

Institute of Architecture of Application Systems

University of Stuttgart
Universitätsstraße 38
D-70569 Stuttgart

Masterarbeit

**A platform for collaborative
quality assurance of
bibliographical references and
notes**

Tamara Müller

Course of Study: Softwaretechnik

Examiner: Prof. Dr. Dr. Frank Leymann

Supervisor: Dr. Oliver Kopp,
Dr. Uwe Breitenbücher

Commenced: May 17, 2017

Completed: November 17, 2017

Abstract

In scientific work it is common to work in teams, while dealing with an abundance of literature. With the help of a literature management software these can be collected and managed as well as exported in bibliographies. Many online sources offer functionalities to import references into reference management tools. However, the entries are often incomplete or faulty. Hence, this thesis covers a concept for a reference management platform that supports collaborative work and quality assurance of references. The main ideas are to use a group functionality to support collaboration and a rating system for the quality assurance. Additionally, the users can edit reference entries to improve their quality. As part of this thesis, a prototype was implemented which supports both aspects.

Kurzfassung

Beim wissenschaftlichen Arbeiten wird üblicherweise in Teams gearbeitet. Dabei befasst man sich mit einer Menge von Literatur. Diese kann mit Hilfe von einem Literaturverwaltungsprogramm gesammelt und verwaltet, sowie in Literaturlisten exportiert werden. Viele Onlinequellen bieten die Möglichkeit an Referenzen in Literaturverwaltungsprogramme zu importieren. Jedoch sind diese Referenzen oft unvollständig oder fehlerhaft. Deshalb befasst sich diese Arbeit mit einem Konzept für eine Literaturverwaltungsplattform welche gemeinschaftliches Arbeiten und Qualitätssicherung von Literaturangaben unterstützt. Die Hauptideen sind eine Gruppenfunktion für das Zusammenarbeiten von mehreren Personen und ein Bewertungssystem zur Qualitätssicherung. Darüber hinaus können die Benutzer eine Quellenangabe bearbeiten und somit verbessern. Im Rahmen dieser Arbeit wurde ein Prototyp implementiert welche beide Aspekte unterstützt.

Contents

1	Introduction	15
2	Related Work	17
2.1	Requirements for a Reference Management Software for Collaborative Work with Quality Assurance of References	17
2.2	Aigaion	18
2.3	BibSonomy	20
2.4	Citavi	22
2.5	CiteULike	24
2.6	Colwiz	27
2.7	Docear	29
2.8	EndNote	31
2.9	F1000Workspace	33
2.10	JabRef	35
2.11	RefWorks	37
2.12	Comparison of the Reference Management Tools	40
2.13	Other Related Work	40
3	Concept	45
3.1	Collaborative Work	45
3.2	Quality Assurance of Bibliographical References	46
3.3	Managing Bibliographical References	48
3.4	Managing Comments, Users, and Ratings	61
3.5	System	61
4	Architecture and Implementation	69
4.1	Architecture	69
4.2	Design Decisions	69
4.3	Implementation	71
5	Conclusion and Future Work	75
	Bibliography	77

List of Figures

2.1	Aigaion screenshot.	19
2.2	BibSonomy screenshot.	21
2.3	Citavi screenshot.	23
2.4	CiteULike screenshot.	25
2.5	Colwiz screenshot.	28
2.6	Docear screenshot.	30
2.7	EndNote X8 screenshot.	32
2.8	F1000Workspace screenshot.	34
2.9	JabRef screenshot.	36
2.10	RefWorks screenshot.	38
3.1	BPMN process of rating a reference.	46
3.2	BPMN process of rating a suggestion for modification of a reference.	47
3.3	BPMN process of the additional functionalities for a user with the role maintainer.	48
3.4	Git repository example of CloudRef.	49
3.5	Page for creating a new user account in CloudRef.	62
3.6	Screenshot of the login page of CloudRef.	62
3.7	CloudRef screenshot of the page for the manual insertion of a new reference.	63
3.8	CloudRef screenshot of the .bib file import page.	64
3.9	CloudRef screenshot of the table with all references.	65
3.10	CloudRef screenshot of a reference entry.	65
3.11	CloudRef screenshot which shows the view of suggestions to modify a reference.	66
3.12	Screenshot of the CloudRef view of a PDF file and corresponding comments.	67
4.1	Architecture of the CloudRef platform.	70
4.2	Architecture of the CloudRef platform with implementation details.	71
4.3	Database schema of the CloudRef platform.	73

List of Tables

2.1	Comparison of the reference management tools.	42
2.2	Comparison of the reference management tools (cont.).	43
2.3	Fulfillment of the requirements by the reference management tools.	44
3.1	Table that shows which examples of the cases use the Algorithm 3.1 to resolve occurred merge conflicts.	61

List of Listings

3.1	Content of a .bib file of the reference with the BibTeX key “test-merge”. . .	50
3.2	First suggestion to modify the reference with the BibTeX key “test-merge”. . .	50
3.3	Second suggestion to modify the reference with the BibTeX key “test-merge”. . .	50
3.4	Merge of the second suggestion with merge strategy “recursive” and the option “theirs” of Git.	51
3.5	Correct merge of the second suggestion.	51
3.6	Content of the test-merge.bib file after merging second suggestion with strategy “recursive” of Git.	52
3.7	Result of a Git “diff” operation on the second suggestion and the parent commit on the master.	52
3.8	The reference with BibTeX key “test-merge” is added.	54
3.9	Example for Case 1.	55
3.10	Example for Case 2.	55
3.11	Example for Case 3.	55
3.12	First part of the example for Case 4.1.	56
3.13	Second part of the example for Case 4.1.	56
3.14	Result after merging second suggestion of Case 4.1.	56
3.15	First part of the example for Case 5.1.	56
3.16	Second part of the example for Case 5.1.	56
3.17	Result after merging second suggestion of Case 5.1.	57
3.18	First part of the example for Case 5.2.	57
3.19	Second part of the example for Case 5.2.	57
3.20	Result after merging second suggestion of Case 5.2.	57
3.21	Example for Case 6.1.	58
3.22	First part of the example for Case 6.2.	58
3.23	Second part of the example for Case 6.2.	58
3.24	Result after merging second suggestion of Case 6.2.	58
3.25	First part of the example for Case 7.	59
3.26	Second part of the example for Case 7.	59
3.27	Result after merging second suggestion of Case 7.	59
3.28	First part of the example for Case 8.	59
3.29	Second part of the example for Case 8.	59
3.30	Result after merging second suggestion of Case 8.	60
3.31	First part of the example for Case 9.1.	60
3.32	Second part of the example for Case 9.1.	60
3.33	Result after merging second suggestion of Case 9.1.	60
3.34	First part of the example for Case 9.2.	60

3.35 Second part of the example for Case 9.2.	60
3.36 Result after merging second suggestion of Case 9.2.	61

List of Algorithms

3.1	Algorithm for resolving merge conflicts.	53
-----	--	----

1 Introduction

When writing a scientific paper, like an essay on a seminar topic, a thesis like this, or a scientific article for a prestigious conference, one has always to deal with an abundance of literature on the topic. Reference management software assists people in scientific work. The tools are used to collect literature, manage references, and export bibliographies. They provide an efficient way to keep an overview of a large amount of literature. Numerous tools provide the opportunity to manage knowledge about references inside comments, notes, or tags. In contrast to annotating and highlighting text inside a PDF reader, this has the advantage that the entire knowledge base can be searched within the software. Additionally, doing research is often a collaborative task, on which several people work together. Thus, literature management programs should ideally support collaboration. This includes sharing references and comments with other users or people who use another or no literature management tool at all.

There are multiple resources on the web, where people can search for literature such as Google Scholar¹, IEEE Xplore², Springer³, DBLP⁴, arXiv⁵, ResearchGate⁶, or Crossref⁷. Many of them offer the functionality to import a reference into the preferred reference management software. However, they often provide incomplete or faulty reference entries [Kop16]. A correct and complete entry is required for a correct reference list and that is on the other hand a prerequisite for publication. The publishers have different guidelines on how a bibliography has to be printed and which fields are necessary. For example, ACM (Association for Computing Machinery) requests the page numbers which are on the other hand not needed for IEEE (Institute of Electrical and Electronics Engineers), and IEEE has different journal abbreviations than others. To meet this dissimilar guidelines a faultless and complete reference list is required which than can be adapted to the rules of the publisher. Hence, a good quality of the reference entries is important. Many programs for managing references provide a mechanism to detect missing required fields and highlight this entries to show the user that they are incomplete. However, this is not sufficient because wrong information is not detected. The users have to check each reference entry by their own to ensure correctness.

¹<https://scholar.google.com>

²<http://ieeexplore.ieee.org>

³<https://link.springer.com>

⁴<http://dblp.uni-trier.de>

⁵<https://arxiv.org>

⁶<https://www.researchgate.net>

⁷<https://search.crossref.org>

The goal of this thesis is the development of a cloud-based web application for collaborative reference management. To support the cooperation of several people it should be possible to post comments to literature at different levels of visibility. Furthermore, the platform should provide quality assurance for bibliographical references to ensure complete and faultless references.

Structure

This work is structured as follows:

Chapter 2 – Related Work presents related work and several reference management tools which are compared in regard to collaborative work and quality assurance of bibliographical references. In addition, requirements which support these functionalities are formulated.

Chapter 3 – Concept introduces the approach to realize a software for reference management which supports collaborative work and quality assurance of bibliographical references. Furthermore, it explains the user interface of the system and evaluates which of the previously defined requirements are fulfilled.

Chapter 4 – Architecture and Implementation gives an overview of the architecture of the developed system and an insight into the implementation.

Chapter 5 – Conclusion and Future Work summarizes the work and provides an outlook for future work.

2 Related Work

At the beginning of this chapter 20 requirements are defined for a reference management software that supports collaborative work and quality assurance (Section 2.1). Afterwards, Sections 2.2 to 2.11 present several tools for reference management and evaluate them against the previously defined requirements. Section 2.12 provides an overview of the features of the programs and which requirements they fulfill. Finally, Section 2.13 presents other related works.

2.1 Requirements for a Reference Management Software for Collaborative Work with Quality Assurance of References

This thesis covers a concept for an literature management system that supports collaborative work and quality assurance of references. In the following requirements are defined for such a tool. The requirements are grouped. For each requirement, first the description is presented followed by the number of the requirement in brackets.

1. *Comments to literature:*

The users can write comments to literature within the software (Req. 1.1). Furthermore, comments in uploaded PDF files can be viewed (Req. 1.2) and extracted from the program (Req. 1.3). The user who uploads the PDF file can decide on what level of visibility the file (Req. 1.4) and comments (Req 1.5) are published.

2. *Visibility levels for comments:*

Comments can be published at different visibility levels: private (Req. 2.1), for single selected users (Req. 2.2), for multiple groups of users (Req. 2.3), and public that all users of the software can see it (Req. 2.4). The user can decide for each comment on what level of visibility it will be published (Req. 2.5).

3. *Collaborative reference management:*

Every user can create groups¹ (Req. 3.1) and invite several other users into these groups (Req. 3.2). Each group member is allowed to add new references (Req. 3.3), to upload files (Req. 3.4), and to write comments to literature (Req. 3.5).

¹or something similar, e.g., a shared database

4. *Quality assurance of references:*

References with missing information that is required are marked or correct references are highlighted (Req. 4.1). The software is also able to detect duplicate entries of references (Req. 4.2). Furthermore, a consistent notation of conferences (Req. 4.3), authors (Req. 4.4), and abbreviations (Req. 4.5) is ensured.

2.2 Aigaion

The open source reference management program Aigaion is a web based software written in PHP and MySQL [Aig13]. The latest version 2.2.b was released in 2010. To use the system it has to be installed and configured for example on a webspace. The administrator is able to create accounts for other users and manage their rights. After the installation there are the user groups admins, readers, editors, and guests. In the following the features of Aigaion are evaluated for users which are editors.

2.2.1 Import and Export

With Aigaion users can import and export BibTeX and RIS files [Aig13]. For importing the format REFER is available as well. Aigaion does not provide an opportunity to import references from web pages, search in other databases or a full-text search.

2.2.2 Data Input and Editing

New references can also be added manually. There are 13 available document types and the users can cross-reference entries for example reprints. Attachment of documents such as PDF files is also possible and in the site configuration new entry fields can be defined. However, the software does not allow new definitions of document types. During the import there is also the option to check for duplicates but only the titles are compared.

2.2.3 Data Display and Search

All references can be viewed in a list view (see Figure 2.1, center) and a single entry with all its fields by clicking on the title. Publications can be sorted by author, title, type/journal, year, and recently added in the navigation bar (see Figure 2.1, left). There is no support of a search history.

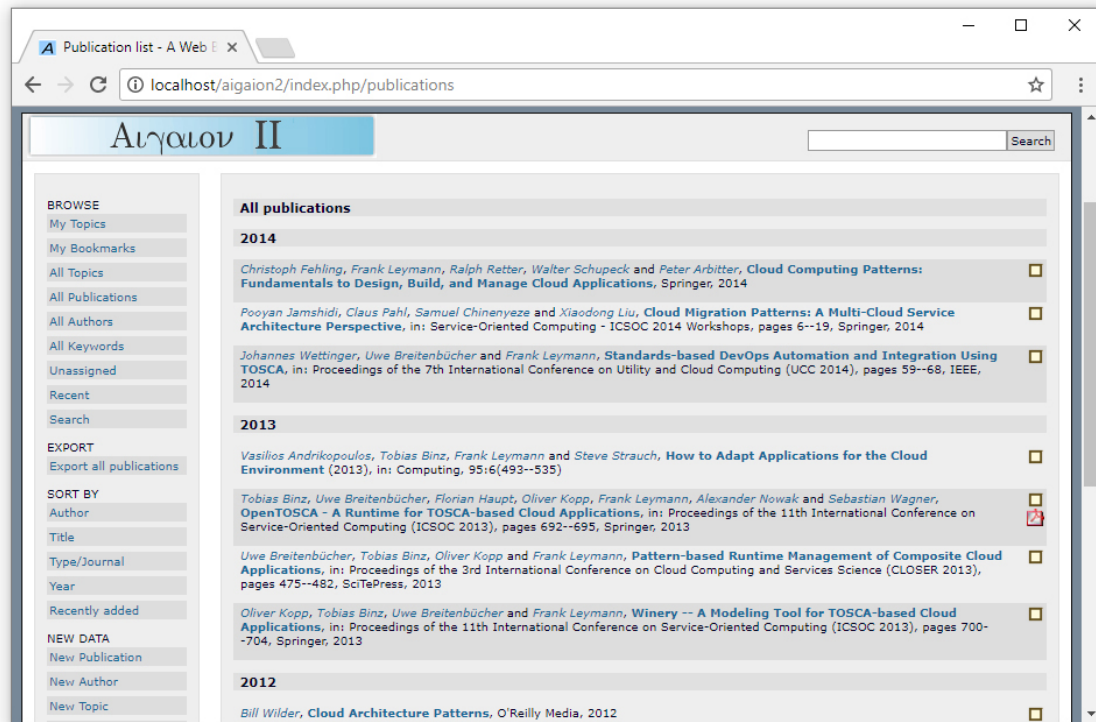


Figure 2.1: Aigaion 2.2.b screenshot. The references are listed in the center, sorted by the year of publication. At the navigation bar on the left side the sorting can be changed.

2.2.4 Cooperation

With the software multiple users can cooperate. For users of the group editors it is possible to create private, public or intern notes, references, and attachment of files. Public means that also anonymous users can see the object but they are not accessible without account. The access level intern is for all users which are not anonymous and private only for the object owner. For each item the user can decide which rights are applied. The access level is visualized with a traffic light next to the object.

2.2.5 Citation and Literature Lists

The software supports some citation styles but there is no opportunity to define or adapt styles. Furthermore, there is no support for word processing programs and literature lists can only be created statically.

2.2.6 Organization of Knowledge

To manage the knowledge Aigaion supports tags, notes and hierarchical groups of references called topics.

2.2.7 Fulfillment of the Requirements

The Requirements 1.1, 1.4 and 1.5 are met by Aigaion. From the second group of the requirements all requirements are fulfilled except for Requirement 2.2. The Requirements 3.1 to 3.5 are fulfilled completely because everyone can install the software and invite others to use it collaboratively. To support all these requirements every user needs suitable rights allocated by the administrator. Furthermore, the Requirement 4.4 is supported and partly Requirement 4.2 because only at the import a duplicate check is possible and only the titles get compared. Therefore, Aigaion provides nearly 14 of the 20 requirements.

2.3 BibSonomy

The open source [Bib17f] web application BibSonomy is a free tool to save and share publications and bookmarks.

2.3.1 Import and Export

BibSonomy can import RIS, BibTeX and other files. Besides, the export supports RIS [Bib17b], BibTeX, and other formats [Bib17b; Bib17c]. With an add-on new references can be imported directly from the browser [Bib17c]. Furthermore, a database search of their own database is available within the software and also a full-text search.

2.3.2 Data Input and Editing

References can be added manually as well. There are 20 different document types available but neither new document types nor new entry fields can be defined. The software also provides a search for metadata via DOI, ISBN, and other information. During the import of publications the references can be checked for duplicates afterwards there is no duplicate detection support. Uploading files to references is also possible [Bib17a]. Different references cannot be cross-referenced in BibSonomy and there is also no functionality for processing and evaluation of full texts.

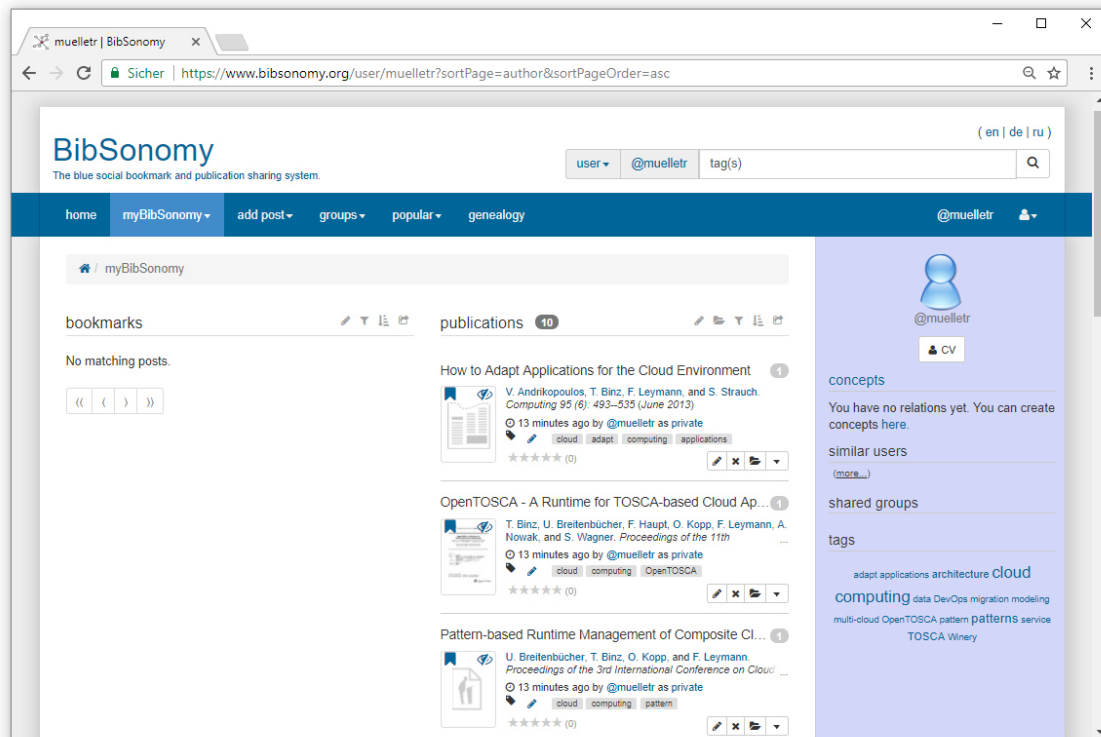


Figure 2.2: Screenshot of the BibSonomy system. In the middle the references are shown, to the left all bookmarks and to the right the publications. On the right-hand side of the screenshot are information of the user like his groups and a tag cloud.

2.3.3 Data Display and Search

BibSonomy provides a list view of all entries (see Figure 2.2, center). Thereby, bookmarks are shown on the left-hand side, whereas publications are located on the right-hand side. The entries can be sorted by date and title at the top of the list view. Advanced sorting options are available via adding parameters to the URL (see Figure 2.2, top) [Bib17e]. Each publication can be opened in a new site. There the most important information is shown and at the bottom of the page also a citation display and the BibTeX source. In the edit mode of a reference the user can see a full view of all entry fields. Furthermore, search histories are saved in BibSonomy.

2.3.4 Cooperation

References can be shared with friends or groups while adding a new reference or in the edit mode [Bib17d]. It is also possible to share a reference with multiple groups using the system tag “for:<group name>”. Every user can create a group but it has to be accepted by the system administrator. Similarly, joining a group has to be accepted by the group

administrator. The group functionality allows collaborative work with BibSonomy [Bib17c]. Comments can be shared with the group members or friends and it is possible to write private and anonymous notes [Bib17d]. In addition, there is a review functionality where users can write a review for a reference and rate it.

2.3.5 Citation and Literature Lists

The application supports different citation styles but no self-definition or adaptation of citation styles. A beta version of a Google Docs add-on is provided [Bib15]. Literature lists can be created in a static manner.

2.3.6 Organization of Knowledge

To organize the knowledge the user can create tags [Bib17c] and notes. The tags are shown in a tag cloud (see Figure 2.2, bottom right). Clicking on a tag shows all references containing the selected tag in a list view. It is also possible to select multiple tags.

2.3.7 Fulfillment of the Requirements

BibSonomy complies the Requirements 1.1 and 1.2 as well as Requirements 2.1 and 2.3 to 2.5. The Requirement 2.2 is not supported. Instead the software has a visibility level for comments that all friends of the user can see them. Additionally, the all requirements of group 3 are fulfilled and partly the Requirement 4.2 because duplicates can only be detected at the import. In summary, BibSonomy meets approximately 12 of 20 requirements.

2.4 Citavi

Citavi is a commercial reference management software for Windows [AMSW16; TUM16]. The available licenses can be categorized into Citavi Free, Citavi for Windows, and Citavi for DBServer. Citavi Free can be used without charge for small projects up to 100 references. The current version 5.6.0.2 of Citavi for Windows is considered in the following.

2.4.1 Import and Export

BibTeX and RIS are two of the supported import and export formats [TUM16]. Additionally, references can be added with the Citavi Picker directly from the browsers Mozilla Firefox, Internet Explorer, and Google Chrome [AMSW16; TUM16]. Within the software, database search is provided at arXiv, Crossref, IEEE Xplore, Springer, and others Furthermore, the user can search for full-texts [AMSW16; TUM16].

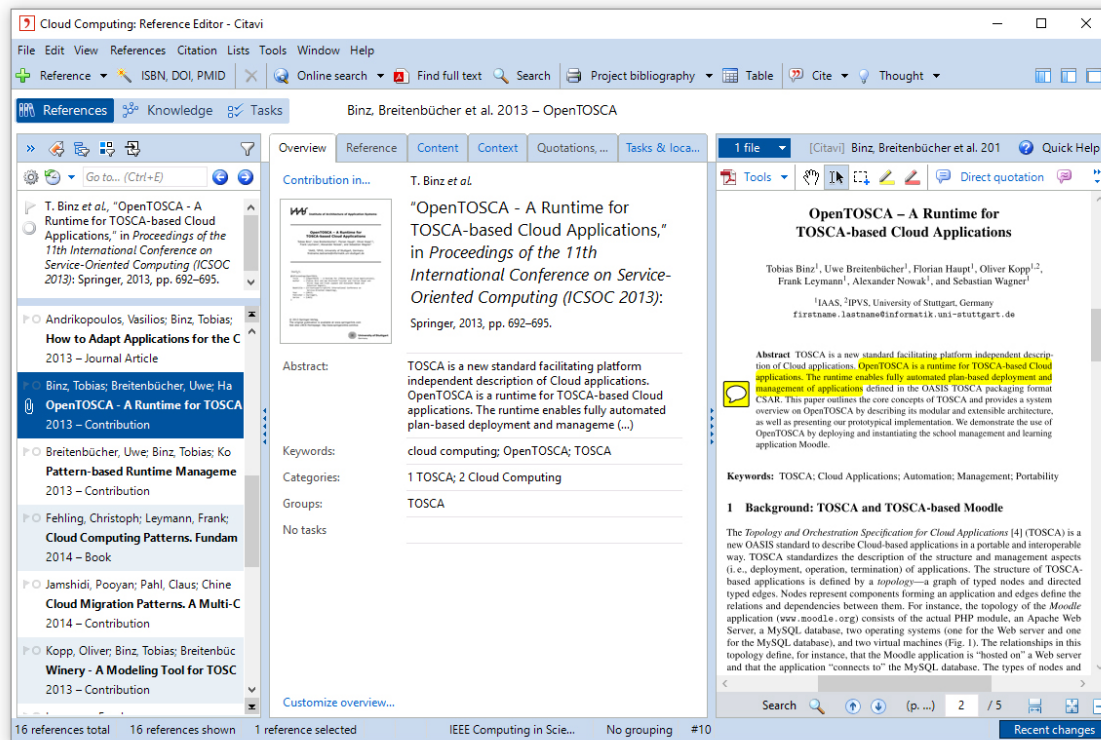


Figure 2.3: Citavi 5 screenshot. On the left-hand side all references are listed. In the center the selected entry is shown and on the right-hand side the corresponding PDF file of the reference.

2.4.2 Data Input and Editing

In addition to all the import capabilities, the user can also add a reference manually [AMSW16]. 35 document types are supported by Citavi [AMSW16; TUM16] but the user cannot define new ones [TUM16]. On the other hand, new entry fields can be specified by the user [AMSW16; TUM16]. The user can link references, e.g. reprints and upload files to references. Metadata can be completed automatically from PDF files with DOI and via the update references button [TUM16]. Additionally, a duplicate check and processing and evaluation of full-texts is provided by Citavi [AMSW16; TUM16].

2.4.3 Data Display and Search

The main window in Figure 2.3 shows a list view of all references [TUM16] on the left, an overview of the selected reference in the middle and the attached PDF file of the chosen reference on the right. The middle of the view has multiple tabs where all filled fields are presented [TUM16]. Above the list view the selected reference is shown with a citation style [AMSW16]. Moreover, all entries can be shown in a configurable table view in an

additional window [TUM16]. The references can be sorted by the user and a search history is available [AMSW16; TUM16].

2.4.4 Cooperation

With Citavi for Windows the cooperation of small teams is supported [Cit16d; TUM16]. The project has to be saved on a location where each user can access it and all members must run the same version of Citavi. It is recommended that only three persons use the project at once. If two users edit the same field of a reference concurrently the last change is saved and overwrites the other one [Cit16b]. For larger groups Citavi for DBServer has to be used where the project is stored on an SQL server [Cit16a; TUM16].

2.4.5 Citation and Literature Lists

There are many available citation styles in Citavi [AMSW16; TUM16]. In addition, the user can create own styles and adapt existing ones [AMSW16; Cit16c; TUM16]. The references can be inserted into the word processing programs Microsoft Word, LibreOffice, OpenOffice, and also in several TeX editors [AMSW16; TUM16]. A static creation of the literature list is supported and a dynamic creation for Microsoft Word and LaTeX.

2.4.6 Organization of Knowledge

The software has a knowledge organizer tab where thoughts [AMSW16], direct and indirect quotations [AMSW16], summaries, comments, and red highlighted text of references is shown. All of them and all references can be categorized, added to a group, and provided with keywords. Furthermore, the user can generate tasks, e.g. verify bibliographic information, and assign a date, priority, notes, and status to them.

2.4.7 Fulfillment of the Requirements

The Requirements 1.1, 1.4, 1.5, 2.1, and 2.2 are met. The third group of the requirements is achieved but in Citavi for Windows collaborative work can be done only with small groups. Besides, the Requirements 4.2 to 4.4 are met. In summary, Citavi fulfills 13 out of 20 requirements.

2.5 CiteULike

CiteULike is a free web application to manage references [Cit17b]. A Gold version provides paying users additional features like annotating PDF files [Cit17a].

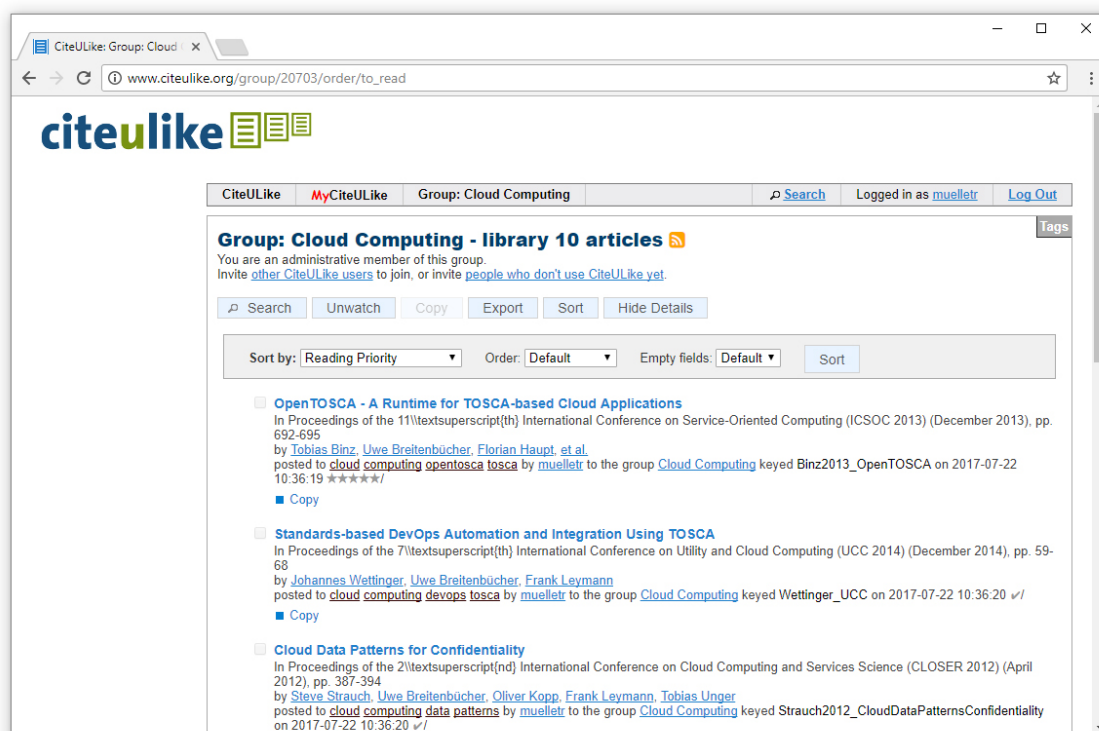


Figure 2.4: CiteULike screenshot. The references are shown in a list view. Above this list are several sorting options.

2.5.1 Import and Export

The import of CiteULike supports BibTeX and RIS files [Cit17b]. During the import a duplicate check can be executed. For exporting the formats BibTeX and RIS are also provided and some additional ones. Furthermore, it is possible to add references from the browser through a browser button. An installation is not required. CiteULike has an own database with entries from their users which can be searched by anyone. However, there is no search for full-texts available.

2.5.2 Data Input and Editing

In addition to the import options, users can also add references manually [Cit17b]. 17 document types are available but no definition of new types or entry fields. Besides, entries cannot be linked and adding metadata to existing references via DOI or ISBN is not supported. The attachment of files is limited to 2 documents and 5 images per reference. Gold users of CiteULike are able to run a duplicate check also after the import and they can annotate PDF files [Cit17a].

2.5.3 Data Display and Search

Figure 2.4 shows a group of references in CiteULike which are presented in a list view. By clicking on one entry it is shown in a short overview. It is also possible to show the BibTeX source and the reference in a citation style. A full view of all fields is available in the edit mode. Like in Figure 2.4 the entries can be sorted for example after reading priority. There is no search history available.

2.5.4 Cooperation

The software provides groups for collaborative work [Cit17b]. These groups can be visible for all users or only for members of the group. In the settings it is also possible to define who is allowed to enter the group. Either everyone can join the group, or users have to ask for permit to enter, or only invited users can join the group. Furthermore, the rights of the members can be chosen. They can have all rights or restricted ones where for example no references can be added. The creation of notes is always allowed.

2.5.5 Citation and Literature Lists

There are different citation styles available but the users cannot create their own. A word processing program is also not supported but static literature lists can be exported.

2.5.6 Organization of Knowledge

The users can create notes, tags, and groups to organize their knowledge. The notes can be published private or public.

2.5.7 Fulfillment of the Requirements

For the fulfillment of the requirements the features of Gold users are evaluated. All requirements of the first and third group are met. In addition, the requirements of group 2 are fulfilled except Requirement 2.3. Of the last requirement group only Requirement 4.2 is met by CiteULike. Overall, 15 of 20 requirements are fulfilled.

2.6 Colwiz

Colwiz is a free tool for managing bibliographical references. It is limited to 5000 publications per account and a maximum online storage of 30 GB [TUM16]. After the registration users have 2 GB storage which can be increased for example through inviting friends and completing the profile details. There is a desktop application and a similar looking web client. Below, the features of the desktop version 3.17.0606 are explained.

2.6.1 Import and Export

In Colwiz users can import and export BibTeX, RIS, and other formats [TUM16]. A Web Importer is provided which allows adding references from many web sites directly from the browser. Furthermore, web search with Google Scholar, arXiv, IEEE Xplore, DBLP, Springer, Crossref, the programs own database, and others can be executed within the software. Besides, full-text search is possible.

2.6.2 Data Input and Editing

In addition, users can add references manually. 15 publication types are supported by Colwiz. However, the users cannot create their own types or entry fields [TUM16]. Metadata of entries can be completed via DOI, ISBN, and other information and a duplicate check is executed automatically by the software. There is no possibility to link different references like reprints but the users can attach files to publications and annotate full-texts.

2.6.3 Data Display and Search

Colwiz provides a list view (see Figure 2.5, center) and a table view for multiple references [TUM16]. If an entry is selected a short overview of it is shown (see Figure 2.5, right) and in the edit mode the users can look at all fields. Furthermore, there are several sorting options and a search history [TUM16].

2.6.4 Cooperation

User groups can be created with the visibility levels private, public, and hidden. Users can copy their references to groups and files can be shared via Colwiz Drive [col14]. Additionally, the annotation of multiple members is supported for PDF files, Microsoft Word and Powerpoint documents.

2 Related Work

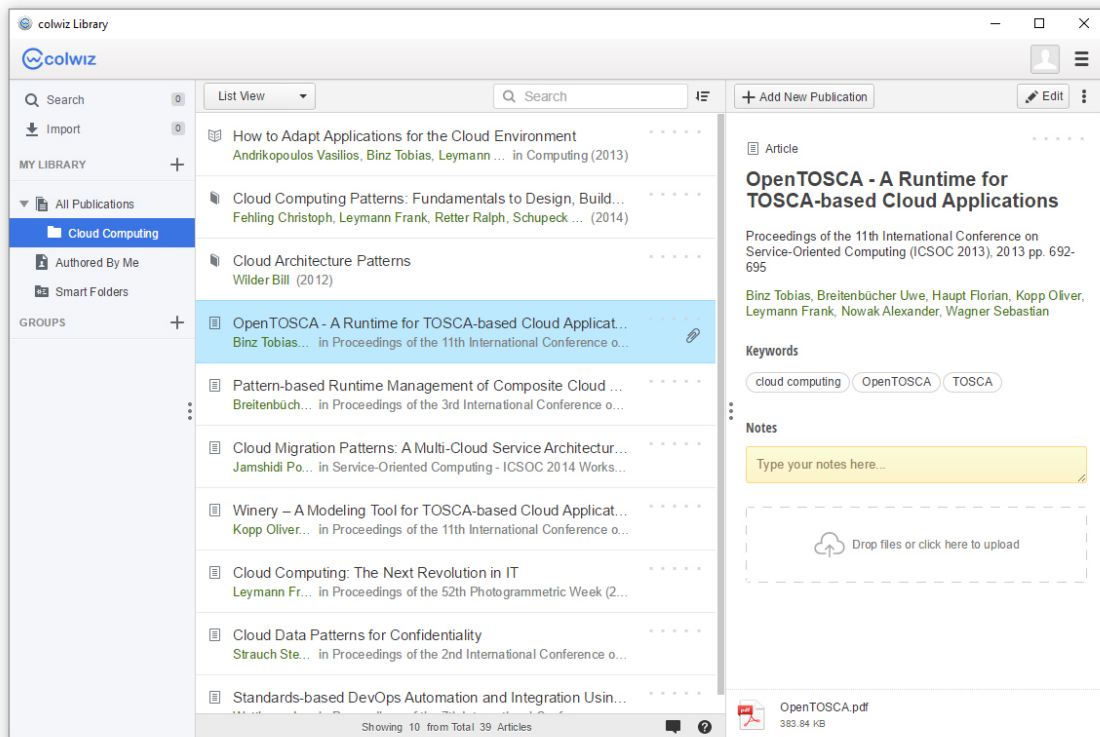


Figure 2.5: Colwiz 3.17 screenshot. At the center is a list view of references. The selected reference of this list is shown on the right-hand side in more detail. Additionally, on the left-hand side users can search and import literature and navigate inside their groups.

2.6.5 Citation and Literature Lists

Colwiz supports different citation styles and also the creation and adaption of styles [TUM16]. Dynamic creation of literature lists is supported for TeX editors, Microsoft Word, LibreOffice, OpenOffice [TUM16], and Google Docs. Besides, static literature lists are supported through the export.

2.6.6 Organization of Knowledge

For the organization of knowledge users can create notes, groups [TUM16], and keywords. In addition, smart groups are supported by the software.

2.6.7 Fulfillment of the Requirements

Colwiz fulfills the third group of the requirements completely if the group members have administrator rights. The other groups are only partly supported. The Requirements 1.1,

1.4 and 1.5 as well as Requirements 2.1, 2.2 and 2.5 are provided. Furthermore, Requirement 4.2 is fulfilled by an automatic duplicate check. 12 of 20 requirements are met by the application.

2.7 Docear

With the open source software Docear users can manage their literature and knowledge using a mind map [Doc17b]. It uses the open source tools Freeplane² for the mind map functionality and JabRef to manage the references. A web client is not available yet but a simple prototype is partly implemented [Doc17b]. The current version of Docear is 1.2.0.

2.7.1 Import and Export

In Docear users can import BibTeX files to new projects and export references to BibTeX, RIS, and other formats. There is no add-on to import references from web pages, the user has to copy the BibTeX source manually [Bee14]. In addition, currently there is no online search provided within the program. Besides, full-text search does not exist [Doc17b; Doc17c].

2.7.2 Data Input and Editing

The references can be added manually by the user and cross-referenced. Docear supports 19 document types but no definition for new ones. On the other hand, new entry fields can be defined by the user. Furthermore, files can be attached and comments extracted (see result at Figure 2.6, center) [Doc17c]. The software provides a metadata search on Google Scholar for PDF files. A duplicate check for references can be activated in the settings.

2.7.3 Data Display and Search

In Docear all bibliographical references can be displayed in a table view (see Figure 2.6, right) and sorted by clicking on a column heading. All fields of a single entry are visible in the edit mode where also a tab for the BibTeX source is available. At the lower right corner of Figure 2.6 the preview of the selected entry is shown and can also be printed. Docear provides a search history within one session and allows also to search in the mind map.

²<https://www.freeplane.org>

2 Related Work

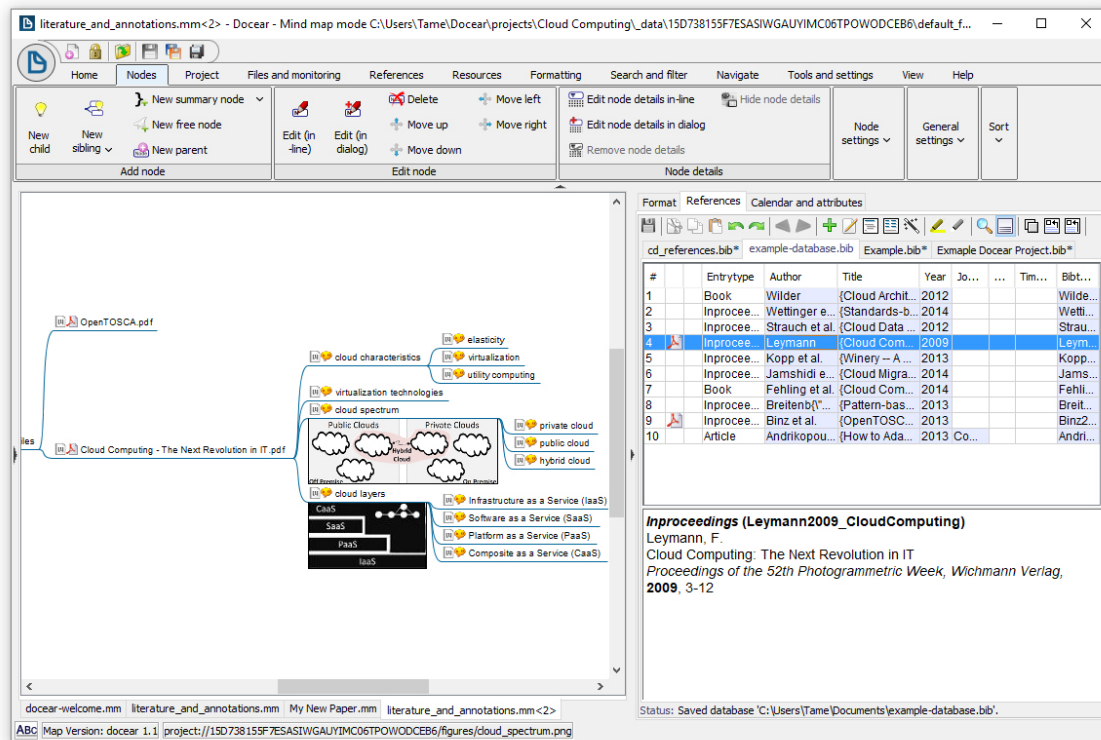


Figure 2.6: Docear 1.2.0 screenshot. On the left-hand side the mind map feature of the software is shown and on the right-hand side the stored references.

2.7.4 Cooperation

Users can copy their project [Doc17c] or the BibTeX file and share them with others but there is no support for collaborative work yet [Doc17b].

2.7.5 Citation and Literature Lists

Multiple citation styles are provided [Doc17a; Doc17b; Doc17c] and the creation of new styles is also possible [Doc17a]. Furthermore, a Microsoft Word add-on is available for Windows users and TeX editors are supported [Doc17b]. Literature lists can be created dynamically for them and static literature lists are also supported.

2.7.6 Organization of Knowledge

Knowledge of the user about the literature can be managed in Docear with a mind map feature. Comments can be extracted from PDF files and a manual creation of nodes is also possible. All nodes can be moved by drag and drop interactions and arranged hierarchically.

Furthermore, the user can add images and references to the mind map. The references itself can be categorized into groups and have keywords.

2.7.7 Fulfillment of the Requirements

All partial requirements of 1 are fulfilled. There are no different visibility levels, as everything is published private, thus only Requirement 2.1 is met of the second group of requirements. Docear does not support collaborative work hence all requirements of group 3 are not supported. Besides, the Requirements 4.1, 4.2 and 4.4 are met. Requirement 4.4 is realized with an auto-complete functionality for different fields like author. In summary, Docear fulfills 9 of the 20 requirements.

2.8 EndNote

EndNote X8 is a commercial desktop reference management software [AMSW16; TUM16]. There also exists a free version called EndNote basic which is a web application with limited functionality, references, and storage capacity. In the following, only version X8 is considered.

2.8.1 Import and Export

EndNote provides database search with Crossref and many other search engines within the program (see Figure 2.7, left). Additionally, there exists a tool to capture references from websites [AMSW16; TUM16]. The import of EndNote supports RIS files and many other file formats [TUM16], but no BibTeX. The export on the other hand supports RIS and BibTeX in addition to other formats [TUM16]. A full text search is available via OpenURL [AMSW16; TUM16].

2.8.2 Data Input and Editing

New references also can be entered manual by the user [AMSW16]. There are 51 different document types supported by the program [AMSW16; TUM16] and the user can define new document types [TUM16] and entry fields [AMSW16; TUM16]. Multiple references, e.g. translations or reprints, cannot be interlinked in EndNote. Furthermore, metadata can be added from PDF files with DOI information and via the update reference button of the software [TUM16]. Attachment of files is also provided by EndNote and PDF files and their annotations can be viewed and edited within the program (see Figure 2.7, bottom) [AMSW16; TUM16]. There is also a duplicate detection for references available.

2 Related Work

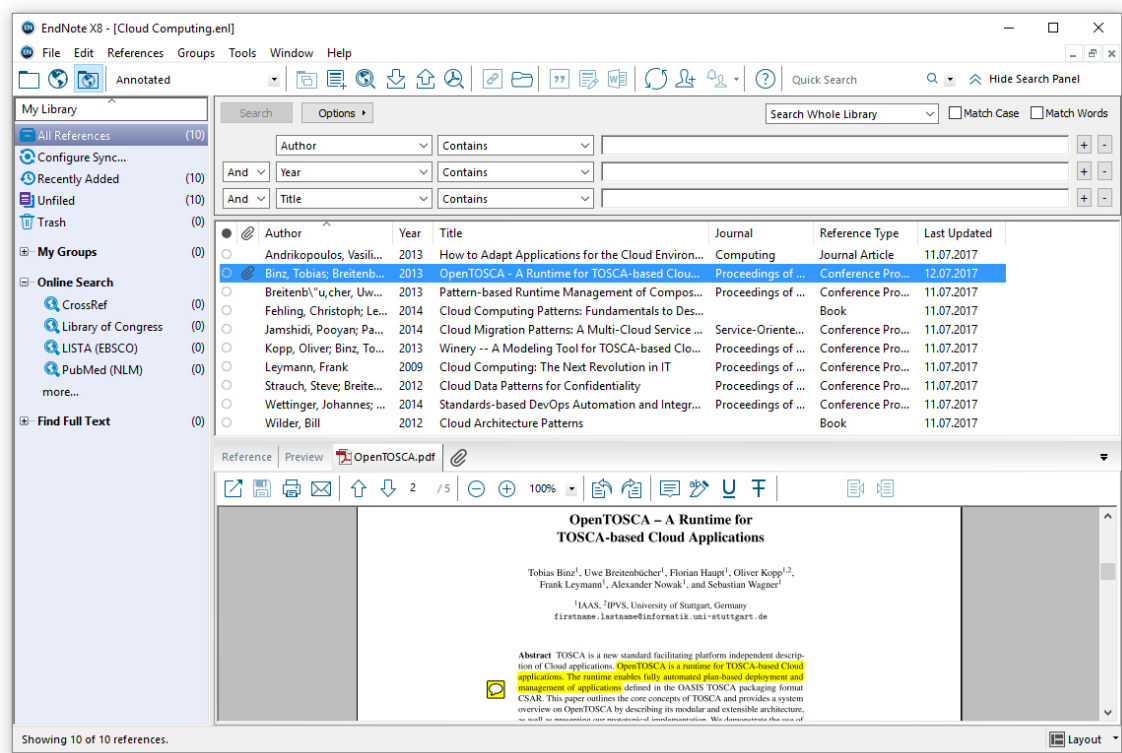


Figure 2.7: EndNote X8 screenshot. At the upper part of the center the references are listed. Below the PDF file of the selected reference is shown.

2.8.3 Data Display and Search

Multiple references can be viewed in a table (see Figure 2.7, center) and also a view of all fields of a reference is available [AMSW16; TUM16]. In addition, EndNote shows a reference with the selected citation style in the preview tab. In the table view the rows can be sorted by clicking on the column headings. The software also provides a search history.

2.8.4 Cooperation

A library can be shared in EndNote X8 with up to 100 users but an online account is required for each user [End17]. With that account the user has to synchronize his library first and share it with others afterwards. Each member of a shared library can add references, comments, and files. EndNote neither has a limited library size which can be shared nor does the users have to make any additional payments to use this functionality.

2.8.5 Citation and Literature Lists

Many citation styles are installed with EndNote X8 and the user can download more from the website³ [AMSW16; TUM16]. There are over 6000 styles in total. It is also possible to create a new citation style or to adapt an existing one. EndNote provides add-ins for Microsoft Word and OpenOffice. The literature lists can be created static or dynamic.

2.8.6 Organization of Knowledge

Knowledge can be organized in EndNote with tags, notes [AMSW16], and groups of references. The group functionality also supports smart groups where the user for example can define which word should be contained in the title. All matching references from the library are added automatically to this group.

2.8.7 Fulfillment of the Requirements

The requirements of the first and third group are completely fulfilled by EndNote. However, the requirements of the other groups are only partly met. The Requirements 2.1 and 2.2 as well as Requirements 4.2 and 4.4 are provided. Overall, 14 of 20 requirements are fulfilled.

2.9 F1000Workspace

F1000Workspace is a commercial reference manager on the web which offers different types of accounts [F1017a]. There is also a free subscription with amongst other things limited storage and projects.

2.9.1 Import and Export

The software provides import and export of BibTeX, RIS, and other file formats [F1017a; F1017b]. Furthermore, importing directly from websites is supported through an add-on for the browser [F1017a]. However, there is no full-text search and database search within F1000Workspace but the user can insert references via DOI, ISBN, and other identifiers.

³Citation styles for EndNote: <http://endnote.com/downloads/styles>

2 Related Work

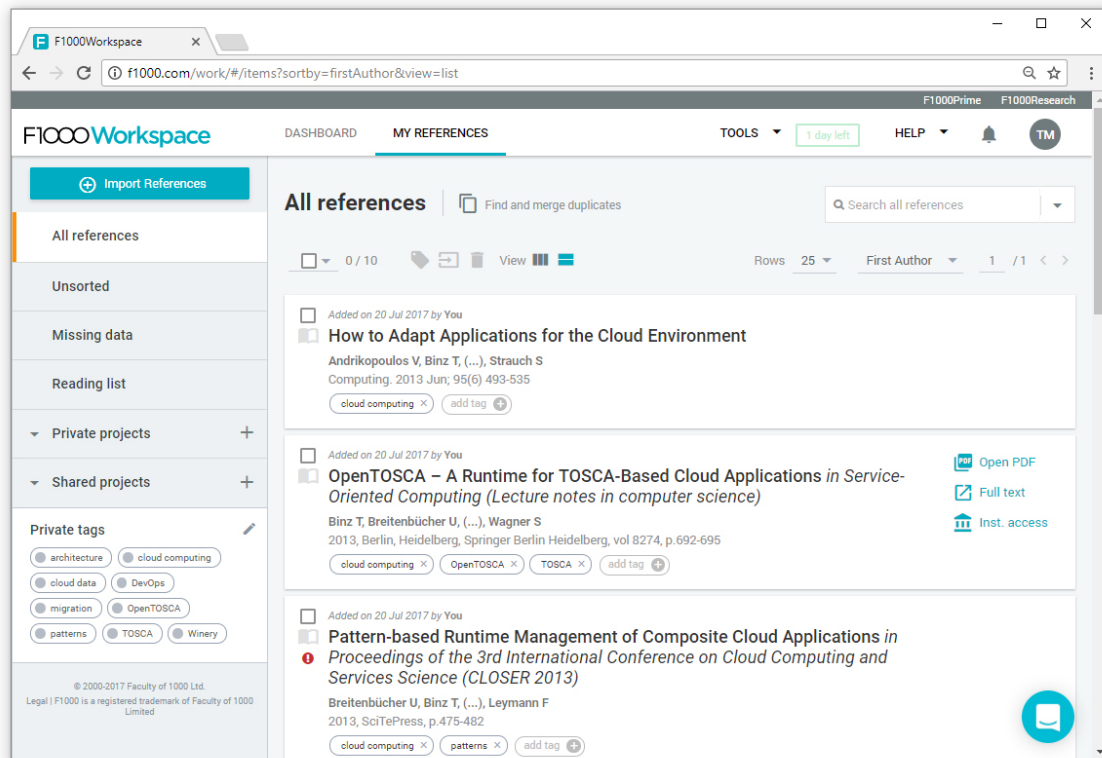


Figure 2.8: F1000Workspace screenshot. References are listed at the center of the screen. On the left side the user can navigate into projects where also collaborative work is supported.

2.9.2 Data Input and Editing

Adding references can also be done manually by the user [F1017a]. The program supports 11 document types but no definition of new types or entry fields. References with missing information are marked with a red exclamation point (see Figure 2.8, third entry) and the missing metadata can be added via DOI, ISBN, and further options [F1017b]. Furthermore, documents can be uploaded and a duplicate check is available [F1017a; F1017b]. Besides, processing and evaluation of full texts is possible with a browser add-on [F1017a].

2.9.3 Data Display and Search

F1000Workspace displays multiple references in a list view (see Figure 2.8, center) which can also be switched to a configurable table view. There are different sorting options for the data displays available [F1017a]. By clicking on a reference it is shown in a short overview with the most important information. All entry fields can be seen in the edit mode. Furthermore, F1000Workspace provides a search history for the users.

2.9.4 Cooperation

Users can create shared projects and invite others to them [F1017a]. All members are allowed to add, edit, and delete references. Additionally, they can read notes from others and insert their own. Only private notes are not visible to other project collaborators.

2.9.5 Citation and Literature Lists

F1000Workspace provides multiple citation styles [F1017a]. However, an adaption or definition of new styles is not supported. Microsoft Word and Google Docs can be used with static literature lists which can be updated through a button click.

2.9.6 Organization of Knowledge

For the organization of knowledge F1000Workspace provides notes, tags, and project groups for references [F1017a]. The projects can also have subprojects.

2.9.7 Fulfillment of the Requirements

The requirements of group 1 and 3 are completely fulfilled. Requirements 2.1, 2.2, 2.5, 4.1 and 4.2 are also met. In summary, 15 of 20 requirements are fulfilled by F1000Workspace.

2.10 JabRef

JabRef is an open source software for managing bibliographical references [AMSW16; TUM16]. It is written in Java and thus a platform independent desktop application. This section analyses the features of JabRef version 4.0.0.

2.10.1 Import and Export

BibTeX, RIS and many other formats are supported by the import and export of the program [TUM16]. An add-on for Firefox⁴ allows to import references directly from the browser. Furthermore, new references can be imported via web search within JabRef (see Figure 2.9, left) [AMSW16; TUM16]. Supported search engines are arXiv, Crossref, DBLP, Google Scholar [Jab17a], IEEE Xplore, Springer, and others [Jab17a; TUM16]. Besides, a full text search is available in the software [AMSW16; TUM16].

⁴Firefox add-on JabFox: <https://addons.mozilla.org/de/firefox/addon/jabfox/>

2 Related Work

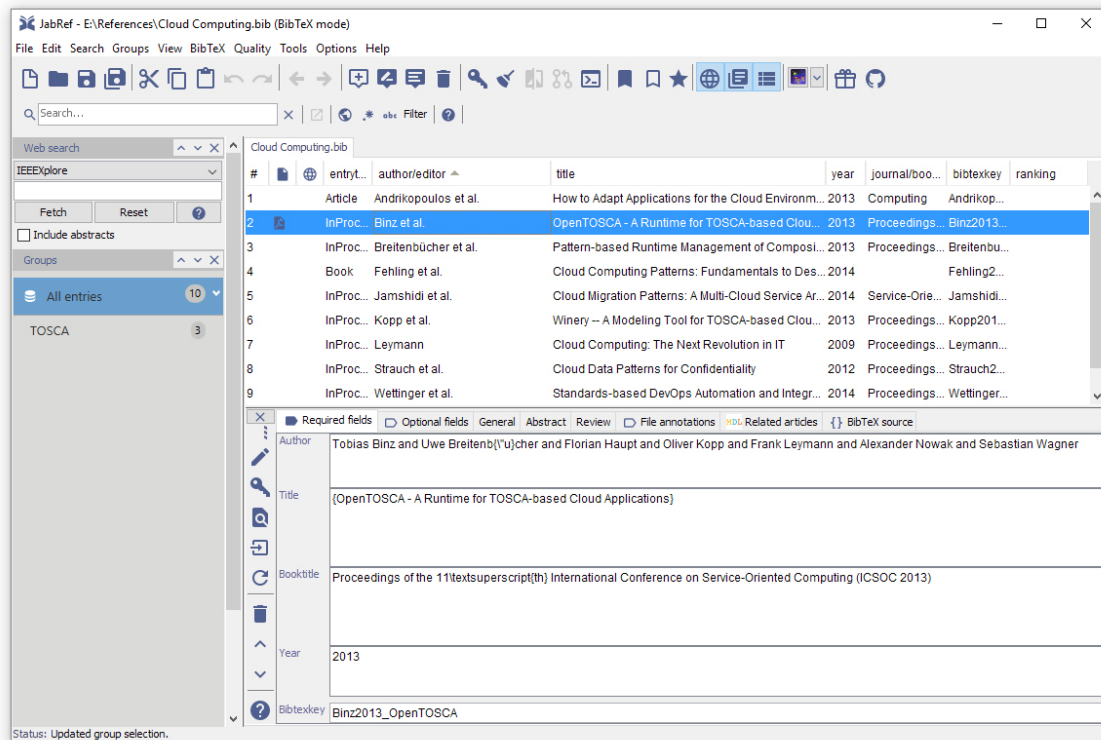


Figure 2.9: JabRef 4.0.0 screenshot. On the left side is a search bar to access the supported web searches. In the center the references are listed in a table. Below that table the currently selected reference entry is shown in detail.

2.10.2 Data Input and Editing

In addition to the import options, the user can add a reference manually [AMSW16]. There are 19 available document types [AMSW16; TUM16] and the user can define own types [TUM16] and entry fields [AMSW16; TUM16]. Furthermore, references can be linked in JabRef with the *Crossref* field and file attachments of many formats are supported. Metadata can be added via DOI, ISBN, and from the metadata of PDF files [TUM16]. A duplicate check is also available [AMSW16; TUM16]. Processing and evaluation of full texts is not supported by JabRef.

2.10.3 Data Display and Search

Figure 2.9 shows the main view of JabRef which consists of a table view in the center. The columns can be configured by the user and the table can be sorted by clicking on the column headings. At the bottom of Figure 2.9 all fields of the selected entry are shown in multiple tabs and can be edited. The last tab shows the BibTeX source code of the reference. If the full view of an entry is closed the user can see a short preview of the

entry or the reference printed in the selected citation style. JabRef also provides a search history [AMSW16; TUM16].

2.10.4 Cooperation

Users can share the BibTeX file or a SQL database with other users [AMSW16; Jab16b]. A shared SQL database allows collaborative work with JabRef [Jab16a]. A prerequisite to use this feature is a remote database. The user can choose between PostgreSQL, MySQL, and Oracle as database type. Changes of the reference library are automatically updated and merged if possible.

2.10.5 Citation and Literature Lists

Different citation styles are supported by JabRef including individual definition and adaptation of styles [AMSW16]. The word processing programs Microsoft Word [TUM16], OpenOffice [AMSW16], and LibreOffice are supported as well as TeX editors [AMSW16; TUM16]. The creation of literature lists can be done statically or dynamically [AMSW16].

2.10.6 Organization of Knowledge

For the organization of knowledge the software provides creation of hierarchical groups of references [TUM16], smart groups, tags [AMSW16], and special fields [Jab16c]. The special fields allow rating literature, and setting a priority and a read status. The user can also toggle the fields relevance, quality assured, and print status. In the settings of JabRef the user can choose if the special fields should be synchronized with keywords or written to separate fields of the BibTeX source [Jab16c].

2.10.7 Fulfillment of the Requirements

JabRef fulfills completely the groups 1 and 3 of the defined requirements. Furthermore, the Requirements 2.1 and 2.3 as well as the Requirements 4.1, 4.2, 4.4 and 4.5 are met. Altogether, 16 of the 20 requirements are supported.

2.11 RefWorks

RefWorks is a web application [TUM16]. The version 2.0 is the legacy software of RefWorks [Ref17d] and has two different types of accounts: individual subscriber and organizational subscriber [Ref17c]. An account via an organization enables additional features like uploading files [Ref17a] and cooperative work [TUM16].

2 Related Work

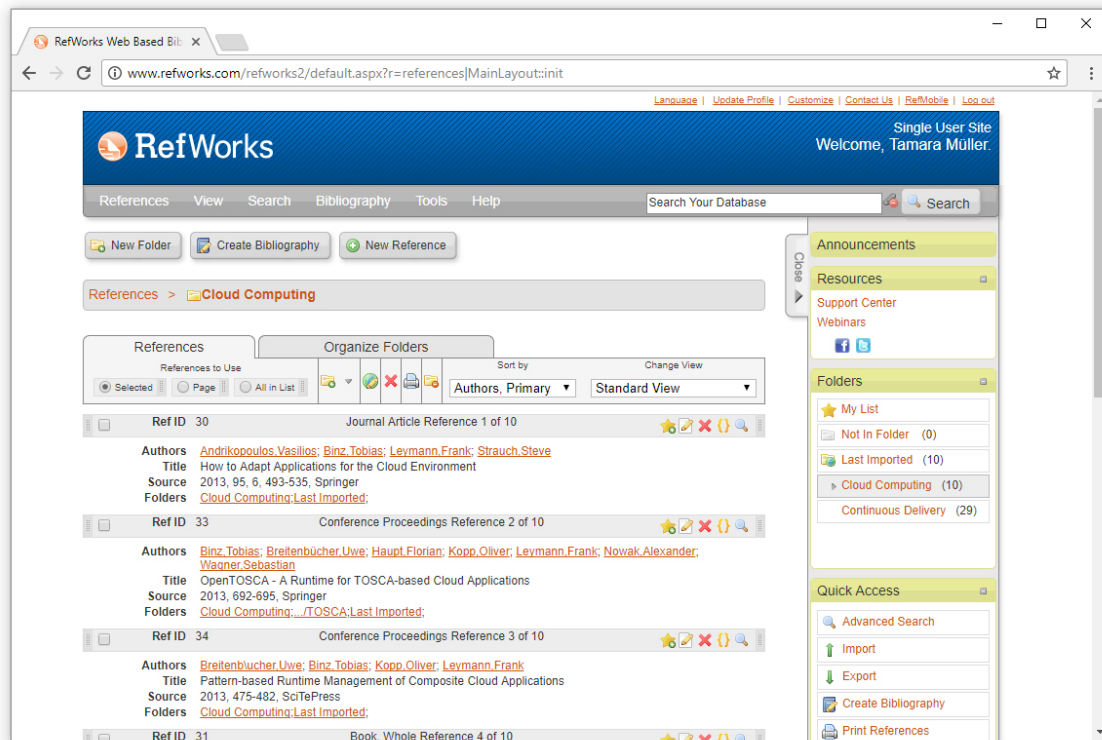


Figure 2.10: RefWorks 2.0 screenshot. The references are listed in the center of the window. On the right-hand side several functionalities are provided like importing and exporting references or to navigate into folders of references.

2.11.1 Import and Export

The import and export support BibTeX, RIS and many other formats [TUM16]. Furthermore, a database search is available within the software and an import from web pages by use of the RefGrab-It tool [Ref17b; TUM16]. With an individual subscription there is no access to full-text search [TUM16]. For that purpose an organizational account is required and the organization has to configure the full-text search [Ref17f].

2.11.2 Data Input and Editing

The user can also add references by hand. RefWorks supports 31 different document types and the definition of new entry fields [TUM16]. However, the user cannot create own document types and link references. For adding documents an organizational account is required [Ref17a]. In addition, processing and evaluation of full texts is not supported [TUM16]. On the other hand, checking for duplicates is provided.

2.11.3 Data Display and Search

Different views are available in RefWorks [Ref17f; TUM16]. The standard view (see Figure 2.10, center) is a list view of multiple references. The user can switch to a short or full view where less or all information is shown to each reference. All data displays can also be sorted and there exists a history for search queries [TUM16].

2.11.4 Cooperation

Sharing records with other users and cooperative work is not supported with an individual account. Therefore an organizational subscription is required [TUM16].

2.11.5 Citation and Literature Lists

The application provides multiple citation styles and allows the users to edit them or create new ones from scratch [TUM16]. There exists a Write-N-Cite add-in for Microsoft Word which can insert reference lists dynamically. Besides, the creation of static literature lists is possible.

2.11.6 Organization of Knowledge

To manage the knowledge users can group references hierarchically [TUM16], add notes, and write into additional free text fields.

2.11.7 Fulfillment of the Requirements

The Requirements 1.1, 1.4 and 1.5 are fulfilled. Furthermore, the user can add references, files, and comments to his private library and share a folder or the library via URL with others or with the entire organization if he uses an organizational subscription [Ref17e]. In the settings of a shared folder the user can enable comments of other persons but they are not allowed to edit references. So the Requirements 2.1, 2.2, 2.3, 3.1, 3.2, and 3.5 are fulfilled. Furthermore, the Requirements 4.2 and 4.4 are realized in the software. Overall, 11 of 20 requirements are met by RefWorks.

2.12 Comparison of the Reference Management Tools

The tools in the previous sections provide different functionalities and are developed for different scenarios. The Tables 2.1 and 2.2 give an overview of the features supported by each software. Most of the tools support BibTeX and RIS to import and export references. This allows to transfer references from one tool to another, so there is no vendor lock-in. Organization of knowledge is similar at the most tools. They provide tags, notes, and groups to organize knowledge on literature. Docear is the only software which additionally allows to organize the information in a mind map. However, Docear is the only tool which does not support collaborative work.

The fulfillment of the requirements from Section 2.1 are listed for each software in Table 2.3. JabRef meets the most requirements with 16 of 20, followed by CiteULike and F1000Workspace which support one requirement less. None of the programs fulfills all requirements for quality assurance of references. Each provides some mechanism for duplicate detection but the other partial requirements are only implemented rarely. Hence, the next chapter introduces a concept for a reference management software which supports all of the requirements. The realization of the collaborative and quality assurance requirements are especially considered.

2.13 Other Related Work

The focus of this chapter is on reference management tools but there are also other related works. Sturm and Sunyaev [SS17a; SS17b] research a system for systematic literature search. First, they introduce the requirements for such a kind of system – comprehensiveness, precision, and reproducibility. Afterwards, design principles are developed and also a prototype [SS17a; SS17b; SSS15]. Beel and Langer [BL15] as well as Adomavicius and Tuzhilin [AT05] analyze approaches for recommendation systems which show users related articles in their research field. Such systems are for example used in JabRef [FSG+17] and Docear [BLGN13].

Furthermore, there are other tools which can support people in scientific work. One example is Hypothesis⁵ – an open source tool for annotations on web pages [Hyp17a]. It does not need any implementation of the page owner to allow annotating a site, it uses a layer on top of a website to provide this functionality. This enables annotating each page on the web like a news article, a blog entry or a book. Everyone can create a free account and add Hypothesis to the browser either by installing an add-on for Google Chrome or through adding a button to the bookmarks of the browser [Hyp17c]. Furthermore, Hypothesis supports collaborative work [Hyp17b]. Other people can be invited to cooperate by sharing a link. There are different types of annotations – highlights, text notes, page notes, and replies [Ude16]. Highlighted text passages are only visible for the user himself, they cannot

⁵<https://hypothes.is>

be shared. The text notes on the other hand are highlighted text with an additional note. They can be tagged and published on different visibility levels. Page notes are similar to text notes but do not refer to a text passage instead they provide notes for the entire web page. With a reply a user can comment a note or another reply. Besides, tags and visibility levels are supported by replies. However, Hypothesis does not provide a functionality to store references.

2 Related Work

Software	Open Source	Web client	Import filter	Export	Full-text search	Link references
Aigaion	yes	yes	BibTeX, RIS, others	BibTeX, RIS	no	yes
BibSonomy	yes	yes	BibTeX, RIS, others	BibTeX, RIS, others	yes	no
Citavi	no	no	BibTeX, RIS, others	BibTeX, RIS, others	yes	yes
CiteULike	no	yes	BibTeX, RIS	BibTeX, RIS, others	no	no
Colwiz	no	yes	BibTeX, RIS, others	BibTeX, RIS, others	yes	no
Docear	yes	no	BibTeX	BibTeX, RIS, others	no	yes
EndNote	no	yes	RIS, others	BibTeX, RIS, others	yes	no
F1000Workspace	no	yes	BibTeX, RIS, others	BibTeX, RIS, others	no	no
JabRef	yes	no	BibTeX, RIS, others	BibTeX, RIS, others	yes	yes
RefWorks	no	yes	BibTeX, RIS, others	BibTeX, RIS, others	yes (L)	no

Table 2.1: Comparison of the reference management tools from the previous sections.

Processing and evaluation of full texts	Sharing records with other users	Cooperative use	Self-definition or adaptation of citation styles	Organization of knowledge
no	yes	yes	no	groups, notes, tags
no	yes	yes	no	notes, tags
yes	yes	yes	yes	free text fields, groups, notes, tags
yes	yes	yes	no	groups, notes, tags
yes	yes	yes	yes	groups, notes, tags
yes	yes	no	yes	groups, mind map, notes, tags
yes	yes	yes	yes	groups, notes, tags
yes	yes	yes	no	groups, notes, tags
no	yes	yes	yes	groups, special fields, tags
no	yes (L)	yes (L)	yes	groups, free text fields, notes

Table 2.2: Comparison of the reference management tools from the previous sections (cont.). Full table available at <https://ultimate-comparisons.github.io/ultimate-reference-management-software-comparison>.

Software	Requirements																			
	1.1	1.2	1.3	1.4	1.5	2.1	2.2	2.3	2.4	2.5	3.1	3.2	3.3	3.4	3.5	4.1	4.2	4.3	4.4	4.5
Aigaion	✓	✗	✗	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✗	(✓)	✗	✓	✗
BibSonomy	✓	✓	✗	✗	✗	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✗	(✓)	✗	✗	✗
Citavi	✓	✗	✗	✓	✓	✓	✓	✗	✗	✗	✓	✓	✓	✓	✓	✗	✓	✓	✓	✗
CiteULike	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	✗	✓	✗	✗	✗
Colwiz	✓	✗	✗	✓	✓	✓	✓	✗	✗	✓	✓	✓	✓	✓	✓	✗	✓	✗	✗	✗
Docear	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✗	✓	✗
EndNote	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗	✓	✓	✓	✓	✓	✗	✓	✗	✓	✗
F1000Workspace	✓	✓	✓	✓	✓	✓	✓	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗
JabRef	✓	✓	✓	✓	✓	✓	✗	✓	✗	✗	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓
RefWorks	✓	✗	✗	✓	✓	✓	✓	✓	✗	✗	✓	✓	✗	✗	✓	✗	✓	✗	✓	✗

Table 2.3: Fulfillment of the requirements defined in Section 2.1 by the reference management tools from the previous sections.

3 Concept

The previous chapter evaluated 10 reference management tools against 20 requirements for such software that supports collaborative work and quality assurance of references. It shows that none of the programs fulfills all requirements. Therefore, this chapter introduces a concept for a new system which aims to cover all requirements (Sections 3.1 to 3.4). Finally, in Section 3.5 the implemented platform “CloudRef” gets presented and evaluated against the requirements of Section 2.1.

3.1 Collaborative Work

For the work with bibliographical references in a group of multiple persons like at scientific work it is necessary to share a list of references with others. Therefore, a group functionality can be used which allows every user to create groups and invite other users into them. Inside such a group all members can add references and PDF files. To store and exchange knowledge about the literature the system should provide a comment functionality and a mechanism to share these comments with others. Different visibility levels for the comments can achieve this. They allow the users to write private notes only visible for themselves or to share a comment with other selected users, multiple groups, or all users of the system. The users should decide for each comment separately on which level of visibility it gets published.

Comments are often written in the PDF file of a reference. In a single user system they can be integrated by showing the PDF and the corresponding comments in a frame inside the reference management program similar as in a PDF reader. However, the concept of this chapter aims to provide a system that supports collaborative work, hence this is not enough because every group member can have own notes in a separate PDF file. The comments have to be extracted from the PDF files so that the users do not have to copy their comments to the platform. A tool which can be used for this is Scimappr¹. Furthermore, the users should be able to decide for each comment inside the PDF on which visibility level it gets published. However, it should also be possible to show an uploaded PDF file for a reference inside the software so that not every user has to download the corresponding file and show it next to the system to work efficiently. Comments are not saved at a PDF file, because users can create private ones which nobody except themselves should see. The separate storage also allows filtering comments for example to show all comments of a user.

¹<https://github.com/koppor/scimappr>

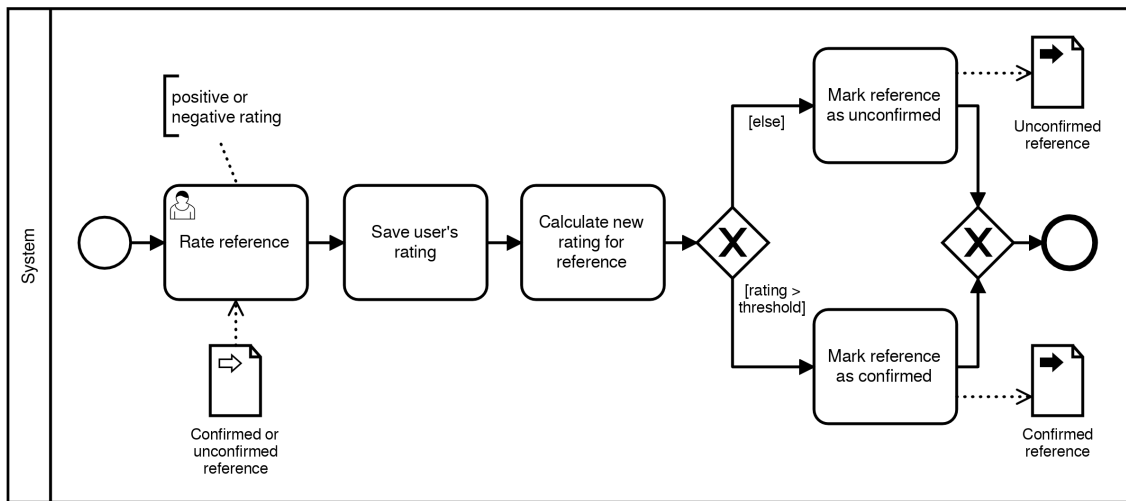


Figure 3.1: BPMN process of rating a reference either positive or negative. If a threshold is reached the reference gets marked as confirmed otherwise it is unconfirmed. The state confirmed is an indicator that the quality of the reference is good.

3.2 Quality Assurance of Bibliographical References

Bibliographical references are often incomplete or faulty [Kop16]. If references are used for a publication it is important that the required fields are provided in the correct way. For example, ACM asks for a page number which is not needed for IEEE, and IEEE has different journal abbreviations than others. A prerequisite to provide a correct bibliography after the guidelines of the publishers is to have complete and correct reference entries.

To ensure a good quality of the bibliographies in the reference management platform a rating system is used. The rating system allows the users to rate a reference positive, for example if they think all necessary information is provided and the data is correct. A reference can also be rated negative, for example if the user thinks something is missing or faulty. A flow of this activity is depicted in Figure 3.1. Afterwards a user rates a bibliographical reference the system calculates the new rating of this reference. Initially the total rating is zero. Each positive vote adds one to the sum and each negative subtracts one. If a positive threshold is reached, the reference is classified as complete and correct, and gets marked as confirmed. This confirmed status means that many users rated this reference positive which is an indicator that the quality of this entry is good. To provide such a quality assurance all users of the system must work on the same references which can be voted by everyone. Furthermore, the references require unique BibTeX keys in order to be clearly identified and to provide an export of all entries in a single, valid .bib file.

Furthermore, if a user detects a mistake at a bibliography he should be able to edit the reference in the system whether the reference was marked as confirmed before or not. The changes do not overwrite the old information immediately but get published as suggestion for modification. All users can rate such suggestions positive or negative (see Figure 3.2) like the rating of a reference explained before. After each rating of a suggestion the system

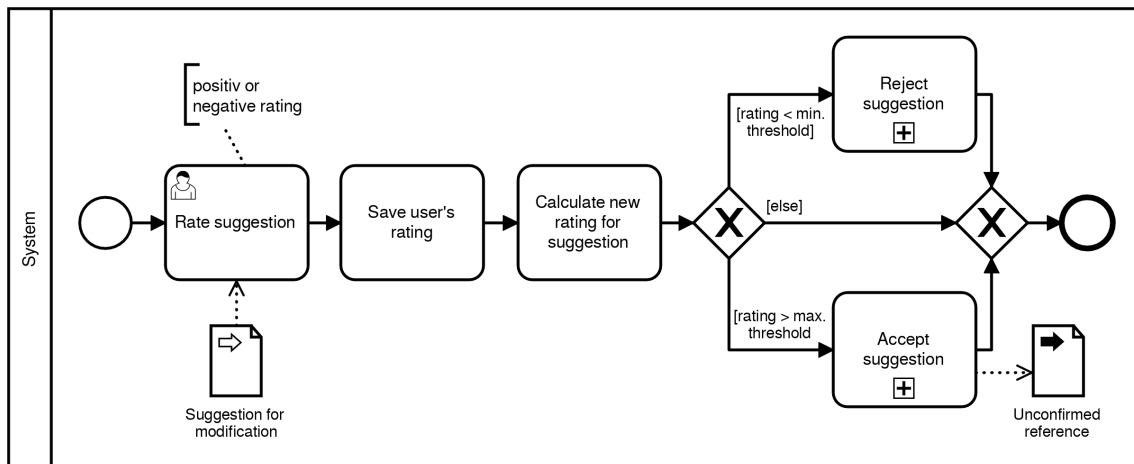


Figure 3.2: BPMN process of rating a suggestion for modification of a reference. If a positive or negative threshold is reached the suggestion gets either accepted or rejected otherwise it stays in the system as suggestion.

calculates the sum of all votes and if the result is lower than a given minimum threshold the suggestion gets rejected. On the other hand, if it is over another maximum threshold, the suggestion is accepted by the users which means that the old reference entry adopt the changes. If the total rating is between these both threshold the suggestion stays unaltered in the system accessible for further ratings of users.

If a suggestion fixes for example a spelling mistake it would be useful to have a mechanism to accept these change faster than with the rating system. For this purpose a special user role is introduced – the maintainer. As illustrated in Figure 3.3 he can accept or reject a suggestion directly and is also allowed to edit a suggestion. If a maintainer edits a suggestion for modification the previous ratings of the users are discarded automatically by the system because they refer to an outdated version of the suggestion.

For reference management tools it is important to have a duplicate detection because users fill in input fields differently [HJSS06]. Thus, a reference can be added multiple times to the system with slight variations. Duplicate detection can be done with an edit distance algorithm as in JabRef [Jab17b]. The fields author, editor, title, and journal have more weight than the others to improve the accuracy of detecting real duplicates. Furthermore, to support consistent notations of authors, conferences, and abbreviations the data has to be stored so that it can be reused in new reference entries. A autocomplete functionality at the user interface allows the users to select previous information or to add new data. To avoid multiple author entries of the same person, for example one entry with the middle name and one without, additional information can be saved. Many publications provide the e-mail addresses of the authors which can be used to identify a person. Because an e-mail address can change – for example if the person changes the company – multiple addresses should be able to be added. Likewise, different abbreviations can be stored for a text. This provides the opportunity to define other abbreviations for IEEE which can be used for the export of a bibliography to match the publisher’s guidelines.

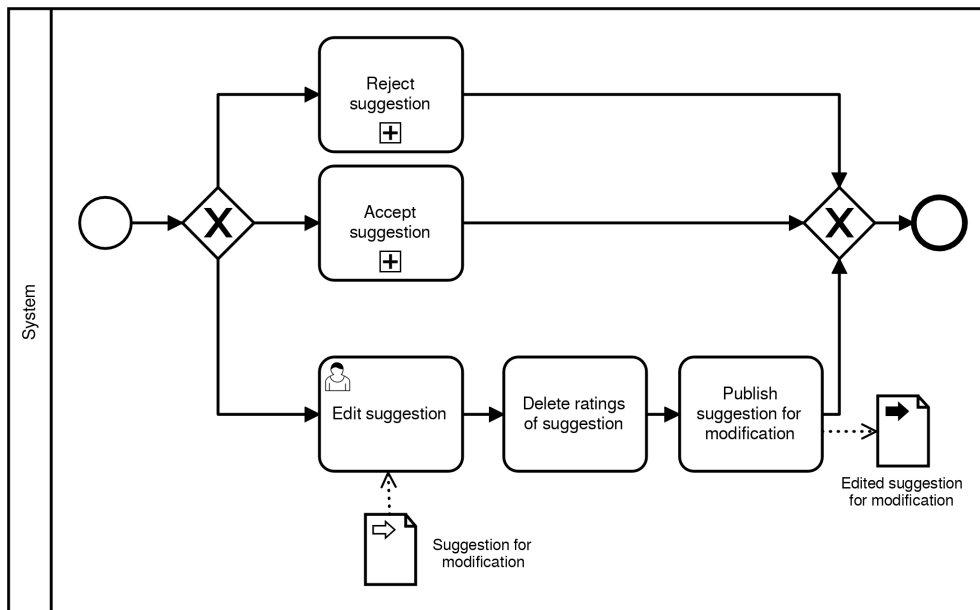


Figure 3.3: A maintainer has additional options on a suggestion for modification. He can induce that a suggestion is applied or rejected directly and he is also able to edit a suggestion.

3.3 Managing Bibliographical References

The references which are stored in CloudRef are saved each in a separate `.bib` file on the hard disk because a single file with all reference can get very huge. Furthermore, the references are under version control to support traceability of changes made by suggestions to modify a reference. References which are inserted in the system are added inside a new `.bib` file to the master branch. There is no validation of the completeness or correctness of the reference. Each suggestion for modification is stored at a new branch of the repository and if it is accepted or rejected the branch gets merged into the master branch, where all references except suggestions are stored.

Figure 3.4 shows an example of a Git repository generated by CloudRef. On the left side the branches are visualized with the commits on each branch as a circle. The commits are sorted by the time they were created, with the first commit at the bottom. The corresponding commit information is printed to the right of a circle. The first value inside the brackets is the name of the branch, the following text to the hyphen is the commit message, and the text after the hyphen the author of the commit. The committer – the one who applies the commit – is always the CloudRef system. Below this information the added and modified files of the commit are listed, with the added files marked by a green square and the changed ones with a yellow square. The blue branch stands for the master branch and the first commit there adds a reference with the BibTeX key “test”. The following two commits change these reference, each on a separate branch because they are two different suggestions by users. The fourth commit changes the first suggestion for the reference

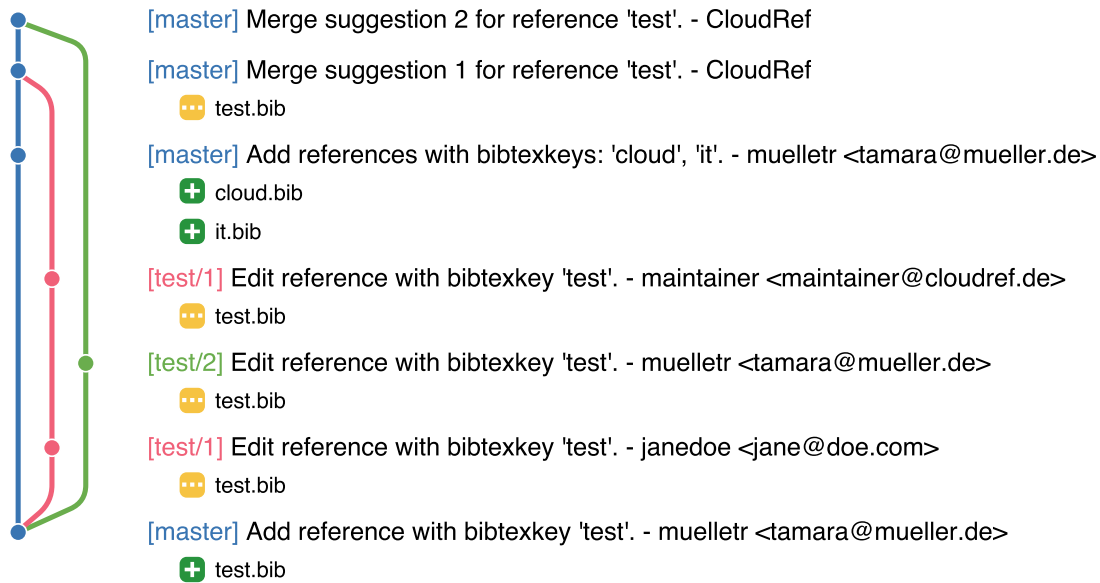


Figure 3.4: Git repository example of CloudRef. On the left side the branches are visualized with the commits as circles. To the right of each circle more information about the commit is provided. Added files are indicated with a green square and modified ones with a yellow square. The commits are sorted by the time they were created with the first commit at the bottom.

with the BibTeX key “test”. The user who made this change has the user role maintainer because only such users are allowed to modify suggestions (see Section 3.2). The next commit illustrates an import of multiple references from a .bib file. This is the only case where multiple files are added. Afterwards, the first suggestion is accepted by the users and the corresponding branch “test/1” is merged from the CloudRef system into the master. The last commit of the figure shows how a suggestion is rejected by the users. The branch “test/2” is merged into the master but no files are modified because the changes of the branch are rejected and are not allowed to change anything on the master.

3.3.1 Merging Suggestions

For merging a rejection of a suggestion the “ours” strategy of Git can be used which ignores all changes made on the branch and uses only the tree of the master [Git17]. A correct merge of an accepted suggestion takes all changes made on the branch by a user and applies them to the reference on the master because the suggestion was accepted by the users and should replace the current reference. Also if a field was meanwhile changed at the master if it was changed on the branch the version of the branch should be preferred. However, if a field was not changed at the branch but meanwhile on the master the version of the master should be taken. Accepted suggestions cannot always use a available merge strategy that results in a correct merge without conflicts in the sense of the CloudRef application. This is explained by a simple example in the following.

3 Concept

Listing 3.1 Content of a .bib file which contains a reference with the BibTeX key “test-merge”.

```
@article{test-merge,
  title = {Test reference to show merge strategies.},
  year = {2017},
  keywords = {test, merge strategy}
}
```

Listing 3.1 shows a reference which has a type, BibTeX key, title, year, and keywords. In Listing 3.2 an author is added to this reference as suggestion to modify the reference and Listing 3.3 shows another suggestion where the month is added. Both suggestions are on a different branch of the Git repository. The first suggestion of Listing 3.2 can be merged with the “recursive” strategy – the default strategy for merging a branch in Git [Git17] – without generating any conflict because meanwhile the reference on the branch has not changed. The file content of the reference on the master will look like at the branch before. On the other hand, if we afterwards merge the other suggestion of Listing 3.3 with the same strategy into the master we get a conflict because the reference on the master was changed by the first suggestion. Git does not know which of the inserted lines it should take because they are at the same location of the file.

Listing 3.2 Suggestion to modify the reference with the BibTeX-key “test-merge”. The author is added to the reference.

```
@article{test-merge,
  title = {Test reference to show merge strategies.},
+  author = {Tamara Mueller},
  year = {2017},
  keywords = {test, merge strategy}
}
```

Listing 3.3 Suggestion to modify the reference with the BibTeX-key “test-merge”. The month is added to the reference.

```
@article{test-merge,
  title = {Test reference to show merge strategies.},
+  month = {oct},
  year = {2017},
  keywords = {test, merge strategy}
}
```

To resolve merge conflicts automatically, Git provides the opportunity to add the option “ours” or “theirs” to the recursive strategy which prefers the chosen version if conflicts occur [Git17]. The result of merging the second suggestion with this strategy and the option “theirs” is shown in Listing 3.4. The line which contains the author from the first suggestion we merged first gets replaced by the line of the second suggestion which contains the month. This is because we prefer from these two conflicting lines the version of the branch which contains the second suggestion. Hence, this strategy neither solves the merge problem of CloudRef. A correct result of merging both suggestions one after the other is shown in Listing 3.5. There are both new fields – author and month – added to the reference entry.

The order of the fields is not important, for example if the month is listed before the author it would also be a correct merge. To achieve this result, an own algorithm for resolving merge conflicts must be used because out of the box solutions from Git do not work like presented before. How this algorithm works is explained in the following.

Listing 3.4 Merge of the second suggestion with the merge strategy “recursive” and the option “theirs” of Git.

```
@article{test-merge,
  title = {Test reference to show merge strategies.},
-  author = {Tamara Mueller},
+  month = {oct},
  year = {2017},
  keywords = {test, merge strategy}
}
```

Listing 3.5 Correct merge of the second suggestion.

```
@article{test-merge,
  title = {Test reference to show merge strategies.},
  author = {Tamara Mueller},
+  month = {oct},
  year = {2017},
  keywords = {test, merge strategy}
}
```

Listing 3.6 shows the content of the `test-merge.bib` file after merging the second suggestion with the merge strategy “recursive” of Git. The file contains the occurred conflict. The beginning of a conflicting region is labeled in Git with “<<<<<<” followed by the version of the first input tree which is in our case the master branch [Git17]. Then a separation of the first and second input tree is indicated by “=====” followed by the version of the second input tree. Afterwards, “>>>>>>” marks the end of the conflict. At the example in Listing 3.6 the lines 1, 2, 8, 9, and 10 do not contain a conflict because they are the same at both suggestions. On the other hand, the lines 3 to 7 contain a conflict where line 4 presents the version at the master branch and line 6 the version of the branch of the second suggestion. This information is used by CloudRef to resolve the merge conflicts through an algorithm. The idea is to use all lines of the `.bib` file where no conflicts are. In conflicting areas all parts of the branch are used if they were edited in the suggestion for modification. Additionally, all fields of the reference which are on the master and not modified on the branch are taken because they come from meanwhile changes of the reference entry through other accepted suggestions.

Algorithm 3.1 depicts the pseudo code for resolving the merge conflicts. At line 3 all fields which are changed on the branch by the user are fetched. This is done by getting the differences between the `.bib` file from the head of the branch and the parent commit on the master – the commit the suggestion was created from. Such functionality is supported by the “diff” operation of Git. The output of the “diff” for our example is shown in Listing 3.7. Added and deleted lines are marked inside the file with a single plus or minus sign at the beginning of the line (see Listing 3.7, line 8). Thus, changes can be searched inside the

3 Concept

Listing 3.6 Content of the `test-merge.bib` file after merging the second suggestion with strategy “recursive” of Git. The file contains conflicts which must be resolved.

```
1 @article{test-merge,
2   title = {Test reference to show merge strategies.},
3   <<<<<<< HEAD
4     author = {Tamara Mueller},
5   =====
6     month = {oct},
7   >>>>>> refs/heads/test-merge/6
8     year = {2017},
9     keywords = {test, merge strategy}
10 }
```

Listing 3.7 Result of a Git “diff” operation on the second suggestion and the parent commit on the master.

```
1 diff --git a/test-merge.bib b/test-merge.bib
2 index d5a3c38..ff87f77 100644
3 --- a/test-merge.bib
4 +++ b/test-merge.bib
5 @@ -1,5 +1,6 @@
6 @article{test-merge,
7   title = {Test reference to show merge strategies.},
8 +   month = {oct},
9   year = {2017},
10  keywords = {test, merge strategy}
11 }
```

“diff” output. If a change is found, the corresponding name of the field is added to a set or if the line contains the type and BibTeX key of the reference the string “type” is added to indicate that the first line of the reference has been changed. Furthermore, if the changed line only includes a added or deleted comma at the end of a line it is not added to the set because the user did not change the value. Afterwards, the set with all changes is returned to Algorithm 3.1. In the example this would be the field “month”.

From line 4 to 34 the Algorithm 3.1 reads each line from the `.bib` file (line 5) of the merge result (see Listing 3.6). Lines which only contain a closing curly bracket are skipped because such a line can only occur at the end of the file and is added manually at the end of the algorithm in line 35. Furthermore, if the line does not belong to a conflicting area it is directly added to the result (line 7 and 8), otherwise the algorithm checks if it has to use the version of the master or the branch. Lines 10 to 20 check for lines which contain a field of the reference, not the type and BibTeX key, if it belongs to the version of the branch or the master and if it has to be added to the result or not. A line is added to the result if it was changed at the branch and the line belongs to the branch (line 12 to 15) or if it was not changed on the branch and belongs to the master (line 16 to 20). The second case refers to meanwhile changes on the master which must be adopted if the branch does not change them. Similarly, the lines 21 to 30 check if a line has to be added which contains the type and BibTeX key. If the algorithm has read all lines of the file it adds a closing curly bracket to the result (line 35) which is always the last line of a BibTeX entry. Afterwards, the result

Algorithm 3.1 Algorithm for resolving merge conflicts.

```

1: procedure RESOLVECONFLICTS(file, bibtexkey, idsuggestion)
2:   result  $\leftarrow \emptyset$ 
3:   changed  $\leftarrow$  GETCHANGEDFIELDSOFBRANCH(bibtexkey, idsuggestion)
4:   while not reached end of file do
5:     line  $\leftarrow$  GETNEXTLINE(file)
6:     if line not contains single closing curly bracket then // at end of a BibTeX entry
7:       if line not in conflicting region of file then
8:         add line to result
9:       else
10:        if line contains field then
11:          fieldname  $\leftarrow$  GETFIELDNAMEFROMLINE(line)
12:          if changed contains fieldname then
13:            if line is version of branch then
14:              add line to result
15:            end if
16:          else // field not changed on branch, hence use version of master
17:            if line is version of master then
18:              add line to result
19:            end if
20:          end if
21:        else // line contains type and BibTeX key
22:          if changed contains “type” then
23:            if line is version of branch then
24:              add line to result
25:            end if
26:          else // type not changed on branch, hence use version of master
27:            if line is version of master then
28:              add line to result
29:            end if
30:          end if
31:        end if
32:      end if
33:    end if
34:  end while
35:  add single closing curly bracket to result // end of BibTeX entry
36:  // add line breaks and ensure correct use of commas between lines
37:  WRITERESULTTOFILE(result, file)
38: end procedure

```

is used to overwrite the .bib file of the reference which contains the conflict. Line breaks are generated after each element of the result and commas are added or deleted at the end of each line if necessary to match a correct BibTeX file without a comma after the last field of the reference.

3.3.2 Case Distinction for Merging Suggestions

The list below enumerates all cases that can occur when accepting change requests in the system which are created from the same version of a reference. It will look at change suggestions from one and two users, as well as all possible combinations of adding, changing, and deleting information at the same or a different location. The location refers to a field or the type of the reference, not the position in the .bib file.

- Case 1** User A adds information, user B does nothing.
- Case 2** User A changes information, user B does nothing.
- Case 3** User A deletes information, user B does nothing.
- Case 4.1** User A and user B add information at the same location.
- Case 4.2** User A and user B add information at a different location.
- Case 5.1** User A and user B change information at the same location.
- Case 5.2** User A and user B change information at a different location.
- Case 6.1** User A and user B delete information at the same location.
- Case 6.2** User A and user B delete information at a different location.
- Case 7** User A adds information and user B changes information at a different location.
- Case 8** User A adds information and user B deletes information at a different location.
- Case 9.1** User A changes information and user B deletes information at the same location.
- Case 9.2** User A changes information and user B deletes information at a different location.

In the following, an example for each case is presented. Listing 3.8 shows the content of the reference which is used as base for the first example. It is a new inserted reference in the system with the type “article”, the BibTeX key “test-merge”, and the fields title and keywords. All other examples are based on each other. The cases 1, 2, and 3 consider changes made by one user without any simultaneous changes of the reference.

Listing 3.8 The reference with BibTeX key “test-merge” is added.

```
+ @article{test-merge,  
+   title = {Test reference to show merge strategies.},  
+   keywords = {test, merge strategy}  
+ }
```

Listing 3.9 is an example for Case 1. A user adds a note to the reference. In the listing also the line with the keywords is changed because a trailing comma between this line and the new inserted field “notes” is needed. This example can be merged by the “recursive” strategy of Git without producing any conflict because there are no meanwhile changes on the master branch. The result looks like in Listing 3.9. Case 2 is visualized in Listing 3.10, there the note is changed by a user. This suggestion can also be merged with the default strategy “recursive” and the resulting reference file looks like at the branch before. Similarly, Case 3 (see Listing 3.11) can be merged because only one user modified the reference so there are no conflicts during the merge.

Listing 3.9 Example for Case 1.

```
@article{test-merge,
  title = {Test reference to show merge strategies.},
- keywords = {test, merge strategy}
+ keywords = {test, merge strategy},
+ note = {This is a reference to show merge strategies.}
}
```

Listing 3.10 Example for Case 2.

```
@article{test-merge,
  title = {Test reference to show merge strategies.},
  keywords = {test, merge strategy},
- note = {This is a reference to show merge strategies.}
+ note = {This is a reference to show different merge strategies.}
}
```

Listing 3.11 Example for Case 3.

```
@article{test-merge,
  title = {Test reference to show merge strategies.},
- keywords = {test, merge strategy},
- note = {This is a reference to show different merge strategies.}
+ keywords = {test, merge strategy}
}
```

Case 4.1 is the first case where two users modify the reference simultaneously. In the example both users add the field “year” with a different value (see Listings 3.12 and 3.13). If both suggestions get merged on after the other a merge conflict occurs which gets resolved by Algorithm 3.1. The first merged suggestion with the year “2016” is replaced in Listing 3.14 by the second suggestion with the year “2017” because changes made by the user in a suggestion overwrite always the value on the master if the suggestion was accepted by the users.

An example for Case 4.2 is presented in the previous section. Listing 3.2 adds an author and Listing 3.3 a month to the reference. The first suggestion can be merged with the “recursive” strategy, however the second merge results in a conflict. After resolving these conflict with Algorithm 3.1 both new fields are added to the reference (see Listing 3.5).

3 Concept

Listing 3.12 First part of the example for Case 4.1.

```
@article{test-merge,  
  title = {Test reference to show merge strategies.},  
+  year = {2016},  
  keywords = {test, merge strategy}  
}
```

Listing 3.13 Second part of the example for Case 4.1.

```
@article{test-merge,  
  title = {Test reference to show merge strategies.},  
+  year = {2017},  
  keywords = {test, merge strategy}  
}
```

Listing 3.14 Result after merging second suggestion of Case 4.1.

```
@article{test-merge,  
  title = {Test reference to show merge strategies.},  
-  year = {2016},  
+  year = {2017},  
  keywords = {test, merge strategy}  
}
```

In Case 5.1 two users change the same information about the reference to a different value (see Listings 3.15 and 3.16). Merging the second suggestion results in a conflict like at the example before because the “recursive” strategy of Git does not know which of the changed values it should take as new value. The Algorithm 3.1 takes always the latest accepted changes as new value. In the example this is the second suggestion because it was last merged. The resulting reference is shown in Listing 3.17.

Listing 3.15 First part of the example for Case 5.1.

```
@article{test-merge,  
  title = {Test reference to show merge strategies.},  
  author = {Tamara Mueller},  
-  month = {oct},  
+  month = {10},  
  year = {2017},  
  keywords = {test, merge strategy}  
}
```

Listing 3.16 Second part of the example for Case 5.1.

```
@article{test-merge,  
  title = {Test reference to show merge strategies.},  
  author = {Tamara Mueller},  
-  month = {oct},  
+  month = {nov},  
  year = {2017},  
  keywords = {test, merge strategy}  
}
```

Listing 3.17 Result after merging second suggestion of Case 5.1.

```

@article{test-merge,
  title = {Test reference to show merge strategies.},
  author = {Tamara Mueller},
-  month = {10},
+  month = {nov},
  year = {2017},
  keywords = {test, merge strategy}
}

```

In the example for Case 5.2 user A changes the author (see Listing 3.18) and user B the keywords (see Listing 3.19). This suggestions can be merged without conflicts by the “recursive” strategy of Git because they change values of different fields. The result of the reference is illustrated in Listing 3.20.

Listing 3.18 First part of the example for Case 5.2.

```

@article{test-merge,
  title = {Test reference to show merge strategies.},
-  author = {Tamara Mueller},
+  author = {Tamara M\{u\}ller},
  month = {nov},
  year = {2017},
  keywords = {test, merge strategy}
}

```

Listing 3.19 Second part of the example for Case 5.2.

```

@article{test-merge,
  title = {Test reference to show merge strategies.},
  author = {Tamara Mueller},
  month = {nov},
  year = {2017},
-  keywords = {test, merge strategy}
+  keywords = {test, merge strategy, recursive, theirs, ours}
}

```

Listing 3.20 Result after merging second suggestion of Case 5.2.

```

@article{test-merge,
  title = {Test reference to show merge strategies.},
  author = {Tamara M\{u\}ller},
  month = {nov},
  year = {2017},
-  keywords = {test, merge strategy}
+  keywords = {test, merge strategy, recursive, theirs, ours}
}

```

Two user delete the same information in Case 6.1, for example the month like in Listing 3.21. Because both users suggest the same change there does not occur a merge conflict. Furthermore, the file of the reference is not modified by the second merge because the changes are already adopted by merging the first suggestion. Afterwards, the reference looks like in Listing 3.21.

3 Concept

Listing 3.21 Example for Case 6.1.

```
@article{test-merge,
  title = {Test reference to show merge strategies.},
  author = {Tamara M\{u}ller},
-  month = {nov},
  year = {2017},
  keywords = {test, merge strategy, recursive, theirs, ours}
}
```

In Listing 3.22 user A deletes the information about the year and in Listing 3.23 user B the keywords. This is an example for Case 6.2. Merging the second suggestion ends with a conflict because this suggestion changes also the line with the year (see Listing 3.23, line 4 and 5). The year was not changed by the user but the system deleted the comma at the end of the line because it is the last field of the reference after the keywords are deleted. The field year is already deleted on the master branch because the first suggestion was merged. The “recursive” merge strategy of Git does not know if the line should be changed or deleted, hence a conflict occurs. Algorithm 3.1 can resolve these kind of conflict. After the algorithm is finished the reference on the master looks like Listing 3.24, both fields – year and keywords – are deleted.

Listing 3.22 First part of the example for Case 6.2.

```
@article{test-merge,
  title = {Test reference to show merge strategies.},
  author = {Tamara M\{u}ller},
-  year = {2017},
  keywords = {test, merge strategy, recursive, theirs, ours}
}
```

Listing 3.23 Second part of the example for Case 6.2.

```
@article{test-merge,
  title = {Test reference to show merge strategies.},
  author = {Tamara M\{u}ller},
-  year = {2017},
+  year = {2017}
-  keywords = {test, merge strategy, recursive, theirs, ours}
}
```

Listing 3.24 Result after merging second suggestion of Case 6.2.

```
@article{test-merge,
  title = {Test reference to show merge strategies.},
-  author = {Tamara M\{u}ller},
-  keywords = {test, merge strategy, recursive, theirs, ours}
+  author = {Tamara M\{u}ller}
}
```

For Case 7 first user A adds the year (see Listing 3.25) and then the type of the reference is changed by user B (see Listing 3.26). Merging both suggestions does not produce a conflict because the changes are on different parts of the .bib file. The result is shown in Listing 3.27.

Listing 3.25 First part of the example for Case 7.

```
@article{test-merge,
  title = {Test reference to show merge strategies.},
-  author = {Tamara M\{u}ller}
+  author = {Tamara M\{u}ller},
+  year = {2017}
}
```

Listing 3.26 Second part of the example for Case 7.

```
- @article{test-merge,
+ @misc{test-merge,
  title = {Test reference to show merge strategies.},
  author = {Tamara M\{u}ller}
}
```

Listing 3.27 Result after merging second suggestion of Case 7.

```
- @article{test-merge,
+ @misc{test-merge,
  title = {Test reference to show merge strategies.},
  author = {Tamara M\{u}ller},
  year = {2017}
}
```

In Case 8 user A also adds a field. In the example at Listing 3.28 this is the field “month”. On the other hand, user B deletes the author of the bibliographical reference (see Listing 3.29). This changes result in a merge conflict which is resolved by Algorithm 3.1 to Listing 3.30.

Listing 3.28 First part of the example for Case 8.

```
@misc{test-merge,
  title = {Test reference to show merge strategies.},
  author = {Tamara M\{u}ller},
+  month = {nov},
  year = {2017}
}
```

Listing 3.29 Second part of the example for Case 8.

```
@misc{test-merge,
  title = {Test reference to show merge strategies.},
-  author = {Tamara M\{u}ller},
  year = {2017}
}
```

The last two cases deal with changing and deleting a field. In Case 9.1 both fields are the same. The example is shown in Listings 3.31 and 3.32 where the month is edited and deleted. Merging these suggestions results again in a merge conflict because they affect the same field and the “recursive” algorithm of Git does not know which change is correct. The reference after executing Algorithm 3.1 is shown in Listing 3.33.

3 Concept

Listing 3.30 Result after merging second suggestion of Case 8.

```
@misc{test-merge,
  title = {Test reference to show merge strategies.},
-  author = {Tamara M\{u}ller},
  month = {nov},
  year = {2017}
}
```

Listing 3.31 First part of the example for Case 9.1.

```
@misc{test-merge,
  title = {Test reference to show merge strategies.},
-  month = {nov},
+  month = {11},
  year = {2017}
}
```

Listing 3.32 Second part of the example for Case 9.1.

```
@misc{test-merge,
  title = {Test reference to show merge strategies.},
-  month = {nov},
  year = {2017}
}
```

Listing 3.33 Result after merging second suggestion of Case 9.1.

```
@misc{test-merge,
  title = {Test reference to show merge strategies.},
-  month = {11},
  year = {2017}
}
```

In the last example for Case 9.2 user A changes the type and user B deletes the year of the reference entry (see Listings 3.34 and 3.35). Like before, merging ends with a conflict that is resolved by Algorithm 3.1. The final reference after the merge is shown in Listing 3.36.

Listing 3.34 First part of the example for Case 9.2.

```
- @misc{test-merge,
+ @article{test-merge,
  title = {Test reference to show merge strategies.},
  year = {2017}
}
```

Listing 3.35 Second part of the example for Case 9.2.

```
@misc{test-merge,
-  title = {Test reference to show merge strategies.},
-  year = {2017}
+  title = {Test reference to show merge strategies.}
}
```

Listing 3.36 Result after merging second suggestion of Case 9.2.

```

@article{test-merge,
-   title = {Test reference to show merge strategies.},
-   year = {2017}
+   title = {Test reference to show merge strategies.}
}

```

Table 3.1 gives an overview of which examples only use the “recursive” strategy of Git to merge the suggestions and which additionally need Algorithm 3.1 to resolve occurred merge conflicts. This information does not fit to each possible alternative of a case because not only the case is relevant if suggestions can be merged without conflicts or not, also the position of the changed information in the .bib file plays a role.

Example for Case	1	2	3	4.1	4.2	5.1	5.2	6.1	6.2	7	8	9.1	9.2
Algorithm 3.1	✗	✗	✗	✓	✓	✓	✗	✗	✓	✗	✓	✓	✓

Table 3.1: Shows which examples of the cases use the Algorithm 3.1 to resolve occurred merge conflicts. Cases that have a cross use only the “recursive” strategy of Git without producing conflicts.

3.4 Managing Comments, Users, and Ratings

CloudRef stores users, comments to literature, and ratings of references and suggestions in a database. The platform needs user accounts to assign the comments and ratings to a specific user. Furthermore, each user is allowed to rate a reference or a suggestion only once. As described in Section 3.2 a user has a role which can be user or maintainer. This must also be saved in the account of the user.

To relate the comments with the references which are saved in .bib files on the hard disk the BibTeX key of the corresponding reference is saved in the table where the comments are stored. Additionally, the visibility level of each comment is managed in the database. The mapping of the ratings to the corresponding reference or suggestion is also done by the BibTeX key and additionally by an identifier for the different suggestions.

3.5 System

This section introduces the developed reference management platform CloudRef and explains all its functionalities. At the end there is also a section about the fulfillment of the requirements stated in Section 2.1.

3 Concept

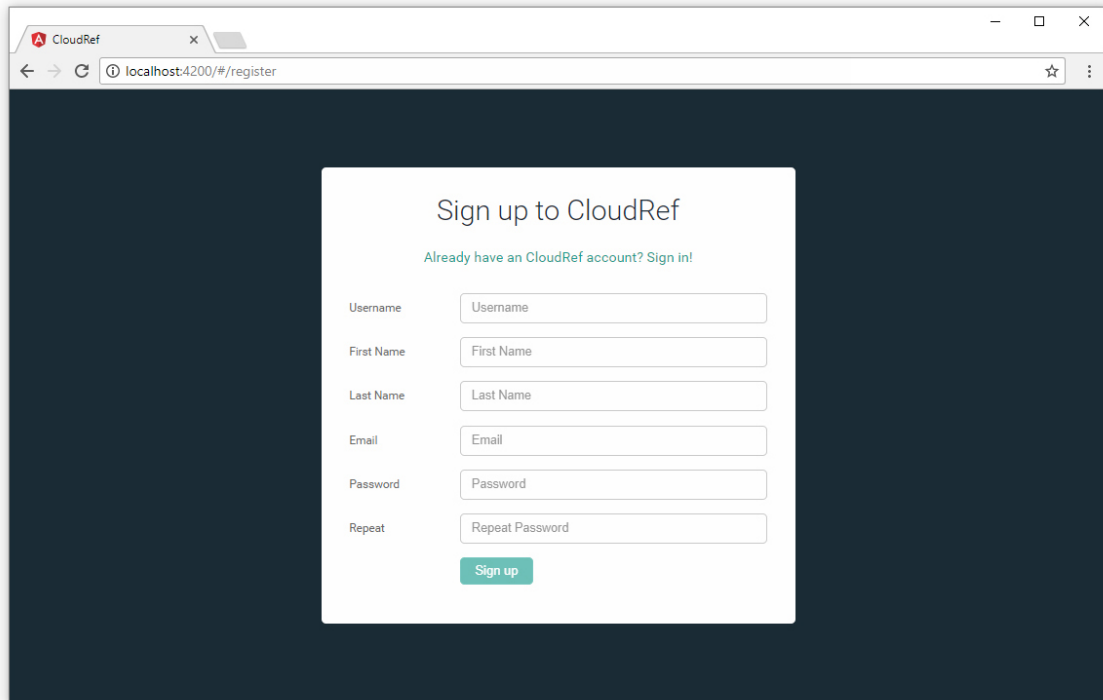


Figure 3.5: Page for creating a new user account in CloudRef.

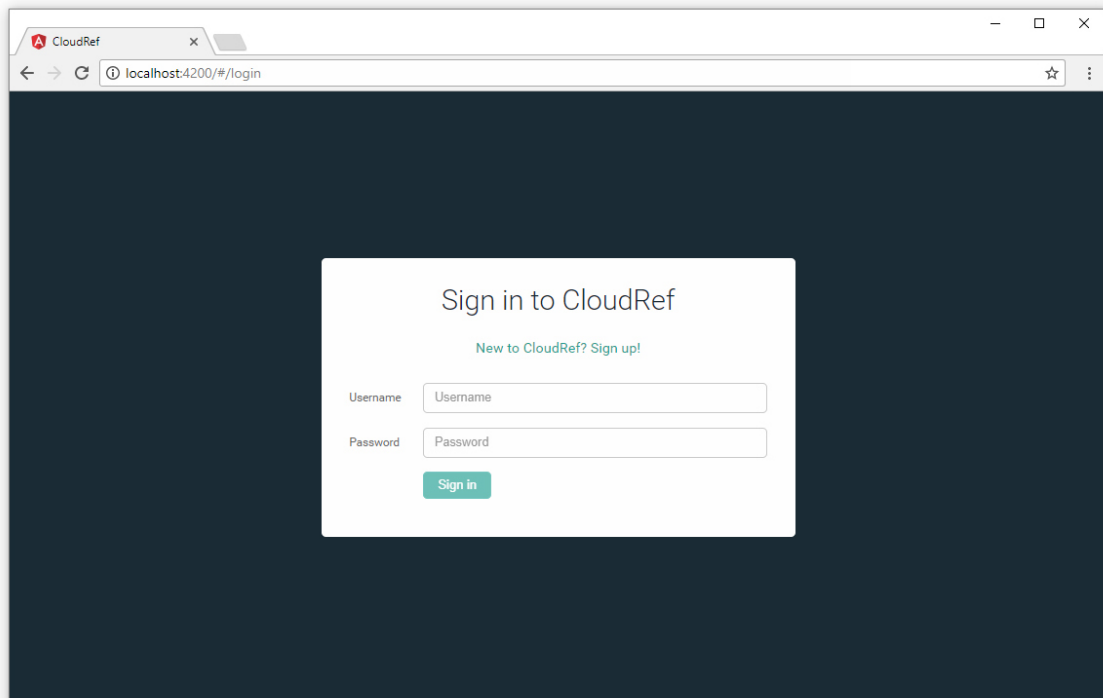
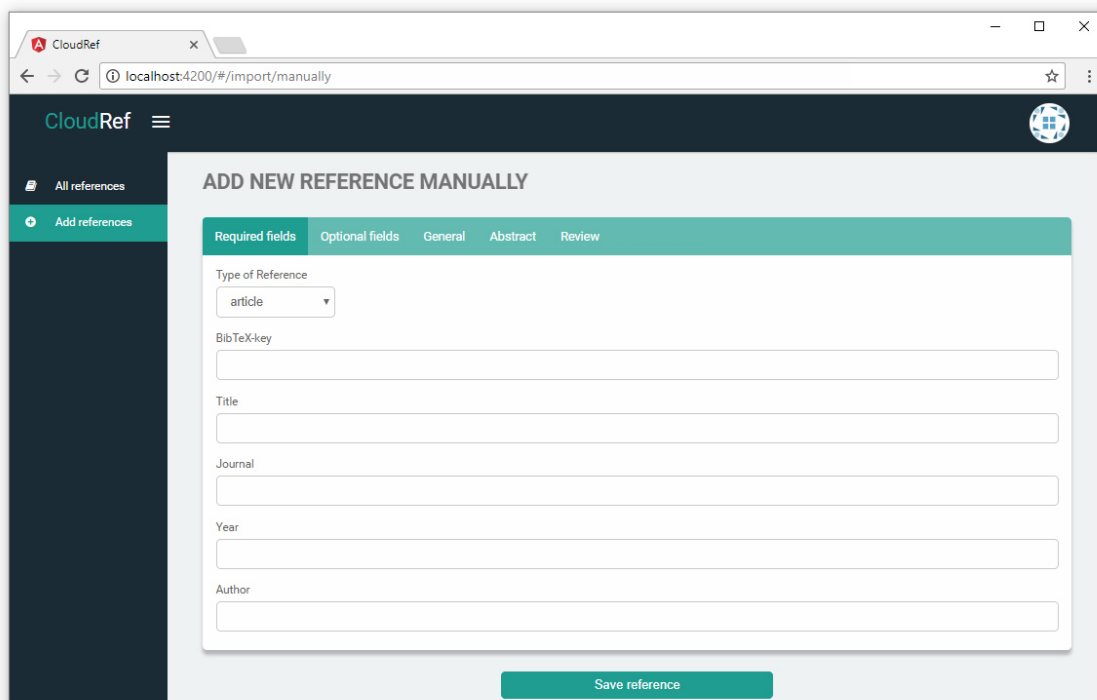


Figure 3.6: Screenshot of the login page of CloudRef.



The screenshot shows a web browser window with the URL `localhost:4200/#/import/manually`. The page title is "ADD NEW REFERENCE MANUALLY". The form has five tabs: "Required fields", "Optional fields", "General", "Abstract", and "Review". The "Required fields" tab is selected. It contains a "Type of Reference" dropdown menu with "article" selected, and five text input fields labeled "BibTeX-key", "Title", "Journal", "Year", and "Author". A green "Save reference" button is at the bottom right.

Figure 3.7: CloudRef screenshot of the page for the manual insertion of a new reference. Different document types are supported and also an upload for PDF files.

3.5.1 Registration and Login

If a person uses the system the first time, he has to create a new account with a unique user name. Furthermore, the first name, last name, and e-mail address of the person are required to register at the system and the user has to choose a password (see Figure 3.5). After the registration, the user gets logged in automatically and can use the reference management software. For each login in the future the user name and password have to be entered to access the system (see Figure 3.6). Every user who registers on the user interface has the role “user”. The role of a registered user can be upgraded to “maintainer” in the database.

3.5.2 Adding References

A user of the CloudRef software can insert new references either by entering them manually via a form (see Figure 3.7) or through uploading a `.bib` file (see Figure 3.8). The form for the manual insertion distinguishes between the reference types: article, book, booklet, conference, inbook, incollection, inproceedings, manual, mastersthesis, misc, phdthesis, proceedings, techreport, unpublished, and standard. For each reference a type and BibTeX key are required, all other fields can be empty. In addition, the BibTeX key must be unique and may contain numbers (0-9), letters without special characters (a-z, A-Z), and hyphens.

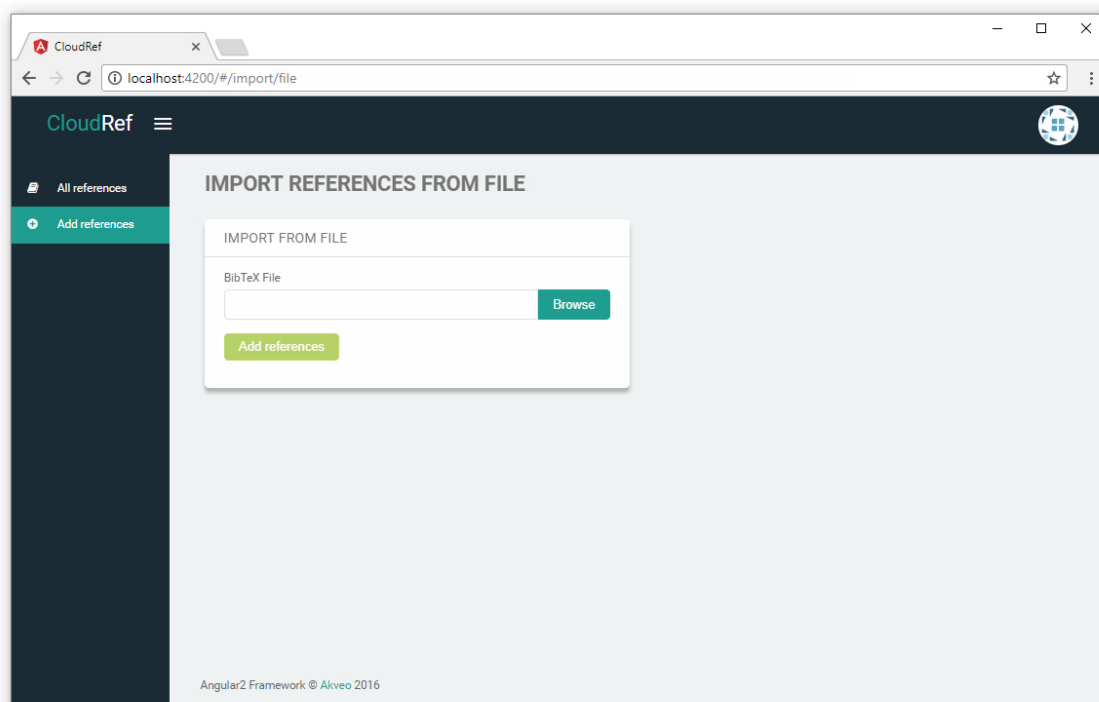


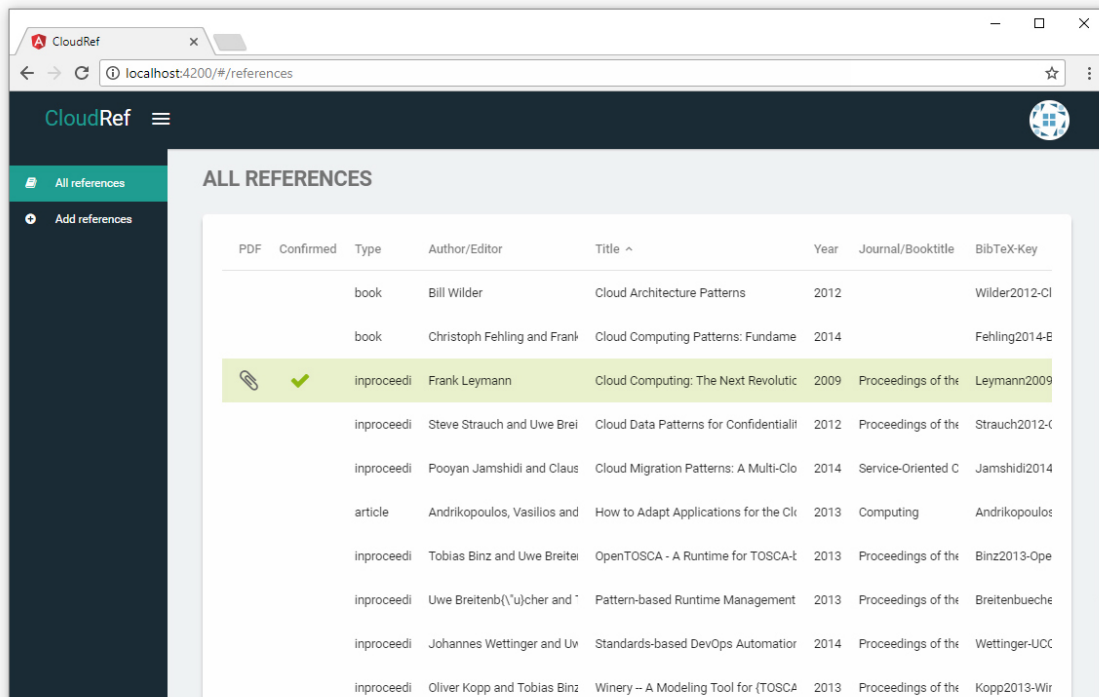
Figure 3.8: CloudRef screenshot of the `.bib` file import page. References can be modified during the import to ensure unique BibTeX keys and match their naming schema.

The form is divided into multiple tabs like in JabRef (see Section 2.10) to increase the clarity. It is also possible to link a cross reference and to upload a PDF file. References and their PDF files are visible to every user of the CloudRef system.

The import of a `.bib` file ensures that the BibTeX keys in the system are unique, this means that if an key exists already in the system the new inserted reference key gets modified during the import process. The BibTeX key also changes if the naming schema explained in the previous paragraph is not met. After a successful insertion of the bibliographical references the user gets redirected to a table view of all references of the CloudRef system (see Figure 3.9).

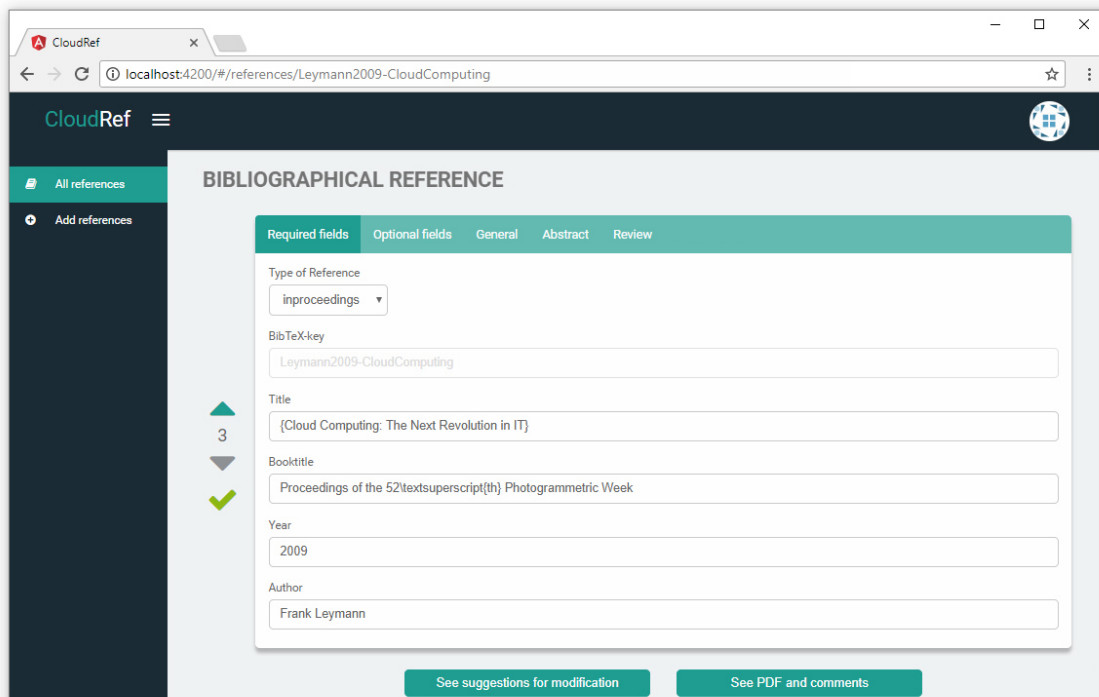
3.5.3 References

All references which are saved in CloudRef are listed in a table (see Figure 3.9). This table supports pagination, that only 12 references are shown at the same time. Furthermore, the entries can be sorted through clicking on the column titles and the width of the columns can be adjusted by the user. CloudRef has a feature to allow users to classify the quality of a reference as good, such a confirmed reference is highlighted in the table with green background color and has a check mark at the confirmed column like the third reference in Figure 3.9.



PDF	Confirmed	Type	Author/Editor	Title ^	Year	Journal/Booktitle	BibTeX-Key
		book	Bill Wilder	Cloud Architecture Patterns	2012		Wilder2012-CI
		book	Christoph Fehling and Frank	Cloud Computing Patterns: Fundame	2014		Fehling2014-E
		inproceedi	Frank Leymann	Cloud Computing: The Next Revolutic	2009	Proceedings of the	Leymann2009
		inproceedi	Steve Strauch and Uwe Brei	Cloud Data Patterns for Confidentialit	2012	Proceedings of the	Strauch2012-C
		inproceedi	Pooyan Jamshidi and Claus	Cloud Migration Patterns: A Multi-Clo	2014	Service-Oriented C	Jamshidi2014
		article	Andrikopoulos, Vasilios and	How to Adapt Applications for the Clk	2013	Computing	Andrikopoulos
		inproceedi	Tobias Binz and Uwe Breite	OpenTOSCA - A Runtime for TOSCA-t	2013	Proceedings of the	Binz2013-Ope
		inproceedi	Uwe Breitenb(\u)cher and	Pattern-based Runtime Management	2013	Proceedings of the	Breitenbueche
		inproceedi	Johannes Wettinger and Uw	Standards-based DevOps Automatior	2014	Proceedings of the	Wettinger-UCC
		inproceedi	Oliver Kopp and Tobias Binz	Winery - A Modeling Tool for (TOSCA	2013	Proceedings of the	Kopp2013-Wir

Figure 3.9: CloudRef screenshot of the table with all references. Confirmed references indicate that the quality of this entry is good.



BIBLIOGRAPHICAL REFERENCE

Required fields | Optional fields | General | Abstract | Review

Type of Reference: inproceedings

BibTeX-key: Leymann2009-CloudComputing

Title: (Cloud Computing: The Next Revolution in IT)

Booktitle: Proceedings of the 52nd Photogrammetric Week

Year: 2009

Author: Frank Leymann

3 (Rating)

See suggestions for modification | See PDF and comments

Figure 3.10: CloudRef screenshot of a reference entry. On the left side are rating buttons and an check mark if the reference is already confirmed.

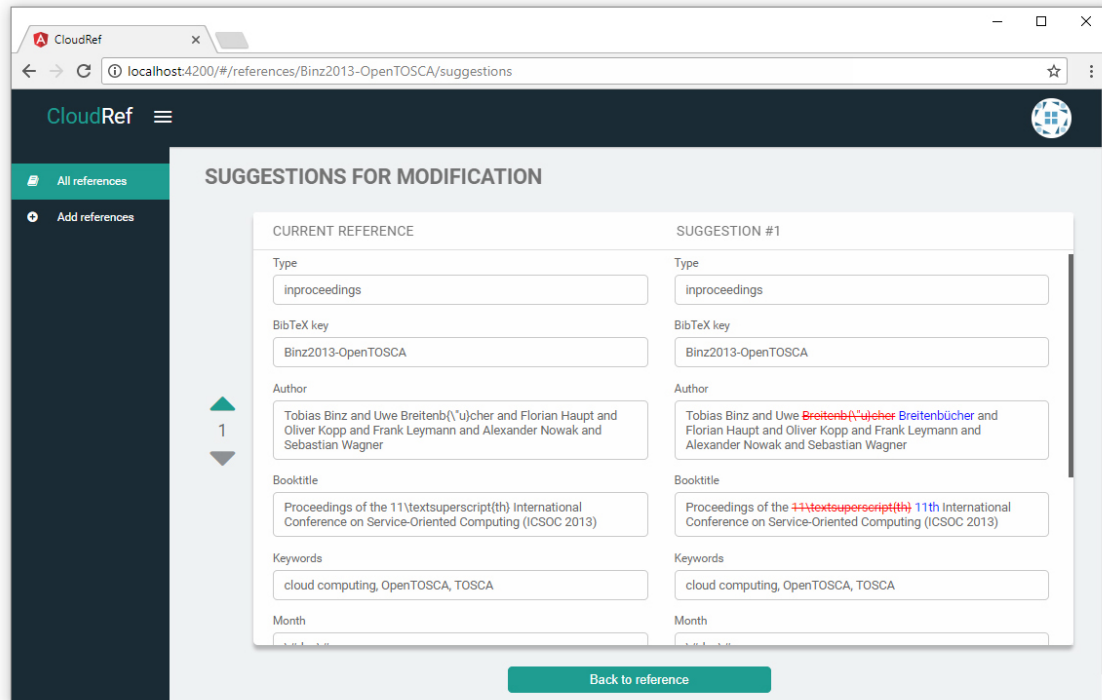


Figure 3.11: CloudRef screenshot which shows the view of suggestions to modify a reference. On the left side the current reference entry is printed and on the right side the changed version. The differences are highlighted on the modified version. To the left of this version comparison are rating buttons and between them the current rating of all users.

If the user selects a reference in the table he gets redirected to the page of this reference (see Figure 3.10) which looks similar to the manual import on Figure 3.7. There the user can upload a PDF file if none has been uploaded before. Furthermore, he can edit the reference and publish the changes as a suggestion to modify the bibliography or if he does not want to provide a suggestion but thinks the entry is incomplete or faulty he can just rate the reference negative. On the other hand, if the user thinks the reference is good he can rate it positive. The value of the rating is initially zero. Each positive vote increases the value by one and all negative votes decrease it also by one. If the user voted already the button of his decision is colored but it is also possible to change the voting. Between the buttons the current rating of all users is shown. If the rating reaches a positive threshold the reference is classified as confirmed (see Section 3.1). A green check mark is shown below the rating buttons if the reference is confirmed by the users. At the bottom of the page are buttons to navigate to the suggestions which are available for this reference and to a PDF and comments view if a PDF file was uploaded.

Figure 3.11 shows the page of the suggestions for a reference. The reference and the suggestions to modify it are shown side by side. At the changed version, on the right side, the differences are highlighted. Removed parts are red and crossed out, added text on

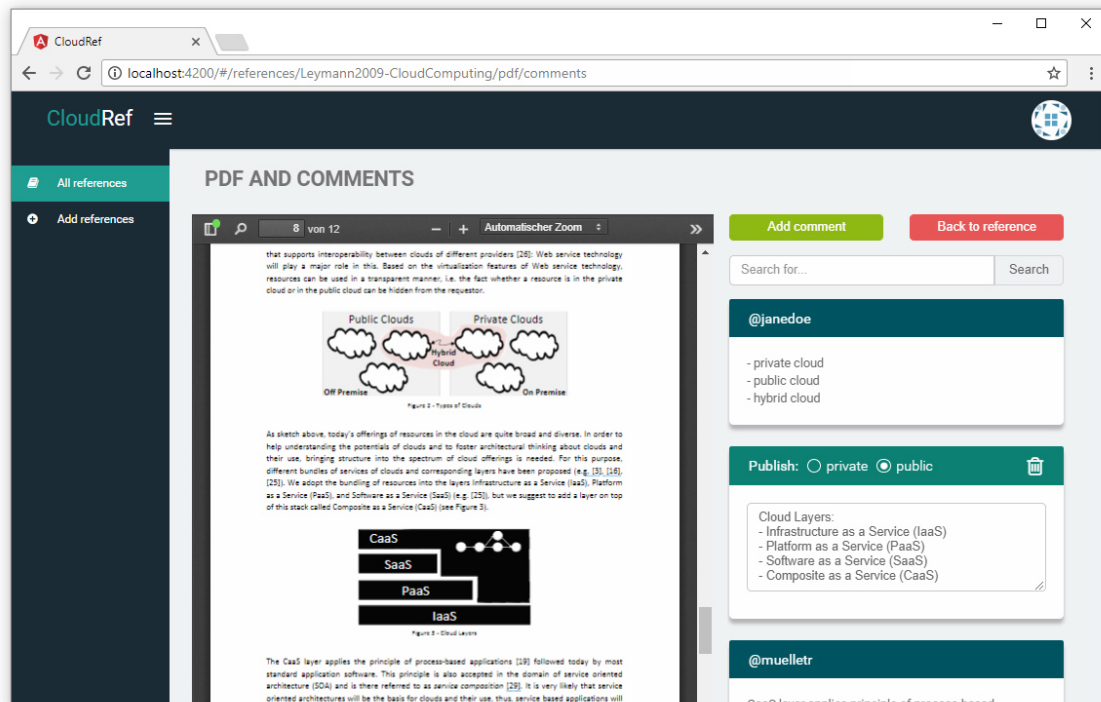


Figure 3.12: CloudRef screenshot which shows a PDF file of a reference and the corresponding comments of the users. Only the comments of the current PDF page are visible. Above the comments are buttons to create a new comment or to navigate to the reference view.

the other hand is blue. To the left of the comparison of both references are buttons to rate the suggestion positive or negative like at the view of a single reference. Between the buttons also the current rating of all users is shown. If this value reaches a positive or negative threshold the suggestion gets accepted or rejected automatically by the system. Accepting means that the current reference is adapted by the suggestion and rejecting that the suggestion is removed from the suggestions view (see Section 3.2). If the signed in user has the role maintainer, below the rating buttons are three buttons for modifying the suggestion or for accepting or rejecting it immediately (see Section 3.2).

3.5.4 PDF and Comments

The PDF and comments view in Figure 3.12 shows the PDF file of a reference and all public published comments as well as all comments of the user who is logged in. New comments can be written and published private or visible for all users of the system. Only the comments of the current PDF page are visible for the user. If the user scrolls in the PDF to another page, the currently shown comments fade out and the comments of the new selected page fade in. The comments initially look the same no matter who is the author of the comment like the first and third comment on the figure. A signed in user

can click on his comments to edit them (see Figure 3.12, second comment) similar than in the Adobe PDF reader. He can change the visibility level as well as the content of the comment. Furthermore, users can delete their own comments. This action requires verification through a confirmation dialog to avoid deleting comments by mistake.

3.5.5 Fulfillment of the Requirements

Section 2.1 defines 20 requirements for a cooperative reference management software with quality assurance of references. CloudRef supports the Requirements 1.1 and 1.2 through allowing users to add PDF files to references. If an uploaded PDF contains comments they are shown inside the frame of the file. Extracting these comments (Requirement 1.3) and the visibility level of the file (Requirement 1.4) and the comments within the PDF (Requirement 1.5) cannot be changed, they are always visible for all users of the platform. The visibility levels private and public are available in CloudRef for comments, thus the Requirements 2.1 and 2.4 are fulfilled. Additionally, the Requirement 2.5 is met because the users can decide for each comment separately who is allowed to see it.

The third group of the requirements is about cooperative use which is supported by CloudRef but there is no group functionality implemented. It is only possible to share content with all other users of the platform. However, the platform can be set up by every person and he can invite his colleagues to use it. Thus, the platform supports something similar than described in Requirements 3.1 and 3.2. Additionally, the Requirements 3.3 to 3.5 are supported because every user is allowed to add references and PDF files as well as write comments to literature. Of the quality assurance requirements only Requirement 4.1 is implemented in the software by highlighting confirmed references in the table with all requirements as well as in the view of a single reference entry. Overall, CloudRef provides 11 of the 20 requirements.

4 Architecture and Implementation

This chapter describes the architecture of the CloudRef system and how the components work together. Afterwards, the design decisions are evaluated and finally, there is a deeper insight into the implementation of CloudRef to show how the platform is realized.

4.1 Architecture

The CloudRef platform consists of two parts, a user interface and a back end application. The architecture is illustrated as Fundamental Modeling Concepts¹ block diagram in Figure 4.1. The back end provides a REST interface with that the user interface can interact through HTTP requests. As described in Sections 3.3 and 3.4 the platform uses a local version control system and a relational database to store information. The bibliographical references saved in CloudRef are stored in separate `.bib` files on the hard disk, as well as the corresponding PDF files. Furthermore, the files containing the references are under version control. All other information, like comments to literature, are stored in a relational database. The back end has read and write access to these storages which is provided through a data access layer.

4.2 Design Decisions

The references can be stored each in a single `.bib` file or all in one file. At the beginning we decided to store them in separate files to enable multi user access. A file can only be written by one user, meanwhile all other accesses are blocked – reading as well as writing. Additionally, a single file can get very huge if all references are stored there which is another reason for storing all in an own file. For the naming of the files we decided to use the unique BibTeX keys of the references. Not every character is supported for the naming of a file like a slash or colon. Hence, the BibTeX keys can only contain numbers (0-9), letters without special characters (a-z, A-Z), and hyphens. Furthermore, to support multiple versions of references and to provide traceability for them we decided to use a version control system instead of a plain file system. Suggestions to modify a reference are stored each on a new branch because they are new versions of a reference which do not immediately replace the current version (see Section 3.2). The branches are named

¹<http://www.fmc-modeling.org>

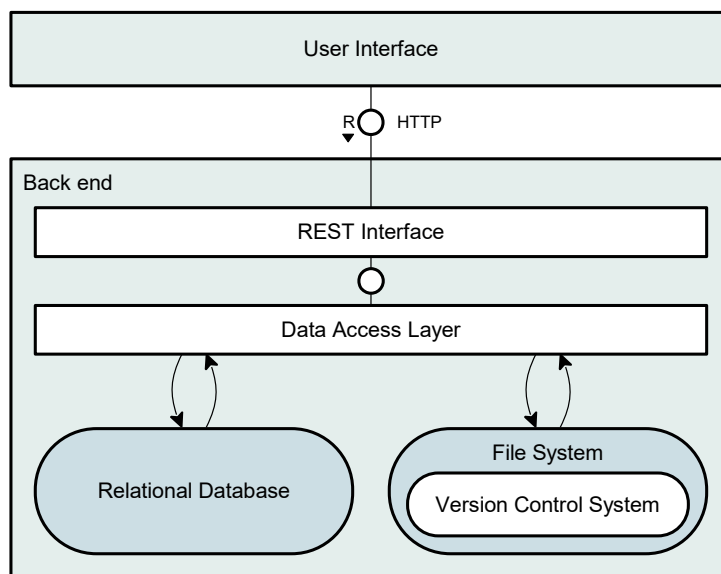


Figure 4.1: Architecture of the CloudRef platform. It consists of two parts, a user interface and a back end. The back end provides a REST interface with that the user interface can interact. Furthermore, the back end has read and write access to a relational database and the file system through a data access layer. The file system has also a version control system which handles different versions of a reference entry.

by the schema $\{\text{BibTeX_key}\}/\{\text{suggestion_id}\}$ to clearly separate the BibTeX key from the suggestion identifier. If such a suggestion is rejected by the users the branch could be deleted or merged without changing the current version of the reference. To provide full traceability rejected suggestions are merged and not deleted.

Comments to literature can be stored in the reference entry or in a separate file. Furthermore, a combination of both can be used where public comments are stored in the reference entry and private comments in a separate file. All three approaches have the drawback that consistency is hard to achieve. For example comments can be deleted in the meanwhile. On the other hand, the comments must be numbered consecutively. To overcome these issues a database management system is used instead. There are several possibilities for a primary key. A composite variant could use the BibTeX key and the comment identifier. Additionally, the username can be added to this composite primary key. On the other hand, also the comment identifier on its own can be used but this has the drawback that the identifier is increased by each comment. With the first strategy the identifier can be increased for comments on each reference individually. This is used because it matches the REST style of our resource where each reference has a consecutively increasing number for the comments of all users.

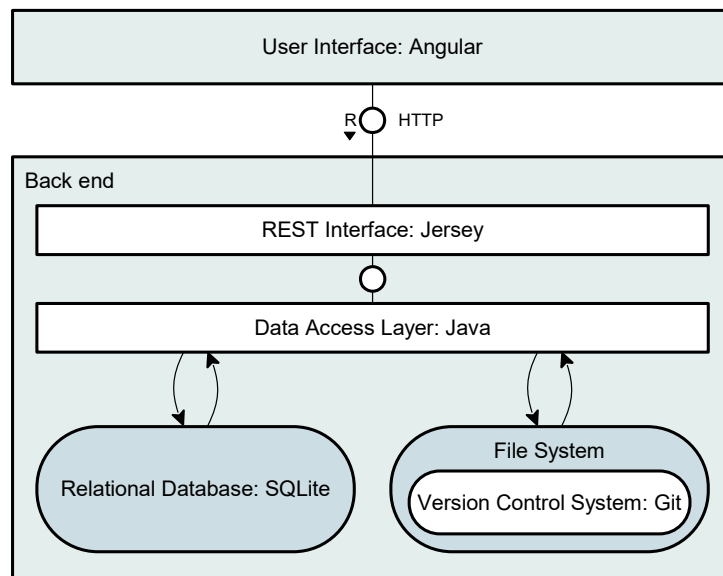


Figure 4.2: Architecture of the CloudRef platform with implementation details. The user interface is a Angular application and the back end a Java program. The REST interface uses Jersey and the version control system is Git. The data are stored in a SQLite database.

4.3 Implementation

The front end of CloudRef – the user interface – is an Angular application and the back end a Java program (see Figure 4.2). The implementation of the REST interface uses Jersey which is a framework to program RESTful web services in Java and supports the JAX-RS API² [Jer17; TESW15]. The code at the front end which calls the REST API is generated with Swagger Codegen³ but some parts of it had to be adapted to work properly. For storing user accounts, comments to literature, and ratings of references and suggestions for modification a relational SQLite database is used. Furthermore, Git is the version control system in CloudRef which allows to manage different versions of a reference entry.

4.3.1 Front end

The Angular application at the front end uses the Angular and Bootstrap 4 Admin Template “Ng2-Admin”⁴ from Akveo. To show the PDF files at the front end PDF.js⁵ is used (see Figure 3.12) which is the standard PDF viewer in Firefox. The library Jdenticon⁶ enables

²<https://github.com/jax-rs>

³<https://swagger.io/swagger-codegen>

⁴<https://akveo.github.io/ng2-admin>

⁵https://mozilla.github.io/pdf.js/getting_started

⁶<https://jdenticon.com>

to show an avatar for each user generated from his username hash value (see Figure 3.12, top right). Finally, the Angular component ngx-datatable⁷ is used to show the overview table of all references (see Figure 3.9). The table comes with many features like column resizing, sorting, and pagination which are used for CloudRef.

4.3.2 Back end

To read and write the BibTeX files from hard disk the Java library JBibTeX⁸ is used. It allows to convert each reference into an “BibTeXEntry” Java object which gets send to the front end to show a reference entry and from the front end to the back end to save or update an entry. The fields of an “BibTeXEntry” contain an abstract class, hence the type of the field has to be added in the JSON representation to enable unmarshalling of the received objects from the front end. Furthermore, empty constructors have to be added to some classes of the library.

As explained in Section 3.3 the CloudRef platform uses a local Git repository to manage different versions of reference entries. The Java library JGit⁹ provides access to this repository from the source code. To merge a rejected suggestion the merge strategy “ours” of JGit is used which works like the “ours” strategy of Git, it takes the tree of the master and ignores the changes from the branch [ecl17; Git17]. This is exactly what we want if the changes are not accepted by the users. On the other hand, if the changes are accepted the changes made on the branch have to be adopted by the master. It is achieved by CloudRef through merging the branch with the “recursive” strategy into the master and resolving the occurred conflicts with Algorithm 3.1. The “recursive” strategy is used to merge the accepted suggestions because there is no strategy which can resolve all conflicts automatically in the right way (see Section 3.3) and “recursive” is the default strategy in Git to merge a branch [Git17].

In CloudRef the users, comments on literature, and ratings of references are stored in a relational SQLite Database. To access the database from the Java source code, the library Hibernate is used. It enables a Object/Relational Mapping which means that it maps Java Objects into the relational database and vice versa [Hib17a; Hib17b]. The database schema is depicted in Figure 4.3.

To authenticate the users of the platform HTTP basic authentication is used. To access a resource at the back end the username and password have to be provided in the HTTP request from the front end application [TESW15]. This information is encoded with base64 which represents the text in letters without special characters (a-z, A-Z), numbers (0-9), plus signs, and slashes. This encoding is used to send texts over channels that accept only a basic character set, the data is not encrypted. The back end decodes this information and

⁷<https://github.com/swimlane/ngx-datatable>

⁸<https://github.com/jbibtex/jbibtex>

⁹<https://www.eclipse.org/jgit>

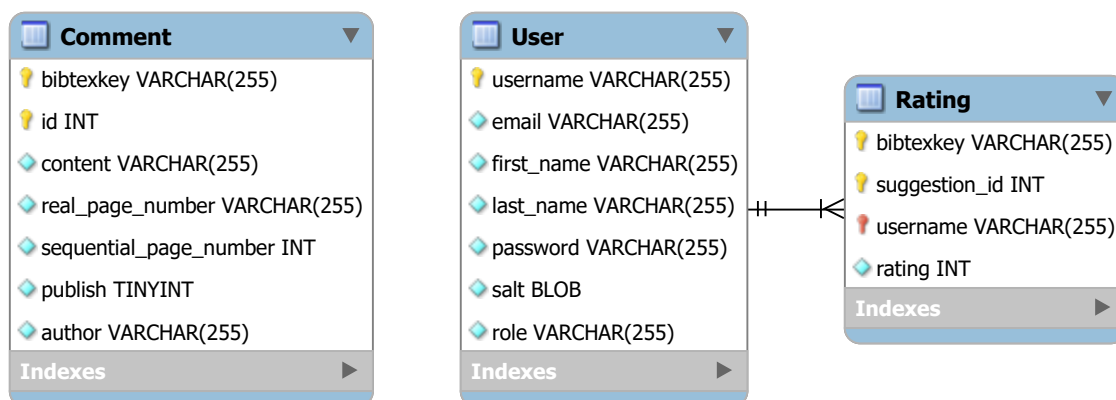


Figure 4.3: Database schema of CloudRef.

extracts the username and password. Afterwards, it can check if the user exists and provide him access to the resource.

To provide a login functionality all registered users are stored in the database (see Figure 4.3, center). The password of a user account is encrypted and not saved in plain text. Therefore the hash function “PBKDF2” is used [Def17]. Furthermore, to avoid that same passwords are stored with the same hash value a random string – called salt – is added to the password before hashing it [Def17; Wor16]. Another salt is used for each password. This prevents that lookup tables and rainbow tables can be used to crack the hash [Def17]. The salt is generated with a cryptographically secure pseudo-random number generator which is in Java provided by the class “SecureRandom”¹⁰. Such a generator is needed to prevent that the salt is predictable which would decrease the security. Afterwards, the hashed password with the salt and the salt itself are stored in the database (see Figure 4.3, center) [Def17; Wor16]. The salt has to be stored to enable checking if the user information is correct. This is done by hashing the provided password of the user with the same hash function as before and the salt from the database. The password is valid if the result of this hash function call and the password stored in the database match.

Comments to references are also stored in the database (see Figure 4.3, left). The author column of a comment stores only the username and has no foreign key relation to the user table because the mapped Java objects of these table are sent via the REST interface to the front end, hence a comment should not contain all information of the author like his email address or password. The sequential page number is additionally saved because the real page number can be empty for example if the reference is a book which has some empty pages after the cover.

The ratings of references and suggestions are saved in another table of the database (see Figure 4.3, right). There the mapping to the corresponding reference or suggestion is done by the BibTeX key and additionally by an identifier for the different suggestions. If the

¹⁰<https://docs.oracle.com/javase/8/docs/api/java/security/SecureRandom.html>

4 Architecture and Implementation

rating belongs to a reference, not a suggestion, the identifier is zero. The ratings of the references and suggestions are saved in one table because all columns except the identifier of the suggestions are the same.

5 Conclusion and Future Work

Literature management programs are used for scientific work to collect, manage, and export bibliographical references. They provide the functionality to store knowledge about literature in form of comments, tags, or inside the corresponding PDF file. In scientific work it is common to work together with other persons, hence a reference management software should support collaborative work. For that purpose, references and comments must be able to be shared with other users or people who use another or no literature management software at all. In addition, in scientific work the bibliographies are used for publications. The publishers have different guidelines about which information is needed for a reference entry. Furthermore, a correct and complete bibliography is required for publication. References from online sources like Google Scholar are often faulty or incomplete [Kop16], therefore a quality assurance of references inside the reference management software is desirable. Many tools provide a mechanism to detect missing required fields and highlight these entries as incomplete. However, this is not sufficient because wrong information is not detected.

This thesis covers a concept how to realize a reference management software that supports collaborative work and quality assurance of references. At the beginning, 20 requirements are defined for such a software (Section 2.1). Chapter 3 presents how to achieve these requirements. The first part of the chapter deals with the realization of the requirements for collaborative work. A group functionality can be used which allows the users to share references and comments with other group members. Furthermore, it is possible to share them with all users of the system through publishing them public or on the other hand make comments private that they are only visible for the user himself. For each comment the visibility level can be set separately. The second part of Chapter 3 deals with the quality assurance of references. This is achieved by a rating system which allows the users to rate reference entries positive or negative. If a rating of a reference reaches a positive threshold the entry is marked as confirmed. Such a confirmed status is an indicator for that the quality of the reference is good. Additionally, reference entries can be improved through editing them and publish this changed version as suggestion for modification. These suggestions can be rated like the references before. If a positive threshold is reached the suggestion replaces the reference and on the other hand if a negative threshold is reached than the suggestion is rejected. A prototype – named “CloudRef” – was implemented as part of this thesis. It implements several of the introduced concepts and fulfills 11 of the 20 defined requirements.

Future Work

CloudRef does not implement all presented concepts and therefore not all defined requirements are fulfilled by the platform. Further steps should expand the functionalities of the system with the aim to meet all 20 requirements. This includes first especially the requirements for quality assurance of bibliographical references. The collaborative aspects are already fulfilled roughly and fine granular features can be integrated at a later point in time. If the quality assurance parts – detection of duplicates and consistent notation of conferences, authors, and abbreviations – as well as an export for bibliographies are implemented the platform can be used and evaluated. Such an evaluation can uncover if the system fulfills its purpose, which parts should be adjusted, and if there are missing functionalities.

Afterwards, there can be added additional features like replies to comments. This can be useful in collaborative work to allow discussions about literature. Furthermore, the platform can be expanded by a recommender system which shows the users related articles to a reference. Another idea would be to insert a mind map functionality like in Docear (see Section 2.7).

Bibliography

- [Aig13] Aigaion. *Web based bibliography management system*. 2013. URL: <https://sourceforge.net/projects/aigaion/> (cit. on p. 18).
- [AMSW16] M. Adam, J. Musiat, M. Stöhr, C. Wenzel. *Literaturverwaltungsprogramme im Überblick*. SLUB Dresden. Dec. 6, 2016. URL: <http://nbn-resolving.de/urn:nbn:de:bsz:14-qucosa2-77350> (cit. on pp. 22–24, 31–33, 35–37).
- [AT05] G. Adomavicius, A. Tuzhilin. “Towards the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions.” In: *IEEE Transactions on Knowledge and Data Engineering* 17.6 (2005), pp. 734–749 (cit. on p. 40).
- [Bee14] J. Beel. *Docear – Howto: Import references from webpages (e.g. PubMed, IEEE, ACM, ...)* July 22, 2014. URL: <https://www.docear.org/2014/07/22/howto-import-references-from-webpages-e-g-pubmed-ieee-acm/> (cit. on p. 29).
- [Bib15] BibSonomy. *Wiki: BibSonomy GoogleDocs Add-on*. Jan. 28, 2015. URL: <https://bitbucket.org/fwhkoenig/bibsonomy-googledocs-add-on/wiki/Home> (cit. on p. 22).
- [Bib17a] BibSonomy. *Attach a private document to publications*. 2017. URL: https://www.bibsonomy.org/help_en/AttachDocuments (cit. on p. 20).
- [Bib17b] BibSonomy. *BibSonomy – Export*. 2017. URL: <https://www.bibsonomy.org/export> (cit. on p. 20).
- [Bib17c] BibSonomy. *Getting started with BibSonomy*. 2017. URL: <https://www.bibsonomy.org/gettingStarted?lang=en> (cit. on pp. 20, 22).
- [Bib17d] BibSonomy. *Group functions*. 2017. URL: https://www.bibsonomy.org/help_en/GroupFunctions (cit. on pp. 21, 22).
- [Bib17e] BibSonomy. *Sorting publications*. 2017. URL: https://www.bibsonomy.org/help_en/SortingPublications (cit. on p. 21).
- [Bib17f] BibSonomy. *Source*. 2017. URL: <https://bitbucket.org/bibsonomy/bibsonomy/src> (cit. on p. 20).
- [BL15] J. Beel, S. Langer. “A Comparison of Offline Evaluations, Online Evaluations, and User Studies in the Context of Research-Paper Recommender Systems.” In: *TPDL*. Vol. 9316. Lecture Notes in Computer Science. Springer, 2015, pp. 153–168 (cit. on p. 40).
- [BLGN13] J. Beel, S. Langer, M. Genzmehr, A. Nürnberger. “Introducing Docear’s research paper recommender system.” In: *JCDL*. ACM, 2013, pp. 459–460 (cit. on p. 40).

Bibliography

- [Cit16a] Citavi. *Citavi for DBServer*. 2016. URL: https://www.citavi.com/sub/manual5/en/citavi_for_db_server_overview.html (cit. on p. 24).
- [Cit16b] Citavi. *Conflict Management*. 2016. URL: https://www.citavi.com/sub/manual5/en/team_conflict_management.html (cit. on p. 24).
- [Cit16c] Citavi. *Creating Citation Styles*. 2016. URL: https://www.citavi.com/sub/manual5/en/cse_creating_citation_styles_overview.html (cit. on p. 24).
- [Cit16d] Citavi. *Overview of Team Features*. 2016. URL: https://www.citavi.com/sub/manual5/en/team_functions_overview.html (cit. on p. 24).
- [Cit17a] CiteULike. *CiteULike Gold*. 2017. URL: <http://www.citeulike.org/gold> (cit. on pp. 24, 25).
- [Cit17b] CiteULike. *Frequently Asked Questions*. 2017. URL: <http://www.citeulike.org/faq/faq.adp> (cit. on pp. 24–26).
- [col14] colwiz Team. *New Feature: Collectively read and annotate documents through the colwiz Drive*. Sept. 1, 2014. URL: <http://blog.colwiz.com/2014/09/01/new-feature-collectively-read-and-annotate-docs-through-colwiz-drive/> (cit. on p. 27).
- [Def17] Defuse Security. *Salted Password Hashing – Doing it Right*. Aug. 1, 2017. URL: <https://crackstation.net/hashing-security.htm> (cit. on p. 73).
- [Doc17a] Docear. *Citation Styles*. 2017. URL: <http://www.docear.org/software/add-ons/docear4word/citation-styles/> (cit. on p. 30).
- [Doc17b] Docear. *Details & Features*. 2017. URL: <http://www.docear.org/software/details/> (cit. on pp. 29, 30).
- [Doc17c] Docear. *User Manual*. 2017. URL: <http://www.docear.org/support/user-manual/> (cit. on pp. 29, 30).
- [ecl17] eclipse JGit Project. *Class MergeStrategy*. 2017. URL: <http://download.eclipse.org/jgit/site/4.9.0.201710071750-r/apidocs/org/eclipse/jgit/merge/MergeStrategy.html> (cit. on p. 72).
- [End17] EndNote. *Library Sharing*. 2017. URL: http://endnote.com/product-details/library-sharing?utm_source=en-online&utm_medium=referral&utm_campaign=en-online-banner (cit. on p. 32).
- [F1017a] F1000Workspace. *F1000Workspace user guide*. Jan. 27, 2017. URL: http://f1000.com/resources/Workspace_User_Manual.FINAL.pdf (cit. on pp. 33–35).
- [F1017b] F1000Workspace. *Help guide – Working with references*. 2017. URL: <http://f1000.com/work/faq/references-in-f1000> (cit. on pp. 33, 34).
- [FSG+17] S. Feyer, S. Siebert, B. Gipp, A. Aizawa, J. Beel. “Integration of the Scientific Recommender System Mr. DLib into the Reference Manager JabRef.” In: *ECIR*. Vol. 10193. Lecture Notes in Computer Science. 2017, pp. 770–774 (cit. on p. 40).

- [Git17] Git. *Git Merge*. Oct. 23, 2017. URL: <https://git-scm.com/docs/git-merge> (cit. on pp. 49–51, 72).
- [Hib17a] Hibernate. *Hibernate ORM – What is Object/Relational Mapping?* 2017. URL: <http://hibernate.org/orm/what-is-an-orm/> (cit. on p. 72).
- [Hib17b] Hibernate. *Hibernate ORM – Your relational data. Objectively.* 2017. URL: <http://hibernate.org/orm/> (cit. on p. 72).
- [HJSS06] A. Hotho, R. Jäschke, C. Schmitz, G. Stumme. “BibSonomy: A Social Bookmark and Publication Sharing System.” In: *Proceedings of the Conceptual Structures Tool Interoperability Workshop at the 14th International Conference on Conceptual Structures*. Vol. 87. 2006, pp. 87–102 (cit. on p. 47).
- [Hyp17a] Hypothesis. *About Us*. 2017. URL: <https://web.hypothes.is/about/> (cit. on p. 40).
- [Hyp17b] Hypothesis. *Annotating with Groups*. 2017. URL: <https://web.hypothes.is/annotating-with-groups/> (cit. on p. 40).
- [Hyp17c] Hypothesis. *Get started*. 2017. URL: <https://web.hypothes.is/start/> (cit. on p. 40).
- [Jab16a] JabRef. *Shared SQL Database*. Nov. 18, 2016. URL: <http://help.jabref.org/en/SQLDatabase> (cit. on p. 37).
- [Jab16b] JabRef. *Sharing a Bib(La)TeX Database*. Sept. 26, 2016. URL: <http://help.jabref.org/en/SharedBibFile> (cit. on p. 37).
- [Jab16c] JabRef. *Special Fields*. Oct. 27, 2016. URL: <http://help.jabref.org/en/SpecialFields> (cit. on p. 37).
- [Jab17a] JabRef. *Fetching entries from the web*. Jan. 4, 2017. URL: <http://help.jabref.org/en/#fetching-entries-from-the-web> (cit. on p. 35).
- [Jab17b] JabRef. *Find duplicates*. Sept. 4, 2017. URL: <http://help.jabref.org/en/FindDuplicates> (cit. on p. 47).
- [Jer17] Jersey. *Jersey – RESTful Web Services in Java*. June 9, 2017. URL: <https://jersey.github.io/> (cit. on p. 71).
- [Kop16] O. Kopp. *JabCloud*. June 9, 2016. URL: <https://github.com/JabRef/jabcloud/wiki> (cit. on pp. 15, 46, 75).
- [Ref17a] RefWorks. *Attaching Files To A Reference*. 2017. URL: https://www.refworks.com/refworks/help/Attaching_Files_To_A_Reference.htm (cit. on pp. 37, 38).
- [Ref17b] RefWorks. *Capturing Web Page Data With RefGrab-It*. 2017. URL: https://www.refworks.com/refworks/help/Using_RefGrab-It_to_Capture_Web_Page_Data.htm (cit. on p. 38).
- [Ref17c] RefWorks. *Creating Your Account*. 2017. URL: http://www.refworks.com/rwathens/help/Setting_up_Your_Account.htm (cit. on p. 37).
- [Ref17d] RefWorks. *RefWorks*. 2017. URL: <http://www.proquest.com/products-services/research-tools/refworks.html> (cit. on p. 37).

- [Ref17e] RefWorks. *Sharing A Folder Or Your Entire Database*. 2017. URL: http://www.refworks.com/refworks/help/sharing_your_database.htm (cit. on p. 39).
- [Ref17f] RefWorks. *Viewing References*. 2017. URL: http://www.refworks.com/rwathens/help/Viewing_References.htm (cit. on pp. 38, 39).
- [SS17a] B. Sturm, A. Sunyaev. “If You Want Your Research Done Right, Do You Have to Do It All Yourself? Developing Design Principles for Systematic Literature Search Systems.” In: *Designing the Digital Transformation: DESRIST 2017 Research in Progress Proceedings of the 12th International Conference on Design Science Research in Information Systems and Technology*. Karlsruhe, Germany. 30 May-1 Jun. Ed. by A. Maedche, J. v. Brocke, A. Hevner. Karlsruher Institut für Technologie (KIT). 2017, pp. 138–146 (cit. on p. 40).
- [SS17b] B. Sturm, A. Sunyaev. “You Can’t Make Bricks Without Straw: Designing Systematic Literature Search Systems.” In: (2017). Ed. by A. for Information Systems, pp. 1–9 (cit. on p. 40).
- [SSS15] B. Sturm, S. Schneider, A. Sunyaev. “Leave No Stone Unturned: Introducing a Revolutionary Meta-search Tool for Rigorous and Efficient Systematic Literature Searches.” In: *Proceedings of the 23rd European Conference on Information Systems ECIS*. 2015, pp. 1–10 (cit. on p. 40).
- [TESW15] S. Tilkov, M. Eigenbrodt, S. Schreier, O. Wolf. *REST und HTTP: Entwicklung und Integration nach dem Architekturstil des Web*. dpunkt.verlag GmbH, Apr. 28, 2015. ISBN: 9783864901201 (cit. on pp. 71, 72).
- [TUM16] Universitätsbibliothek, Technische Universität München. *Literaturverwaltungsprogramme im Vergleich*. Technische Universität München, Universitätsbibliothek. Aug. 3, 2016. URL: <https://mediatum.ub.tum.de/1316333> (cit. on pp. 22–24, 27, 28, 31–33, 35–39).
- [Ude16] J. Udell. *An Illustrated Taxonomy of Annotation Types*. Jan. 6, 2016. URL: <https://web.hypothes.is/blog/varieties-of-hypothesis-annotations-and-their-uses/> (cit. on p. 40).
- [Wor16] Wordfence. *Password Authentication and Password Cracking*. Feb. 15, 2016. URL: <https://www.wordfence.com/learn/how-passwords-work-and-cracking-passwords> (cit. on p. 73).

All links were last followed on November 16, 2017.

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

place, date, signature