

Christian Markus HOFER

Reference Database for Scientific Literature Organization

Bachelor Thesis



Institute for Genomics and Bioinformatics

Graz University of Technology

Petersgasse 14, 8010 Graz

Supervisor:

Dipl.- Ing. Bettina Halwachs

Evaluator:

Dr. Gerhard Thallinger

Graz, July, 2013

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Graz,

(date)

.....

(signature)

Abstract

The number of journal articles, books, proceedings and other resources of scientific literature are continuously increasing. Partially, these resources get digitalized and further increase the already huge number of digital resources flooding scientist's desktops. Therefore, adequate literature organization is absolutely indispensable. The market recognized this issue and offers useful reference management software tools using huge bibliographic databases for an efficient way of management. Despite the large number of reference management applications available it is not ensured that the needed functional scope is covered. Thus, sometimes individual solutions have to be developed. The main focus of this thesis was the customized re-implementation of the Reference Database for the Institute for Genomics and Bioinformatics providing enhancement in processing and improved management capabilities such as edit and delete functions to simplify the everyday life of scientists. Also the importance of online databases providing information about resources and their close connection to reference management software is pointed out. Additionally, several familiar software applications within this context are mentioned and partially compared. Finally, the benefits of the developed in-house application compared to popular existing tools are discussed. The Reference Database is realized using web technologies such as HTML, JS and PHP providing high portability and license free development. A MySQL databases was implemented for proper storage of resource information. Finally, a well customized Reference Database meeting the requirements of the Institute for Genomics and Bioinformatics was developed. The functional scope and the usability in comparison to the predecessor were significantly increased.

Keywords: Literature Organization, Reference Management, Citation, Web-Development, Management Application

Abstract

German

Die Anzahl der Journale, Bücher, Proceedings und anderer wissenschaftlicher Dokumente steigt kontinuierlich. Diese Schriftstücke werden des Öfteren digitalisiert und vergrößern zusätzlich die beträchtlichen Mengen an digitalen Ressourcen der Forscher. Aus diesem Grund ist eine vernünftige Literaturverwaltung unabdingbar. Der Markt reagierte auf diesen Bedarf mit einer Vielzahl von verschiedenen Softwarelösungen um die Literaturverwaltung effizient und einfach zu gestalten. Diese Charakteristiken werden zusätzlich durch den Zugriff auf bibliographischen Datenbanken verstärkt. Trotz der Mannigfaltigkeit an verfügbarer Software wird der Bedarf nicht vollständig gedeckt und individuelle Softwarelösungen müssen entwickelt werden. Das Hauptziel dieser Bachelorarbeit bestand darin, eine Reimplementierung und Funktionserweiterung der bestehenden Referenzdatenbank am Institute for Genomics and Bioinformatics durchzuführen. Die Verwaltungsmöglichkeiten wurden u.a. durch das Hinzufügen von Editier- und Löschfunktionen erweitert. Zudem wird in dieser Arbeit die Signifikanz von Onlinedatenbanken, welche Metadaten von digitalen Ressourcen für u.a. Literaturverwaltungssoftware zur Verfügung stellen, aufgezeigt. Am Markt etablierte Softwarelösungen für Literaturmanagement werden in unterschiedlichen Kategorien miteinander verglichen. Die im Zuge dieser Arbeit erstellte Referenzdatenbank wurde durch den Einsatz von lizenzfreien Webtechnologien wie z.B. HTML, JS und PHP entwickelt, um auf diversen Geräten und Systemen eingesetzt werden zu können. Eine MySQL Datenbank wurde für die Speicherung der Dokumenteninformationen designt und implementiert. Das Resultat dieser Arbeit ist eine den Anforderungen entsprechende Referenzdatenbank, welche den Funktionsumfang des Vorgängers beträchtlich erweitert und Verbesserungen in Bezug auf Benutzerfreundlichkeit bietet.

Stichwörter: Literaturverwaltung, Referenzmanagement, Zitation, Webentwicklung, Managementapplikation

Table of contents

LIST OF FIGURES	III
LIST OF TABLES	IV
LISTINGS	V
GLOSSARY	VI
1 INTRODUCTION	1
2 METHODS	4
2.1 Technologies	4
2.1.1 Hypertext Transfer Protocol (HTTP)	4
2.1.2 Transmission Control Protocol / Internet Protocol (TCP/IP)	5
2.1.3 HyperText Markup Language (HTML)	5
2.1.4 Cascading Style Sheets (CSS).....	5
2.1.5 Document Object Model (DOM)	6
2.1.6 JavaScript (JS)	6
2.1.7 Document Type Declaration (DOCTYPE)	7
2.1.8 Extensible Markup Language (XML)	7
2.1.9 JavaScript Object Notation (JSON)	7
2.1.10 Asynchronous JavaScript and XML (AJAX).....	8
2.1.11 PHP: Hypertext Preprocessor (PHP)	8
2.1.12 Relational Database Management System (RDBMS).....	9
2.1.13 My Structured Query Language (MySQL)	9
2.2 Tools.....	9
2.2.1 phpMyAdmin	9
2.2.2 phpDocumentor.....	10
2.2.3 Apache HTTP Server Project.....	10
2.2.4 Netbeans.....	10
2.2.5 Mockingbird.....	11
2.2.6 CrossRef Meta Data Search	11
2.2.7 Entrez Programming Utilities (eUtils)	11
2.3 Libraries and plugins	12
2.3.1 JQuery	12

2.3.2	JQuery UI.....	12
2.3.3	Uploadify	12
2.3.4	SimpleXML	13
2.3.5	Mysqli	13
3	RESULTS.....	14
3.1	Web application	14
3.1.1	Search interface.....	15
3.1.2	Upload interface	18
3.1.3	Export interface.....	20
3.2	Design and implementation	21
3.2.1	MySQL database.....	21
3.2.2	PHP structure and functionalities	22
3.2.3	Structure and design of the user interface	26
3.2.4	Communication between server and client	27
3.2.5	JS structure.....	28
4	DISCUSSION	30
4.1	Comparison to the previous version of the Reference DB	30
4.2	Comparison to bibliographic libraries and RM software	31
4.3	Conclusion and Outlook.....	32
5	BIBLIOGRAPHY	33
6	APPENDIX	36
6.1	Functional scope of the previous Reference DB.....	36
6.2	User requirements document (URD)	38
6.3	Reference types	41

List of figures

Figure 2.1: HTTP Communication between client and server	4
Figure 3.1: Main screen and search interface.....	15
Figure 3.2: Multiple search.	16
Figure 3.3: Results screen.	16
Figure 3.4: Reference editor.....	18
Figure 3.5: Upload screen.	19
Figure 3.6: Manual upload screen.	20
Figure 3.7: Export user interface.	21
Figure 3.8: Relational database schema of the Reference DB.	22
Figure 3.9: Simplified UML class diagram.....	24

List of tables

Table 1.1: Feature comparison between RM software.....	3
Table 3.1: Icon description of the document entries in the result screen	17

Listings

Listing 2.1: HTML element example	5
Listing 2.2: CSS rule example.....	6
Listing 2.3: DOCTYPE example	7
Listing 2.4: JSON example	8
Listing 2.5: Code snippet showing the method “getStartPage” and its DocBlock.	10
Listing 2.6: CrossRef metadata request example	11
Listing 2.7: eUtils metadata request example.	12
Listing 2.8 XML snippet of a PubMed response	13
Listing 2.9: Access to the PMID element via SimpleXML	13
Listing 3.1: HTML-structure of the manual book upload screen.....	27

Glossary

AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
CSS	Cascading Style Sheets
DB	Database
DBMS	DB Management System
DNA	Deoxyribonucleic Acid
DNS	Domain Name Service
DOCTYPE	Document Type Declaration
DOI	Digital Object Identifier
DOM	Document Object Model
DTD	Document Type Definition
GUI	Graphical UI
HT	HyperText
HTML	HT Markup Language
HTTP	HT Transfer Protocol
ID	Identifier
IF	Impact Factor
IP	Internet Protocol
ISBN	International Standard Book Number
JRE	Java Runtime Environment
JS	JavaScript
JSON	JS Object Notation
IDE	Integrated Development Environment
LO	Literature Organization
MEDLINE	Medical Literature Analysis and Retrieval System Online
MySQL	My Structured Query Language
NCBI	National Center for Biotechnology Information
OO	Object-Oriented
OOP	OO Programming
OS	Operating System
PDF	Portable Document Format
PMID	PubMed ID

RDBMS	Relational DB Management System
RM	Reference Management
TCP	Transmission Control Protocol
UI	User Interface
UML	Unified Modeling Language
URD	User Requirements Document
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
W3C	WWW Consortium
WWW	World Wide Web
XML	Extensible Markup Language

1 Introduction

Literature Organization (LO) is very important for scientists because of the high and continuously increasing number of (digital) resources. That is why applications providing LO are absolutely mandatory in scientific disciplines. Within this thesis digital LO was required to meet the following needs:

- Uploaded resources should be managed (upload, search, delete, edit) and read.
- Metadata (data about data, e.g. author name, book title, journal issue of a resource) should be automatically retrieved from (external) databases like PubMed or eventually from the PDF itself. If they are not available they have to be manually provided by the user.
- The user has to be able to change the stored metadata. Therefore, an editor has to be available.
- Reference export functionality for various types of reference managers such as EndNote has to be offered.
- These functionalities have to be accessible in an intuitive and efficient way, that is, easy to use and the execution times should be short. Access has to be granted via a web interface in the intranet or via the web using an authentication mechanism.

Databases providing the metadata of documents are important. They help to keep the stored documents manage- and maintainable. Much time can be saved because the metadata is retrieved automatically and has not to be typed in manually. Thus, users can devote their precious time to their main issues. Important databases for such a purpose within this thesis are PubMed and CrossRef.

PubMed [1] is a digital library provided by the National Center for Biotechnology Information (NCBI). It contains more than 22 million citations from more than 19 000 life science journals for biomedical articles. Usually the abstract and the link to the full-text are provided. The full-text is not always freely available. PubMed Central for instance, which is a subset of PubMed provides only articles from PubMed whose full-text is freely available. The available metadata

of the publications on PubMed can be automatically exported via the entrez Programming Utilities (eUtils), see section 2.2.7 to Extensible Markup Language (XML) and Medical Literature Analysis and Retrieval System Online (MEDLINE) flat-file [1–3].

CrossRef [4] is a membership association and was founded by scholarly publishers. It helps publishers to work collectively to provide users access to primary research content. Furthermore, it is the Digital Object Identifier (DOI) link registration agency. The citation linking network of CrossRef contains about 60 million registered DOI links. The metadata of the documents is stored in a database (DB) which can be queried for free. In contrary to PubMed the abstract is not contained in the DB. Like the full-text, the abstract remains at the site of the publisher who decides about the access to the contents [4–6].

The reference formats are used by different *Reference Management (RM) applications*. RM software fulfills a variety of helpful tasks, which vary with the program used. Organizing, searching or sorting operations within the bibliography of the RM software are important features. The support for word processor integration is essential in writing publications. Such a feature generates automatically the bibliography of an article by citing the paragraphs to their corresponding references. Import functions to integrate metadata from bibliographic DBs are comfortable and time saving. Obviously, such a feature is not supported by all the RM applications. The possibility to annotate references, the ability to share the stored data and data migration possibilities have to be considered by choosing RM software. Well-known RM applications are for example *EndNote* [7], *Mendeley* [8], *Reference Manager* [9], *RefWorks* [10] and *Zotero* [11] [12, 13].

These RM solutions cover all the features mentioned above. Great differences are not visible on first sight. Their respective purchase costs may be a relevant factor in deciding what to use. Mendeley and Zotero are freely available. Web-based solutions are important in that they guarantee platform independency. They offer reference sharing functionalities and users may access their database from every device that supports a web browser. A web-based version is not available for the Reference Manager, which is usable with Microsoft Windows only. A general overview of the features supported by the different RM applications is shown in table 1.1 [14].

	Endnote	Mendeley	RefWorks	Zotero
Supported bibliographic DBs				
PubMed	X		X	X
Scopus			X	
Web of Science	X			X
Bookmarklet	X	X	X	X
Online storage, sharing				
PDF files		X	X	X
Public folders		X	X	X
WWW	X	X	X	X
Desktop versions (supported operating systems)				
Windows	X	X	X	X
Mac	X	X		X
Linux		X		X
PDF support				
Extract metadata	X	X		X
Full-text search	X	X	X	X
PDF viewer		X		
File organizer		X		
Supported applications for word processing				
Microsoft Word	X	X	X	X
Open Office	X	X		X
Latex		X		

Table 1.1: Feature comparison between RM software [15]

The aim of this thesis was to create a Reference DB for scientific literature organization providing

- the full-text of the literature resource,
- automatic metadata retrieval from the external PubMed and CrossRef DBs to simplify the upload of new sources,
- manual upload feature for more independency and flexibility,
- an easy to use front end for quick access and modification of the references,
- support of exporting mechanisms to Reference Manager, EndNote and Biblatex.

The requirements are summarized in detail within the User Requirements Document (URD), see section 6.2.

The implemented Reference DB has the purpose to be only used in-house. Advantages of such an in-house system are that only relevant documents and their references are stored and that the access can be controlled within the organization.

2 Methods

2.1 Technologies

2.1.1 HyperText Transfer Protocol (HTTP)

The hypertext transfer protocol is responsible for data transmission through the World Wide Web (WWW) according to the client server model, see figure 2.1. Every browser uses this protocol as client to retrieve information from a related host. HTTP ensures that the integrity of the sent data remains safe [16].

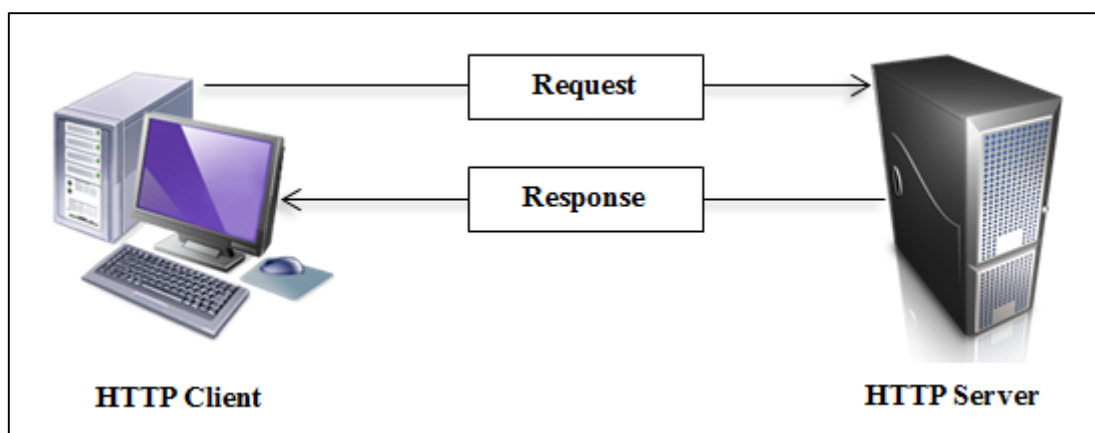


Figure 2.1: HTTP communication between client and server: The client sends an HTTP request to the server, which corresponds to the rules of the HTTP standard. The request consists of the resource the client wants to retrieve and eventually additional information. The server reads and interprets the message and sends an adequate response to the client. The response contains information about the status of the request and in case of a successful processing the requested resource [17].

In the following, the communication between client and server is explained in more detail. The client sends an HTTP request to the server. This message corresponds to the rules of the HTTP standard. The resource which the client wants to retrieve as well as additional information for the server is included. The message is read and interpreted by the server. The actions necessary for an adequate response to the client are taken by the server. This message contains the information about the success of the request as well as the content of the requested source in case of a successful response [17].

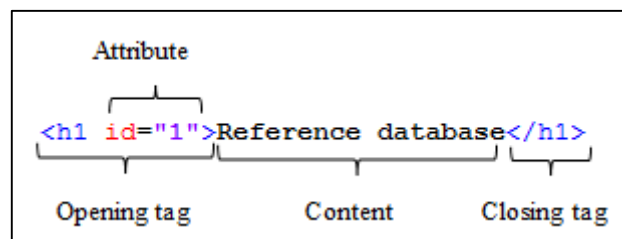
HTTP is based on the Transmission Control Protocol / Internet Protocol, see section 2.1.2, which regulates the network communication [16].

2.1.2 Transmission Control Protocol / Internet Protocol (TCP/IP)

HTTP needs TCP/IP for its message transport from the client to the server or vice versa. IP addresses and ports are used to establish the connection. The Uniform Resource Locator (URL) given in the browser is converted by the Domain Name Service (DNS) into the corresponding IP address. If no port in the URL is given, port 80 is assumed. The purpose of the domain is that it is much easier to comprehend for humans than hard to interpret IP addresses. TCP provides an error free transportation and is responsible for the network communication [16].

2.1.3 HyperText Markup Language (HTML)

HTML is the most widely used language on the internet. It is a markup language, which means that it is used to structure the content of web pages. In general HTML is presented by a web browser. Through the structure of the document it is possible to distinguish between headings, sections, paragraphs, tables, etc. These elements are called tags. Most tags consist of an opening and a closing tag. Together with the content, which is located between the opening and the closing tag an element is created. Attributes, which can be defined in the opening tag, provide additional information about the element. Important attributes are, for example *class*, *id* or *title*. Listing 2.1 illustrates an HTML element [18].

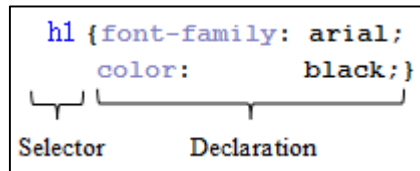


Listing 2.1: HTML element <h1> example: For the predefined heading 1 tag, id “1” is specified. The heading itself is “Reference database”, which is the content of the element and finally shown in the browser. The heading content is surrounded by its opening and closing tags. The opening tag may contain attributes providing additional information about the element [18].

2.1.4 Cascading Style Sheets (CSS)

In preceding versions of HTML it was common to include the style markup in HTML. By the time HTML 4 was released the World Wide Web Consortium (W3C) decided to introduce a new language to be able to distinguish between the structure and the appearance of a web page. CSS was born. The separation of structure and style allows the programmer to adjust the format of the content in a comfortable way. For example, the appearance of headings can be specified,

see listing 2.2. The selector states the elements, which are formatted in the style of the declaration. In this example the appearance of heading 1 (<h1>) is specified. Each <h1> element will appear in a black Arial style in the browser [18, 19].



Listing 2.2: Definition of a CSS rule for heading 1: All heading 1 elements will be displayed in a black “Arial” font. The selector of a CSS rule contains the elements which have to apply to the rule. This selector is followed by the declaration of the CSS rule defining properties for the element [19].

CSS can be applied to change the appearance of huge web sites in very short time. Another advantage is the possibility to adapt the presentation to different types of devices like smart phones or tablets quite easily. Typically CSS specifications are located in a separate file [18, 19]

2.1.5 Document Object Model (DOM)

The DOM is a cross platform and language independent user interface. This interface offers the access and modification of the structure and the style of a document using the DOM. The DOM is used in HTML and XML. It may present these types of files in a structured tree for a good overview of the document. Within the UI of the Reference DB the DOM is manipulated several times through various JS functions [18, 20, 21].

2.1.6 JavaScript (JS)

JS is a scripting language which partially supports the object-oriented (OO) paradigm. It is a cross-platform, lightweight language. JS is predestinated to be embedded in other applications. Typically such a host environment is in a web browser. JS may access objects provided by the host environment to gain more control. The use of JS allows the programmer to generate a dynamic web page by processing user input. The role of JS is neither the structure nor the presentation, but the organization of the behavior of the web page. JS can be defined in an HTML document using the script tag or in an external file. JS runs both, on the client and the server. On both, it extends the core language by making objects available. On the client’s side this means for example that via these objects the DOM may be manipulated. Other important examples are reactions to events generated by the user when they submit a form, push a button, etc. On the server JS provides a broad range of functionalities for usual server activities such

as DB access and file management. This project applies JS on the client side and is implemented in a functional manner [18, 22, 23].

2.1.7 Document Type Declaration (DOCTYPE)

The DOCTYPE provides space for the Document Type Definition (DTD) and the root element of the document. The DTD explains the valid structure of the document. It may reference to an external file and / or may be declared within the document. The DOCTYPE is very important because it specifies how the content of the document should be interpreted by the browser. Listing 2.3 shows two example DOCTYPES [24–26].

HTML 5	{	<code><!DOCTYPE html></code>
XHTML 1.0 Strict	{	<code><!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd"></code>

Listing 2.3: DOCTYPE declaration examples: In an HTML file the DOCTYPE is the first appearing element. This listing illustrates the declarations for HTML 5 and XHTML 1.0 Strict [25].

2.1.8 Extensible Markup Language (XML)

As HTML, XML is a markup language to structure data. XML looks very similar to HTML. The big difference between these two languages is, that XML is a meta markup language, which means, that it does not have a fixed number of valid tags like HTML. It is not used for presenting or viewing data. It is mainly used for data exchange, storing and describing different sorts of data. The data can be easily read and manipulated by humans and computers. Computer programs may exchange data by using XML documents. XML documents may be easily extended, which makes it very flexible. Any XML reader which could read the former version of an XML file will read the extended one without any modifications. The new inserted information will not be recognized but this can easily be adapted. An XML file can assume any shape because of the potentially unlimited number of valid tags [27].

2.1.9 JavaScript Object Notation (JSON)

The JSON notation is completely language independent and used like XML to interchange data. The data encoded in JSON is very lightweight, because data can only be sent in two different structures: On one hand as a list of values (array) and on the other hand as a collection of key value pairs. JSON data can be easily read and written by humans as well as computers. In the Reference DB this notation was chosen because of its simplicity and is used for sending data

between the server and the client. Listing 2.4 illustrates the structure of a JSON object using key value pairs [28].

```
{
  "title": "reference database",
  "author": {
    "first_name": "Max",
    "last_name": "Mustermann"
  }
}
```

Listing 2.4: JSON example using a collection of key value pairs: The key of a JSON object is located on the left. “title” and “first_name” are the keys for the values “reference database” and “Max”, which are followed by the key. The key value pairs may be interleaved as “author” shows [28].

2.1.10 Asynchronous JavaScript and XML (AJAX)

The acronym AJAX reveals that it is a collection of different technologies. The asynchronous part means that the browser is not on hold during data transfers from the server. Synchronous would mean that the browser has to wait until the data arrives from the server and this could result in a problem e.g. with a slow internet connection. JavaScript takes a significant position, because it handles the transferred data. The XML part signals that the data sent from the server is encoded in XML, but it is possible to return different text types. AJAX creates a more desktop like feeling for the user. This desktop feeling is caused by the asynchronous mechanism of AJAX. Instead of reloading or loading the whole site (synchronous), new content only gets pasted into the current site (asynchronous) [29].

2.1.11 PHP: Hypertext Preprocessor (PHP)

PHP is a server side programming language which means that the files are executed on the server after the HTTP request from the client. During this process the HTML or PHP file is generated and after the completion the file is returned to the client. Like JS, PHP makes HTML files dynamic. PHP code may be entered directly into an HTML file or separated into own files within the `<?php` tag. The file extension has to be `php` because the web server has to know that it is PHP code that has to be interpreted. The code is replaced by the output of the script. Web development using PHP is comfortable, because a lot of useful functions and libraries are included in the framework, which help for example accessing a DB or manipulating XML files. Due to the fact that PHP does not store data it is common to combine this technology with DBs. My Structured Query Language (MySQL) DBs work very well in combination with PHP. The

teamwork of PHP and MySQL is intended by the developers and was considered during development. Benefits of both technologies are that they run on different operating systems (OS) and that they are freely available. PHP may be programmed in a procedural or OO way [30].

2.1.12 Relational Database Management System (RDBMS)

The DB represents the stored data, which is organized in tables. Tables consist of columns and rows, where each row represents a data record. A data record may consist of various pieces of information and each piece corresponds to a column of the table. “Relational” means, that information stored in one table may relate to stored data in another table. A management system allows the user to insert, modify, delete or retrieve information [31].

2.1.13 My Structured Query Language (MySQL)

MySQL is a free multi-threaded and multi-user RDBMS, which works well in combination with PHP. SQL is the most popular language for data manipulation in DBs. “Multi-threaded” means, that different back ends, client applications and libraries, programming interfaces and administration tools are supported. Multiple users may access multiple databases. MySQL supports a number of various storage engines, which gives the developer the opportunity to select the appropriate engine with the desired properties. Furthermore, it offers SQL server, client programs for accessing the server and libraries for accessing the DB through user programs. The tools provided enable various functions such as storing, sorting, searching and retrieving data [30–32].

2.2 Tools

2.2.1 phpMyAdmin

phpMyAdmin¹ is a free browser-based DB administration tool. The tool helps managing MySQL DBs. It provides an intuitive graphical user interface (GUI). With the phpMyAdmin it is easy to create, modify, delete, import, query or import the DBs, tables, entries, fields and much more. All types of queries can be executed in the SQL tab where individual MySQL queries can be built and executed. Additionally, simple queries such as viewing or sorting the data of a table are already integrated into the GUI and can be easily executed by a single click [33].

¹ http://www.phpmyadmin.net/home_page/index.php

2.2.2 phpDocumentor

This valuable tool helps to document the PHP source code. It generates from an external documentation and the source code a useful documentation. The tool parses the logical structure of the code e.g. classes, methods, constants, etc. and generates an interactive documentation. The phpDocumentor¹ also parses the so called DocBlocks and integrates them into the final documentation. DocBlocks are comments implemented by programmers, which provide helpful information for users. Listing 2.5 illustrates a DocBlock for a method. Through converters the documentation can be stored e.g. as a PDF [34].

DocBlock describing the method getStartPage	}	<pre>/** * returns the start page of the specified journal xml. * @param SimpleXMLElement \$journal * @return string start_page */ private function getStartPage(\$journal) { \$xml = \$journal->MedlineCitation->Article->Pagination; if(!isset(\$xml)) return ''; \$pieces = explode("-", \$xml->MedlinePgn); return \$pieces[0]; }</pre>
Method getStartPage		}

Listing 2.5: Code snippet of the Reference DB showing the method “getStartPage” and its corresponding DocBlock describing the method. For a more precise description tags may be used. A tag is introduced by the “@”. In this snippet the types of the parameter and the return value are declared using the tags “@param” and “@return”. The method gets a SimpleXMLElement and returns a string [34].

2.2.3 Apache HTTP Server Project

This is a free HTTP/1.1 compliant web server, which can be applied on different operating systems including Windows and UNIX. The web server provides the client(s) with the requested data. The Apache web server interprets the PHP code. Modules which extend the functionality of the server allow a faster processing of the PHP code. The advantages of the Apache web server are that it is free, its unrestrictive license and the accessible source code [30, 35].

2.2.4 Netbeans

This integrated development environment (IDE) is programmed in Java and freely available. It offers cross platform support, multiple programming languages and is expandable through different plugins. It is possible to gain access to DBs like MySQL and software revision systems like Apache Subversion (SVN) [36].

¹ <http://phpdoc.org/>

2.2.5 Mockingbird

Mockingbird¹ is a browser-based mockup application. A restricted version of this application is freely available. In this mode the creation of two or more projects is not granted. The mockup can be created quickly by drag and drop of a bunch of tools like buttons, pictures, input fields, etc. The created wireframes may be stored online or exported into a PNG or PDF file [37].

2.2.6 CrossRef Meta Data Search

The CrossRef Meta Data Search² allows the user to query available metadata information about any kind of digital resources such as books, journals, music, pictures, etc. Within the scope of the Reference DB this specific term could be for example the author of a book, the title of a journal or the Digital Object Identifier (DOI). The DOI is a persistent unique identification for all kinds of objects. CrossRef offers programmers an application programming interface (API) to query the DB via a specific URL. Listing 2.6 shows the structure of the URL. The database is searched and the CrossRef server will immediately respond an XML file containing the sorted metadata for the requested resource [4, 38].

```
http://api.labs.crossref.org/10.1016/S1097-2765(03)00225-9.xml
```

The diagram shows the URL `http://api.labs.crossref.org/10.1016/S1097-2765(03)00225-9.xml` with three brackets underneath. The first bracket is under `api.labs.crossref.org` and labeled "Domain". The second bracket is under `10.1016/S1097-2765(03)00225-9` and labeled "DOI". The third bracket is under `.xml` and labeled ".xml".

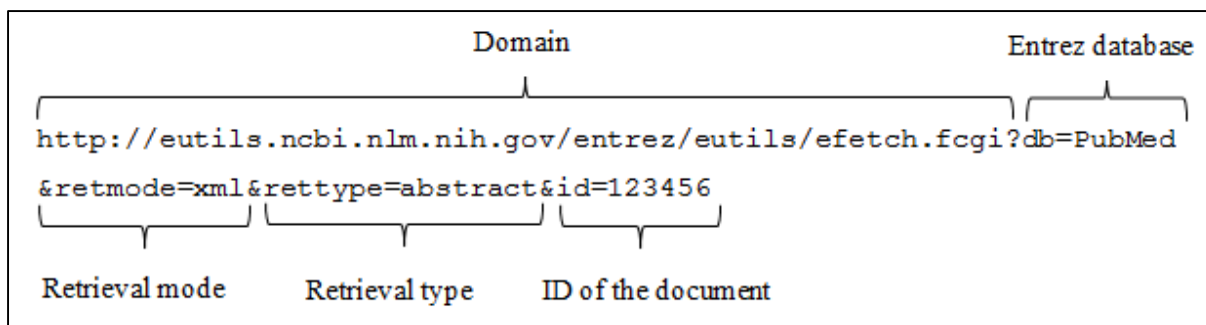
Listing 2.6: CrossRef metadata request example: The request starts with the domain of the API. Followed by the slash and the DOI of interest and “.xml” the meta-information of the documented may be accessed [4].

2.2.7 Entrez Programming Utilities (eUtils)

eUtils is an interface for programmers to query the different DBs of the NCBI. It is the primary system regarding to search and retrieval. Through this service the access to PubMed is granted. PubMed offers tens of millions citations of literature including the abstract quite often. Via a specific URL, see Listing 2.7, followed by the PubMed Identifier (ID) of a document its metadata can be queried. After the HTTP request, the eUtils translate the received parameters into the values needed for the NCBI software. The requested data is retrieved from the database and eUtils returns the XML document containing the queried data [1, 39, 40].

¹ <https://gomockingbird.com>

² <http://search.labs.crossref.org>



Listing 2.7: eUtils metadata request example: The meta-information of a document can be requested via the illustrated URL. The domain is followed by optional parameters. “db” declares the DB (PubMed). “Retmode” declares the format of the response (XML). The retrieval type is used to get access to the abstract of the document. At last the ID, in this case the PubMed ID (PMID) has to be declared. This request returns the metadata and the abstract of the document with the PMID “123456” in the XML format. A response snippet is illustrated in listing 2.8 [39].

2.3 Libraries and plugins

2.3.1 JQuery

JQuery is a JavaScript library which simplifies the writing of scripts on the client side. JQuery is freely available. A great advantage of this library is the support for cross-browser development. Due this property the use of AJAX within the JQuery library is simplified, because the AJAX API is implemented differently in various browsers. Furthermore, the selector enables a simple manipulation of the DOM. Effects, animations or hiding content can be applied comfortably. Extensions like JQuery UI make this library even more powerful [41, 42].

2.3.2 JQuery UI

The JQuery UI extends the functionality of the regular JQuery library. Extended functionalities in the categories interactions, widgets, effects and utilities are offered. Dialog boxes, date pickers, autocompletion (during user input word proposals are made for faster input) and progress bars are just some of the features the JQuery UI is offering. Like JQuery, JQuery UI is available for free [43].

2.3.3 Uploadify

Uploadify¹ is a useful tool for multiple file upload based on the JQuery library. There is a HTML5 and a flash version available. In this project the flash version of Uploadify is used, because the HTML5 version is not available for free. The requirements of this tool are of course

¹ <http://www.uploadify.com/>

the JQuery library (1.4.x or greater), the Flash Player (9.0.24 or greater) and a server side handler in this project realized in PHP, which parses the incoming data [44].

2.3.4 SimpleXML

This PHP library is used in the reference DB for reading the XML files returned by CrossRef and PubMed. The advantage of this library is its ease of use. An object is created of an XML file by calling the constructor of the library and passing the path of the XML file. Listing 2.8 and listing 2.9 (xml is a SimpleXML object) illustrate the simplicity of addressing an element [45, 46].

```
<?xml version="1.0"?>
<!DOCTYPE PubmedArticleSet PUBLIC "-//NLM//DTD PubM
<PubmedArticleSet>
<PubmedArticle>
  <MedlineCitation Owner="NLM" Status="MEDLINE">
    <PMID Version="1">123456</PMID>
```

Listing 2.8 XML snippet of a PubMed response

```
xml->PubmedArticle->MedlineCitation->PMID
```

Listing 2.9: Access to the PMID element of a PubMed XML object via SimpleXML

2.3.5 Mysqli

The Mysqli library allows the user to access the functionalities of the MySQL instance. Through the constructor of this PHP extension a connection to the DB can be established. The constructor is used when an object-oriented programming style is applied. Otherwise, the connect function is called. An object of this class may be used for manipulations of the DB in a comfortable way. A lot of options are such as automatic commitment, prepared statements, multi queries, etc. are offered for a safe and comfortable use of the MySQL DB connection [47].

3 Results

In this section design as well as functionality of the developed Reference DB is described in more detail. The Reference DB implementation was realized using only freely available web technologies such as HTML, CSS, JavaScript and MySQL. The application allows users to manage their scientific literature, comprising uploading, reading, modifying and deleting functionalities for six different kinds of literature resources (books, book chapters, conference proceedings, journal articles, online sources, theses/dissertations). The upload supports automatic metadata retrieval from the DBs of PubMed and CrossRef, if the corresponding ID of the document is available. A comfortable search function allows the user to query the DB efficiently. Every stored source can be exported for further use e.g. in various RM software such as Reference Manager and EndNote. The available formats are ENW, RIS and TEX. The following sections describe the Literature DB in detail. Section 3.1 offers a description of the features and the handling; section 3.2 explains the structure of the web application.

3.1 Web application

In this section the UI of the Reference DB is described in more detail. The web interface is based on the former version of the Reference DB. Since several features have been added the appearance changed, but it was tried to keep the look and feel of the primary program. The application is divided into three different areas, which are reachable by using the tabs on the top, see figure 3.1 (1). Below the tabs the content of the selected tab is shown.

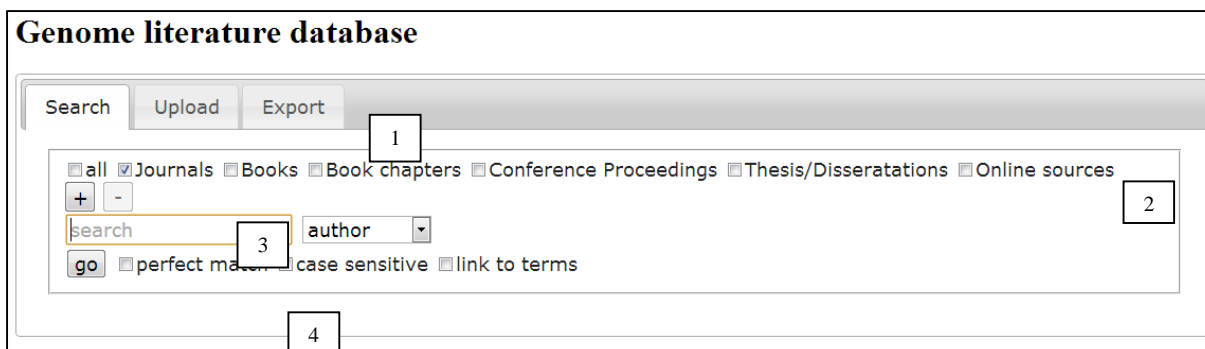


Figure 3.1: Main screen and search interface: The bar on the top contains the tabs linking to the three main areas of the application (1). The content of the selected tab is located below the tab bar. Search interface: Specific reference types may be selected for search (2). A keyword for search may be entered into the input box (3). Additional search fields may be added or removed by using the plus or minus buttons located above the input field. Next to the input box a dropdown field is located to define the category for search. The checkboxes (4) below the search field offer additional options. Beside these boxes the “go” button is located. By pressing this button the data is sent to the server which submits the query.

Figure 3.1 illustrates the home screen of the Reference DB comprising its three main features, *Search*, *Upload* and *Export*. These sections are working independently from each other. In other words a busy procedure in one tab does not block actions in another. For example, users are able to continue DB search, although an export is in progress or resources are imported.

3.1.1 Search interface

Since the centerpiece of the application is the search, the first screen after loading the program is the view of the search interface, see figure 3.1. The input box in the center is used to search for a keyword in the DB. The category can be selected via the drop down menu. The following categories are available for search: *Author*, *PMID*, *Journal*, *Title*, *Publisher*, *Editor*, *Institution*, *Keywords*, *Comments* and *Year*. To get a more specific result the check boxes above the search field allow querying the DB for specific types of references. The checkboxes at the bottom are adapted from the former Reference DB. “*Perfect match*” means that the attribute in the DB has to be identical to the entered string. “*Case sensitive*” performs an upper and lower case sensitive query. “*Link to terms*” highlights the keywords in the PDF file, which is available in the result section appearing after a successful query. Using the plus or the minus button it is possible to add up to eight search fields to get a more precise result, see figure 3.2.

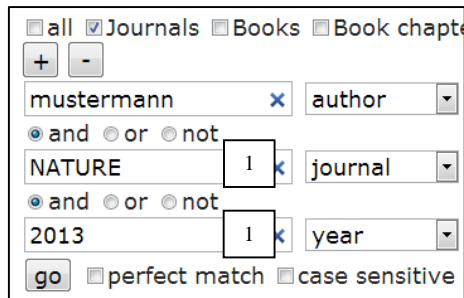


Figure 3.2: Multiple search: Logical connectives are offered if one or more input fields are added (1) AND, OR and NOT are available for refinement of the query.

Between the multiple search input boxes logical connectives can be chosen to refine the query, see figure 3.2. A click on the “go” button below the input boxes sends the request to the server and the query is executed. The results are shown on the same page below the search form, see figure 3.3.

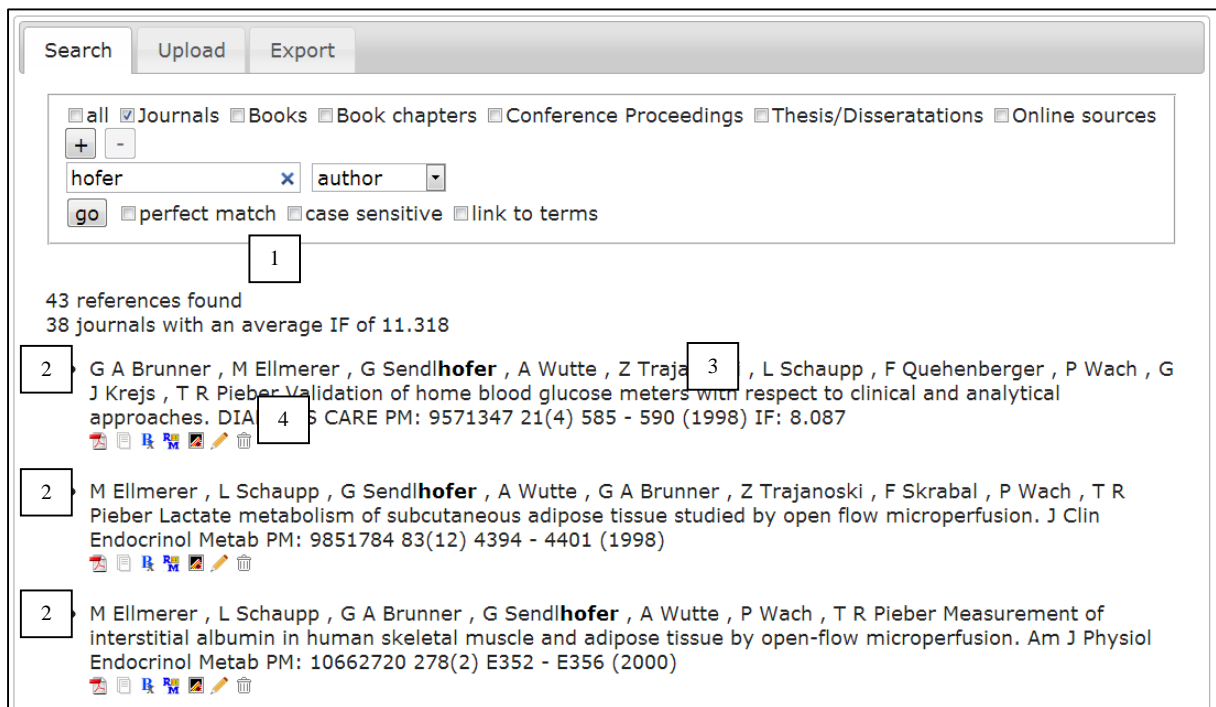


Figure 3.3: Results screen after a successful query: The summary of the query is located on the top (1). Below the summary the references found are shown (2). Each reference contains meta-information (3). Below the metadata an icon set for further options is displayed (4). Table 3.1 explains the meaning of each icon.

On the top of the results screen, see figure 3.3 the result summary is shown. It contains the number of found references and the mean impact factor (IF). The mean IF comprises journals which are rated in this way. The summary is followed by the list of found references. Each list element consists of two parts. The first one is the main part, which contains the reference meta-information, while the second is a set of icons beneath it. Depending on the document type

various attributes are displayed as meta-information. Table 3.1 describes the icons and their functions in more details.















Icon	Description
	Opens the corresponding PDF file in a new tab
	Opens the URL of an online source in a new tab
	Signals that there is no comment for the reference
	Comments are available and may be viewed by resting the mouse over the symbol
	A click induces TEX download of the reference
	A click induces RIS download of the reference
	A click induces ENW download of the reference
	Opens the editor of the reference below the reference
	Deletes the reference after a positive confirmation reply

Table 3.1: Icon description of the document entries in the result screen

Figure 3.4 shows the metadata editor of a Reference DB entry. It is possible to open multiple editors at a time. Mandatory fields, which means that these types of fields have to be filled, are highlighted with a blue margin. Fields such as “*Link*” are greyed out, signaling that this kind of field cannot be changed by the user. A click on the *tick* or the *cross* at the bottom on the right side closes the editor. The *tick* induces the actualization of the data in the browser and the DB on the server. The *cross* icon cancels the modifications made. The icons for the editor and the deletion of documents are only available if the IP address of the user is contained in a simple text file. From this it follows, that several IP addresses in the text file determine the administrators of the Reference DB.

• G A Brunner , M Ellmerer , G Sendlhofer , A Wutte , Z Trajanoski , L Schaupp , F Quehenberger , P Wach , G J Krejs , T R Pieber Validation of home blood glucose meters with respect to clinical and analytical approaches. DIABETES CARE PM: 9571347 21(4) 585 - 590 (1998) IF: 8.087

Title:

Authors:

Journal title:

Volume:

Issue: 1

Start page:

End page:

Pub date:

Impact factor:

Link: 2

Comments:

Keywords:

Abstract:

3

Figure 3.4: The editor of a reference: The editor contains all the information stored in the DB. The properties and its information are editable. The blue input boxes (1) have to be filled out otherwise the changes are not accepted. (2) Properties such as the "Link" are not editable and greyed out. The changes may be confirmed or canceled by the tick or cross icon at the bottom of the editor (3).

3.1.2 Upload interface

The upload interface, see figure 3.5 offers a comfortable way to add new resources to the DB.

It is currently possible to upload six different types of documents. These types are

- whole books,
- book chapters,
- conference proceedings,
- theses or dissertations,
- journal articles and
- online sources.

The metadata of documents dedicated to a PMID is automatically fetched from the server if it is provided by the filename or the corresponding input field. In the case that both include a PMID the input field has higher priority and is used for the metadata retrieval.

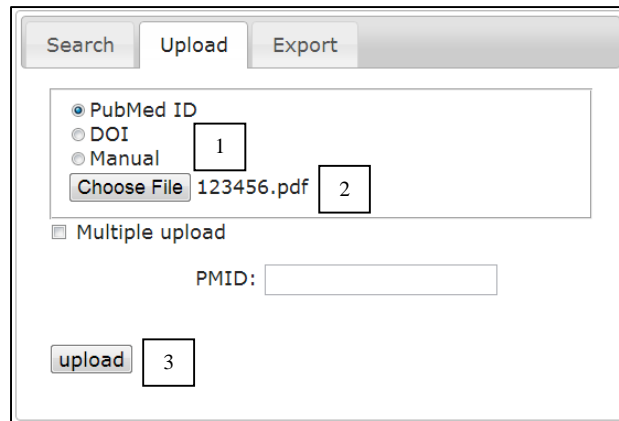


Figure 3.5: Upload screen: The upload offers three different upload methods that can be chosen via the radio buttons (1). The PDF file for upload has to be specified by using the “choose file” element button (2). The upload is committed by clicking the “upload” button located at the bottom of the screen (3).

The multiple upload is offered for PubMed journals only. This upload can be reached by clicking on the checkbox. The *choose file* element is greyed out if *multiple upload* is checked. A grey button appears at the bottom of the page and allows the selection of one or more PDF files. The files for the upload have to be named according to their PMID otherwise the multiple upload fails.

Documents dedicated by a DOI can be uploaded by a click on the DOI radio button. This feature uses CrossRef for metadata retrieval. The DB of CrossRef does not contain the metadata of every document identified by a DOI and it is possible that the requested metadata is not available. In this case a manual upload has to be considered.

Figure 3.6: Manual upload screen: The type of the document has to be assigned in the drop-down menu at the top (1). The available properties for the chosen document type are shown (2). Mandatory fields are signed by a blue border and have to be filled out (3).

Figure 3.6 shows the manual upload user interface. The document type has to be chosen. Beneath the drop-down menu the properties can be entered. As in the editor the blue margined input boxes are mandatory and have to be filled.

No matter which upload type is chosen a file has to be selected in the choose file element. Exceptions are the upload of an online source and the multiple upload of PubMed documents. In these cases the element is greyed out. The upload is committed by a click on the upload button at the bottom of the screen.

3.1.3 Export interface

The export interface offers the possibility to download the whole Reference DB in the desired reference format. These formats may be imported into various RM applications such as End-Note, Bibtex or Reference Manager. It is possible to sort the export file by author, date or title.

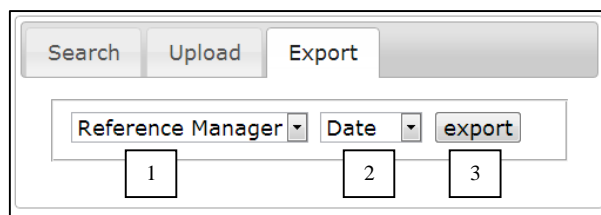


Figure 3.7: Export user interface: The type of the export format may be selected on the left side of the screen (1). The export file may be sorted by the date, author names or the title of the reference and is selectable by using a drop-down menu (2). The export button induces the download of the desired format (3).

Figure 3.7 illustrates that these options are selected via drop down menus. The export button starts the conversion process. Currently about 16000 references are stored in the DB. Therefore, the operation takes a while and during processing the export form is greyed out signaling the user that the procedure is in progress. After the export has finished a file download is started automatically.

3.2 Design and implementation

The Reference DB comprises a client and a server side. The server runs the MySQL instance and the PHP environment. This part of the application is responsible for a proper storage of the data provided by the client. The PHP layer manages the communication between the client and the DB. HTML, CSS and JS are used on the client side for structuring, representing and operating on this important part of the application. It follows a detailed description of the different parts, see section 3.2.1 – section 3.2.5.

3.2.1 MySQL database

The previous version of the Reference DB uses a simple text document for data organization. Thus, a clever design for data storage had to be reconsidered and adjusted. The implemented design of the DB had been kept simple for easier access as well as to create simple queries avoiding lots of joins to receive query result as fast as possible. On this account a MySQL DB comprising three main tables was created.

- A table for the different types of documents named *doc_types*.
- A second one, which consists of the different journals known from PubMed named *journals*.

- Finally, the main table, where the whole information of the documents is stored named *docs*.

Figure 3.8 shows the relational schema of the DB. The *InnoDB storage engine*¹ was used, which makes it possible to declare foreign keys to maintain referential integrity. This feature does not allow broken links between two tables. An example is the field *id_doctype* in the table *docs*. This field contains a foreign key from the field *id* of the table *doc_types*. Referential integrity ensures that *id_doctype* references always to a legal value of *id*.

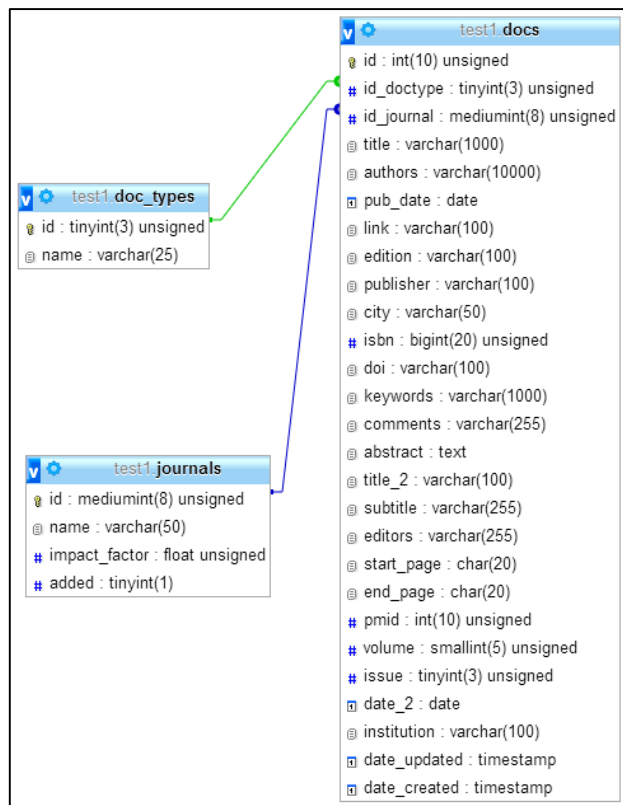


Figure 3.8: Relational DB schema of the Reference DB: The three main tables (*docs*, *doc_types*, *journals*) with their structure are shown. The tables “*doc_types*” and “*journals*” are related to the main “*docs*” table.

3.2.2 PHP structure and functionalities

The object-oriented programming (OOP) paradigm was applied to the PHP implementation. Each reference type has its own class, where the appropriate properties of the type are mapped. The different types of documents share some common properties like title, authors, etc. Therefore, a base class was created, comprising the shared properties. The document types are derived

¹ <http://dev.mysql.com/doc/refman/5.0/en/innodb-storage-engine.html>

from this class and extended by the unique properties needed. The controller on the other hand is implemented in a procedural manner. The controller uses the implemented classes, creates objects, communicates with the client and represents the “brain” on the server side. A simple switch-case handler grabs the information provided by the client and decides the follow up of the request. For this reason a procedural programming paradigm was used for the controller.

The access to the meta-information received from PubMed and CrossRef is granted by separate classes. A PubMed object grabs the XML file via the use of the eUtils service by providing the PMID. A CrossRef object gains access to the XML file via the CrossRef metadata search by providing the DOI. These classes fetch the information from the XML files and save the metadata of the documents in the adequate model, which is represented by the various document type classes. The XML file was read by the use of the *SimpleXML* library.

The application generates three different export formats. The supported formats are ENW, RIS and TEX and are used by various RM applications. Each format has its distinct class. An instanced object creates the file containing the information of a specific document from the data stored in the MySQL DB.

The DB access is accomplished via a singleton class. This class offers various methods for easy DB requests. Methods for modification, deletion and insertion of metadata are prepared. Additionally, some methods for system queries are implemented, such as, for example, a method that retrieves all stored journal names. The DB class is able to handle objects derived from the reference class. Finally, methods for migration of the former DB information are located in this class.

The email transmission system is based on a class named *email* which dynamically generates the message string sent to the subscribers of a particular mailing list. A simple handover of the reference object and the class creates a string which later represents the content of the email.

Finally, noteworthy elements of the PHP server side are the class including the constants of the application and the converter class. *Constants.inc.php* allows the adjustment of the Reference DB. The DB connection information, path of the PDF documents, recipients for the email func-

tionality and similar Constants may be adjusted in this file. *Converter.inc.php* converts the provided document data from the client into the convenient document type format required by the operations on the server side.

Figure 3.9 illustrates the class diagram implemented in PHP on the server.

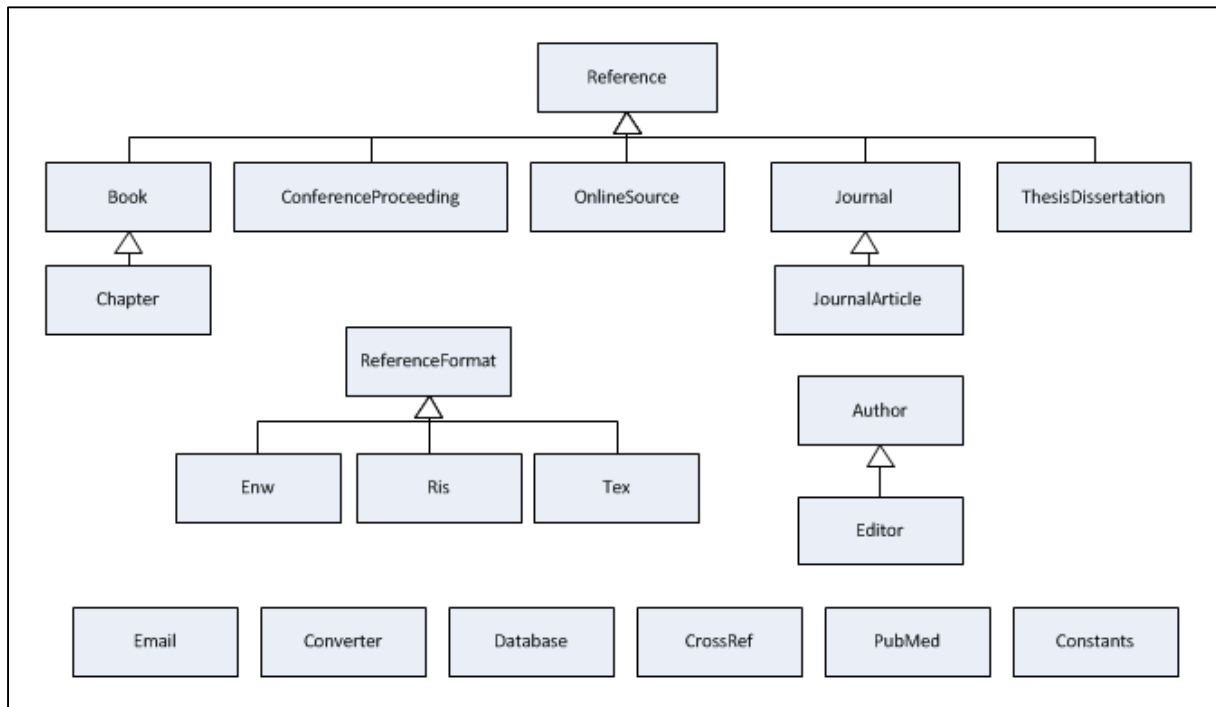


Figure 3.9: Simplified UML (Unified Modeling Language) class diagram

The classes illustrated in figure 3.9 in more detail:

- Reference:** This is the abstract base class for all document types. It contains the common properties shared by all types of documents such as comments, title and authors.
- Journal:** This abstract class is derived from *Referenc.inc.php* and contains the unique attributes of a journal such as volume and issue.
- JournalArticle:** *JournalArticle.inc.php* extends the properties of *Journal.inc.php*. For example, attributes for start and end pages were added. This class represents one of the six reference classes which may be instantiated.

- Book:** Book is derived from *Reference.inc.php*. It contains additional properties such as ISBN, subtitle and editor. This class represents the base class for the book chapter class.
- Chapter:** This class represents a book chapter. Additional properties are for example the start and end pages.
- ConferenceProceeding:** *ConferenceProceeding.inc.php* is like all types of documents derived from the *Reference* class. Unique properties such the proceedings title and the conference title were added.
- OnlineSource:** This class is derived from *Reference.inc.php* and is the only reference type where no additional properties were added. The base class contains all of the attributes needed.
- ThesisDissertation:** The base class of *ThesisDissertation.inc.php* is the class *Reference*. New attributes such as institution and publication were added.
- ReferenceFormat:** *ReferenceFormat.inc.php* represents an interface for the *Ris*, *Enw* and *Tex* classes. Objects of these classes generate the content of the export formats used by the RM software. A public create method is declared which is supposed to call one of the declared protected create methods existing for each reference type.
- Ris:** Implements *Reference.inc.php*. The constructor expects a reference object. A call of `createRef()` generates the content of the RIS export format. `createFile()` generates a new file where the generated content is written. `writeIntoFile()` appends the generated content to the file.
- Tex:** Similar to *Ris.inc.php*.
- Enw:** Similar to *Ris.inc.php*.
- Email.inc.php:** The constructor expects an object of the reference class. The content of the mail is created by calling the `createEmail()` method. A call of the method `sendMail()` sends the mail to the recipients defined in *Constants.inc.php*.
- CrossRef:** An instance of this class creates a reference object. Therefore, the constructor needs the DOI of the document. With use of the API of

the CrossRef metadata search the XML file containing the meta-information is fetched if available. The generated reference may be fetched by calling *getDocument()*.

- PubMed:** Similar to *CrossRef.inc.php*. Instead of the CrossRef metadata search the eUtils are used to fetch the XML file.
- Converter:** This class offers conversion methods. For example a fetched array from MySQL is converted to a reference type.
- Database:** This implemented singleton class provides all the methods for accessing and manipulating the MySQL DB. The user does not need to know SQL. *getInstance()* creates / returns the object of this class. The object provides several methods for querying the DB.
- Constants:** The class *Constants* consists of *const* properties only. No methods are available. The main purpose of this class is to simplify adjustments of the client side. For example DB access information is stored in this class which is used by *Database.inc.php*. Properties for *Email.inc.php* such as sender and receiver are adjustable.

3.2.3 Structure and design of the user interface (UI)

For an intuitive access to several functionalities and the possibility to perform different tasks at the same time, the application is split up into logical sections by using the JQuery UI tabs widget. The implementation is realized in the *index.php* where the particular divisions representing the tabs are located. This file contains all structured information and together with the main CSS file, it represents the complete UI of the application.

A well designed data structure for the HTML part had to be implemented, because the application has to be able to access the data for further treatment e.g. formatting styles with CSS, handover information to the server, etc. To gain a better understanding of the structure listing 3.1 explains how the data is organized by taking the example of the manual upload area.

```

<a class="title">
  <label>Title: </label>
  <input size="60" type="text" class="mandatory" name="title" id="title" required="required">
  <br>
</a>
<a class="link" style="display: none;">
  <label>Link: </label>
  <input size="60" type="text" name="link" id="link">
  <br>
</a>
<a class="authors">
  <label>Authors: </label>
  <input size="60" type="text" class="mandatory" title="Format: surname givenname given..." name="authors">
  <br>
</a>
<a class="pub_date">
  <label>Pub date: </label>
  <input size="60" type="text" class="mandatory" title="Format: yyyy-mm-dd (year-month-date)" name="pub_date">
  <br>
</a>

```

Listing 3.1: HTML-structure of the manual book upload screen: Each property of the book and its labels are grouped by using HTML classes using intuitive names. By knowing the class name simple access to the values of the properties is ensured. The input fields also contain classes to declare mandatory fields.

The HTML structure shown in listing 3.1 and similar structures allow simple data access. Each property of the selected document type and its labels are grouped by using HTML classes using intuitive names. By knowing the class name simple access to the values of the properties is ensured. The input fields also contain classes to declare mandatory fields. As an example, CSS gains access to mandatory input fields like the title of journal articles, authors of books, etc. and is able to highlight them and the user knows that the corresponding input box has to be filled out. This and similar structures are generated dynamically using JS and the related library JQuery.

3.2.4 Communication between server and client

A JSON object is returned from the server, which encodes mainly information of the reference types. The different document types and the attributes describing the documents are transmitted using this format. Less complex information like successful operation responses is returned via simple string outputs from the server. Most of the time the client sends objects used as associative arrays using AJAX to the server. The earlier mentioned controller of the server continues the process by decoding the submitted data. Depending on the submitted data the controller decides what has to be done. The controller decides for example if an XML is fetched, a DB entry is made and uses the methods provided by the classes explained in section 3.2.2.

3.2.5 JS structure

JS is implemented in a procedural manner. This is the main reason why the client side is not as well structured as the PHP part of the application. In contrast to the PHP server unit, the JS section does not contain a main controller where all decisions are made. Each main part of the client has its own controller and knows what has to be done.

According to a clear arrangement the JS part was divided up into three parts named *main*, *search* and *upload*. Each part corresponds to a specific file, where each file affiliates to a certain area of the application.

In the *main part* several constants of the program are defined. Furthermore, the initialization is started which fetches the available reference types and its properties from the servers and stores them into an object. At the same time the various journal titles are requested and stored in an array, which is used at a later point in time for the automatic completion of the user input.

The name of the so-called *search unit* of the JS compilation is a bit misleading, because it conceals functions which are not applied on the search form. Basically the search file features every operation needed by the search form of the UI. This includes transmitting operations from and to the server, where most of the time AJAX is used for transmission. The HTML structure of the result and editor screens is built as well. Furthermore, applied changes by the user in the editor are sent to the server and refreshed on the result screen. Additionally, user input is validated by the use of JQuery and regular expressions. JQuery handles the occurrence of events like clicks on buttons, etc. As mentioned before this part of the code is also responsible for other forms of the application. For example, it signals the main JS file to execute the program initialization and contains functionality of the multiple upload. For the implementation of the multiple upload a tool called *Uploadify* is used.

Finally, export functions of the corresponding export form are completely included in the search unit. Initially, the intension was to separate the export functions in an own JS file, but after some redesign caused by exceptional behavior of the Internet Explorer in comparison to other web browsers, this design was considered and implemented to prevent future problems.

The *upload unit*, as the name implies, offers all functions used uploading new reference documents. It signals the controller via AJAX whether a manual, PubMed or CrossRef upload was selected and transmits the rest of all needed information. Regular expressions and JQuery were used to validate the user input.

4 Discussion

In the scope of this thesis a re-implementation and extension of a Reference DB for scientific literature organization used by the Institute for Genomics and Bioinformatics was realized. Therefore, the migration of the data from the previous version of the Reference DB was accomplished. An analysis of the requirements was performed. The export functionalities were identified as main requirements. ENW, RIS and TEX export formats were implemented to support common RM applications such as Endnote and Reference Manager. This feature makes the Reference DB flexible because a wide range of RM software is covered. Automated metadata retrieval from PubMed and CrossRef was included to enable a quick and comfortable upload of new resources. A manual reference upload, editor functionality and deletion functions for the references were mandatory requirements to improve and extend the management features of the former Reference DB. On the server side a MySQL DB was designed, which is accessed by PHP scripts implemented mainly in an OO style. The OO programming paradigm allows better maintenance and extensibility on condition that it is applied adequately. HTML and JS were used on the client side mainly for presentation and communication with the server. The JS library JQUERY was intensively used for updating the presentation interface, communicating via AJAX to the server and some minor features like auto completion, date picker, etc. JQUERY was used because all the aforementioned features could be implemented in very short time in comparison to pure JS. The scripts on the client side were realized in a procedural way to guarantee a short implementation phase. All the mentioned methods and tools are freely available. Additionally, licensing issues are completely avoided and users may access the application without installing additional software on the client: Applications based on these techniques may be accessed by all kinds of devices and systems providing a web browser.

4.1 Comparison to the previous version of the Reference DB

The previous version supported the upload of documents from PubMed only. Currently the new version allows the upload of six different types, supporting automated metadata retrieval from CrossRef and PubMed. The application was implemented in a structured manner, increasing extensibility and flexibility: Hence, a new reference type can be added without much effort. The migration to the MySQL DB provides much better performance in speed regarding the

results of queries and creation of complete reference lists. The reduction of search time enhances the usability of the Reference DB. The multiple upload issues which caused a malfunction after the release of the latest Java JRE (Java Runtime Environment) were resolved by the use of the tool *Uploadify*. The UI was improved to enable a more efficient LO by introducing three tabs for the main purposes of the application which can be switched during processing. The comprehensive editor capabilities increase the management functionalities of the stored references tremendously. In the previous Reference DB it was necessary to manually edit a text file, in order to apply changes. Deleting documents is executed by two clicks (click on the corresponding icon and confirmation). Previously, several steps had to be made completely manually. The editor and the delete functionalities are only available if the user is an administrator.

4.2 Comparison to bibliographic libraries and RM software

Various DBs and programs which have a similar purpose like the application in this thesis already exist. Some examples would be Mendeley or Zotero. These tools are very useful but do not fully comply with the requirements specified by the Institute for Genomics and Bioinformatics. PubMed for example does not always provide a link where the full-text is provided for free and the topics cover mainly life science and biomedical areas. Mendeley provides a lot of features that the implemented reference DB supports. It allows the user to include the full-text and it is also possible to upload their own documents. Options for sharing are offered and simplify the distribution of the resources among interested users. The great advantage of this version of the Reference DB is that it was implemented in-house. This allows the organization to adapt the application to their own needs and fully adjust it to their requirements. Bug fixes, extensions, updates, etc. may be implemented independently in short time. Another asset is that the stored documents remain within the organization, which ensures full control over all stored resources. The access to the documents from the outside may be restricted. Sharing is possible by granting access to the Reference DB. The resources and its corresponding metadata are stored on the server and may be easily retrieved by users with permission.

In summary, it can be stated that the great benefits compared to Mendeley are the independency, extensibility of the application and the full control over the sources. Other RM applications do not always offer an option for sharing. Additionally, only relevant documents are stored in such

an in-house system. This reference DBs only comprises references used by the staff of the organization.

4.3 Conclusion and Outlook

This thesis pointed out that Reference DBs and RM software are important for scientists regarding the organization of literature. The objective of implementing such a useful tool complying with the requirements, see section 6.2 was achieved. The management possibilities of the documents were extended in comparison to the previous version of the reference DB by

- adding a reference editor,
- enabling deletion of references in the UI,
- supporting six instead of one reference type and
- including CrossRef for automatic metadata retrieval.

Enhancements in speed of processing were achieved. This resulted in better usability which got even more improved by the addition of navigation elements (tabs) and auto completion for journal names. Problems with the existing multiple upload were fixed. Due to this version being an in-house application it offers benefits like the full control over sources, independency and the possibility to implement new required features compared to other DBs which was mentioned in this chapter.

Finally the Reference DB developed in this thesis serves as a tool that provides increased comfort in managing and creating scientific documents.

An interesting point for improvement of the current version of the Reference DB would be the implementation of linking references to the user, who uploaded them. Such a feature would result in giving the user a better overview of its own resources implying an extension and simplification of the management functionalities. Assigning users to groups seems also reasonable. Providing this additional information queries and exports can get more precise by entering the user or group of interest. A quicker access to the data is possible by using this option because the number of references queried would be significantly lowered in comparison to a full DB query.

5 Bibliography

1. **PubMed** [<http://www.ncbi.nlm.nih.gov/pubmed>] 2013
2. Hull D, Pettifer SR, Kell BK: **Defrosting the Digital Library: Bibliographic Tools for the Next Generation Web**. *PLoS Comput Biol* 2008, **4**:e1000204.
3. Wheeler DL, Barrett T, Benson, Dennis A.Bryant SH: **Database resources of the National Center for Biotechnology Information**. *Nucleic Acids Res* 2007, **35**(Database issue):D5–D12.
4. **CrossRef.org** [<http://www.crossref.org/>] 2013
5. **CrossRef: A Collaborative Linking Network** [<http://webdoc.sub.gwdg.de/edoc/aw/ucsb/istl/01-winter/article1.html>] 2013
6. Blake ME, Knudson FL: **Metadata and reference linking**. *Library Collections, Acquisitions, and Technical Services* 2002, **26**:219–230.
7. **EndNote Online User Guide** [<http://endnote.com/if/online-user-manual>] 2013
8. **Mendeley Getting started guide** [<http://blog.mendeley.com/tag/mendeley-getting-started-guide/>] 2013
9. **Reference Manager** [<http://www.refman.com/>] 2013
10. **Introduction to RefWorks** [<http://refworks.libguides.com/home>] 2013
11. **Zotero** [<http://www.zotero.org/>] 2013
12. **Reference Management Software: a Comparative Analysis of Four Products** [<http://www.istl.org/11-summer/refereed2.html>] 2013
13. **EndNote, RefWorks, Zotero, Mendeley Help** [<http://www.lib.berkeley.edu/PUBL/endnote.html#Zotero>] 2013
14. Fenner M: **Reference Management meets Web 2.0**. *Cellular Therapy and Transplantation* 2010, **2**:1–3.
15. **Reference Manager Overview** [<http://blogs.plos.org/mfenner/reference-manager-overview/>] 2013
16. Gourley D, Totty B: *HTTP: The Definitive Guide*. 1st edition. Beijing: O'Reilly; 2002.
17. **The TCP/IP Guide** [http://www.tcpipguide.com/free/t_HTTPOperationalModelandClientServerCommunication.htm] 2013

18. Duckett J: *Beginning Web Programming with HTML, XHTML and CSS*. 1st edition. New Jersey: John Wiley & Sons; 2004.
19. Deloach S: *CSS to the Point*. Atlanta, GA: Clickstart; 2008.
20. **Document Object Model** [<http://www.w3.org/DOM/>] 2013
21. **XML DOM** [http://www.w3schools.com/xml/xml_dom.asp] 2013
22. Flanagan D: *JavaScript: The Definitive Guide*. 6th ed. Beijing: O'Reilly; 2010.
23. **JavaScript Overview** [https://developer.mozilla.org/en-US/docs/JavaScript/Guide/JavaScript_Overview] 2013
24. **DTD Tutorial** [<http://www.w3schools.com/dtd/default.asp>] 2013
25. **Was ist eigentlich der DocType?** [<http://www.professorweb.de/html/was-ist-eigentlich-der-doctype.html>] 2013
26. **DOCTYPE Declaration** [DOCTYPE Declaration] 2013
27. Hunter D: *Beginning XML*. 4th ed. Indianapolis, Ind.: Wiley; 2007.
28. **Introducing JSON** [<http://json.org/>] 2013
29. Holzner S: *Ajax for Dummies*. Hoboken, N.J.: Wiley Pub.; 2006.
30. Davis ME, Phillips JA: *Learning PHP and MySQL*. 2nd ed. Farnham: O'Reilly; 2007.
31. DuBois P: *MySQL*. Indianapolis, IN: New Riders Publishing; 1999.
32. Biswas BG: *MySQL Is the Open Source Data Management System: Study of Its Feature and Functions*. :1–21.
33. **Bringing MySQL to the web** [http://www.phpmyadmin.net/home_page/] 2013
34. **phpDocumentor Quickstart** [http://manual.phpdoc.org/HTMLSmartyConverter/HandS/phpDocumentor/tutorial_phpDocumentor.quickstart.pkg.html] 2013
35. **Apache HTTP Server Project** [<http://httpd.apache.org/>] 2013
36. **NetBeans IDE** [<http://netbeans.org/>] 2013
37. **Mockingbird** [<https://gomockingbird.com/>] 2013
38. **The DOI System** [http://www.doi.org/doi_handbook/1_Introduction.html] 2013
39. **Entrez Help** [<http://www.ncbi.nlm.nih.gov/books/NBK3837/>] 2013

40. **A General Introduction to the E-utilities**
[<http://www.ncbi.nlm.nih.gov/books/NBK25497/>] 2013
41. Lindley C: *jQuery Cookbook*. 1st ed. Beijing: O'Reilly Media; 2010.
42. **jQuery** [<http://jquery.com/>] 2013
43. **jQuery user interface** [<http://jqueryui.com/>] 2013
44. **About Uploadify** [<http://www.uploadify.com/about/>] 2013
45. **SimpleXML** [<http://www.php.net/manual/en/book.simplexml.php>] 2013
46. **SimpleXml Introduction** [<http://www.php.net/manual/en/intro.simplexml.php>] 2013
47. **MySQL Improved Extension** [<http://www.php.net/manual/en/book.mysql.php>] 2013

6 Appendix

6.1 Functional scope of the previous Reference DB

The functionalities are divided up into upload, query, results and export:

6.1.1 Upload:

- Upload of a new reference is only possible with an existing PMID and the full-text document in PDF format.
- The PMID has to be entered into a text field.
- The metadata of a document is fetched from the external DB called PubMed via its corresponding ID.
- A Java applet allows the upload of multiple documents. All documents have to be named with its respective PMID. The applet does not work with Java version 7 or later.
- A click on “*Datei auswählen*” opens the explorer, which allows selecting the document one wishes to upload.
- By clicking on “*Upload*” the selected file will be uploaded. In course of the upload the metadata for the document is acquired from PubMed with the provided ID.
- Every member of the mailing list [papers] gets informed about the new document in the DB after the successful upload.

6.1.2 Query:

- It is possible to search the DB with one or more keywords in a text field.
- Keywords can be linked via logical connectives. These are OR, AND and NOT. One logical connective per query may be used.
- The options “*perfect match*”, “*case sensitive*” and “*link to terms in PDF*” (note: keywords are highlighted in the PDF) can be used on the query and these options can also be combined among each other.
- The keywords may be searched in the categories *Author*, *Journal*, *Volume*, *First page*, *Last page*, *Year*, *PMID*, *comment* or in all of them. The specific fields can be selected via a drop down box.
- The query is started by a click on the search button.

- A click on the clear button deletes the entered keywords and resets the search form to its default.

6.1.3 Results of the query:

- The result of the query shows every source in the DB below the search area of the interface.
- A full text export of every document in the format of PDF is possible.
- The metadata of every document may be downloaded for BibLatex, Endnote and Reference Manager.
- Comments made by other users are view- and editable for every document.
- The available metadata for every source is shown.
- A click on an author of a document starts a new query with the authors name as a new keyword and a logical AND connection. The query searches in the category “author” and the options “case sensitive” and “perfect match” are used.
- The PMID of the results is a link, which refers to the site of the National Center for Biotechnology Information, where further information about the document can be viewed.
- The DOI of a document is also a link, where the user gets all information of the source.
- A small summary of the references can be found at the end of the list: The number of references found, the sum of the IFs and the mean IF are displayed.

6.1.4 Export:

- A reference list of the whole DB may be exported into the formats of BibLatex Endnote and Reference Manager.
- It is possible to decide in which way this reference list is sorted. A sorting according to *Author, PMID, First page, Last page, Year, Volume, Journal or Title* is selectable.

6.2 User requirements document (URD)

The assignment is to create a scientific literature organization application. This application provides the possibility to manage different types of scientific references, which are saved in a DB. This means, that it is possible to upload, search and download scientific literature of different types. Additionally, all information of the scientific documents e.g. author and publisher may be edited. All in all CRUD (Create, Update, Read and Delete) functionality has to be enabled for all entries of the management system. Furthermore, the design and creation of the DB are also part of the task.

6.2.1 Guidelines

- The application has to run in the currently most popular web browsers to ensure more flexibility as well as platform independence. These browsers are the Internet Explorer, Chrome, Firefox and Safari. If the application runs perfectly in these web browsers, there should be no problem in any other browser.
- The programming languages used for the main part are HTML and PHP. Use of different script languages like JS and AJAX may also be necessary.
- The DB is required to be programmed in MySQL.
- Apart from already supported features from the existing reference DB described in section 6.1 the application will be upgraded by new features.
- All the information saved in the existing reference DB is required to be migrated to the new one.

6.2.2 Upload:

- The upload of different types of literature sources has to be possible with or without PMID or DOI.
- If the scientific reference has no PMID or DOI, the user is obliged to add the necessary metadata, of the literature manually via the online form (manual upload).
- If a PMID or DOI exists, the number has to be entered by the user. The metadata will be fetched from the PubMed or CrossRef DB and stored in the Reference DB.
- The upload is executed by a click on the upload button.
- The application displays a message about the success of the upload operation.

- Every member of the mailing list [papers] gets informed about the new document in the DB after the successful upload.

6.2.3 Manual upload:

- The type of document has to be selected by the user, if the PMID or DOI is missing. The existing document types in the DB are selectable via a drop-down-menu.
- If the type of the document does not exist in the DB, it may be added by the user and the new type will be available in the drop-down-menu next time.
- Depending on the document type the online form will display the corresponding mandatory and optional fields, where the available metadata has to be entered.
- The entered data gets checked on correctness and completeness. The latter means that the mandatory metadata has to be entered, otherwise the upload will be declined. An example for correctness would be that it is not possible to enter a letter into the date field.
- Multiple upload has to be available for documents of PubMed. These documents need a PMID for fetching the metadata. Without this identifier the upload has to be done one by one in the manual upload form.

6.2.4 Query:

- Possibility to query the DB with one or more keywords.
- The options “*perfect match*”, “*case sensitive*” and “*link to terms in PDF*” (note: keywords are highlighted in the PDF) can be used on the query and these options can also be combined among each other.
- The keywords may be searched in the different available categories like *Author*, *Publisher* and so on. It will be possible to combine the search in two or more categories to refine the search.

6.2.5 Results of the query:

- The result of the query will be displayed below the main screen, where a new query can be executed to be able to search the DB as fast as possible.
- The full text may be viewed of every document in the format of PDF. The icon will be displayed next to the document.

- The export of a particular reference to the format of BibLatex, Endnote or Reference Manager can be achieved by a click on the appropriate icon next to the document. Necessary metadata is then converted into the respective reference style.
- Comments may be added for each document.
- Available comments are viewed- by resting with the mouse cursor on the corresponding icon.
- The available metadata in the DB for every source is shown.
- By clicking on an author of a document, a new query will start and the result will show every document, which was written by the selected author.
- The PMID of the results will be a link, which refers to the site of the PubMed, where further information about the document can be viewed.
- Every DOI of a document listed in result will also be a link to the CrossRef website, where information about the literature is stored and may be viewed.
- There can be found a small summary of the references at the end of the list: The number of references found, the sum of the impact factors and the mean impact factor are displayed.
- Every document may be deleted by a click on a specific icon next to it.
- The edit-mode can be opened by a click on a document.

6.2.6 Editor:

- The editor is displayed below the selected document in the results screen.
- All the metadata of a document may be changed. The new information has to be entered in the corresponding text field and can be saved by a click on the save icon, located at the bottom of the editor.

6.2.7 Other features:

- The export option allows the user to export the whole reference DB into the three available reference formats of BibLatex, Endnote or Reference Manager.
- Everywhere on screen will be a quick help available. If the mouse cursor rests on a button or text field the user will get informed about how the option is used in a proper way.

6.3 Reference types

The various reference types are described by different properties. These properties may be split up into mandatory and optional properties. The following overview illustrates the properties. Attributes in bold are mandatory, the rest is optional.

Whole book: **title, authors, publication date, link to the full text, edition, publisher, city**, ISBN, DOI, keywords, comments, subtitle, abstract

Book chapter: **title (chapter), authors, publication date, link to the full text, title (book), publisher, city, editors, start page, end page**, ISBN, DOI, keywords, comments, subtitle, abstract

Scientific journal article: **title, authors, publication date, link to the full text, journal title, volume, issue, start page, end page, PMID**, keywords, comments, abstract

Online source: **title, link to the full text, access date**, authors, keywords, comments

Conference proceedings: **title, authors, publication date, link to the full text, proceedings title, publisher, city, date of the conference, start page, end page, editors**, DOI, keywords, comments, abstract

Thesis/Dissertation: **title, authors, approval date, link to the full text, publication place (city), institution, abstract**, DOI, keywords, comments