikator to deal with subcollections. This has already been on the Publikator roadmap for quite a while, but had not been implemented yet. So the integration of subcollections into the Publikator was started within this thesis. Essentially, the integration reuses existing collections from the data model. The GUI decides, if an item of type collection is found, to treat it as a subcollection. Every subcollection is a new RDF metadata file with an unique storage ID, similar to the top level collection. The only references to a subcollection in the parent collection are its ID and its type, for example:

```
dariahstorage:100 a dariah:Collection ;
    dc:title "parent collection" ;
    dcterms:hasPart dariahstorage:200.

dariahstorage:200 a dariah:Collection.
```

The other information of this subcollection, such as its title and filename or further subcollection references, are stored in a separate file, in this example a file with the storage ID 200.

If the JavaScript part of the collection edit view finds a reference to a subcollection, it loads its contents and adds them to the metadata model.

Important for the implementation of subcollections is a representation of the nested collection structure which is easy to navigate by the user. It was decided to represent this structure in a tree view, as the concept of browsing a hierarchy of files and collections should already be known to the user from file system browsers like Windows Explorer. For implementing the treeview, the Javascript library jsTree[126] was chosen, as it offers plugins which add functionality to the treeview like drag and drop or popup menus. These plugins could be useful if the functionality of the tree view needs to be extended in the future. To connect the jsTree data model and the Ractive.js data model, some glue code was needed. On the one hand, Ractive.js needs to be notified if the tree view selection changes to show the according metadata entry and hide the other. On the other hand, the tree view needs to know if the data model changes, like a new title for an object or if new leafs like subcollections or files are added. For the latter, the Ractive.js concept of observers[127] and key paths[128] came in handy, which allows to observe the data model. If data changes on a specific key path the jsTree is changed in this case. To notify Ractive.js of selection changes an event handler for jsTree was used.

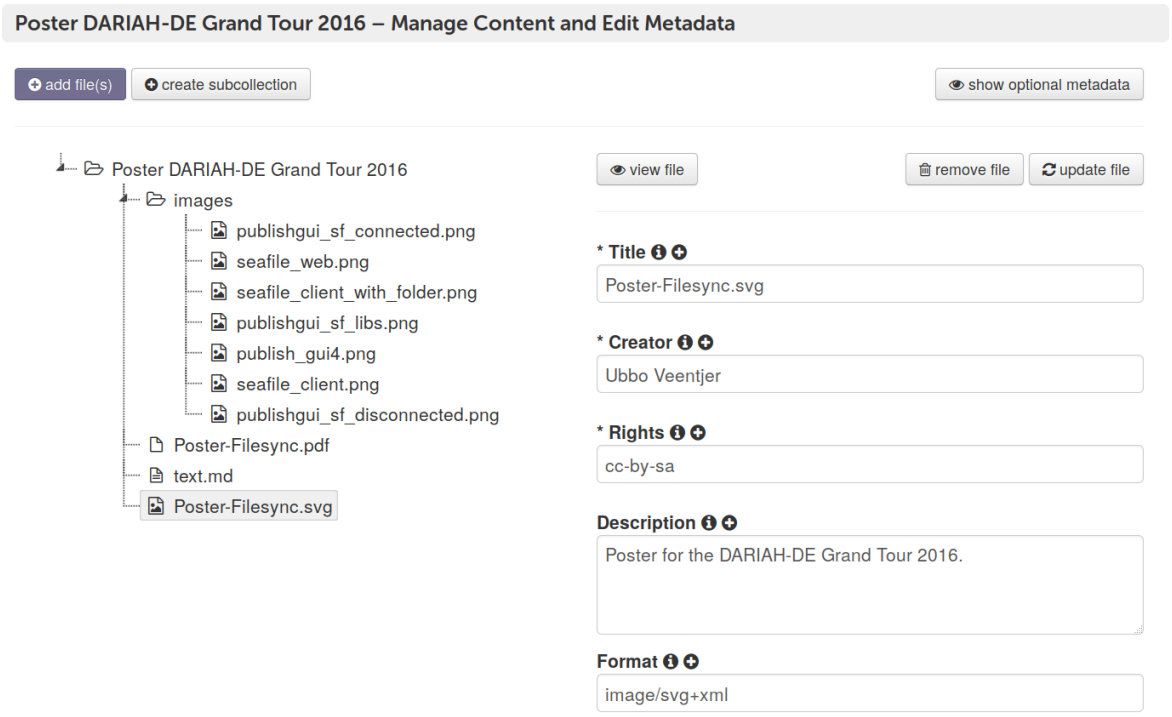

Figure 15: DARIAH-DE Publikator - Treeview for subcollections

[126] jsTree (2006). https://www.jstree.com/
[127] Ractive.js (2016b). http://docs.ractivejs.org/latest/observers
[128] Ractive.js (2016a). http://docs.ractivejs.org/latest/keypaths

5.4.1 Seafile Specific Additions for Subcollection Handling

Subdirectories on the users hard disk, synchronized with Seafile, are mapped to subcollections in the Publikator. This is reflected in the RDF metadata by an entry like:

```
dariah:describesSeafileLibraryPath "/subdir/"
```

The root collection just has the "/" set as library path. Additionally, the property dariah:describesSeafileLibrary is set for every collection, so the metadata for a subcollection may be represented like this:

```
dariahstorage:377555 a dariah:Collection ;
    dariah:describesSeafileLibrary
              seafile:a16ea413-e7ff-4337-92d7-ecee62a24dd7 ;
    dariah:describesSeafileLibraryPath "/subdir/" ;
    dc:title  "subdir".
```

So the REST service which handles the RDF data loading[129] knows which subfolder of a Seafile library needs to be checked for changes.

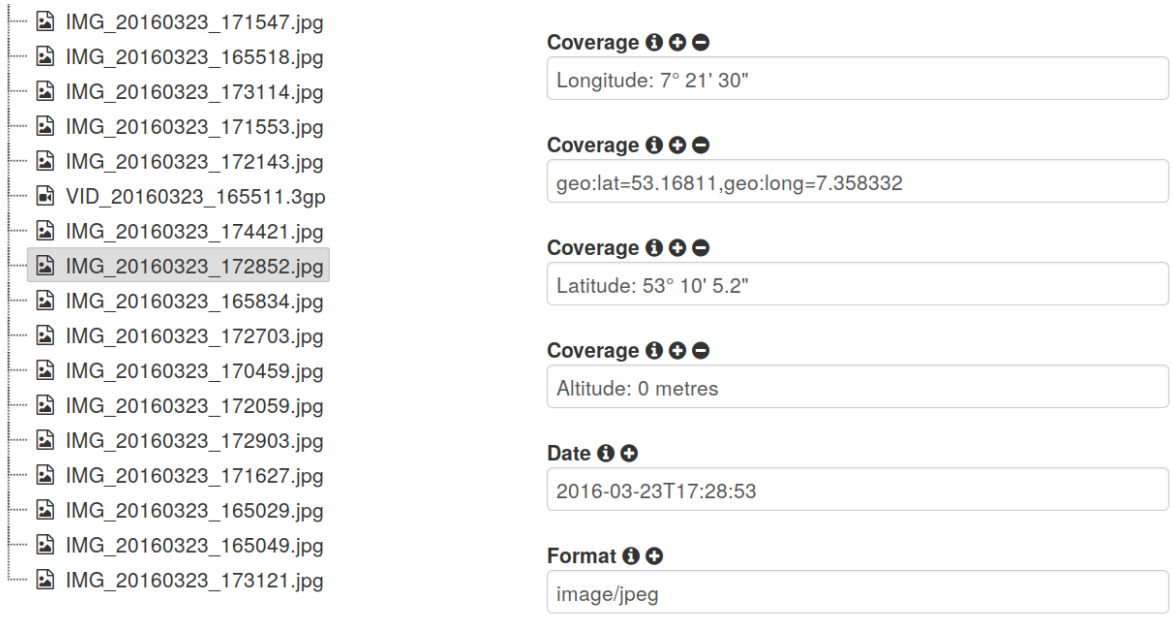

Figure 16: DARIAH-DE Publikator - extracted image metadata

5.5 Automated Metadata Extraction

As use case 1. describes the benefits of automated metadata extraction, the implementation was added to the Publikator within this thesis (fig. 16). The implementation uses Apache Tika[130] for extracting

[129] see chapter 5.3
[130] Apache Tika (2016a)

metadata of files referenced within a collection. On adding a file in the Publikator the file is parsed by a Tika autoparser, trying to extract information like dc:title, dc:creator and GPS data from the file. If metatada is extracted, it is added with the corresponding Dublin Core metadata terms to the data object in the Publikator. Tika provides metadata extraction for various file formats out of the box.[131] Office documents and PDF files often contain metadata of the title, the creator and the date as metadata, while photos may contain GPS information and a timestamp, which can be extracted by Tika.

5.6 Publish to the DARIAH-DE Repository

To allow the publication of data from Seafile, a RESTful webservice named seafile-dhpublish-connector was written.[132] It uses the JAX-RS implementation of CXF and is setup with Spring. It is mainly used to provide files managed by Seafile in a downloadable form for the DH-publish service. The service has the URL `https://sftest.de.dariah.eu/files/` and the following main method annotation:

```
@GET
@Path("/{seafileId}")
public Response getFile(
        @PathParam("seafileId") String seafileId,
        @HeaderParam("Authorization") String authHeader,
        @QueryParam("token") String token
        )
```

So it takes the parameter seafileId from the URL, and also an authorization token, which may either be in a query parameter `token` or be transferred in the authorization header of the HTTP call. If the Seafile access token is send in the authorization header, it is written in the following form:

```
Authorization: bearer THIsIsAsECRETtOKEN
```

This is similar to the way OAuth2 access tokens are transferred in between services utilizing the PDP in DARIAH-DE.[133]

If a user clicks on "publish" in the publish GUI, the Seafile authorization token is handed over to DH-publish. DH-publish had to be modified, so that it knows that this token has to be used if a file from `https://sftest.de.dariah.eu/files/` is retrieved for publication. This implementation on DH-publish side was done by its maintainer Stefan E. Funk.

When DH-publish finds a reference to a data file like `seafile:rEALLYlONGiD`, it requests this file from `https://sftest.de.dariah.eu/files/rEALLYlONGiD` and adds the authorization token to the header. The seafile-dhpublish-connector service can then gather the information in which Seafile

[131] Apache Tika (2016b)
[132] Veentjer (2016a). https://projects.gwdg.de/projects/seafile-dhpublish-connector/repository
[133] see chapter 2.3 and Haase et al. (2016)

library and which path the file resides from the ID.[134] The service decodes the authentication token with its secret key and then gets the download link from the Seafile REST API to stream the file to DH-publish.

The seafile-dhpublish-connector is the final missing piece which had to be implemented to have the complete possibility to manage and publish data from Seafile to the DARIAH-DE repository. All the other functionality like getting PIDs for the data or registering the publication with the collection registry just works the same way as with data residing in the DARIAH-DE storage.

6 Validation of the Implementation

In this chapter the use cases from the beginning are revisited, to check if all requirements from the use cases are matched by the implementation done within this thesis.

6.1 Use Case 1 - Data Publication of Existing Research Data

Bob wants to do a data publication of files located on his hard disk. Using the Seafile integration for the DARIAH-DE Publikator eases the process for him. After installing the Seafile client software, he adds the server address for the DARIAH-DE Seafile service. Then he logs in with his DARAH-DE Shibboleth credentials. Now he is ready to take a folder from his hard disk and drop it onto the drop zone of his Seafile desktop client. He is asked for a library name, and his files and subfolders get synchronized with the Seafile server.

After synchronization is finished, Bob opens the DARIAH-DE Publikator and is able to select his newly created Seafile library for metadata annotation. He finds all his files and the directory structure organized in subcollections. Some descriptive metadata, like dc:creator and dc:title, is already extracted from the files (e.g. PDF documents). He is now able to change these metadata fields or add even more metadata. He is always able to look into the source documents from the Publikator hitting the "view file" button. Doing this, Bob finds a spelling mistake in one of his files. He opens the document with his spreadsheet program, and corrects the mistake. Seafile takes care of synchronizing the new version of the file, so it reaches the Publikator without further interaction by Bob necessary.

Finally, he publishes his data into the DARIAH-DE repository using the "publish collection" button from the Publikator. After publication is finished, he can use the PIDs to reference his data from his publication.

So the implementation done within this thesis helps Bob to quicker reach his goal of publishing his research data. He benefits from tools like metadata extraction, and synchronization of changes in his documents.

[134] see chapter 4.1.1

6.2 Use Case 2 - Backup, Metadata for Research Data and Metadata Extraction

Alice wants to add metadata to her research data files early in the research process. She wants to have the newest version of the file available in the Publikator without caring too much about the upload process. The embedded GPS coordinates and timestamps of the photos she takes with her smartphone shall be usable in the Publikator. Also she needs a backup solution for her files.

Alice installs the Seafile desktop client on her notebook and the Seafile mobile client on her smartphone. This allows her to drop the folder with her research data from her notebook into the Seafile client, to have it available in the Publikator. In the mobile client she defines a subfolder of her research data library as a target for automated photo upload.

Now Seafile takes care about synchronizing the data from her notebook and her mobile phone every time Alice has network access. With network access, she is able to describe her files with appropriate metadata. The GPS location and time stamp information extracted from her photos help her to put the files in context. Knowing that the data is stored at a provider she trusts, and the internet connection is secured using strong SSL encryption, gives Alice a secure feeling.

Back from her research trip she has already done the groundwork for her data publication.

The implementation done within this thesis helps Alice, as she has a cloud-based backup solution for her files, which at the same time allowed her to start adding metadata to her data early in the research process. Using file synchronization is a benefit in her case, as she did not have to find out herself which data was already copied to the cloud, when she gained network access.

Years later, another researcher is able to locate Alice's research data in the DARIAH-DE repository, as the geographical coordinates and the Dublin Core terms and keywords from the metadata allow very precise searching. He is able to cite her data using PIDs and has the guarantee that the data referenced by these URIs stays accessible. So the high quality of the metadata and the persistent nature of the DARIAH-DE repository enriches the research process far beyond Alice's research project.

6.3 Use Case 3 - Collaboration

For collaborating on a paper, Bob needs access to the data Alice collects in Vietnam. Alice logs into the Seafile web interface, opens her Vietnam research data library and clicks on "share". Now she is able to identify Bob by his DARIAH-DE account name, and give him access to her library. This library shows up in Bobs Seafile desktop client now, and he is able to synchronize it to his hard disk. Unfortunately, he is not able to view the DC metadata Alice added to the data, as the DARIAH-DE Publikator does not yet allow collaboration in groups, and Alice did not publish her data yet.

So the Publikator Seafile connection partially helped the collaboration of Alice and Bob, but would be much more convenient if Bob also was able to view the metadata Alice added in the Publikator. To implement group management in the Publikator, the DARIAH-DE AAI would need group management for the DARIAH-DE own storage beforehand. This is on the roadmap for the AAI, so collaboration in groups may be added to the Publikator later on. Seafile already has group support, it is yet to find out if these may also be Shibboleth groups.

6.4 Possible Future Work

6.4.1 Tracking Renaming and Removal of Files and Subfolders

The handling of moving, removing or renaming files implemented within this thesis is not very user friendly. If in he current implementation a file or subcollection is not found with its former path or name this is handled as a removal. In consequence the metadata for this file is silently removed from the storage without any user interaction or notification. The worst case which may occur here would be a user who has manually added metadata to a bunch of files renaming a subcollection, in which case all the work would have to be done again, although the files are still there.

A better implementation is possible and may rely on the Seafile API for file history[135]. A little test of this API and a directory renaming from *subfolder1* to *subfolder2* shows the following history for a file *test1.txt* residing in this directory:

```
$ curl -H 'Authorization: Token sECRETtOKEN' \
  https://sftest.de.dariah.eu/api2/repos/95c7bdc2-c93d-4fbb-a9cf\
  /file/history/?p=/subfolder2/test1.txt

  {"commits": [
      {   ...
          "desc": "Renamed \"subfolder1\".\n",
          ...
          "rev_renamed_old_path": "subfolder1/test1.txt",
          ...
      },{ ...
          "desc": "Added or modified \"test1.txt\".\n",
          ...
          "rev_renamed_old_path": null,
          ...
      }
  ]}
```

The previous path of the file *subfolder1/test1.txt* is tracked in the history of *subfolder2/test1.txt* in the property `rev_renamed_old_path`. A renaming of *test1.txt* to *test2.txt* is also trackable with the `rev_renamed_old_path` JSON key.

If a subfolder (or a file) is renamed, the Publikator-Seafile backend finds some deleted and some new entries. By checking the history of the new files it can find the references to the old entries and re-add the matching metadata accordingly.

This still would not help if a file or folder is really removed from Seafile. Possibly, the user moved it out accidentally or on purpose to re-add it later. In this case it would also be nicer of the Publikator

[135] Seafile (2016e). http://manual.seafile.com/develop/web_api.html#get-file-history

to not drop the metadata. The service could, instead of dropping the metadata silently, in this case communicate with the user and give the possibility to re-attach the unbound metadata to files in a later stage, or really drop it. This would require some further work on the GUI.

6.4.2 Seafile Drive Client

While this thesis was being finalized, Seafile Ltd. announced the availability of their Drive client:

> "Conceptually, the Drive client extends your local disk space with the massive storage capacity on your Seafile server."[136]

The basic idea is that the Seafile libraries are mapped to a virtual storage drive in the file system of the user. But different from the way the software works until now, not all files are downloaded to the users drive. While the users are able so see the whole remote file system, only the files they decide to work on get downloaded. Other than with network drive solutions like WebDAV, NFS or Windows shares the data is still editable without network connection and gets synchronized to the server when back online.

Dropbox announced a similar concept called "Project Infinite" in April 2016.[137]. The solution from Dropbox requires a module running in kernel mode[138], which means that it gains access to the whole system, while the normal Dropbox client only runs with the permissions of the user who installed it. Seafile tries to achieve the same goals with the software still running in user mode, which is performing nicer with data privacy and security.

The Seafile drive client should enhance the usability Seafile on client side, and also allows for further interesting use cases like "Writing large scientific data directly to Seafile server."[139] As it is a client side development, using the Seafile drive client with the Publikator does not need further implementation on the Publikator side. Users should be able to use the Seafile drive client instead of or additionally to the normal Seafile desktop client.

6.4.3 Client Side Encryption

It would be a big benefit in regards to privacy and security, if the Publikator could work with Seafile libraries which are saved on the synchronization server with client side encryption enabled. There was no further research done on this issue within this thesis, as the use of client side encrypted libraries and data publication seem to be contradictionary on the first look. But with the goal to allow metadata annotation as early as possible in the data collection phase, there may be the legitimate requirement to keep the data locked until publication.

[136] Seafile (2016a)
[137] Dropbox Inc. (2016a). https://blogs.dropbox.com/business/2016/04/announcing-project-infinite/
[138] Clarke (2016). http://www.theregister.co.uk/2016/05/26/dropbox_kernel_access/
[139] Seafile (2016a)

Seafile allows client side encryption, which means that the data may only bee seen on the client were a password is entered, but it is not visible to the server administrator. As some testing has shown it is possible to view the folder structure and the filenames with the Seafile web API.[140] This may be a drawback for some use cases of client side encryption, as file names may tell a lot about what content a user is synchronizing. Still for the implementation of a publishing workflow this behavior may be useful, as also encrypted files may be annotated with metadata in this case.

Still there are further things to be investigated, as how to seamlessly decrypt files on publication. In this case a password would need to be entered and handed over to the DH-publish service. Also the "feature" this would be based on is a bug on the tracker[141], so the metadata leakage, such an approach would be based on, may be removed in future.

Also a security audit would have to take place, to give a proper statement how secure the usage of client side encryption is and what limitations may apply.

7 Conclusion

For the abstract idea of a research data lifecycle to be put into practice the perspective of the information science has to meet the requirements of the target group, in this case digitally working researchers in the Arts and Humanities. This may look like "more work" for the researcher in the first place. But it is an agreed fact that reuse of scientific data fosters research on a long run. A task for the information sciences is to provide tools for researchers to ease the process of data preparation for publication. This could only be done by looking into the requirements of the target group, and observe how it is working with digital data. The Publikator in the DARIAH-DE project was build for this reasons. The development of this tool was based on theoretical ground works done within DARIAH-DE, like the definition of a research data lifecycle and how it matches the workflow of digitally working humanists.[142]

This thesis took a deeper look at one crucial part of the research data lifecycle, how and when to prepare the data for publication. The outcome of this thesis is an extension of the publishing workflow, in a way that research data can be enriched with metadata for publication earlier in the research process. While the software developed in this thesis is in a prototype state, it covers large parts of the use cases and allows testing the initial idea. To have files from hard disk available for metadata addition early in the research process serves a requirement of the researchers. Being able to view the actual state of the files and work on their metadata any time in the research process, and not having to manually synchronize the state on hard disk with that in the web browser is a large benefit. This is a task file synchronization software is already doing very well, as its explicitly good in tracking state changes in different locations.

[140] see ef4 (2016)
[141] ef4 (2016)
[142] see chapter 2.6

Abbreviations

AAI - Authorization and Authentication Infrastructure

AES - Advanced Encryption Standard

AJAX - Asynchronous JavaScript and XML

API - Application Programming Interface

DARIAH - Digital Research Infrastructure for the Arts and Humanities

CD-ROM - Compact Disc Read-Only Memory

CMS - Content Management System

CPU - Central Processing Unit

DC - Dublin Core

DFN - Deutsches Forschungsnetzwerk

DFG - Deutsche Forschungsgemeinschaft

DVD - Digital Versatile Disc

EPIC - European Persistent Identifier Consortium

GUI - Graphical User Interface

GPS - Global Positioning System

GWDG - Gesellschaft für wissenschaftliche Datenverarbeitung Göttingen

HTTP - Hypertext Transfer Protocol

ID - Identifier

IDP - Identity Provider

IRI - Internationalized Resource Identifier

JSF - Java Server Faces

JSON - JavaScript Object Notation

JWT - JSON Web Token

LDAP - Lightweight Directory Access Protocol

NFS - Network File System

NSA - National Security Agency

OASIS - Organization for the Advancement of Structured Information Standards

OCS - Open Collaboration Services

PC - Personal Computer

PDF - Portable Document Format

PID - Persistent Identifier

POJO - Plain Old Java Object

RAM - Random Access Memory

RDF - Resource Description Framework

REST - Representational State Transfer

SHA - Secure Hash Algorithm

SP - Service Provider

SSL - Secure Sockets Layer (old, now: TLS)

SSO - Single Sign On

SAML - Security Assertion Markup Language

TLS - Transport Layer Security

URI - Uniform Resource Identifier

URL - Uniform Resource Locator

W3C - World Wide Web Consortium

XML - Extensible Markup Language

List of Figures

Bibliography

Apache Software Foundation. (2004). Apache License Version 2.0. Retrieved September 13, 2016, from https://www.apache.org/licenses/LICENSE-2.0

Apache Tika. (2016a). Apache Tika Website. Retrieved September 7, 2016, from https://tika.apache.org/

Apache Tika. (2016b). Apache Tika: Supported Document Formats. Retrieved September 7, 2016, from https://tika.apache.org/1.12/formats.html

Balasubramaniam, S., & Pierce, B. C. (1998). What is a file synchronizer? In *Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom '98)*. Retrieved from http://www.cis.upenn.edu/~bcpierce/papers/snc.ps

Becket, D., Berners-Lee, T., Prud'hommeaux, E., & Carothers, G. (2014). RDF 1.1 Turtle - Terse RDF Triple Language. Retrieved September 7, 2016, from http://www.w3.org/TR/turtle/

Bray, T. (2014). *The JavaScript Object Notation (JSON) Data Interchange Format* (RFC No. 7159). RFC Editor; Internet Requests for Comments; RFC Editor. Retrieved from http://www.rfc-editor.org/rfc/rfc7159.txt

Brickley, D., & Guha, R. (2014). RDF Schema 1.1. Retrieved September 7, 2016, from https://www.w3.org/TR/rdf-schema/

Clarke, G. (2016). Dropbox gets all up in your kernel with Project Infinite. *The Register*. Retrieved from http://www.theregister.co.uk/2016/05/26/dropbox_kernel_access

DARIAH-DE. (2016a). About DARIAH-DE. Retrieved August 15, 2016, from https://de.dariah.eu/dariah-de-english

DARIAH-DE. (2016b). Collection Registry. Retrieved September 7, 2016, from https://de.dariah.eu/collection-registry

DARIAH-DE. (2016c). DARIAH-DE Publikator website. Retrieved September 7, 2016, from https://de.dariah.eu/publish

DARIAH-DE. (2016d). DARIAH-DE Website. Retrieved September 7, 2016, from https://de.dariah.eu

DARIAH-DE. (2016e). DARIAH-DE-Wiki: DARIAH AAI Documentation. Retrieved September 7, 2016, from https://wiki.de.dariah.eu/display/publicde/DARIAH+AAI+Documentation

DARIAH-DE. (2016f). DARIAH-DE: Autorisierungs- und Authentifizierungs-Infrastruktur. Retrieved September 7, 2016, from https://de.dariah.eu/aai

DARIAH-DE. (2016g). DARIAH-DE: Forschungsdaten in DARIAH-DE. Retrieved September 15, 2016, from https://de.dariah.eu/forschungsdaten

DARIAH-DE. (2016h). DARIAH-DE: Information Flyer - English. Retrieved September 15, 2016, from https://de.dariah.eu/documents/10180/411593/DARIAH-Pentagon-Klappflyer-EN-1.5.pdf

DARIAH-DE. (2016i). DARIAH-DE: Research Data Lifecycle – Der Forschungsdatenzyklus in DARIAH-DE.

Retrieved September 15, 2016, from https://de.dariah.eu/research-data-lifecycle

DARIAH-DE. (2016j). Das DARIAH-DE Repositorium. Retrieved September 21, 2016, from https://wiki.de.dariah.eu/display/publicde/Das+DARIAH-DE+Repositorium

DARIAH-DE. (2016k). DataObjects RDF-Schema for Publikator. Retrieved September 7, 2016, from http://de.dariah.eu/rdf/dataobjects/terms/

DARIAH-DE. (2016l). Generische Suche. Retrieved September 7, 2016, from https://de.dariah.eu/generische-suche

DARIAH-DE. (2016m). Git repository for the Publikator. Retrieved September 13, 2016, from https://projects.gwdg.de/projects/publish-gui-portlet/repository

DARIAH-DE. (2016n). Kriterien für die Integration von Tools und Diensten in die DARIAH-DE-Infrastruktur. Retrieved September 7, 2016, from https://de.dariah.eu/kriterien-toolintegration

DARIAH-DE. (2016o). Nachhaltige Referenzierung von Digitalen Objekten mit Hilfe von persistenten Identifikatoren (PID). Retrieved September 7, 2016, from https://de.dariah.eu/pid-service

DARIAH-DE. (2016p). Nachhaltige Referenzierung von Digitalen Objekten mit Hilfe von persistenten Identifikatoren (PID). Retrieved September 25, 2016, from https://de.dariah.eu/pid-service

DARIAH-EU. (2016). DARIAH in a Nutshell: By Researchers for Researchers. Retrieved September 25, 2016, from http://dariah.eu/about.html

DCMI Usage Board. (2012). DCMI Metadata Terms. Retrieved September 7, 2016, from http://dublincore.org/documents/2012/06/14/dcmi-terms/

DDI Structural Reform Group. (2004). DDI Version 3.0 Conceptual Model. Retrieved September 8, 2016, from http://opendatafoundation.org/ddi/srg/Papers/DDIModel_v_4.pdf

Deutsche Forschungsgemeinschaft (DFG). (2014). Cloud-Dienste — Addendum zu den Empfehlungen der Kommission für IT Infrastruktur. Retrieved September 8, 2016, from http://www.dfg.de/download/pdf/foerderung/programme/wgi/addendum_cloud_dienste_kfr_2014.pdf

Deutsche Nationalbibliothek (DNB). (2016). Datendienst "Bibliografische Dienstleistungen" - Normdaten in RDF. Retrieved September 17, 2016, from http://datendienst.dnb.de/cgi-bin/mabit.pl?userID=opendata&pass=opendata&cmd=login

Deutsches Forschungsnetz (DFN). (2016). DFN-AAI - Authentifikations- und Autorisierungs-Infrastruktur. Retrieved September 12, 2016, from https://www.aai.dfn.de/

Django Software Foundation. (2016). django Website. Retrieved September 17, 2016, from https://www.djangoproject.com/

Donauer, J. (2015). Benchmark: ownCloud gegen Seafile – wer synchronisiert schneller? *BITblokes*. Blog. Retrieved from https://www.bitblokes.de/2015/01/benchmark-owncloud-gegen-seafile-wer-synchronisiert-schneller/

Dropbox Inc. (2016a). A revolutionary new way to access all your files – Dropbox announcement of Projectb Infinite. Retrieved September 26, 2016, from https://blogs.dropbox.com/business/2016/04/

announcing-project-infinite/

Dropbox Inc. (2016b). Dropbox Blog: Celebrating half a billion users. Retrieved September 20, 2016, from https://blogs.dropbox.com/dropbox/2016/03/500-million/

Dropbox Inc. (2016c). Dropbox Help: Can I have Dropbox sync files outside my Dropbox folder? Retrieved September 17, 2016, from https://www.dropbox.com/help/12

Dropbox Inc. (2016d). Dropbox Help: Can I specify my own private key for my Dropbox? Retrieved September 17, 2016, from https://www.dropbox.com/help/28

Dropbox Inc. (2016e). Dropbox Help: How secure is Dropbox? Retrieved September 17, 2016, from https://www.dropbox.com/help/27

Dropbox Inc. (2016f). Dropbox Help: What's a conflicted copy? Retrieved September 17, 2016, from https://www.dropbox.com/help/36

Dropbox Inc. (2016g). Dropbox Website. Retrieved September 17, 2016, from https://www.dropbox.com/

Dropbox Inc. (2016h). Dropbox: Core API. Retrieved September 17, 2016, from https://www.dropbox.com/developers-v1/core/docs

Dunn, S. (2008). Dropbox File Sync Service. *The Washington Post*. Retrieved from http://www.washingtonpost.com/wp-dyn/content/article/2008/08/01/AR2008080100260.html

Dyllick-Brenzinger, C. (2015). ownCloud vs Seafile – Performance. *ionas Blog*. Blog. Retrieved from https://www.ionas-server.com/blog/owncloud-vs-seafile-performance/

ECMA International. (2016). *Standard ECMA-262 - ECMAScript 2015 Language Specification* (6.0). Retrieved from http://www.ecma-international.org/ecma-262/6.0/

ef4. (2016). Encrypted libraries leak lots of information — Seafile bug tracker. Retrieved September 25, 2016, from https://github.com/haiwen/seafile/issues/350

Funk, S. E. (2010). Digitale Objekte und Formate. In H. Neuroth, A. Oßwald, Scheffel R, S. Strathmann, & K. Huth (Eds.), *nestor Handbuch: Eine kleine Enzyklopädie der digitalen Langzeitarchivierung* (pp. 140–145). Boizenburg: Verlag Werner Hülsbusch,

Funk, S. E., & Schmunk, S. (2015). DARIAH-DE Repositorium – Prototyp (M 4.3.2.1). Retrieved August 9, 2016, from https://dev2.dariah.eu/wiki/download/attachments/14651583/M%204.3.2.1-DARIAH-Repositorium-Prototyp-final.pdf

Funk, S. E., Gietz, P., Haase, M., Harms, P., Aschenbrenner, A., Tonne, D., & Rybicki, J. (2012). DARIAH Storage API – A Basic Storage Service API on Bit Preservation Level. Retrieved September 7, 2016, from

https://dev2.dariah.eu/wiki/download/attachments/10618851/DARIAH-Storage-API-v1.0_final.pdf

Gartner, Inc. (2016). Gartner Says Worldwide Smartphone Sales Grew 3.9 Percent in First Quarter of 2016. Retrieved September 17, 2016, from http://www.gartner.com/newsroom/id/3323017

Gesellschaft für wissenschaftliche Datenverarbeitung Göttingen (GWDG). (2016). GWDG Homepage. Retrieved September 24, 2016, from https://www.gwdg.de/

Göbel, M., Grupe, N., Heise, C., Köhlmann, M., Meyer, K., Neuschäfer, M., … Söring, S. (2015). DARIAH-DE und TextGrid - Disseminationsstrategie inklusive Marketingkonzept sowie DARIAH-DE Open Mission Statement und Publikationsstrategie. Retrieved from https://dev2.dariah.eu/wiki/download/attachments/14651583/DARIAH-TextGrid-Disseminationskonzept.pdf

Greenwald, G., & MacAskill, E. (2013). NSA Prism program taps in to user data of Apple, Google and others. *The Guardian*. Retrieved from http://www.theguardian.com/world/2013/jun/06/us-tech-giants-nsa-data

Guy, M., Powell, A., & Day, M. (2004). Improving the quality of metadata in Eprint archives. *Ariadne*, (38). Retrieved from http://www.ariadne.ac.uk/issue38/guy

Haase, M., Gietz, P., Widmer, M., Funk, S. E., & Veentjer, U. (2016). *Einbindung von RESTlike Web Services in eine SAML-basierte Föderation mit OAuth2*. DARIAH-DE; Draft; not yet published as of 23-September-2016; DAASI.

Heckel, P. C. (2012). Minimizing remote storage usage and synchronization time using deduplication and multichunking: Syncany as an example. Retrieved September 12, 2016, from https://web.archive.org/web/20130508153942/http://www.philippheckel.com/files/syncany-heckel-thesis.pdf

Jackson Project. (2016). Jackson Project Home @github. Retrieved September 17, 2016, from https://github.com/FasterXML/jackson

JCP. (2003). JSR 168: Portlet Specification. Retrieved September 7, 2016, from https://jcp.org/en/jsr/detail?id=168

Jones, M., Bradley, J., & Sakimura, N. (2015). *JSON Web Token (JWT)* (RFC No. 7519). RFC Editor; Internet Requests for Comments; RFC Editor. Retrieved from http://www.rfc-editor.org/rfc/rfc7519.txt

Josefsson, S. (2006). *The Base16, Base32, and Base64 Data Encodings* (RFC No. 4648). RFC Editor; Internet Requests for Comments; RFC Editor. Retrieved from http://www.rfc-editor.org/rfc/rfc4648.txt

jsTree. (2006). jsTree Website. Retrieved September 8, 2016, from https://www.jstree.com/

Karlitschek, F. (2013). Draft Specification: Open Collaboration Services v1.7. Retrieved March 25, 2016, from https://www.freedesktop.org/wiki/Specifications/open-collaboration-services-1.7/

Karlitschek, F. (2016). Nextcloud. *Blog of Frank Karlitschek*. Blog. Retrieved from http://karlitschek.de/2016/06/nextcloud/

Liferay Inc. (2016). Liferay website. Retrieved September 7, 2016, from https://www.liferay.com/

Nextcloud GmbH. (2016). Nextcloud Homepage. Retrieved September 25, 2016, from https://nextcloud.

com/

NISO. (2004). Understanding Metadata. Retrieved September 18, 2016, from http://www.niso.org/publications/press/UnderstandingMetadata.pdf

OASIS. (2016). OASIS:SAML. Retrieved September 12, 2016, from http://www.oasis-open.org/committees/security

OAuth. (2016a). OAuth 2.0. Retrieved September 12, 2016, from https://oauth.net/2/

OAuth. (2016b). OAuth Homepage. Retrieved September 12, 2016, from https://oauth.net/

Oracle. (2016). Deflater Java API. Retrieved September 13, 2016, from https://docs.oracle.com/javase/8/docs/api/java/util/zip/Deflater.html

ownCloud. (2016a). Encryption Configuration — ownCloud 8.1 Server Administration Manual. Retrieved September 25, 2016, from https://doc.owncloud.org/server/8.1/admin_manual/configuration_files/encryption_configuration.html

ownCloud. (2016b). External API — ownCloud Developer Manual 8.1 documentation. Retrieved September 25, 2016, from https://doc.owncloud.org/server/8.1/developer_manual/core/externalapi.html

ownCloud. (2016c). Features — ownCloud Website. Retrieved September 25, 2016, from https://owncloud.org/features/

ownCloud. (2016d). ownCloud Website. Retrieved September 25, 2016, from https://owncloud.org/

ownCloud GmbH. (2016). Features — ownCloud.com Website. Retrieved September 25, 2016, from https://owncloud.com/features/

Pivotal Software, Inc. (2016). Spring Website. Retrieved September 7, 2016, from https://spring.io/

Puhl, J., Andorfer, P., Höckendorff, M., Schmunk, S., Stiller, J., & Thoden, K. (2015). Diskussion und Definition eines Research Data LifeCycle für die digitalen Geisteswissenschaften. Retrieved from http://webdoc.sub.gwdg.de/pub/mon/dariah-de/dwp-2015-11.pdf

Ractive.js. (2016a). Ractive.js Documentation — Keypaths. Retrieved September 8, 2016, from http://docs.ractivejs.org/latest/keypaths

Ractive.js. (2016b). Ractive.js Documentation — Observers. Retrieved September 8, 2016, from http://docs.ractivejs.org/latest/observers

Ractive.js. (2016c). Ractive.js Website. Retrieved September 8, 2016, from http://www.ractivejs.org/

Riebl, H. (2016). Git repository for tgForms. Retrieved September 7, 2016, from https://github.com/hriebl/tgForms/

Romanello, M., Stiller, J., & Thoden, K. (2015). Usability Criteria for External Requests of Collaboration (R 1.2.2/R 7.5). Retrieved September 7, 2016, from https://dev2.dariah.eu/wiki/download/attachments/

14651583/R1.2.2_Usability_Criteria_for_External_Requests_of_Collaboration.pdf

Schießle, B. (2015). Encryption 2.0 in ownCloud Server 8.1. *ownCloud Blog*. Blog. Retrieved from https://owncloud.org/blog/encryption-2-0-in-owncloud-server-8-1/

Schreiber, G., & Raimond, Y. (2014). RDF 1.1 Primer. Retrieved September 7, 2016, from http://www.w3.org/TR/rdf11-primer/

Seafile. (2016a). Announcing Seafile Drive client, a new way to map Seafile storage as virtual drive. Retrieved September 26, 2016, from https://blogs.seafile.com/2016/09/02/announcing-seafile-drive-client-a-new-way-to-map-seafile-storage-as-virtual-drive/

Seafile. (2016b). Components of Seafile Server — Seafile Documentation. Retrieved September 7, 2016, from https://manual.seafile.com/develop/server-components.html

Seafile. (2016c). Github Repository — Seafile. Retrieved September 7, 2016, from https://github.com/haiwen/seafile

Seafile. (2016d). Reliable and High Speed File Sync and Share — Seafile Website. Retrieved September 7, 2016, from https://www.seafile.com/en/home/

Seafile. (2016e). Seafile Web API. Retrieved September 7, 2016, from https://manual.seafile.com/develop/web_api.html

Seafile. (2016f). Seafile-roadmap. Retrieved August 11, 2016, from https://seacloud.cc/group/3/wiki/seafile-roadmap/

Seafile. (2016g). Security Questions — Seafile Documentation. Retrieved September 7, 2016, from https://manual.seafile.com/security/security_features.html

Seafile. (2016h). SHA1 encoding of Seafile IDs - Source reference. Retrieved September 7, 2016, from https://github.com/haiwen/seafile/blob/d14b4f6ce8a9baefc642fd15bc75101f5b973206/common/fs-mgr.c#L453

Seafile. (2016i). Synchronization algorithm — Seafile Documentation. Retrieved September 25, 2016, from https://manual.seafile.com/develop/sync_algorithm.html

Seafile-Forum. (2015). How to sync same library with 2 Seafile servers? — Seafile Forum. Retrieved September 7, 2016, from https://forum.seafile-server.org/t/how-to-sync-same-library-with-2-seafile-servers/2071

Shibboleth. (2016). Shibboleth Website. Retrieved September 7, 2016, from https://shibboleth.net/

Stiller, J., Thoden, K., Leganovic, O., Heise, C., Höckendorff, M., & Gnadt, T. (2015). Nutzungsverhalten in den Digital Humanities (R 1.2.1/ M 7.6). Retrieved September 7, 2016, from https://dev2.dariah.eu/wiki/

download/attachments/14651583/Report1.2.1-final3.pdf

Syncplicity Blog. (2009). Spring Website. Retrieved September 12, 2016, from https://web.archive.org/web/20160304061348/https://www.syncplicity.com/blog/why-delta-sync-doesn-t-matter

Text Encoding Initiative (TEI). (2016). TEI Website. Retrieved September 18, 2016, from http://www.tei-c.org/

Tridgell, A. (1996). First release of rsync - rcp replacement. Retrieved September 7, 2016, from https://groups.google.com/forum/#!msg/comp.os.linux.announce/tZE1qtTcQaU/IF8GhGQ_uTsJ

University of Wuppertal. (2016). Tipps und Hinweise zum Datenschutz. Retrieved September 8, 2016, from http://www.uni-wuppertal.de/de/universitaet/struktur-institutionen/behoerdliche-datenschutzbeauftragte/tipps-und-hinweise-zum-datenschutz.html

Veentjer, U. (2016a). Git repository for the Seafile DH-publish Connector. Retrieved September 24, 2016, from https://projects.gwdg.de/projects/seafile-dhpublish-connector/repository

Veentjer, U. (2016b). Test server for the Publikator with Seafile. Retrieved September 26, 2016, from https://portal.sftest.de.dariah.eu/

Veentjer, U. (2016c). Using the DARIAH-DE Seafile service with the Publikator. Retrieved September 25, 2016, from https://sftest.de.dariah.eu/docs/

Verborgh, R. (2016). Git repository for n3.js. Retrieved September 7, 2016, from https://github.com/RubenVerborgh/N3.js/

Wikipedia. (2016a). Comparison of file synchronization software — Wikipedia, The Free Encyclopedia. Retrieved September 7, 2016, from https://en.wikipedia.org/wiki/Comparison_of_file_synchronization_software

Wikipedia. (2016b). Dropbox (service). Retrieved September 17, 2016, from https://en.wikipedia.org/wiki/Dropbox_%28service%29

Wikipedia. (2016c). Dropship (software). Retrieved September 17, 2016, from https://en.wikipedia.org/wiki/Dropship_%28software%29

Zafer, T. (2015). Why Client-Side Encryption Is the Next Best Idea in Cloud-Based Data Security. *Information Security Today*. Retrieved from http://www.infosectoday.com/Articles/Client-Side_Encryption.htm