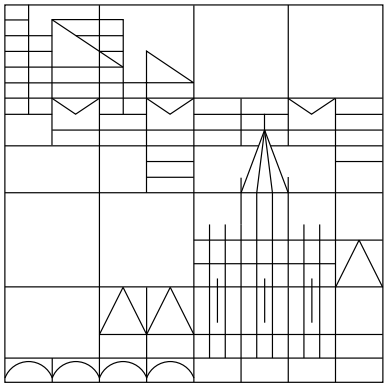
Universität Konstanz



Fachbereich für Informatik und Informationswissenschaft
Lehrstuhl Datenbanken und Informationssysteme

Wissenschaftliche Arbeit
zur Erlangung des Grades eines Bachelor Information Engineering
im Fachbereich Informatik & Informationswissenschaft der Universität Konstanz

Entwicklung eines webbasierten Publikationsverwaltungssystems

Verfasser:

Siniša Avramović

26. September 2006

Gutachter:

Prof. Dr. Marc Scholl

Prof. Dr. Rainer Kuhlen

Universität Konstanz
Fachbereich für Informatik und Informationswissenschaft
D–78457 Konstanz
Germany

Siniša Avramović
Gottlieberstrasse 1
78462 Konstanz
Matr-Nr.:01/495707
E-mail: avramovi@inf.uni-konstanz.de
Entwicklung eines webbasierten Publikations-verwaltungssystems
Bachelorarbeit, Universität Konstanz, 2006.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einleitung

1.1 Aufbau der Arbeit

Die vorliegende Bachelor-Arbeit beschreibt die Konzeption und Implementierung einer Anwendung zur Verwaltung wissenschaftlicher Publikationen. Nach einer einführenden Diskussion der Motivation dieser Arbeit und kurzer Abgrenzung zu bestehenden Implementierungen im ersten Kapitel, werden im zweiten Kapitel Definitionen und Grundlagen der für diese Arbeit relevanter Fachbegriffe aus den Bereichen der Datenbanktechnologie und dem des Information Retrieval dargeboten.
Das dritte Kapitel liefert die Beschreibung der Konzeption der vorliegenden Applikation. Beginnend mit der Anforderungsanalyse und der daraus hervorgehenden Kernpunkte, werden im Anschluss die Überlegungen einer geeigneten Applikationsarchitektur erläutert. Den Erläuterungen des Datenbankentwurfs sowie der Schichtenarchitektur ist aufgrund der Bedeutung für das gesamte Projekt jeweils ein gesonderter Abschnitt gewidmet.
Kapitel vier dieser Arbeit erörtert die Aspekte der konkreten Implementierung der Anwendung. Nach einer einführenden Erläuterung der zugrundeliegenden Technologien zur Datenbankanbindung und der Implementierung der Suchfunktionalität, werden in den darauf folgenden Abschnitten die Anwendungsfunktionen mit den jeweiligen Problemstellung und Lösungsansätzen erläutert. Das abschliessende fünfte Kapitel bietet eine Zusammenfassung der Ergebnisse dieser Arbeit kombiniert mit den Überlegungen zum Ausblick.

1.2 Motivation

Untersuchungen der Wirtschaftsuniversität Wien haben gezeigt, dass sich die Zahl der Wissenschaftler in den letzten 150 Jahren alle fünfzig Jahre in etwa verzehnfacht hat. Die Zahl der wissenschaftlichen Publikationen ist proportional dazu stark gestiegen. Schätzungen zufolge liegt allein die Zahl der neu erschienenen Bücher in deutscher, englischer und französicher Sprache bei jährlich 230.000, die Zahl wissenschaftlicher Artikel

wird weltweit auf ein Volumen von rund drei Millonen Veröffentlichungen geschätzt. Scientometrische Evaluationen der CEST [ces] ergaben für die Universität Konstanz in dem Zeitraum von 1998 bis 2002 eine Zahl von 2510 wissenschaftlichen Publikationen. Als Hilfsmittel zur Bewältigung des enormen Aufwandes der Erfassung, Katalogisierung und Archivierung wissenschaftlicher Publikationen haben sich Informationssysteme etabliert und in hohem Maße bewährt. Dabei reicht die Auswahl von Volltextdatenbanken, welche die Quellen als Textsammlungen lediglich bereithalten, bis zu den Methoden intelligenter Informationserschließung, den modernen Retrieval Systemen. Solche Systeme liefern anhand von Deskriptoren inhaltsbeschreibende und klassifizierende Schlagwörter als Hilfe für die Selektion relevanter Dokumente zu einer bestimmten Anfrage.
Der durch eine vorbereitende Internetrecherche gewonnene Einblick in bestehende Applikationen macht deutlich, dass eine Reihe von Implementierungen die Problematik der Erfassung, Archivierung und Katalogisierung wissenschaftlicher Publikationen bereits abhandelt. Dennoch lässt die in Abschnitt 1.3 beschriebene nähere Betrachtung dieser den Schluss zu, dass keine der betrachteten Anwendungen den Anforderungen der Publikationsverwaltung innerhalb von Lehrstühlen als Organisationseinheiten vollständig gerecht wird.
Aus einem solchen Sachverhalt ergab sich die Motivation, auf Konzeption und Implementierung einer Anwendung einzugehen, welche einen etwas anderen Ansatz im Umgang mit wissenschaftlichen Publikationen verfolgt. Im Gegensatz zu den bestehenden Applikationen, welche meist ein breites Auditorium als Zielgruppe haben, soll die Fokussierung auf die Bedürfnisse eines engeren Benutzerkreises, nämlich der wissenschaftlichen Mitarbeiter und Lehrenden innerhalb von Lehrstühlen als Organisationseinheiten, in erster Linie in der Konzeption hervorgehoben werden. Bisher vermisste Funktionen zur Beschleunigung des Veröffentlichungsprozesses wie Verarbeitung einzelner oder in Listenform vorhandener BibTex Einträge, eine Möglichkeit, die hierarchische Struktur zwischen den einzelnen Mitarbeitern und dem Lehrstuhl als Organisationseinheit abzubilden, komfortable Generierung und Export von Publikationslisten zur Einbindung in eigene Webpräsenzen sind dabei als hauptsächliche Differenzierungsmerkmale zu bestehenden Applikationen zu nennen.

1.3 Abgrenzung

Die bereits erwähnte vorbereitende Internetrecherche lieferte einen Überblick über bestehende Anwendungen zur Publikationsverwaltung. Diese sollen hier in einer kurzen Zusammenfassung beschrieben und auf die in der Motivation aufgeführten Kernanforderungen untersucht werden. In diese Untersuchung wurden ferner Kriterien, welche im Abschnitt 3.1 der Anforderungsanalyse näher beschrieben sind, einbezogen. Jedoch besteht nicht die Absicht eine detaillierte Funktionsübersicht aller erwähnten Anwendungen darzulegen.
In der zur Zeit von der Bibliothek der Universität Konstanz eingesetzten Anwendung zur Verwaltung wissenschftlicher Publikationen *KOPS* [Kop], kommt aufgrund einer

stark von bibliothekarischen Aspekten motivierten Konzeption den Bedürfnissen der Publizierenden als Benutzergruppe eine sekundäre Rolle zu. Gehören hier die Auswahl aus diversen Fachklassifikationen zur Sacherschließung oder detaillierte Suchfunktionen einschließlich der Volltext- und Phrasensuche zu den Leistungsfunktionen, stehen den Publizierenden für die Veröffentlichung der digitalisierten Werke ausschließlich umfangreiche und mehrseitige Onlineformulare zur Verfügung. Möglichkeiten zur Beschleunigung der Eingabeprozesse wie die automatische Übernahme von Attributen aus vorgefertigten Einträgen, wie es Bibtex ist, oder der Verarbeitung einer ganzen Reihe von Publikationen in Form von Listenimporten fehlen gänzlich, ebenso wie die Referenzierung von Dokumenten, welche schon in digitalisierter Form auf externen Servern bereitgestellt sind. Eine Editierung digital publizierter Werke im Sinne einer Korrektur von Fehleingaben oder nachträglichen Hinzufügen von Deskriptoren bleiben den Benutzern verwehrt. Eine Möglichkeit zur Abbildung bestehender hierarchischer Organisation der Lehrstühle besteht ferner nicht. Lediglich die Suche im Publikationsbestand nach Fachbereichen ist hier möglich. Generierung und Export benutzerbezogener Publikationslisten zur weiteren Verwendung oder Einbindung in eigene Webpräsenzen ist nicht gegeben.
Die in [FH03] beschriebene Anwendung ist speziell auf die Publikationsverwaltung wissenschaftlicher Mitarbeiter einer Hochschule ausgerichtet. Hier ist ein automatisierter Import von Publikationslisten, welche in Form von BibTex Einträgen vorliegen, gegeben. Eine Veröffentlichung von Publikationen ausserhalb der Listenimporte ist mangels einer Formularfunktion nicht möglich. Der Export von Publikationslisten ist möglich, jedoch ist diese Option nur für manuelle Exporte gegeben, eine dynamische Listeneinbindung in eigene Webpräsenzen ist nicht vorhanden. Die Formatierung der exportierten Listen ist von den Benutzern mittels eigener `css` Dateien beliebig erweiterbar. Die Abbildung der Lehrstuhlorganisation ist auch hier nicht direkt modelliert. Zwar ist eine Organisation von Benutzergruppen möglich, jedoch ist diese auf eine ständige Pflege durch einen Administrator angewiesen. Die Benutzerverwaltung ist auf die Vergabe von Zugriffsrechten auf Publikationen anderer Benutzer beschränkt, ohne dabei die Abbildung einer Lehrstuhlstruktur zu ermöglichen. Eine Einbindung von Schlagwörtern eines Fachvokabulars zur Deskribierung ist nicht verfügbar.
Eine weitere frei erhältliche Anwendung zur Publikationsverwaltung ist *Aigaion* [Aig]. Auch hier sind zur Beschleunigung von Eingabeprozessen umfangreiche Importmöglichkeiten gegeben. Dabei können Publikationslisten in den Formaten BibTex und RIS automatisiert in den Publikationsbestand eingefügt werden. Die BibTex-Importfunktion ist auf fehlerfreie Publikationslisten Angewiesen. Syntaxfehler innerhalb der Listen erzwingen hier einen Importabbruch an der Fehlerstelle. Möglicherweise korrekte Einträge unterhalb der Fehlerstelle werden dabei ignoriert. Generierung und Export benutzerbezogener Publikationslisten ist nur manuell verfügbar. Der Publikationsbestand kann anhand der Themengebiete in Gruppen organisiert werden. Allerdings eignet sich auch diese Organisationsform nicht zur Abbildung von Lehrstuhlstrukturen. Die Stichwortsuche der Suchfunktion reduziert sich auf Übereinstimmungen im Titelfeld, eine Indexierung der relevanten Felder findet nicht statt. Alternativ ist noch die Option

der Autorensuche in Kombination mit Booleschen Operatoren über den einzelnen Themenfeldern gegeben.
Die unter [Php] ebenso frei verfügbare Anwendung zur Publikationsverwaltung *phpbiblio* bietet einen etwas kleineren Funktionsumfang als die bisher beschriebenen Implementierungen. Auch hier ist eine Abbildung von Lehrstuhlstrukturen nicht möglich, lediglich die Organisation von Veröffentlichungen nach Themengebieten ist gegeben. Das Einbringen von Publikationen ist ausschließlich über die Formularfunktion möglich. Generierung von Publikationslisten und ihr Export in das BibTex Format sind verfügbar, jedoch nur für ein gesamtes Themengebiet. Eine separate Zuordnung zu den Einzelnen Autoren ist nicht möglich. Die Suchfunktion letzlich ist auf die Phrasensuche im Titelfeld reduziert. Eine Indexierung relevanter Publikationsfelder findet nicht statt.

	KOPS	XML-BMS	Aigaion	phpbiblio	Procite	Endnote	Biblioscape	WPVS
Plattformunabhängigkeit	√	√	√	√	-	-	-	√
Lehrstuhlbasierte Benutzer- / Publikations-verwaltung	-	-	-	-	-	-	-	√
Indexieren	√	-	-	-	√	√	√	√
Import v. Publikationslisten	-	√	√	-	√	√	√	√
Export v. Publikationslisten	-	√	√	√	√	√	√	√
Dynamischer Listenexport	-	-	-	-	-	-	-	√

Tabelle 1.1: Vergleich bestehender Anwendungen zur Publikationsverwaltung

Neben den bisher beschriebenen kostenfreien[1] ist eine Reihe kostenpflichtiger Anwendungen zur Publikationsverwaltung verfügbar. Dabei sind in erster Reihe Procite [Pro], Endnote [End] und Biblioscape [Biba] zu erwähnen. Diese Anwendungen bieten umfangreiche Import und Exportfunktionen für Publikationslisten, eine Indexierung relevanter Publikationsfelder und die indexgestützte Suche sowie Einbindung multipler Publikationsdatenbanken. Referenzierung extern verfügbarer Publikationen in digitaler Form gehört ebenso zum Funktionsumfang. Allerdings eignen sich auch diese Anwendungen aufgrund fehlender Möglichkeit lehrstuhlbasierter Benutzer- und Publikationsverwal-

[1] Das auf dem Online Publikationssystem der Universität Stuttgart (OPUS) [Opu] basierende KOPS ist für den Einsatz an Hochschulen kostenfrei.

tung, ihrer Platformabhängigkeit sowie lizenzbedingter Beschränkungen in der Anzahl der Benutzer oder konkurierender Zugriffe nicht für eine Publikationsverwaltung von Lehrstühlen als Organisationseinheiten.

Eine tabellarische Gegenüberstellung der beschriebenen Anwendungen ist in Tabelle 1.1 gegeben. Darin ist die in [FH03] vorgestellte Anwendung als XML-BMS bezeichnet. Die Attribute der Applikation zur Publikationsverwaltung, welche Gegenstand dieser Arbeit ist, sind in der Spalte WPVS (webbasiertes Publikationsverwaltungssystem) ausgewiesen.

Kapitel 2

Grundlagen

Dieses Kapitel soll die Definitionen und Grundlagen einiger Fachbegriffe aus den Gebieten der Datenbanksysteme und dem des Information Retrieval liefern, die in der Konzeption und Implementierung enthalten sind und in der weiteren Ausarbeitung referenziert werden. Es ist nicht das Ziel, detaillierte Elaborationen zu den einzelnen Teilbereichen zu präsentieren, da diese schon in einer Vielzahl von Arbeiten beschrieben wurden. Vielmehr ist es die Absicht, grundlegende Sachverhalte so darzustellen, dass wiederkehrende Nebenerläuterungen in den Kapiteln Applikationsentwurf und Implementierung vermieden und die Begründungen der Entscheidungen für bestimmte Realisierungsansätze transparenter werden.

2.1 Datenbanksysteme

Die wesentlichen Punkte, die im Zusammenhang mit dem Begriff Datenbanken genannt werden müssen, sind in ihrer Definition, welche in [Sch04] anzutreffen ist, genannt:

> *„Eine Datenbank (database), kurz DB, ist eine integrierte und struktuierte Sammlung persistenter Daten, die allen Benutzern eines Anwendungsbereichs als gemeinsame und verlässliche Basis aktueller Information dient. “*

Der Begriff Datenbank hat sich in diesem Kontext im alltäglichen Sprachgebrauch der technischen Welt eingebürgert. In einschlägigen Fachkreisen jedoch kommt eine differenziertere Terminologie zum Einsatz. Hierbei wird zwischen einem Datenbanksystem **DBS**, einem Datenbank−Management−System **DBMS** und einer Datenbank **DB** unterschieden. Unter einem Datenbank-Management-System wird die *Gesamtheit aller Software-Module, die die Verwaltung einer Datenbank übernehmen* [AK04] , verstanden. Der Begriff Datenbank bezeichnet in diesem Zusammenhang lediglich einen strukturierten, vom DBMS verwalteten Datenbestand. Die Kombination aus einer Datenbank und einem DBMS wird als Datenbanksystem, kurz DBS bezeichnet. Für eine Anwendung zur Verwaltung von Publikationen mehrer Benutzer ist eine zentrale Datenhaltung, welche die Datenintegrität, Datenkonsistenz sowie parallelen und effizienten Datenzugriff ermöglichen von zentraler Bedeutung. Genau diese Vorteile bietet der Einsatz eines DBMS.

2.1.1 Relationale Datenbanksysteme

In relationalen Datenbanksystemen werden die Objekttypen der zu modellierenden Anwendungswelt durch Relationenschemata beschrieben. Relationsschemata bestehen aus einer Menge von Attributen, welche die Eigenschaften der zu modellierenden Objekte repräsentieren. Jedem dieser Attribute wird ein Wertebereich, auch Domäne genannt, zugeordnet. Die zu einem Relationenschema aktuell vorhandenen Daten werden als Relation bezeichnet. Einzelne Elemente einer Relation heißen Tupel. Eine Relation kann als eine Tabelle verstanden werden, in der die Attribute des Relationenschemas die Kopfzeile mit den Spaltenüberschriften bilden. Die darauf folgenden Zeilen, die Tupel, bilden die Relation. So organisierte Tabellen können nur atomare Werte enthalten, nicht wiederum Tabellen oder sonstige struktuierte Werte, daher auch die Bezeichnung *flache Relationen*. Eine Menge von Relationenschemata bilden das Datenbankschema. Abbildung 2.1 veranschaulicht die Beziehung der hier erläuterten Begriffe.

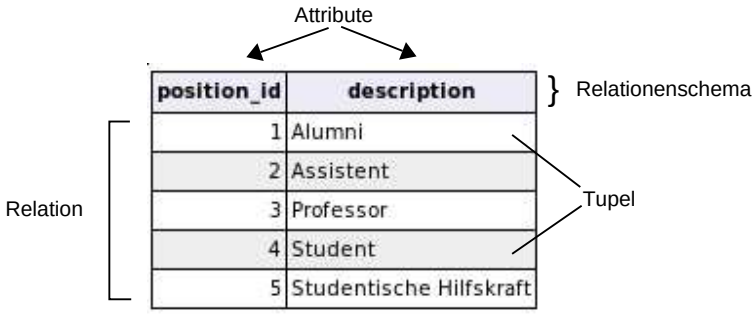


Abbildung 2.1: Relationenschema und Relation

Zur eindeutigen Identifizierung einzelner Einträge (*Tupeln*) einer Tabelle werden diese mit eindeutigen Schlüsselwerten gekennzeichnet, welche auch als Primärschlüssel bezeichnet werden. Diese können sowohl Attribute der realen Welt, künstlicher Natur wie etwa fortlaufende Zahlen oder gar eine Kombination von mehreren Attributen sein. Lediglich die Eindeutigkeit ist zwingend vorgeschrieben. Die Relationen einer Datenbank können wie die Objekte der Anwendungswelt, welche sie repräsentieren in Beziehung zueinander stehen. Die Erhaltung der Datenintegrität auf der Ebene der Beziehungen zwischen Relationen wird dabei als referentielle Integrität bezeichnet, welche von den DBMS mit dem Einsatz von Fremdschlüsseln gewährleistet werden. Als Fremdschlüssel werden dabei Attribute einer Relation bezeichnet, welche in einer anderen, in Beziehung stehenden Relation als Primärschlüssel existieren.
Der Zugriff auf die Daten einer Datenbank zur Datendefinition und Datenmanipulation wie *Einfügen*, *Aktualisiern* und *Löschen*, ferner die Auswahl von Antwortmengen zu bestimmten Anfragen nach bestimmten Selektionsbedingungen werden über eine deklarative Anfragesprache gewährleistet. Deklarativ in dem Sinne, dass die Benutzer definieren, welche Daten gewünscht sind, und nicht, wie die Datenauswertung algorithmisch ablau-

fen soll. Die wohl am meisten verbreitete Anfragesprache derzeit ist SQL[1]. Zu Zwecken der genannten Datenoperationen stellt SQL eine Fülle von Operatoren zur Verfügung welche in einer Vielzahl von Veröffentlichungen ausführlich abgehandelt wurden, wie zum Beispiel in [AH00], daher an dieser Stelle nicht weiter ausgeführt werden.

2.1.2 Transaktionen

Datenbanksysteme stehen im Allgemeinen selten nur einem Benutzer exklusiv zur Verfügung. In einem Mehrbenutzerszenario muss ein gleichzeitiger Zugriff gewährleistet werden, so dass die Benutzer sich idealer Weise nicht bemerken oder gar gegenseitig stören. Dabei soll die Datenkonsistenz gewahrt bleiben. Um diese Forderungen erfüllen zu können, greifen mehrbenutzerfähige DBMS auf das Konzept der Transaktionen zurück. Der Grundgedanke von Transaktionen besteht darin, eine Folge von Operationen systemintern so zu verarbeiten, als würde die Datenbank einem Benutzer exklusiv zur Verfügung stehen. Die zeitgleichen Aufkommen solcher Operationsfolgen müssen vom DBMS in der Ausführung zeitlich verzahnt so synchronisiert werden, dass jede dieser Folgen aus der Sicht der zugreifenden Anwendung, welche diese Operationen auslöst, unzertrennlich und isoliert von den restlichen Abläufen erscheint. Im konkreten Fall einer Anwendung zur Publikationsverwaltung ist ein Szenario denkbar, in welchem mehrere Benutzer gleichzeitig Einfüge− oder Änderungsoperationen an identischen Datensätzen durchführen. Beispeilsweise kann ein solcher Fall während der gleichzeitigen Editierung von Publikationsattributen einer gemeinsam veröffentlichten Publikation durch mehrere Benutzer auftreten. Das Konzept der Tansaktionsverwaltung verhindert in solchen Fällen beispielsweise das gegenseitige „Überschreiben“durchgeführter Änderungen. Einige Funktionen dieser Anwendung, welche im Kapitel 4 näher beschrieben werden, machen Gebrauch von der Transaktionsfunktionalität des DBMS.

2.1.3 Indexstrukturen

Eine der wesentlichen Anforderungen an ein DBMS ist die effiziente Gestaltung von Datenzugriffen. Heutige Datenbanksysteme weisen verschiedene Strategien auf, um den Datenzugriff weitreichend zu optimieren. Ein nativer Vorgang, um die in Tabellen abgespeicherten Daten nach bestimmten Selektionskriterien zu filtern, ist der sequentielle Durchlauf aller darin vorkommenden Daten. Bei großen Datenmengen ist ein solcher Ansatz alleine aus der Tatsache heraus nicht performant, dass in den meisten Fällen eine geringe Teilmenge der Daten einer Relation zur Antwortmenge gehören. Zur Effizienzsteigerung bei Datenzugriffen bieten die meisten heutigen DBMS Indexe an. Den Stichwortverzeichnisen von Büchern ähnlich, bestehen solche Indexe aus Kombinationen von Schlüsselwerten und der physikalischen Adresse an der diese Einträge abgespeichert sind. Dabei ist die Auswahl an Datenstrukturen für die Realisierung solcher Indexe umfangreich.

[1]Das Akronym **SQL** steht für Strucured Query Language

Auch die hier vorliegende Anwendung kann von solchen Hilfsmitteln enorm profitieren. Eine wesentliche Aufgabe der Publikationsverwaltung besteht in der effizienten Auswahl zu einer bestimmten Anfrage relevanter Dokumente. Die zu diesem Zweck implementierte stichwortbasierte Suchfunktion stützt sich auf eine Weiterentwicklung bestehnder Index−Datenstrukturen, dem GIST−Baum[2]. Diese Datenstruktur vereinigt die Vorteile einiger sehr populärer Baumstrukturen wie den B+ Baum in sich. Die Besonderheit der GIST-Datenstruktur liegt jedoch in der Erweiterbarkeit der anwendbaren Operatoren und den unterstützten Datentypen. Diese Option der benutzerdefinierten Operatorenerweiterung macht diese Indexstruktur besonders für den Umgang mit unstruktuierten Daten, wie es auch Texte sind, besonders geeignet. Ein Grundeinstieg in die Thematik der GIST Datenstruktur zusammen mit den Ansätzen der Operatorenerweiterung ist in [JMH95] gegeben.

2.2 Information Retrieval (IR)

Zu den Grundfunktionen einer Anwendung zur Verwaltung von Publikationen gehört zweifelsohne die Möglichkeit, aus dem verwalteten Publikationsbestand diejenigen Dokumente zu extrahieren, welche für eine bestimmte Anfrage von Bedeutung sind. In der Realisierung dieser Funktionlität wird innerhalb der vorliegenden Anwendung auf einige Konzepte aus dem Fachgebiet des Information Retrieval zurückgegriffen. In den nachfolgenden Abschnitten soll daher nach einer Definition auf die für Konzeption und Implementierung der Anwendung relevanten Aspekte des Information Retrieval eingegangen werden.

2.2.1 Einführung und Definition

Die computergestützte Extraktion der für das Informationsbedürfnis von Anwendern zu einer bestimmen Anfrage oder Problemlösung relevanter Information aus großen Mengen unstruktuierter Daten kann als Forschungsgebiet des *Information Retrieval*, zu deutsch *Informations* (*−wieder*) *gewinnung* und der daraus entstandenen Information− Retrieval− Systeme erfasst werden. Gerard Salton bietet in [Sal68] eine folgende Definition des *Information Retrieval* :

> „ *Information retrieval is a field concerned with the structure, analysis, organization, storage, searching, and retrieval of information.*“

Eine weitere Beschreibung bietet die Fachgruppe Information Retrieval der Gesellschaft für Informatik, welche hier im Auszug zitiert wird [Fer03]:

> „*Im Information Retrieval (IR) werden Informationssysteme in Bezug auf ihre Rolle im Prozess des Wissenstransfers vom menschlichen Wissensproduzenten zum Informations- Nachfragenden betrachtet.* “

[2]Das Akronym GIST steht für *generall implemented search tree*

Die Grundcharachteristik und zugleich das Hauptunterscheidungsmerkmal der Anfragen des IR zum Daten Retrieval herkömmlicher Informationssysteme sind ihre Vagheit, Unschärfe und Unsicherheit. Vagheit, da der Benutzer sein Informationsbedürfnis nicht formal präzise definieren kann, wie es zum Beispiel in relationalen Datenbanksystemen und der dazugehörigen Anfragesprache SQL der Fall ist. Die Unsicherheit bezieht sich auf die fehlenden Kenntnisse des Systems über den Inhalt der Dokumente, was zu unscharfen oder zum Teil unvollständigen Antwortmengen führen kann. Im Gegensatz zu RDBMS, welche auf einer syntaktischen Datenverarbeitung basieren (Faktenretrieval), ist diese in den Information−Retrieval−Systemen auf die semantische Analyse fokussiert. Hierzu müssen die Dokumente vorbereitend interpretiert werden. Eine in der Literautur häufig anzutreffende Differenzierung zwischen Information Retrieval und Fakten Retrieval, die Differenzierung nach Van Rijsbergen [Rij79], ist in Tabelle 2.1 gegeben.

	Data Retrieval	**Information Retrieval**
Matching	Exact match	Partial match, best match
Inference	Deduction	Induction
Model	Deterministic	Probabilistic
Classification	Monothetic	Polythetic
Query language	Artificial	Natural
Query specification	Complete	Incomplete
Items wanted	Matching	Relevant
Error response	Sensitive	Insensitive

Tabelle 2.1: Differenzierung von Information Retrieval und Faktenretrieval nach Van Rijsbergen

2.2.2 Information Retrieval Systeme

Ein abstraktes Modell eines Information−Retrieval−Systems, im Folgendem als IRS abgekürzt, kann, wie in Abbildung 2.2 [Bek], gezeigt schematisch dargestellt werden. Daraus werden die grundlegenden Module und Funktionsweisen solcher Systeme deutlich. Wissensproduzenten, also Autoren, stellen den IR−Systemen Wissen in Form von Objekten zur Verfügung. Da in vielen Informationssystemen Wissen über Objekte und nicht die Objekte selbst verwaltet werden, wandeln IR−Systeme in der Inhaltserschließung die eingebrachten Objekte in eine durch das syteminterne Repräsentationsmodell bedingte Form um. Resultat dieser Transformationen ist eine in der Regel umfangreiche Wissensbasis als Objektrepräsentation. Benutzer, welche ihren Informationsbedarf mit dem IR−System zu decken versuchen, formulieren ihre Informationsbedürnisse gegenüber dem IRS, wobei die Art, auf welche diese Bedürfnisse formuliert werden, ebenfalls vom Repräsentationsmodell des IR−Systems abhängig ist. Es ist wichtig anzumerken, dass

die Suchanfragen ebenso wie die Objekte zunächst nach dem gleichen Prinzip verarbeitet und in die „Sprache“ des Systems überführt werden, bevor ein Abgleich mit der Wissensbasis stattfindet. Dieser Ansatz liegt auch in der Suchfunktionalität des in dieser Anwendung verwendeten `Tsearch2` Moduls, welches im Abschnitt 4.3 näher erläutert wird, vor. Die Anfrageformulierung, wie der Abbildung 2.2 zu entnehmen ist geschieht im Modul *Recherche*. Die im Recherchemodul formulierte Anfrage– oder besser ihre Repräsentation– wird durch die Retrievalfunktion mit der Wissensbasis abgeglichen. Das Ergebnis ist eine Menge von Dokumenten, welche vom IRS zur Anfrage als relevant befunden wurde.

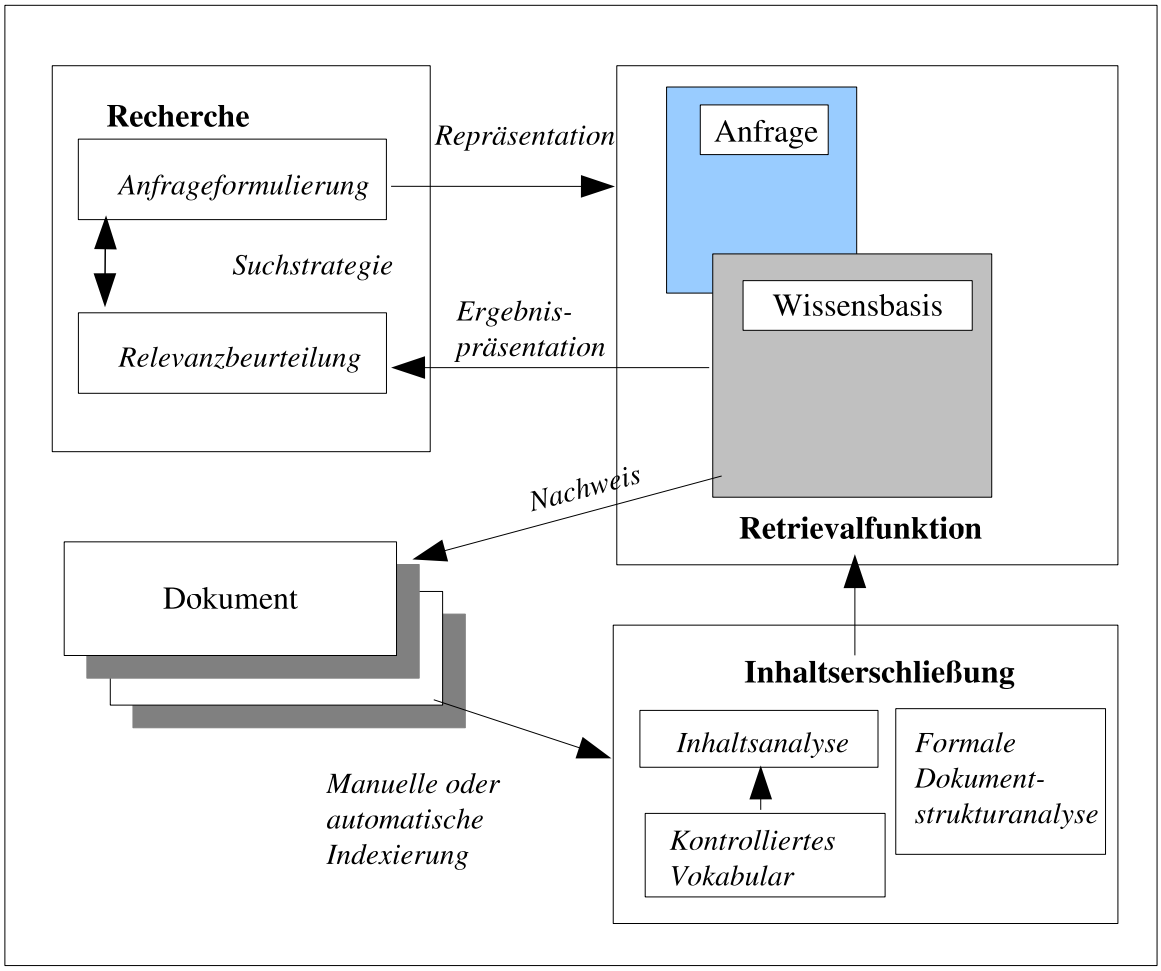


Abbildung 2.2: Information Retrieval System

Modelle des Information Retrieval

Der Abgleich zwischen den Benutzeranfragen und der Wissensbasis in der Retrievalfunktion geschieht durch Anwendung der Regeln, welche vom eingesetzten IR–Modell eines

IRS abhängig ist. Allen IR−Modellen gemeinsam ist die Aufgabe die Vorhersage zu treffen, welche Dokumente für das Informationsbedürfnis eines Benutzers zu einer konkreten Fragestellung relevant sind.

In [Kuh95] wird ein abstraktes IR Modell vorgestellt. Hier wird insbesondere die Zweiteilung auf die *„Verfahren der Inhaltserschließung, Modellierung und Wissensrepräsentationen, die zum Aufbau der rechnerinternen Wissensstrukturen führen“* zum einem, und des Retrieval zum anderen hervorgehoben.

Eine konkrete Aufteilung der IR-Modelle ist die in das Boolesche Modell,Vektormodell und das probabilistische IR−Modell. Abgeleitet von diesen Grundmodellen sind weitere Ansätze wie das erweiterte Boolesche Model,verallgemeinertes Vektormodell und Neuronale Netze, um nur einige aufzuzählen. Die in dieser Anwendung realisierte Suchfunktion basiert auf dem Booleschen Modell, welches im folgenden Abschnitt näher erläutert werden soll.

Boolesches Modell

Dieses auf der klassischen Mengenlehre basierende Modell ist durch die Anwendung von Mengenoperationen auf die Objekte der Wissensrepräsentation, die durch Attributwerte charakterisiert sind, mit dem Ziel, relevante Teilmengen für eine bestimmte Anfrage zu extrahieren, gekennzeichnet. Die Anfragen werden in diesem Modell als Boolesche Ausdrücke formuliert, können somit in der Konjunktiven (KNF) oder Disjunktiven Normalform (DNF) vorliegen. Eine Verknüpfung der elementaren Anfragen ist durch die Operatoren `AND, OR` und `NOT` möglich. Die Konstruktion der Antwortmenge reduziert sich hier auf die Zuweisung der Attribute `true` oder `false`, abhängig davon, ob der gesuchte Term in einem bestimmten Dokument enthalten ist oder nicht. Hier ist leicht zu sehen, dass die Trennschärfe durch die Gewichtung ohne Zwischenstufen mit relevant (1) und nicht relevant (0) sehr hoch ist. Die Antwortmengen können daher relativ groß sein oder überhaupt keine Treffer enthalten.

2.2.3 Indexierung

Die Transformation der wissenstragenden Objekte in eine vom IRS internen Modell abhängige Repräsentationsform, die *Wissensbasis*, kann unter dem Begriff Inhaltserschließung zusammengefasst werden. Vorgänge der Dokumentanalyse die Dokumenten Deskriptoren zuordnen, welche den Inhalt der Dokumente im Index repräsentieren, werden als Indexierung (auch Indizierung) bezeichnet. Die so gewonnenen Deskriptorlisten erfüllen also zwei Funktionen:

- Repräsentation der Dokumente durch stichwortartige Kurzbeschreibung,
- Wiedergabe des Dokumenteninhalts.

Solche Listen können neben den dokumentextbezogenen Inhaltsinformationen auch formale, nicht auf einen Dokumenttext bezogene Informationen wie Autor, Erscheinungsjahr und Verlag beinhalten, wie es auch in der vorliegenden Anwendung der Fall ist. In [GS83] wird der Prozess der Inhaltserschließung als eine der wichtigsten und zugleich schwierigsten Aufgaben der IRS bezeichnet, wobei die Güte der aus dem Indexierungsprozess aufgebauten Wissensbasis die Retrievalqualität maßgeblich diktiert. Ferner weist die gleiche Quelle darauf hin, dass die verwendbaren Retrievalmethoden abhängig vom Indexierungsprozess sind. Grundsätzlich kann bei den Indexierungsverfahren zwischen manuellen und computergestützten automatischen Indexierungsverfahren unterschieden werden. Auch wenn die Rolle der manuellen Indexierung aufgrund ihrer Ergebnisqualität, resultierend aus der Durchführung durch Experten auf bestimmten Wissensgebieten unter Zuhilfenahme verschiedener Hilfsmittel nicht zu verkennen ist, werden heute aufgrund exponentiell steigender Zahlen von Dokumentquellen verstärkt die automatischen Verfahren verbessert und ausgebaut. Die nachfolgenden Ausführungen sollen einen Überblick über die wichtigsten Begriffe und Verfahren aus dem Umfeld des automatischen Indexierens bieten.
Die Termvergabe bei automatischen Verfahren kann anhand der Herkunft der Indexterme aus freiem Vokabular und denen aus einem kontrollierten Vokabular unterteilt werden. Je nach Art des Vokabulars, welches zur Indexierung eingesetzt wird, kann zwischen *singulären* und *kontextbezogenen* Deskriptoren differenziert werden. Nach [GS83] sind kontextbezogene Deskriptoren durch spezifische Relationen verbundene singuläre Deskriptoren. Dabei sind sowohl Einzelbegriffe als auch Mehrwortbegriffe Bestandteil der kontextbezogenen Deskriptoren. Werden erst bei der Suchanfrage die Einzelbegriffe zur thematischen Beschreibung zusammengeführt, wird von *Postkoordination* gesprochen. Der umgekehrte Weg, also die Verwendung kontextbezogener Terme schon bei der Indexierung, wird als *Präkoordination* bezeichnet.
Inzwischen ist eine ganze Reihe von Verfahren zur Deskriptorengewinnung bekannt, welche unterschiedliche Ansätze der Termextraktion und Vergabe verfolgen. Ein vereinfachtes und verbreitetes Verfahren ist hierbei die Eliminierung von Termen mit geringen Beitrag zur Inhaltsbeschreibung unter Verwendung von Stoppwortlisten. Dabei werden die Terme, welche in einer solchen Liste hinterlegt sind, als irrelevant eingestuft und von der Indexierung ausgeschlossen. Ein wesentlicher Nachteil einer solchen Vorgehensweise ist die Betrachtung der Terme als Folge von Zeichenketten und die begrenzte Möglichkeit, verschiedene Ausprägungen eines Wortes als zusammengehörig zusammenzufassen. Eine Verbesserung stellen auf diesem Gebiet die computerlinguistischen Verfahren dar. Dieser Ansatz fasst Terme als bestimmte Formen eines Wortes zusammen, in dem diese durch Anwendung verschiedener linguistischer Regeln auf eine allgemeinere Form gebracht werden. Dabei wird grundsätzlich zwischen der *Grundformreduktion*, also der Reduktion von Wörtern auf ihre grammatikalische Grundform[3], und der *Stammformreduktion*, welche Wörter auf ihren in der Sprache nicht als Wort vorkommenden Stamm zurückführt. Diese Reduktionen führen unter anderem auch dazu, dass :

[3]bei Substantiven auf den Nominativ Singular, bei Verben auf den Infinitiv

- invertierte Listen oder ähnliche Verweisstrukturen aufgrund einer geringeren Termzahl auch deutlich kleiner werden,
- bei der Suche mit zusammengefassten Termen kann auf eine erhöhte Anzahl der gefundene Dokumente gehofft werden.

Inzwischen sind einge Regelwerke für eine solche Reduktion bekannt. Ein Beispiel für ein Regelwerk, mit dem die meisten englischen Begriffe auf ihre Stammform reduziert werden können, ist der Algorithmus von Kuhlen [Kuh77]. Ein weiterer, sehr verbreiteter Algorithmus zur Stammformreduktion (häufig auch als *Stemming* bezeichnet) ist der Porter Algorithmus, welcher zur Deskriptorengewinnung in der hier beschriebenen Applikation zur Anwendung kommt. Eine Algorithmusbeschreibung ist unter [Por] zu finden.

Kapitel 3

Applikationsentwurf

In diesem Kapitel ist die konkrete Konzipierung einer Softwareanwendung zur Publikationsverwaltung beschrieben. Die in der Anforderungsanalyse erfassten erforderlichen Funktionalitäten werden anschliessend als Basis für die Auswahl einer geeigneten Applikationsarchitektur und für den Datenbankentwurf eingesetzt.

3.1 Anforderungsanalyse

Aus der im einführenden Kapitel dargestellten Motivation kann die grundlegende Ausrichtung der hier vorliegenden Applikation erkannt werden. Daraus ergibt sich eine Fokussierung auf den Publikationsprozess aus der Sicht der Wissensproduzenten, ganz konkret der forschenden Personen innerhalb der Lehrstühle als Organisationseinheit. Dies war zugleich auch die Hauptvorgabe für den Entwurf und den Implementierungsprozess. Eine grundlegende und als erfolgsentscheidend angesehene Anforderung war es, ein ausgewogenes Verhältnis von gängigen und somit auch vom Endbenutzer erwarteten Leistungsfunktionen und neuen Funktionen zu finden, um Verbesserungen und Erleichterungen für den Publikationsprozess zu schaffen. Dabei musste sowohl den aufgabenspezifischen als auch den allgemeinen Qualitätsanforderungen gleichermassen Rechnung getragen werden. Die in der Konzeption zu beachtenden allgemeinen Anforderungen können wie folgt hervorgehoben werden :

- *Portabilität und Plattformunabhängigkeit :* Eine grundlegende Erwartung an gut konziperte Anwendungen ist zweifelsohne eine weitgehende Unabhängigkeit von der darunter liegenden Hardwarekonfiguration und den vorliegenden Betriebssystemen.

- *Mehrschichtenmodell :* Benutzerinteraktionen werden über eine graphische Schnittstelle gewährleistet. Diese ist von den darunterliegenden Schichten weitgehend abstrahiert und entkoppelt. Auswirkungen von Änderungen der einzelnen Schichten können somit besser lokal gehalten werden. Dieser Ansatz steigert in hohem Maße die Wartbarkeit und Erweiterungsfähigkeit von Applikationen.

- *Sicherungs- und Wiederherstellungsmechanismen :* An Applikationen wird in der Regel die Anforderung gestellt, in Störfällen angemessen reagieren zu können. Der Schwerpunkt liegt insbesondere auf der Konsistenz und Sicherheit der verwalteten Daten.

Aus der Ausrichtung der Anwendung auf einen speziellen Benutzerkreis resultiert die weitgehend marginale Bedeutung vollständiger bibliothekarischer Klassifikationen. Als wesentlich wurden hingegen Verbesserungen im Veröffentlichungsvorgang erachtet. Diese sollten neben der klassischen Veröffentlichungsvariante von Dokumenten über Formulare durch die Möglichkeit der automatischen Übernahme von Dokumentattributen, welche aus vorgefertigten Einträgen wie etwa Bibtex stammen, erreicht werden. Da die Anzahl der Publikationen in einem bestimmten Zeitraum eine gewisse Größe erreicht, erscheint eine Stapelverarbeitung ganzer Listen solcher Einträge als sinnvoll. Dabei muss die Applikation in diesem Punkt Fehlerrobustheit und Fehlertolleranz aufweisen. In bisher bekannten Anwendungen ist die Dokumentüberlieferung an das System häufig nur über die File-Upload-Option verfügbar. Eine Referenzierung bereits online verfügbarer Dokumente ist an dieser Stelle jedoch von Vorteil, besonders dann, wenn es sich um die Übergabe einer größeren Zahl von Dokumenten in einem Durchlauf handelt.
Für ein effizientes Retrieval der vom System verwalteten Dokumente ist eine Deskribierung mit Schlagwörtern notwendig. Neben der manuellen Vergabe freier Schlagwörter und solcher aus einem kontrollierten Vokabular, ist für die Qualität der Deskribierung eine automatische Indexierung in Ausschlag gebenden Feldern wie Titel und Zusammenfassung (*Abstrakt*) von Vorteil. Eine Suchfunktion ist selbstverständlich unabdingbar, damit das Retrieval überhaupt möglich wird. Da eine ausgiebige Literaturrecherche nicht zu den vorrangigen Zielen der Anwendung gehört, ist der Leistungsumfang der Suchfunktion auf Grundfunktionalität zu beschränken. Dazu gehören Suchmöglichkeiten nach Titel und Autor sowie Schlagwörtern mit der Möglichkeit, diese Attribute mit Booleschen Operatoren zu verbinden. Eine Phrasensuche im Titel erscheint als ausreichend. Die hierarchische Organisation in der Beziehung zwischen dem Lehrstuhl und den zugehörigen Mitarbeitern sollte sich auch im Datenbank- und Anwendungsentwurf niederschlagen. So erscheint die Anforderung logisch, alle von den Lehrstuhlmitarbeitern veröffentlichten Publikationen auch direkt dem jeweiligen Lehrstuhl zuzuordnen. Zudem tritt häufig der Fall auf, dass mehrere Mitarbeiter eines Lehrstuhls gemeinsam Publikationen veröffentlichen. Ein Zugewinn in puncto Komfort ist die automatische Zuordnung von Publikationen zu allen Mitgliedern, welche zu diesem Dokument in einer Koautorenbeziehung stehen. Die Anwendung muss diese Funktionalität auch bei Kooperationen von Mitarbeitern verschiedener Lehrstühle gewährleisten.
Veröffentlichungen werden in der Regel auch in den Webpräsenzen von Lehrstühlen und einzelnen Mitarbeitern aufgeführt und referenziert. Dabei ist die manuelle Pflege solcher Publikationslisten am häufigsten der Fall. Daher ist eine Funktionalität, welche die Generierung und den Export bereits digital veröffentlichter Dokumente in ein für die weitere Verarbeitung geeignetes Format naheliegend. Nicht zuletzt ist auch der Kommunikation zwischen der Anwendung und den Benutzern über eine grafische Schnittstelle

Aufmerksamkeit zu schenken. Diese sollte den Anforderungen einer klaren Gliederung und einer intuitiven Bedienung genügen. Dabei müssen die grundlegenden Konzepte der Mensch-Computer Interaktion beachtet werden.
Teilweise geschützte Daten mit unterschiedlichen Zugriffs- und Modifikationsberechtigungen machen eine differenzierte Zugriffskontrolle erforderlich. Eine weiter Anforderung ist die Ausrichtung der Anwendung auf den Betrieb mit gleichzeitigen Zugriffen mehrer Benutzer.

3.2 Applikationsarchitektur

Die konkrete Applikationsarchitektur orientiert sich an den im vorangegangenen Abschnitt herausgearbeiteten Anforderungen. Um den allgemeinen Qualitätsanforderungen wie Portabilität und Plattformunabhängigkeit zu genügen und die weitgehende Entkopplung unterschiedlicher interagierender Schichten zu erreichen, wurde die Applikation als eine serverseitige webbasierte Anwendung in einer Drei-Schichten-Architektur realisiert. Die Bausteine einer solchen Architektur sind:

- Präsentationsschicht
- Logikschicht
- Datenschicht

Die Präsentationsschicht stellt Funktionalitäten zur Verfügung, welche für die Kommunikation zwischen den Clients und der Applikation zuständig sind. Anfragen der Clients werden durch diese Schicht entgegengenommen und an entsprechende Module zur Weiterverarbeitung gereicht. Die Logikschicht kapselt die allgemeine Prozesslogik, welche für die Verarbeitung von Benutzeranfragen notwendig sind. Die Aufgaben der persistenten und konsistenten Datenhaltung werden von der Datenschicht übernommen. Der Aufbau einer Drei-Schichten Architektur ist in Abbildung 3.1 dargestellt.

3.2.1 Präsentations- und Logikschicht

Für die Implementierung der grafischen Benutzerschnittstelle (*Präsentationsschicht*) wurde aus der Fülle möglicher Technologien die *Java Server Pages* Technologie als Erweiterung des 1997 eingeführten Java Servlet API aus dem Grund ausgewält, da diese Kombination es ermöglicht, eine mächtige objektorientierte Sprache in einer Serverumgebung einzusetzen. Ein bedeutender Vorteil ist hierbei die Möglichkeit bestehende Java API's, wie es zum Beispiel das JDBC API für die Datenbankverbindung ist, einzusetzen. Die darunterliegende Kapselung der Prozesslogik (*Logikschicht*) ist demnach in der Programmiersprache Java realisiert. Die JSP-Technologie lehnt sich prinzipell an die schon bekannten Konzepte der serverseitigen Programmierung, wie es zum Beispiel PHP oder

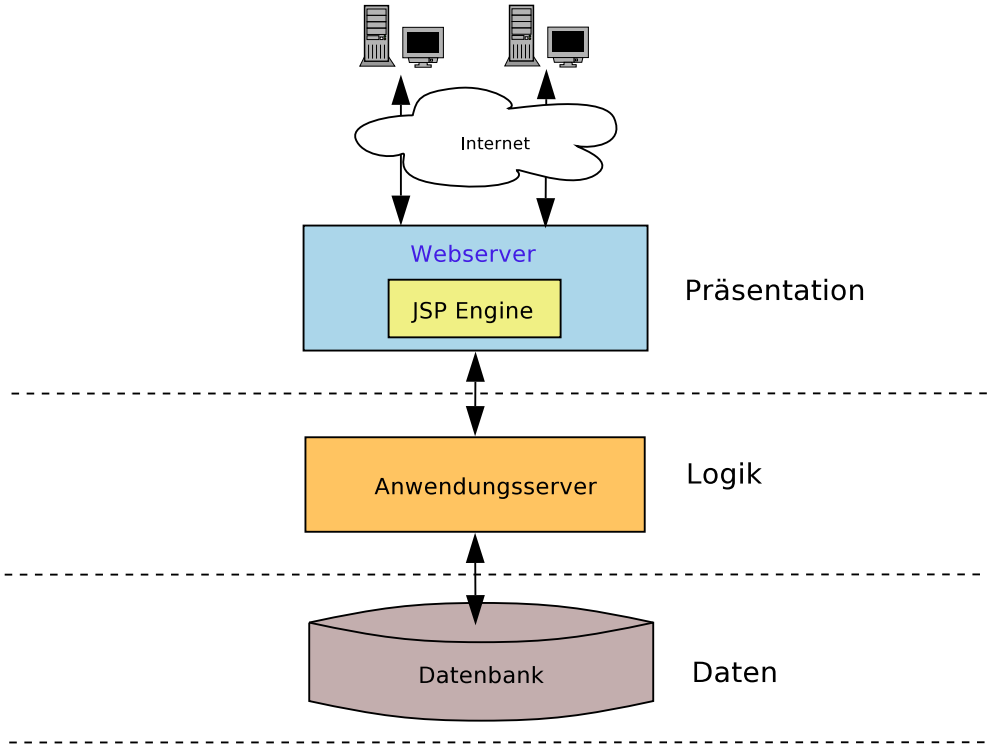


Abbildung 3.1: Die Drei-Schichten-Architektur

ASP sind. Auch hier werden statische HTML-Inhalte durch Einbettung dynamischer Anteile angereichert. Der JSP-Ansatz benutzt für diese Dynamisierung die objektorientierte Programmiersprache Java. In der ursprünglichen Konzeption von JSP erfolgt die Einbettung des Java-Quellcodes direkt in die HTML-Implementierung in Form von Scriptlets. Dieser Ansatz führt jedoch rasch zu einer sehr unübersichtlichen Quellcodelandschaft. Zudem trägt eine solche Vermischung der Darstellungs- und Verarbeitungslogik zu einer wesentlich geringeren Wartbarkeit und gleichzeitig einer erhöhten Fehleranfälligkeit des Quellcodes bei. Die Wiederverwendbarkeit solcher Impelmentierungen sinkt stark mit zunehmender Komplexität. Da die Web-Technologien im wesentlichen keine Trennung der Präsentationslogik und der Verarbeitungsprozesse erzwingen, sind weitere Konstrukte für eine klare Struktur notwendig. Eine solche Struktur ist das im Nachfolgenden beschriebene MVC-Entwurfsmuster.

Das MVC-Entwurfsmuster

Das MVC ist ein aus der Implementierung gewöhnlicher Benutzeroberflächen bekanntes Design Pattern, welches eine strikte Trennung der Darstellung von der Applikationslogik vorsieht. Das Akronym MVC steht dabei für die Begriffe *Model, View und Controller*. Die Kapselung der Prozesslogik (häufig auch als Geschäftslogik bezeichnet), erfolgt in der Modell-Komponente. Die Präsentationsebene des Modells, welche den Zustand der Modell-Komponente wiedergibt, ist in der View-Komponente enthalten. Aufgaben

der Kommunikationssteuerung zwischen Modell- und View-Komponenten sind in der Controller-Komponente enthalten. Dieser Zusammenhang der einzelnen MVC - Komponenten ist zur Verdeutlichung in Abbildung 3.2 dargestellt. Eine kurze Übersicht der MVC-Konstrukte ist in [MVC] gegeben.

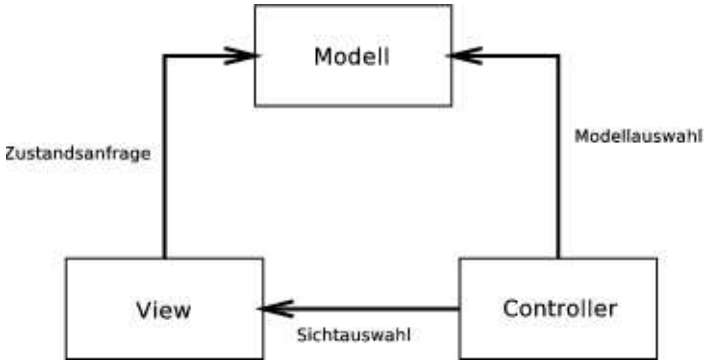


Abbildung 3.2: Das MVC − Entwurfsmuster

Die technische Grundlage der Implementierung nach dem MVC-Design Pattern der hier vorliegenden Anwendung ist eines der prominentesten MVC-Frameworks im Bereich der Javabasierten Webanwendungen, das *Struts Framework*. Seine Grundkomponenten werden im Nachfolgenden näher erläutert.

Das Struts MVC- Framework

Struts stellt ein stabiles Open-Source-Framework zur Entwicklung von Web-Applikationen nach dem MVC-Paradigma dar. Diesem Konzept zu Grunde liegt die Programmiersprache Java mit den Kerntechnologien Java Beans, Java Servlets, Resource Bundles und XML. Dieses von Craigh R. McClanahan initiierte und geleitete Projekt ist seit 2000 Bestandteil der Apache Software Foundation [Str]. Das in Struts verwendete MVC Modell stellt eine für Web-Anwendungen angepasste Version des ursprünglichen Modells dar. Im klassischen MVC-Modell informiert die Prozesslogik (*Model*) die Präsentationsschicht (*View*) über die zu visualisierenden Zustandsänderungen. Da in einer Web-Umgebung die Kommunikation zwischen dem Client und dem Server statuslos ist, sind für eine stets aktuelle Präsenation von Zuständen häufige Serverabfragen durch den Browser notwendig. Ein weiterer Unterschied zum klassischen Modell ist die Tatsache, dass die für die Implementierung der Views verwendete Technologie in der Regel eine andere ist als die der *Model* und *Controller*-Implementierung. Während die Views am häufigsten mit der JSP Technologie realisiert werden, basieren die Implementierungen der letzten zwei Komponenten auf der Programmiersprache Java. Die wesentlichen Bestandteile und die Grundfunktionalität des Struts Frameworks illustriert die Abbildung 3.3

Das Action Servlet nimmt alle mit der `.do`-Endung versehenen Anfragen vom Webbrowser entgegen und interpretiert die übergebenen Parameter. Ist die Anfrage mit Formulardaten versehen, so erzeugt diese Komponente eine ActionForm, ein Java-Objekt, welches

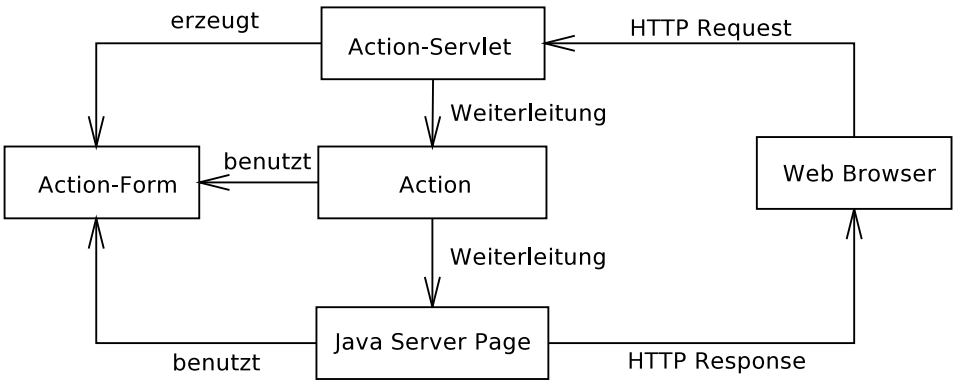


Abbildung 3.3: Die Komponenten von Struts

die Formulardaten speichert. Anschliessend wird aus dem Anfragepfad die entsprechende Action herausgesucht, und die Anfrage zur weiteren Verarbeitung geleitet.
Die Actions implementieren die Funktionalitäten, welche als Prozesse vom Benutzer über die grafische Oberfläche ausgelöst werden können. Dazu zählen zum Beispiel das Schreiben der Formulardaten in die Datenbank. Bei Bedarf besteht hier die Möglichkeit, Daten aus dem zugehörigen Action Form-Objekt auszulesen. Die Resultate der Verarbeitung können weiter in Java-Objekte abgelegt werden, welche wiederrum zur weiteren Verwendung im Request oder im Session-Attribut abgelegt werden können. Die Entscheidung über den Speicherort ist von der weiteren Bedeutung der Resultate abhängig. Sind diese von globaler Bedeutung für den gesamten Interaktionsprozess einer Sitzung, so werden sie in einem Session-Attribut abgelegt. Als View kommen in diesem Fall JSP-Seiten zum Einsatz, welche anhand der von der Action überlieferten Objekte eine HTML-Ausgabe erzeugen, und diese an den Browser des Benutzers weiterleiten.
Die Zuordnung der entsprechenden Actions, Action-Forms und Views erfolgt nicht im Quellcode selbst, sondern in einer in allen Struts-basierten Anwendungen standardmäßig vorhandenen XML-Konfigurationsdatei, der `struts-config.xml`. Diese Vorgehensweise bringt einen entscheidenden Vorteil, da die statische Kodierung der Abbildungen der Requeest-URIs auf die Action-Klassen unterbunden wird, um stattdessen in einer zentralen Datei verwaltet zu werden. Dadurch wird eine schnelle Änderung der Abläufe ermöglicht. Abbildung 3.4 zeigt einen Auszug der `struts-config.xml` Datei mit den wichtigsten Konfigurationselementen. Das *form-beans*-Tag dient der Definition von Action Form Objekten als Strukturdefinition und Container der Formulardaten. Die *global-forwards* definieren allgemein gültige Zuordnungen von Views für die gesamte Web-Applikation. Dieses konkrete Beispiel zeigt die Definiton einer globalen Fehlerseite. Die *action-mappings* nehmen in der Regel den größten Teil der Konfigurationsdatei ein. Diese dienen der Konfiguration der Benutzeraktionen. Jedes Action-Mapping ist genau einer Aktion zugeordnet, wobei die Parameter die Implementationsklasse für die Funktionalität der Benutzeraktion festlegen, den Bereich der Sichtbarkeit also den *Scope* der Komponente (`request, session, application`) definieren, und die Quelle der zu verarbeiteten Formulardaten als Mitteilungen der bestimmten Benutzeraktion übermitteln.

```
<struts-config>
  <form-beans>
    <form-bean
      name="dynaLinkForm"
      type="forms.DynaLinkForm"/>
    </form-beans>
    <global-forwards>
      <forward name="error_report"
      path="/pages/memberarea/error_report.jsp"/>
    </global-forwards>
    <action-mappings>
      <action path="/dynaLinkAction"
       scope="request"
       name="dynaLinkForm"
       type="actions.DynaLinkAction"
       input="DynaLinkOptions.jsp"/>
    </action-mappings>
</struts-config>
```

Abbildung 3.4: Beispiel einer struts-config.xml

Eine Einführung in die Thematik der Java Server Pages ist in [Deh03] gegeben. Eine ausführliche Dokumentation des Struts-Frameworks ist in [HBMW05] bzw. [Str] anzutreffen.

3.2.2 Datenschicht

Die Forderungen nach Datenkonsistenz und Datensicherheit gepaart mit zuverlässigen Wiederherstellungsmechanismen machen den Einsatz eines Datenbank-Management-Systems erforderlich. Bei dieser Komponente, welche in der Datenschicht der Architektur angesiedelt ist, kommt das Open Source DBMS PostgreSQL in der Version 8.0 zum Einsatz. Viele der in kommerziellen Datenbansystemen üblichen Funktionalitäten wie Transaktionssicherheit und Einsatzmöglichkeiten prozeduraler Erweiterungen, sowie die der verschachtelten SQL-Anfragen und umfangreicher Datenstrukturen für die Indexerstellung, sind auch in diesem DBMS anzutreffen. Ein besonderes Augenmerk bei der Auswahl des DBMS war auch die Möglichkeit eine Datenbankgesteuerte Indexierung von Dokumenten. Kombiniert mit der Unterstützung von GIST Indexen für eine effiziente Anfragebearbeitung zu Retrievalzwecken liefert das Postgres DBMS eine gute Grundlage für die Implementierung einer Retrieval Funktion. Nähere Erläuterungen zu den Retrieval Funktionalitäten folgen im Kapitel 4.
Die Anwendung ist in einer wie folgt angelegten Systemumgebung implementiert und getestet worden:

- Betriebssystem : Suse Linux in der Version 9.3

- Web / Anwendungsserver : Apache Tomcat in der Version 5.5.12 [Tom]
- DBMS : PostgreSQL in der Version 8.0 [Pos]

3.3 Datenbankentwurf

Auf die zentrale Bedeutung der Datenbank als gemeinsame Informationsbasis für Benutzer und Anwendungen wurde in dieser Arbeit bereits hingewiesen. Daher ist es besonders wichtig, dem Prozess des inhaltlichen Aufbaus der Datenbank, dem Datenbankentwurf, eine besondere Aufmerksamkeit zukommen zu lassen. Der Entwurfsprozess läuft grundsätzlich in vier Phasen ab :

1. Anforderungsanalyse
2. Konzeptioneller Entwurf
3. Logischer Entwurf
4. Physischer Entwurf

Die Anforderungsanalyse dient der Erfassung der Informationsanforderungen und der Bearbeitungsanforderungen, welche an das zu entwerfende Datenmodell gestellt werden. Zu den Informationsanforderungen zählen alle statischen Daten, welche von der Datenbank für den späteren Betrieb bereitgehalten werden sollen. Dazu zählen die Realwelt-Objekte mit ihren Datentypen, die Objekte klassifizierenden Eigenschaften - Attribute und ihre Wertebereiche sowie die zwischen den Objekten bestehenden Abhängigkeiten.
Die Bearbeitungsanforderungen liefern Aufschlüsse über die dynamischen Aktivitäten und Prozesse, welche während des Betriebes auf der Datenbank ablaufen sollen. Insbesondere sind die Anfragen, die zu beantworten sind, Auswertungen, Datenveränderungen (*Aktualisieren und Löschen*) sowie Berichterstellungen von Bedeutung. Ein weiterer wichtiger Aspekt ist die erwartete Häufigkeit dieser Operationen, welche den Entwurf mitprägen.
Der konzeptionelle Entwurf dient der formalen Beschreibung der Informationsstrukturen nach den in der Anforderungsanalyse erkannten Benutzergruppen und Informationsanforderungen. Diese Ebene des Datenbankentwurfs ist unabhängig vom später eingesetzten DBMS. Dennoch ist es wichtig, für die formale Datenbeschreibung ein Modell zu wählen, welches genaue Regeln für die Übersetzung in ein konkretes Datenbankschema bereitstellt. Da der Einsatz von relationalen DBMS geplant ist, erfolgt die formale Beschreibung typischer Weise im sog. Entity-Relationship-Modell, kurz ER-Modell. Das Resultat dieser Phase ist ein konzeptionelles Schema. Im logischen Entwurf erfolgt dann die Transformation des konzeptionellen Schemas in das Datenmodell eines konkreten DBMS.
Der physische Entwurf betrifft die Entwurfsüberlegungen auf einem sehr tiefen Niveau der Datenorganisation. Hierbei werden Aspekte der Datenorganisation auf Datenträgern

mit den entsprechenden Zugriffsmethoden erwogen, die Auswahl entsprechender Datenspeicherstrukturen sowie eine Kostenanalyse der I/O Zugriffe durchgeführt. Nach [Vos00] ist die zweite Phase, der konzeptionelle Entwurf, eine der wichtigsten und anspruchvollsten Phasen des Datenbankentwurfs. Dementsprechend wird im Folgenden auch primär auf diese Entwurfsphase der dieser Applikation zugrunde liegenden Datenbank eingegangen.

3.3.1 Konzeptioneller Entwurf

Da die Aufgabe der Publikationsverwaltung aus mehreren Teilaufgaben besteht, können auch die zu modellierenden Informationsstukturen in mehrere kleinere und zunächst voneinander unabhängige Datenansichten geteilt werden[1]. Im weiteren Verlauf werden diese Teilaufgaben nach Teilmengen aufgeteilt unter Verwendung des *Entity-Relationship-Modell* modelliert und die Modellierung schematisch dargestellt. Erst im letzten Schritt erfolgt eine Fusion der Teilmodelle in ein Gesamtmodell. Die Kardinalitäten der Beziehungen werden vereinfacht dargestellt
Das erste und grundlegende Datenkonstrukt in der Publikationsverwaltung ist die Menge der zu verwaltenden Publikationsarten mit ihrer Zuordnung zu ihren Autoren. Das ER-Teilmodell dieser Objekte und ihrer Beziehung ist in 3.5 dargestellt.

Die generalisierte Entiät *Publication* kapselt die allen Publikationstypen gemeinsamen Attribute. Diese werden an alle abgeleiteten Entitäten vererbt. Somit beinhalten diese „erbenden“Entitäten nur noch die speziellen Attribute. Diese Art, den Zusammenhang zwischen der abstrakten Publikation und den speziellen Publikationstypen zu modellieren, hat neben der Andeutung einer Teilmengenbeziehung einen weiteren Vorteil. Ein nicht geringer Teil der Publikationsverwaltung besteht aus der Auswahl von Publikationen nach bestimmten Selektionskriterien. Dies kann etwa im Rahmen einer Publikationssuche durch die Benutzer geschehen. Zur effizienten Bearbeitung gehört in einer solchen Anforderungsumgebung eine Datenorganisation, welche eine weniger detaillierte, aber vollständige Übersicht aller verfügbarer Dokumente mit ihren Zuordnungen zu den einzelnen Autoren und den Lehrstühlen zugleich bereithält. Beispielsweise werden alle im System verfügbaren Dokumente für ein effizientes Retrieval gemäß der Erläuterungen im Abschnit 2.2 mit beschreibenden Attributen, den Deskriptoren, versehen. Diese können sowohl frei vom Verfasser vergeben werden als auch aus einem kontrollierten Vokabular stammen. Da diese zum einen das am häufigsten bei Suchanfragen referenzierte Attribut darstellen, zum anderen auch allen Publikationstypen gemeinsam sind, werden diese in der generalisierten Publikationsentität modelliert. Auf diese Weise wird den Effizienzanforderungen in diesem konkreten Fall entsprochen.
Die Auswahl der abgeleiteten Publikationstypen beschränkt sich auf die am häufigsten vetretenen Publikationstypen, Bücher und deren einzelne Kapitel, Aufsätze, Beiträge zu Konferenzen und Abschlussarbeiten wie Master- und PHD-Arbeiten. Als Sammelcontainer für Publikationsarten, welche nicht durch einen der modellierten Typen abgedeckt

[1] Dieser Ansatz ist als „divide and conquer“Entwurfsregel in der Informatik bekannt

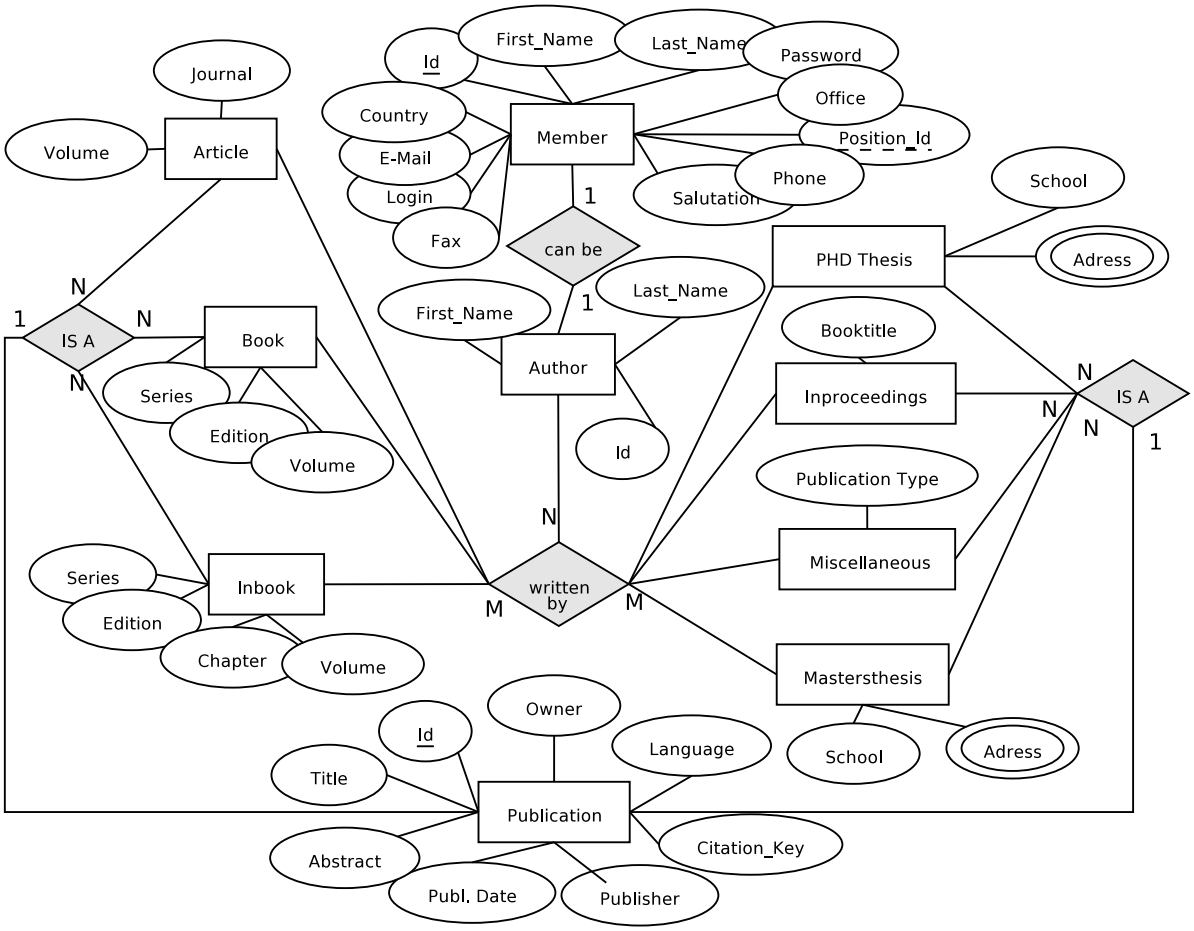


Abbildung 3.5: Teilmodell : Publikationstypen mit Zuordnung zu Autor und Benutzerkonto

sind, fungiert die Publikationsart *Miscellaneous*, in Anlehnung an die gängige Praxis bei BibTex Einträgen.
Die Zuordnung der Dokumente zu den Autoren ist durch die Beziehung *written_by* gegeben. Dabei gilt es zu beachten, dass außer in Sonderfällen (*Abschlussarbeiten*), die Publikationen von einem oder mehreren Autoren verfasst sind. Nicht alle Autoren besitzen zunächst ein Benutzerkonto im System. Daher ist die Beziehung zwischen den Autoren und den Benutzern als eine mögliche Relation dargestellt (*can be*). Die wesentliche Anforderung der automatischen Erkennung von Koautorenbeziehungen bei Erstellung eines Benutzerkontos ist in der *Autor*-Entität modelliert. Dabei ist für alle Publikationen, welche bis dahin unbekannte Autoren besitzen, auch ein entsprechender Eintrag in der Autorenrelation vorgesehen. Um die Konsistenz auch zu den durch BibTex-Einträge importierten Dokumenten zu wahren, ist auch eine BibTex-konforme Namensdarstellung im Attribut *bibtex_format_name* der Autor-Entität bereitgestellt.
Die Zuordnung der Benutzerkonten zu den einzelnen Lehrstühlen sowie der Position, welche der Kontobesitzer in dieser Organisationseinheit innehält, ist in Abbildung 3.6

gezeigt. Da die Modellierung der Entität *Member* schon ausführlich beschrieben wurde, ist diese hier aus Darstellungsgründen ohne ihre Attribute aufgezeigt. Ein Benutzerkonto (dargestellt durch *Member*) ist genau einem Mitglied zugeordent. Die Mitglieder wiederum gehören genau einem Lehrstuhl an, mit genau einer Position innerhalb des Lehrstuhls. Aufgrund der hierarchischen Beziehung zwischen dem Lehrstuhl und den zugehörigen Mitarbeitern, ist zur zentralen Verwaltung aller Mitarbeiterpublikationen für den einzelnen Lehrstuhl jeweils ein Benutzerkonto angedacht. Die Information über ein bestehendes Konto ist im Attribut *Account_created* modelliert.
Der letzte Schritt dieser Entwurfsphase ist die Vereinigung der Teilmodelle in ein gesamtes ER Modell. Dieses ist aus Darstellungsgründen im Anhang A angefügt.

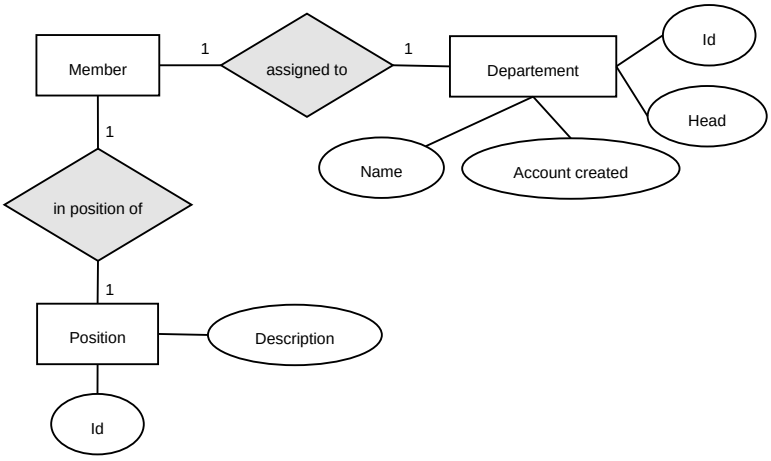


Abbildung 3.6: Teilmodell : Beziehung der Entitäten Benutzerkonto, Position, Abteilung

3.3.2 Logischer Entwurf

Die logische Entwurfsphase dient der Überführung des konzeptionellen Schemas anhand der spezifischen Transformationsregeln in ein konkretes Datenbankschema. Da in diesem konkreten Fall ein relationales DBMS vorliegt, erfolgt die Transformation des ER-Modells in ein Relationenmodell. Das dazu benötigte Grundregelwerk ist in [Vos00] wie folgt beschrieben:

(1) Jeder Entity-Typ wird in ein Relationenschema transformiert.

(2) Jeder Relationship-Typ [2] wird ebenfalls in ein Relationenschema transformiert, es sei denn, es handelt sich um eine zweistellige $1 : 1$ oder $1 : n$ −Beziehung; in diesen Fällen reicht die Hinzunahme von Attributen zu bereits existierenden Relationenschemata.

(3) IS-A-Beziehungen werden allein über IND's[3] ausgedrückt.

[2] Beziehungstyp
[3] Inklusionsabhängigkeiten zur Formalisierung inhärenter Teilmengenabhängigkeiten

Zusätzlich zu diesem Regelwerk sind auch die Aspekte der Primärschlüsselbedingungen und die der referentiellen Integrität, wie in Abschnitt 2.1.1 beschrieben zu beachten. Abbildung 3.5 zeigt stellvertretend für das Gesamtmodell ein Relationenschema als Resultat der Transformation des ER-Teilmodells aus Abbildung 3.7.

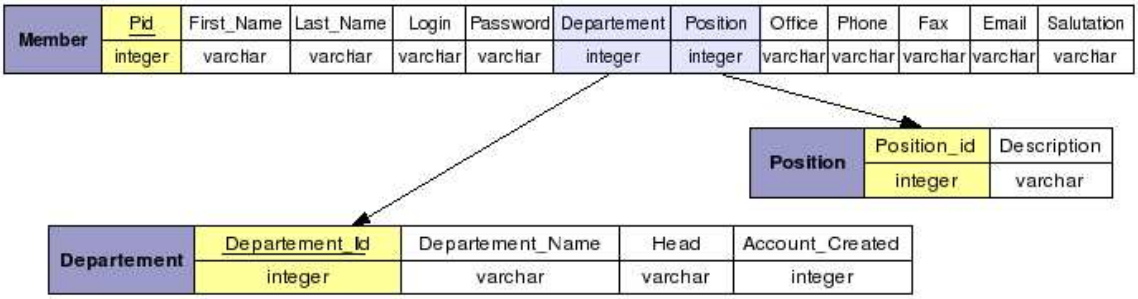


Abbildung 3.7: Relationenschema : Benutzerkonto Position , Abteilung

In den darin abgebildeten Relationenschemata sind die Primärschlüssel unterstrichen dargetsellt. Die Fremdschlüsselattribute sind grau hinterlegt, Pfeile deuten auf die Primärschlüssel der in Beziehung stehenden Relationen, zu welchen eine referentilelle Integritätsbeziehung besteht. Die Wertebereiche der Attribute sind jeweils auf die von dem DBMS unterstützten Typen abgebildet. Die Primärschlüssel als Identifikatoren sind als ganzahlige Datentypen realisiert, was zu einer schnelleren Anfragebearbeitung führt[4].

```
CREATE TABLE author (
    author_id integer DEFAULT nextval('author_id_seq') NOT NULL,
    first_name character varying(32),
    last_name character varying(32),
    organisation text,
    member_id integer,
    bibtex_format_name text
    primary key (author_id),
    foreign key (member_id) references member(pid));
```

Abbildung 3.8: DDL Anweisung : Erzeugen einer Datenbanktabelle

Die Definiton so erhaltener Relationenschemata in der Datenbank erfolgt über eine für die Datendefinition vorgesehene Teilmenge der deklarativen Datenbanksprache SQL, die DDL (Das Akronym DDL steht für *data definition language*). Zuvor müssen selbstverständlich die Datenbank selbst sowie die Benutzer angelegt werden, welche die zu definierenden Tabellen erzeugen und auf diese später zugreifen dürfen. Abbildung 3.8

[4]Vergleichsoperationen auf Zahlentypen sind in der Regel um ein Vielfaches schneller vom DBMS ausgeführt als Stringvergleiche

zeigt eine DDL -Beispielanweisung zur Erzeugung einer Datenbanktabelle. Ähnliche Anweisungen wurden verwendet um die Basisrelationen der Publikationsdatenbank zu generieren.

Kapitel 4

Implementierung

Dieses Kapitel dokumentiert die konkrete Umsetzung der aus der Anforderungsanalyse hervorgehenden Anwendungsfunktionen konform mit der in Kapitel 3 entworfenen Applikationsarchitektur. Der erste Abschnitt bietet einen Einblick in die Paketorganisation des Java-Quellcodes. Die weiteren Abschnitte sind den für die Anwendung essentiellen allgemeinen Modulen wie Datenbankanbindung und Suchfunktionalität gewidmet. Anschließend werden die Anwendungsfunktionen mit den jeweiligen Problemstellungen und Lösungsansätzen erläutert. Eine Gesamtübersicht der wesentlichen und in diesem Kapitel beschriebenen Anwendungsmodule zusammen mit den wichtigsten Beziehungen zwischen diesen ist in Abbildung 4.1 dargestellt.

4.1 Aufbau der Implementierung- Klassenpakete

Eine übliche Praxis in der Java-Implementierung ist die Kapselung zusammenhängender Klassen und Methoden in Pakete. Diese Organisationsform erhöht die Übersicht und die Wiederverwendbarkeit von Code an mehreren Stellen der Applikation, ohne Redundanzen erzeugen zu müssen.
Die Paketstruktur der hier vorliegenden Java-Implementierung orientiert sich gleichwohl an den Modellevorgaben des Struts-Frameworks und der darin verlangten MVC-Strukturen sowie den erfassten Anforderungen. Die Modell-Komponente ist im Paket `Forms` angesiedelt. Die Action-Klassen der Controller-Schicht sind in den nach der gleichen Namenskonvention benannten `Action` Paket anzutreffen. Zudem sind weitere Hilfsklassen notwendig, welche wie folgt gegliedert sind :

- Container-Klassen, welche für die Zuordnung der Datenbankrelationen zu den Java-Objekten der Applikation benotigt werden, sind im Paket `dbmapping` angesiedelt.

- Klassen, welche den Datenbankzugriff koordinieren und spezielle Abläufe auf der Datenbank definieren, sind im Paket `dbactions` zusammegefasst.

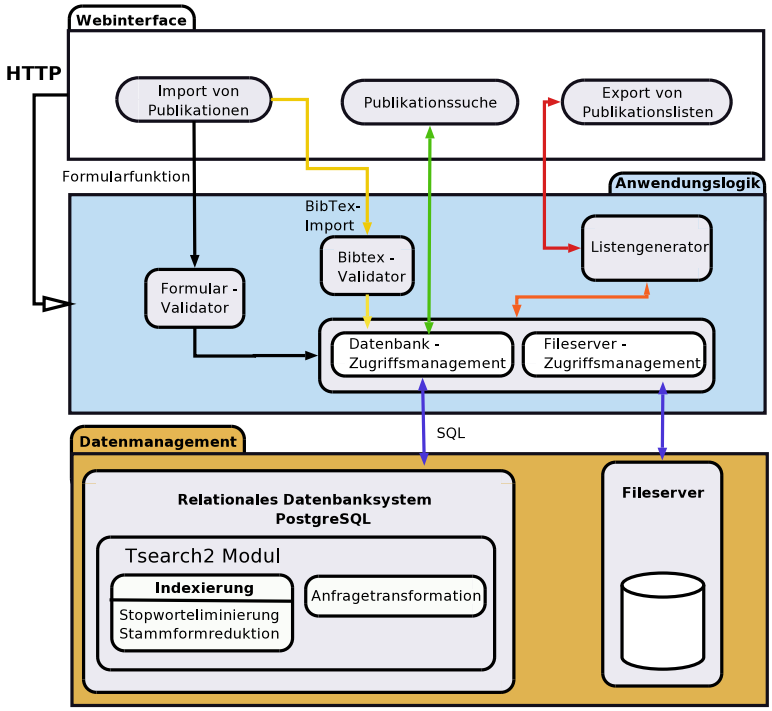


Abbildung 4.1: Übersicht der Anwendungsmodule

- Funktionalitäten, welche die Eingabevalidierung bereitstellen oder den Umgang mit verschiedenen Zeichenkodierungen der die Applikation konnektierenden Clients koordinieren, sind konventionsgemäß im Paket `ressources` angesiedelt.

- Letztlich sind alle Klassen, welche allgemeine Hilfsroutinen wie das Parsing und die Validierung von Bibtexeinträgen bereitstellen oder die Indexierung der Dokumente gewährleisten, im Paket `common_utils` anzutreffen.

4.2 Datenbankanbindung

Die Interaktionen der Benutzer mit dem System sind bis auf wenige Ausnahmen immer auch mit gewissen Operationen auf den Daten der Datenbank verbunden. Daher ist ein effizienter und zuverlässiger Datenbankzugriff von sehr großer Bedeutung. Effizienzverluste, welche durch wiederholtes Aufbauen der Datenbankverbindung entstehen, können durch sogenannte *Connection-Pooling-Mechanismen* umgangen werden. Dabei handelt es sich um einen Ansatz, bei dem eine ganze Reihe von Datenbankverbindungen vorinitialisiert werden. Die Datenbankanbindung dieser Applikation basiert auf dem JDBC API [JDB], einer Datenbankschnittstelle, welche ursprünglich für C++ Applikationen implementiert wurde. Das JDBC API stellt das in JAVA implementierete Gegenstück

dar. Auch das JDBC API bietet Möglichkeiten für den Aufbau vordefinierter Verbindungen. Abbildung 4.2 illiustriert die Definition eines Connection Pools.

```
private void createConnectionPool(){

    Props db = new Props();
    setF("dbacces.properties");
    db.setFile(getF());
    setPropdb(db.getProps());
    setSource(new Jdbc3PoolingDataSource());
    getSource().setDataSourceName(getPropdb().getProperty("DataSourceName"));
    getSource().setServerName(getPropdb().getProperty("ServerName"));
    getSource().setDatabaseName(getPropdb().getProperty("DatabaseName"));
    getSource().setUser(getPropdb().getProperty("User"));
    getSource().setPassword(getPropdb().getProperty("Password"));
    getSource().setMaxConnections(100);

}
```

Abbildung 4.2: Definition eines Connection Pools

Die benötigten Zugangsdaten wie Datenbankname, Serveradresse, Verbindungsport und Authentifizierungsdaten werden aus einer Property-Datei gelesen. Anschließend wird eine Instanz des `Jdbc3PoolingDataSource`-Objekts als Container für die zukünftige Datenbankverbindungen erzeugt. Dieser Instanz werden zur Initialisierung die zuvor gelesenen Verbindungsdaten über die vorhandenen Setter-Methoden mitgeteilt. Im letzten Schritt wird die Anzahl der persistierenden Datenbankverbindungen angegeben. Die hier dargestellte Methode ist Bestandteil der `DbRw`-Bean aus dem Paket `dbactions`. Eine Instanz dieser Bean wird beim Applikationsstart einmalig im Scope *Application* erzeugt. Der Applikation-Scope[1] garantiert die Existenz und Sichtbarkeit der Bean wärend der gesamten Laufzeit der Applikation.
Die in Abbildung 4.3 abgebildete Methode zeigt ein Beispiel der Datenbankverbindung mit Hilfe des Connection-Pooling Mechanismus. Dabei wird zunächst aus der Menge der verwalteten persistenten Verbindungen eine Verbindung angefordert. In diesem Verbindungsobjekt erfolgt dann die Intialisierung eines abstrakten SQL-Statement-Objekts als Behälter für die konkret durchzuführende SQL-Anfrage. Diese wird dem Behälter als eine Zeichenkette übergeben. Die so vorbereitete Anfrage kann nun auf der Datenbank ausgeführt werden. Das Ergebnis der Anfrage wird in das `ResultSet`-Objekt eingebettet. Durch iterativen Durchlauf kann nun auf die einzelnen Tupel der Ergebnismenge zugegriffen werden. Die Anweisung für das Schließen der Datenbankverbindung löst lediglich die Entkopplung der Verbindung vom ausleihenden Prozess und die Rückgabe dieser in den Connection-Pool aus.

[1] *Scope* steht dabei für den bereich des Anwendungscodes, in dem auf eine bestimmte Ressource zugegriffen werden kann. Es handelt sich also um den Sichtbarkeitsbereich einer Ressource.

```
public synchronized int checkLogin(String login,String password){
    int auth=-1;
    String query="select * from checklogin('"+login+"','"+password+"');";
    try {
        setConn(getSource().getConnection());
        java.sql.Statement stmt=getConn().createStatement();
        setRset(stmt.executeQuery(query));
        if(getRset().next()) {
            if (getRset().getInt("checklogin")!=-1){
                auth=getRset().getInt("checklogin");}
            return auth;
        } else return auth;
    }catch (org.postgresql.util.PSQLException pex ) {
        pex.printStackTrace();
        auth=-2;
    return auth;
    }catch (SQLException e) { e.printStackTrace();
    finally {
    try { getConn().close(); } catch(Exception e) {
        System.err.println(e.getMessage()) ; }}
}
```

Abbildung 4.3: Beispiel einer Datenbankverbindung

4.3 Das Tsearch2-Modul

Die Aufbereitung der publizierten Dokumente für das spätere Auffinden mittels der Suchfunktion stellt eine der wesentlichen Aufgaben eines Publikationsverwaltungssystems dar. Dazu werden die Inhalte der Dokumente wie im Abschnit 2.3.3 erläutert anhand von inhaltbeschreibenden Stichwörten, den Deskriptoren, abgebildet. Dieser Vorgang wurde als das Indexieren bezeichnet. Die so gewonnenen Indexe werden für eine effektive Auswertung der Suchanfragen in einer geeigneten Datenstruktur abgelegt. Letztlich sind effiziente Zugriffsmethoden für das Auffinden relevanter Dokumente zu einer bestimmten Suchanfrage notwendig. Die Lösungsansätze dieser drei Aufgabenteile der hier vorliegenden Anwendung basieren auf dem Tsearch2-Modul [Tse]. Dabei handelt es sich um eine optionale Erweiterung der PostgreSQL-Distributionen, welche die benötigten Grundfunktionalitäten für einen effizienten Umgang mit Textdaten bietet. Somit wird die gesamte Funktionalität, welche mit der Indexierung von Textfragmenten, Speicherung dieser Indexe und der Auswertung von Suchanfragen über dem Dokumentbestand verbunden ist, in der Datenbank realisiert. Aufgrund ihrer Bedeutung für die gesamte Anwendung sollen die Grundkonzepte dieses Moduls näher betrachtet werden.
Das Tsearch2-Modul bietet die Möglichkeit der Extraktion in Texten enthaltener Wörter in Listen, es implementiert ferner eine spezielle Datenstruktur für die Speicherung so gewonnener Listen. Die Eliminierung für die Inhaltsbeschreibung irrelevanter Wörter wie Artikel, Konjunktionen und Präpositionen ist mit Hilfe frei konfigurierbarer Stoppwortlisten möglich. Da die Wörter in Texten in abgewandelten Formen anzutreffen sind, ist bei den Indexierungsvorgängen häufig eine Reduktion dieser auf eine allgemeinere Form, die Stammform, zu beobachten. Hier bietet Tsearch2-die Möglichkeit, eine

Stammformreduktion nach dem Porter-Algorithmus [Por] einzubeziehen. Zu diesem Zweck können die Stemmer Algorithmen der Snowball-Projekte [Sno] integriert werden, welche wahlweise als C- oder Java-Implementierung frei verfügbar vorliegen. Der parallele Einsatz mehrerer Stemming-Algorithmen macht eine Sprachunterscheidung und sprachabhängige Stammformreduktion möglich. Dazu ist sebstverständlich auch das Wissen notwendig, um welche Sprache es sich bei einem konkret zu verarbeitenden Textfragment handelt. Die Realisierung der Spracherkennung ist am Ende dieses Abschnitts gegeben.

4.3.1 Indexieren mittels Tsearch2

Die Konversion von Texten in eine Wortliste geschieht durch die Funktion `to_tsvector`(). Dabei wird der vorliegende Text zunächst durch das *Parsen* in *Tokens* zerlegt. Anschliessend erfolgt die Stoppworteliminierung anhand der passenden Stoppwortliste. Die Auswahl der Stoppwortliste erfolgt nach dem übergebenen Parameter, welcher die anzuwendende Sprache definiert. Aus dieser Vorverarbeitung entstandene stoppwort- und redundanzfreie Wortliste wird der Stammformreduktion nach dem Porter-Algorithmus unterzogen. Auch hier ist der anzuwendende Algorithmus sprachabhängig und durch die im Parameter übergebene Sprachdefinition festgelegt. Die Inhaltsbeschreibungen einzelner Dokumente, repräsentiert durch den jeweiligen `tsvector`, werden für einen schnellen Zugriff in einem Suchbaum organisiert, welcher mit der im Abschnitt 2.1.4 beschriebenen GIST-Indexstruktur realisiert ist. Diese Vorgänge sollen am Beispiel der `publication_overview`-Tabelle, wie in Abbildung 4.4 dargestellt, verdeutlicht werden. Die Datenbanktabelle `publication_overview` enthält unter anderem das Feld `title`, welches die Titel aller im System vorhandenen Dokumente beinhaltet. Mit der in Abbildung 4.5 gegebenen Befehlssequenz wird ein weiteres Feld in die Tabelle eingefügt, welches mit dem speziellen Datentyp `tsvector` definiert ist. Es dient der Speicherung von Wortlisten, die durch die `to_tsvector`() erzeugt werden.

```
ALTER TABLE publication_overview ADD COLUMN titleidx tsvector;

UPDATE tblMessages SET titleidx=to_tsvector('default', title);

CREATE INDEX idxFTI_idx ON publication_overview USING gist(titleidx);
       VACUUM FULL ANALYZE;
```

Abbildung 4.4: Einfügen der Tsearch2-Funktionalität

Nach dem Einfügen eines neuen Feldes mit dem Datentyp `tsvector` wird dieses durch die `to_tsvector`- Funktion erzeugten Wortliste aus dem Titelfeld im zugehörigen Datensatz befüllt. Nachdem alle Wortlisten erzeugt sind, erfolgt im dritten Schritt die Generierung eines GIST-Indexes über dieses Feld. Die VACUUM FULL ANALYZE-Anweisung dient lediglich der Aktualisierung der PostgreSQL-Statistiken, welche bei jeder generischen Anfrageoptimierung durch das DBMS konsultiert werden. Dadurch ist sichergestellt, dass

die zukünftigen Anfragen über dieses Feld auch unter Einbeziehung der erzeugten Indizes ausgewertet werden.
Im Gegensatz zu den in der Standarddistribution bekannten Indexstrukturen werden die GIST-Indexe bei Änderungsoperationen nicht vollautomatisch vom DBMS gepflegt, wie es zum Beispiel bei einem Index über ein `integer` Feld der Fall wäre. Die Aktualisierungen können durch die vorhandene `tsearch2` Funktion erzwungen werden. Um diesen Vorgang bei allen Datenänderungen der betroffenen Tabelle zu garantieren, ist der Einsatz von Triggern notwendig. Trigger können als Wächterfunktionen in Datenbanksystemen verstanden werden, welche je nach Definition bestimmte Vorgänge erkennen und entsprechende weitere Reaktionen auslösen, etwa die Ausführung vordefinierter Datenbankfunktionen. Die Kopplung eines Triggers mit der auszulösenden Funktion ist in 4.5 gegeben.

```
CREATE TRIGGER tsvectorupdate BEFORE UPDATE OR INSERT ON publication_overview
          FOR EACH ROW EXECUTE PROCEDURE tsearch2(titleidx, title);
```

Abbildung 4.5: Aktualisierung der GIST Indexe

4.3.2 Transformation von Suchanfragen

Neben der Indexierungsfunktionalität bietet das Tsearch-Modul Operatoren, welche die Konstruktion von Anfragen für das Boolesche Retrieval über die indexierten Dokumente ermöglichen. Dabei ist wichtig zu erwähnen, dass die in den Anfragen selbst enthaltenen Begriffe auf die gleiche Weise vorverarbeitet werden wie die Dokumente selbst. Dieses Vorgehen ist daher essentiell, da Tsearch2 lediglich ein exaktes Übereinstimmen feststellen kann. Unterschiedliche Wortformen könnten auf diese Weise nicht entdeckt werden. Die Anwendung identischer Regeln auf die Suchterme garantiert eine korrekte und vollständige Antwortmenge. Die Konversion der Suchterme erfolgt durch die Funktion `to_tsquery`. Eine Beispielanfrage mittels der `to_tsquery` Funktion mit der erzeugten Ergebnismenge ist in Abbildung 4.6 illustriert.

```
SELECT pub_id,title FROM publication_overview
  WHERE titleidx @@ to_tsquery('default', 'Data');

pub_id |                                    title
--------+-------------------------------------------------------------------------
   1752 | Scalable Visual Data Exploration of Large Data Sets via MultiResolution
   1754 | Data Mining und Data Warehousing
   1755 | HD-Eye: Visual Clustering of High--Dimensional Data
   1757 | Guest Editorial. Data Mining and Knowledge Discovery
```

Abbildung 4.6: Beispiel einer Datenbankabfrage mittels der *to_tsquery* Funktion

4.3.3 Implementierung der Spracherkennung

Für die korrekten Ergebnisse der beschriebenen Indexierungsfunktionalitäten ist das Wissen über die Sprache, in welcher der zu indizierende Text vorliegt, unabdingbar. In der Anforderungsanalyse wurde auf die Beschränkung der Publikationssprachen auf Englisch und Deutsch hingewiesen. Es ist deutlich, dass sowohl für die Stoppworteliminierung als auch für die Stammformreduktion sprachspezifische Algorithmen herangezogen werden müssen. Die umfangreiche Funktionalität des Tsearch2 API kann ebenso für eine einfache Spracherkennung benutzt werden. Dabei ist die Tatsache von Nutzen, dass nach der Umwandlung der vorliegenden Texte die Wortliste, welche im `tsvector` abgelegt ist, nur eine Teilmenge der Wortmenge des ursprünglichen Textes ist. Da die Stoppwortlisten unterschiedlicher Sprachen weitgehend disjunkt sind, ferner zu erwarten ist, dass in einem Text bestimmter Länge eine gewisse Anzahl von Stoppwörtern enthalten ist, kann der Umfang der Reduktion als Indikator für die Spracherkennung genutzt werden. Wird zum Beispiel eine deutsche Stoppwortliste auf einen englischen Text angewandt, so ist zu erwarten, dass die Anzahl der erkannten Stoppwörter gegen Null konvergiert[2]. Die vorliegende Anwendung realisiert die Sparcherkennung mittels der Tsearch2-Funktion für den Längenvergleich von `tsvector`-Einträgen `tsvector_gt`. Dabei werden alle zu indexierenden Texte sowohl in einen `tsvector` unter Anwendung der deutschsprachigen Stoppwortliste, als auch in einen `tsvector` mit englischsprachiger Anlyse konvertiert. Der anschliessende Längenvergleich liefert die kürzere, also stärker komprimierte Wortliste als Indikator für die vorliegende Sprache.

4.4 Aufbau der Benutzeroberfläche

Die Benutzeroberfläche der Anwendung ist logisch in zwei Bereiche aufgeteilt, den öffentlich zugänglichen Bereich und den passwortgeschützten Mitgliederbereich. Die im öffentlichen Bereich zur Verfügung stehenden Funktionen sind die Übersicht aller im System verfügbaren Publikationen sortiert nach Titel oder Jahr, die Suchoberfläche fur die Recherche im Publikationsbestand sowie die Möglichkeit der Erzeugung eines Benutzerkontos.
Der geschütze Mitgliederbereich ist anforderungskonform weiter in den Bereich der einzelnen Mitglieder und den der Publikationsverwaltung von Lehrstühlen unterteilt. Der Funktionsumfang dieser beiden Bereiche bildet die geforderte Hierarchiestruktur der Beziehung zwischen den Lehrstühlen und den zugehörigen wissenschaftlichen Lehrstuhlmitarbeitern ab. Demzufolge operieren die Prozesse, welche aus dem Bereich der Lehrstuhlverwaltung ausgelöst werden, global auf dem Datenbestand aller Benutzerkonten der zugehörigen Lehrstuhlmitglieder. Analog dazu haben die einzelnen Mitglieder nur Zugriff auf Publikationen, welche auch ihnen zugeordnet sind. Durch die schon im Da-

[2] In bestimmten Fällen können Fachwörter wie *Computer* oder *System* ebenso Bestandteil der Stoppwortlisten sein. Diese Internationalismen sind in beiden Sprachen vertreten, werden daher eventuell als Stoppwörter erkannt und eliminiert

tenbankmodell modellierte Beziehung zwischen den Lehrstühlen und ihren zugehörigen wissenschaftlichen Mitarbeitern sind die Effekte der Benutzeraktionen in beiden Benutzerschnittstellen sichtbar. So sind Änderungen des Erscheinungsjahres einer Publikation beispielsweise auch in der Benuzterschnittstelle der betroffenen Mitglieder, den Koautoren dieser Publikation, sofort sichtbar. Der verfügbare Funktionsumfang der beiden Bereiche ist in den Abbildungen 4.8 und 4.7 veranschaulicht.

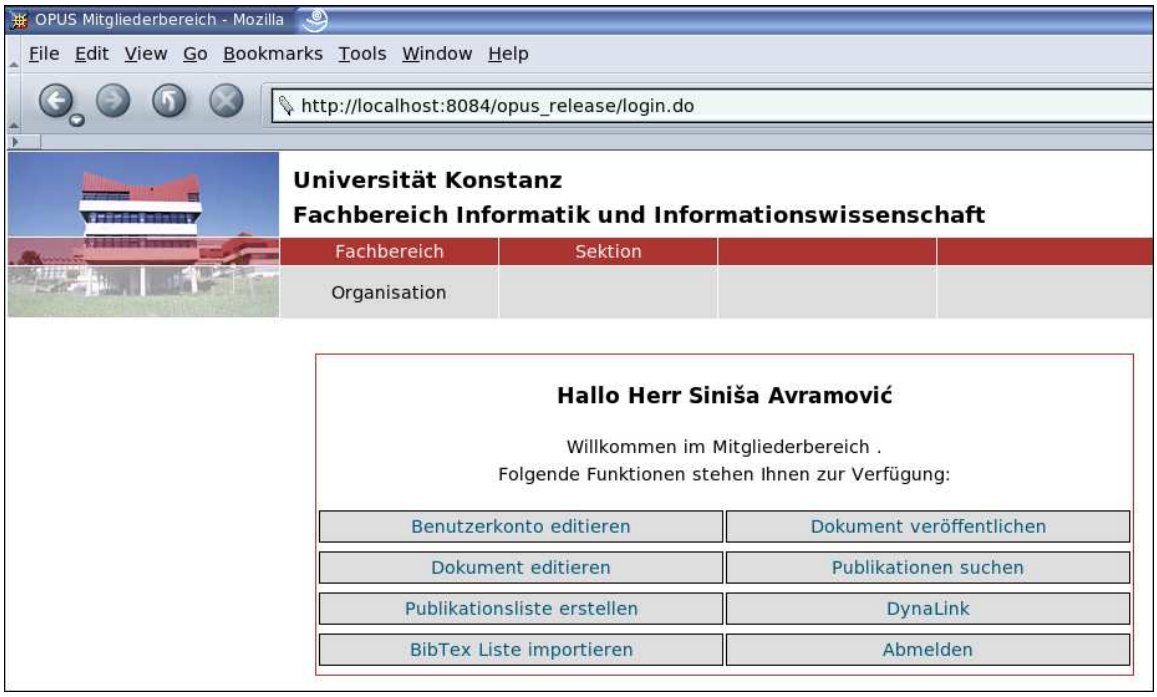


Abbildung 4.7: Funktionsumfang Benutzerbereich

4.5 Erzeugen von Benutzerkonten

Die volle Anwendungsfunktionalität steht ausschliesslich registrierten Benutzern zur Verfügung. Dabei wird, wie im vorrigen Abschnit erläutert, zwischen den Zugängen der einzelnen Lehrstuhlmitglieder und denen für die Publikationsverwaltung von Lehrstühlen unterschieden. Die Anwendung ist so konzipert, daß pro Lehrstuhl nur ein einziger Lehrstuhlzugang existiert. Nachdem ein Lehrstuhlzugang erfolgreich registriert ist, werden weitere Versuche, einen Zugang für denselben Lehrstuhl zu generieren, unterbunden.
Die Generierung von Benutzerkonten beinhaltet neben der reinen Erfassung der persönlichen Zugangsdaten und dem Speichern dieser für weitere Authentifizierungen in der Datenbank auch weitere Funktionalitäten. So geschieht schon bei der Anmeldung eines Benutzers die Zuordnung von Publikationen, zu denen der Benutzer in Koautorenbeziehung steht und die bereits im System vorhanden sind. Diese Zurodnung ist in den

Abbildung 4.8: Funktionsumfang Lehrstuhl-Verwaltungsbereich

Datenbankrelationen `author` und `written_by` gekapselt. Im Laufe des Vorgangs der Publikationsveröffentlichung, welcher im Abschnitt 4.6 detaillierter beschrieben ist, werden stets auch die Relationen aktualisiert. Sollten Publikation zur neu angemeldeten Person im System bereits vorhanden sein, so existiert ein entsprechender Eintrag in der Autorenrelation. Mithilfe einer Verknüpfung der Autorentabelle mit der `written_by`-Relation ist die Gesamtmenge der Dokumente, zu denen die gerade angemeldete Person in Koautorenbeziehung steht, leicht auszumachen. Alle so zugeordneten Publikationen werden zur Generierung von Publikationslisten zu dem neu angemeldeten Benutzer herangezogen, so dass diese sofort nach der Anmeldung auch angezeigt oder gar dynamisch über eine geeignete Schnittstelle in die eigene Webpräsenz eingebunden werden können[3].

4.6 Veröffentlichen von Dokumenten

Eine der zentralen Aufgaben der Publikationsverwaltung besteht in der Bereitstellung von Funktionalitäten für das Einbringen von Dokumenten in das Verwaltungssystem. In Anlehnung an die Anforderungen aus dem Motivationskapitel und dem der Anforderungsanalyse bietet die hier vorgestellte Anwendung eine Auswahl von Möglichkeiten zur Bewältigung dieser Aufgabe. Neben dem klassischen formulargestützten Einbringen von Dokumenten besteht weiter die Möglichkeit der Übergabe vorgefertigter BibTex-Einträge

[3]Diese Funktionalität, in dieser Anwendung als DynaLink bezeichnet, wird im Abschnitt 4.10.2 näher erläutert werden.

sowie die des Imports ganzer Listen solcher Einträge. Die letzten beiden Möglichkeiten stellen eine deutliche Vereinfachung des digitalen Veröffentlichungsprozesses dar. Im Nachfolgenden sollen diese einzelnen Funktionen näher erläutert werden.

4.6.1 Einbringen von Dokumenten mittels der Formularfunktion

Der Vorgang der Veröffentlichung von Dokumenten mithilfe der Formularfunktion ist gemessen an der Anzahl nötiger Interaktionen die umständlichste Art, zu veröffentlichende Dokumente in das Publikationssystem einzubringen. Dieser Vorgang besteht aus mehreren Schritten. Im ersten Schritt erfolgt die Auswahl der zutreffenden Publikationsart. Dabei besteht die Auswahlmöglichkeit aus den am häufigsten anzutreffenden Publikationsformen: Artikel, Buch, Inbuch oder Aufsatz einer Konferenz, sowie den verschiedenen Arten von Abschlussarbeiten. Ist das zu veröffentlichende Dokument durch keine dieser Publikationsarten zu beschreiben, so besteht die Möglichkeit, dieses in der Rubrik *Misscellanoeus* einzubringen. Die Auswahl der Publikationsart definiert die Wahl des Java-Objekts, welches als Datencontainer für Benutzereingaben dient; es legt somit auch die Anzahl und Umfang der nachfolgenden Formularseiten fest. In diesem ersten Schrit erfolgt auch die Festlegung der Anzahl der Autoren, welche an der vorliegenden Publikation mitgewirkt haben. Auch hier ist die Definition von Java-Containerobjekten implizit festgelegt. Wurde eine der Publikationsarten aus dem Bereich der Abschlussarbeiten gewählt, so ist die Festlegung der Autorenanzahl überflüssig, da hier nur jeweils ein Autor zulässig ist. Eventuelle Fehleingaben der Benutzer werden an dieser Stelle unterbunden.
Im zweiten Schritt erfolgt der Aufbau von Formularseiten zur Erfassung benötigter Daten zur gewählten Publikation wie es etwa Titel und Namen der Autoren oder das Erscheinungsjahr sind. Dabei richten sich Aufbau und Inhalt der Formulare zum einen nach der Anzahl der Autoren, zum anderen nach der BibTex-Spezifikation für die Beschreibung der gewählten Publikationsart. Die vorhandenen Pflichtfelder spiegeln im Wesentlichen auch die in den Datenbankrelationen unbedingt benötigten Felder, welche aus ihrer Definition heraus nicht unbeschrieben bleiben dürfen, wieder. Abbildung 4.9 illustriert einen Ausschnit einer solchen Formularseite.

Eine sehr wichtige Information bei der Erfassung der Publikationsdaten ist die Zusammenstellung der Schlagwörter, welche zur Inhaltbeschreibung von den Publizierenden eingesetzt werden. Hierbei besteht die Möglichkeit, die Schlagwörter frei nach uneingeschränkter persönlicher Auswahl zu vergeben oder ein kontrolliertes Vokabular als Hilfestellung einzubeziehen. Die Anwendung bietet darüber hinaus die Möglichkeit, die Schlagwortnormdatei des OSWD über eine Schnittstelle in den Deskriptionsprozess einzubinden. Bei der Schlagwortnormdatei, im Nachfolgenden als SWD abgekürzt, handelt es sich um einen terminologisch kontrollierten, normierten Wortschatz. Dieser wird von der Deutschen Nationalbibliothek in Kooperation mit den einzelnen Bibliotheksverbänden auf Landesebene täglich aktualisert und gepflegt. Über eine vom

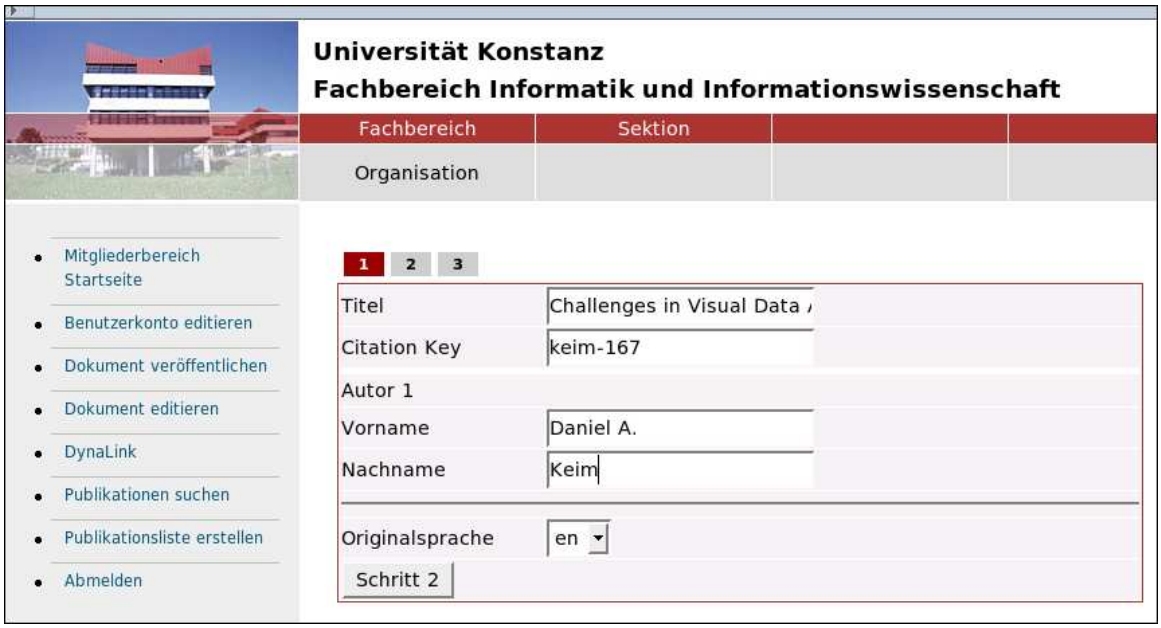


Abbildung 4.9: Beispiel einer Formularseite

Bibliotheksservice-Zentrum Baden-Württemberg zur Verfügung gestellte Webschnittstelle [OSW] ist es möglich, aus dem OSWD-Schlagwortvokabular die zutreffenden Deskriptoren auszuwählen und für die Verschlagwortung einzusetzen. Bei der Auswahl dieser Option werden alle bisher zur Publikation erfassten Daten sowie weitere sitzungsrelevante Attribute im HTTP-Request eingebettet und die Anwendung über eine Weiterleitung an den OSWD-Server kurzzeitig verlassen. Dort werden in Interaktion mit dem Benutzer die zutreffenden Deskriptoren festgelegt und ebenfalls im HTTP-Aufruf gespeichert. Nach Abschluss des Auswahlprozesses erfolgt die Rückkehr in die ursprüngliche Webanwendung. Da im HTTP-Response-Objekt alle sitzungsrelevanten Daten gekapselt sind, ist auch im Mehrbenutzerbetrieb eine Überschneidung ausgeschlossen.
Im nächsten Schritt erfolgt die Übergabe der Dokumentdatei über einen File-Upload-Dialog an das System. Aufgrund der grossen Anzahl der zu unterstützenden Dateiformate sind die zu erwartenden Komplexität und Aufwand der Transformation der Dateien in ein für das Tsearch2 lesbares Format unverhältnismäßig hoch. Daher werden die Dokumente selbst nicht zur Inhaltserschließung herangezogen. Diese werden als Volltexte auf dem Fileserver abgelegt. Hierbei sei angemerkt, dass es sich bei dem erwähnten Fileserver lediglich um einen vom Tomcat Server über eine *Datasource* Einstellung verwalteten Speicherbereich handelt. Die auf diese Art verwalteten Dokumente stehen den Interessenten somit zum Download zur Verfügung. Die Indexierung erfolgt in den einschlägigen Feldern *Titel* und *Abstrakt*. Dabei werden die Formularinhalte und die Funktionen des bereits beschriebenen Tsearch2-Moduls zur Extraktion von Deskriptoren und ihrer entsprechenden Speicherung in Datenbankrelationen an die Indexierungsfunktionalität übergeben. Den so erhaltenen Schlagwörtern werden diejenigen hinzugefügt, welche als

freie Schlagwörter und / oder aus dem kontrollierten Vokabular der SWD zur Beschreibung der Publikation durch den Benutzer vergeben wurden. Bevor die erfassten Daten in die Publikationsdatenbank geschrieben werden, erfolgt an dieser Stelle die Überprüfung der korrekten Zuordnung der Publikation zu den einzelnen Autoren. Zum Einen erfolgt die Überprüfung auf den unwahrscheinlichen Fall, dass Mitglieder im System mit identischen Vor-, Mittel- und Nachnamen angemeldet sind. In solchen Fällen werden die betroffenen Benutzer mit weiteren Attributen wie Lehrstuhlzugehörigkeit und Büroraum zur Auswahl angeboten. Zum anderen wird eine entsprechende Überprüfung der Existenz von Benutzerkonten aller zur Publikation in Koautorenbeziehung aufgeführter Verfasser veranlasst. Für diejenigen Autoren, die erstmals als Autor aufgeführt werden, erfolgt ein Neueintrag in die Autorenrelation. Bei diesem Eintrag wird zeitgleich aus dem vollen Namen eine Initialensequenz erzeugt und ebenfalls in dieser Relation abgelegt. Dieses Vorgehen ist für die fehlerfreie Zuordnung von Publikationen welche über den Import von BibTex-Einträgen in das System eingebracht werden, notwendig. Nach der BibTex-Spezifikation werden die Namen teilweise als Initialien aufgeführt. Bei Autoren welche bereits als Koautoren im System bekannt sind und zu deren Person ein entsprechender Eintrag in der Autorenrelation existiert, ist die eindeutige Zuordnung durch das Datenbankmodell und die Relation `written_by` gegeben. Die letzten zwei Überprüfungen sind notwendig, um im Fall einer Neuregistrierung alle bereits im System vorhandenen Publikationen der neu registrierten Person zuzuordnen. Somit stehen sofort nach Anmeldung ohne weitere Interaktionen Publikationslisten für den Export zur Verfügung.
Nach abgeschlossener Überprüfung und Aufbereitung der Daten folgt das Schreiben aller erfassten Informationen in die Datenbank. Dazu werden die konstruierten Objekte and die `write`-Methode der Klasse `DbRw` übergeben. Die Auswahl der entsprechenden Schreibmethode ist von der ausgewählten Publikationsart abhängig. Da die gesammelten Daten Veränderungen in mehreren Relationen herbeiführen, welche wie im Abschnitt 2.1 beschrieben in einer Beziehung der referentiellen Integrität zueinander stehen, hat aus Integritätsgrunden der gesamte Schreibvorgang innerhalb einer Transaktion zu erfolgen. Dazu werden die einzelnen Einfüge- und Aktualisierungsbefehle konstruiert und in eine Art Liste gespeichert. Um die Überprüfung der Integritätsbedingungen nach jedem einzelnen Befehl zu unterbinden und die Möglichkeit zu behalten, bei Fehlern die gesamte Transaktion unwirksam zu machen, schaltet die Schreibmethode, nachdem eine Datenbankverbindung aus dem Verbindungspool erhalten ist, zunächst den `autocommit`-Modus aus. Nun können die einzeln konstruierten Befehle in der Datenbank ausgeführt und erst im letzten Schrit der Transaktionsabschluß mit einem `commit` Befehl herbeigeführt werden. Auf eventuelle Probleme oder Fehlermeldungen kann mit einer Zurücksetzung der Transaktion reagiert werden.

4.6.2 Publikationsveröffentlichung mithilfe von Bibtex-Einträgen

Das Einbringen von Dokumenten mittels der Formularfunktion erfordert eine Vielzahl von Eingaben durch den Benutzer. Eine deutliche Erleichterung stellt die Möglichkeit

dar, vorgefertigte BibTex-Einträge an das Publikationssystem zu übergeben, welches dann die nötigen Informationen zu der jeweiligen Publikation aus ihnen extrahiert. Der Eingabeprozess reduziert sich dabei auf Validierung und eventuelle Korrekturen vorausgefüllter Formularfelder. Standardfelder wie Titel, Abstrakt, Veröffentlichungsjahr und Autoren sind nach der BibTex-Spezifikation ohnehin in jedem BibTex-Eintrag vorhanden. Abbildung 4.10 zeigt ein Beispiel eines BibTex-Eintrags.

```
@ARTICLE{keim-165,
AUTHOR = "B. Bustos, D. Keim, D. Saupe, T. Schreck, D. Vranic",
TITLE = "An Experimental Effectiveness Comparison of Methods for 3D Similarity Search",
PUBLISHER = {Springer},
JOURNAL = "International Journal on Digital Libraries, Special",
VOLUME = {6},
NUMBER = {1},
PAGES = {39-54},
URL = {infovis.uni-konstanz.de/members/schreck/tsprojects/papers/ijdl05.pdf}
YEAR = {2006}}
```

Abbildung 4.10: BibTex-Eintrag

Das optionale URL-Feld kann in einem BibTex-Eintrag dazu verwendet werden, bereits in digitalisierter Form vorhandene Quellen zu referenzieren, ohne diese erneut über den File-Upload-Dialog dem System übergeben zu müssen.
Die Verarbeitung von BibTex Einträgen erfolgt innerhalb der Anwendung in folgenden Schritten :

- Einlesen der BibTex-Einträge
- Validierung
- Syntaktische Korrekturen
- Extraktion von Informationen mittels externer Module
- Aufbereitung erhaltener Informationen

Die BibTex-Einträge können an das System über ein vordefiniertes Feld des Publikationsformulars mittels Einfügen übergeben werden. Die auf diese Weise transferierten Einträge werden zunächst einer syntaktischen Validierung unterzogen. Für diese Aufgabe werden bestehende, frei verfügbare Module herangezogen. Die Programmiersprache Java bietet für solche Fälle die Option, externe Module mit Übergabeparametern auszuführen und die so gewonnenen Ergebnisse weiter in den Programmablauf zu integrieren. Die Validierung der BibTex-Einträge dieser Anwendung basiert auf dem Modul `bibparse` [bibb]. Ein als syntaktisch für korrekt befundener Eintrag kann dennoch weitere Komplikationen in der Verarbeitung hervorrufen. Auch wenn die Struktur der BibTex-Einträge spezifiziert ist, erfordern zulässige Abweichungen in der Syntax eine entsprechende Reaktion. Beispielsweise können die Namen der Autoren mit dem englischen *and* verknüpft oder

durch einfache Kommata getrennt sein. Weitere Schwierigkeiten bereitet der Umgang mit Umlauten. Nicht entsprechend maskierte Umlaute führen zu einer syntaktischen Invalidität. Eine robuste Applikation muss jedoch in der Lage sein, auf alle genannten Probleme entsprechend zu reagieren. Um die Komplikationen bei der Weiterverarbeitung von BibTex-Einträgen zu minimieren, werden diese vorher zusätzlich durch das Modul `bibclean` [bibc] behandelt. Hierbei handelt es sich um eine Kombination von Funktionalitäten zur Normalisierung von BibTex-Einträgen und einer erweiterten Validierung, Dabei werden die oben genannten Probleme weitgehend beseitigt.
Die bereinigten Einträge können im nächsten Schritt in ein für die weitere Verarbeitung geeignetes Format transformiert werden. Aufgrund vielfältiger Manipulationsmöglichkeiten in der Programmiersprache Java wurde die Transformation in XML-Dateien ausgewählt. Dazu wird zunächst der übergebene BibTex-Eintrag unter Verwendung des extrenen Moduls `bib2xml` [bibd] in eine XML-Datei konvertiert und in eine temporäre Datei gespeichert. Die darin zu einer Publikation enthaltenen Informationen können somit mittels eines XML-Parsers ausgelesen und für die weitere Verarbeitung in ein geeignetes Java-Objekt abgelegt werden. Dieses Java-Objekt wird im weiteren Verlauf zur Konstruktion vorausgefüllter Formularfelder der Publikationsoberfläche eingesetzt. Innerhalb dieser Aufbereitungsphase erfolgt zusätzlich eine Prüfung auf Duplikate. Publikationen, welche schon im System bekannt sind, werden aufgrund dieser Prüfung abgelehnt.
Der abschließende Schritt, das Schreiben der Daten in die Datenbank, lehnt sich an die im vorangegangenen Abschnit beschriebene Funktionalität an. Nach Bestätigung über den Erfolg der einzelnen Abläufe schließt das Löschen temporär gespeicherter Dateien den Publikationsvorgang ab.

4.7 Import von Publikationslisten

Trotz der Möglichkeit, einzelne BibTex-Einträge an das System zu übergeben, würde das Einbringen einer Serie von Publikationen dennoch eine Reihe sich wiederholender Benutzerinteraktionen erfordern. So müsste für jede Publikation ein entsprechender BibTex-Eintrag gesondert in das dazu vorgesehene Formularfeld eingegeben werden. Um den Vorgang des Einbringens mehrerer Publikationen für den Benutzer effizienter zu gestalten, wurde die Funktionalität für die Verarbeitung von Publikationslisten implementiert. Voraussetzung für die Nutzung dieser Funktion ist das Vorliegen einer Publikationsliste als Aneinanderreihung einzelner BibTex-Einträge. Dazu ist es ausreichend, die Einträge der zu übergebenden Publikationen in beliebiger Reihenfolge in eine `.bib`-Datei zusammenzufügen. Die so zusammengesetzte Datei kann in einem File-Upload-Dialog an das Publikationssystem übergeben werden. Diese wird für die Weiterverarbeitung zunächst temporär in einer Datei abgelegt. Durch sequentielles Einlesen der Datei erfolgt die Zerlegung in einzelne BibTex-Einträge. Die so gewonnenen Einzeleinträge durchlaufen, wie im vorangegangenen Abschnitt beschrieben, zunächst die Schritte der syntaktischen Überprüfung und Bereinigung. Einträge, welche in diesem Schritt irreparable Fehler aufweisen, werden von der weiteren Verarbeitung ausgeschlossen und mit dem entsprechenden Feh-

lerreport in eine separat mitgeführte Fehlerliste eingefügt. Da redundante Publikationen schon im Datenbankmodell ausgeschloßen sind, werden Publikationseinträge welche die Validierung bestanden haben einem Abgleich mit bereits im System veröffentlichten Publikationen unterzogen. Im System existierende Publikationen werden in eine weitere für diesen Zweck mitgeführte Liste abgelegt und von der weiteren Verarbeitung ausgeschlossen.
Nachdem die Validierung und Bereinigung einzelner BibTex-Einträge abgeschlossen ist, die Transfomation in eine XML-Datei erfolgreich war und die darin enthaltenen Informationen über die Publikation von der Parser-Klasse herausgelesen und in ein Java Objekt abgelegt wurden, wird dieses Objekt an die `WriteDispatcher` Klasse des `common_utils` Pakets übergeben. Diese Klasse überprüft zunächst, ob es sich bei der vorliegenden Veröffentlichung um eine der untestützten Publikationsformen handelt. Werke, welche dieser Anforderung nicht genügen, werden in eine dritte mitgeführte Liste eingefügt, welche die nicht unterstützten Publikationsarten beherbergt. Andernfalls wird nach der Festlegung der Publikationsform die entsprechende `Write-` Methode aufgerufen, welche innerhalb einer Transaktion alle notwendingen Einfüge- und Aktualisierungsoperationen in der Datenbank dürchführt.

4.7.1 Präsentation der Import-Ergebnisse

Nachdem alle Bibtex-Einträge verarbeitet wurden liefert die Applikation eine Übersicht der Verarbeitungsergebnisse, welche mithilfe der im Verarbeitungsprozess mitgeführten Listen konstruiert wurde. Diese Übersicht liefert Aufschluß über :

- die erfolgreich importierten Publikationen,
- Publikationen, welche bereits im System existieren, somit ihr Import abgelehnt wurde,
- Einträge, welche aufgrund von Fehlern nicht verarbeitet werden konnten,
- Publikationen, welche einer von dem System nicht unterstützten Publikationsart angehören.

Fehlerhafte BibTex-Einträge können anhand des hier gebotenen Fehlerlogs von den Benutzern ausfindig gemacht und korrigiert werden, um anschließend mit einem erneuten Import im System veröffentlicht zu werden. Für die von der Importfunktion nicht unterstützten Publikationsformen besteht die Möglichkeit, diese unter der Rubrik *Sonstige Publikationsarten* in der Formularfunktion zu veröffentlichen. Ein Beipiel eines Importergebnisses ist in Abbildung 4.11 gegeben.

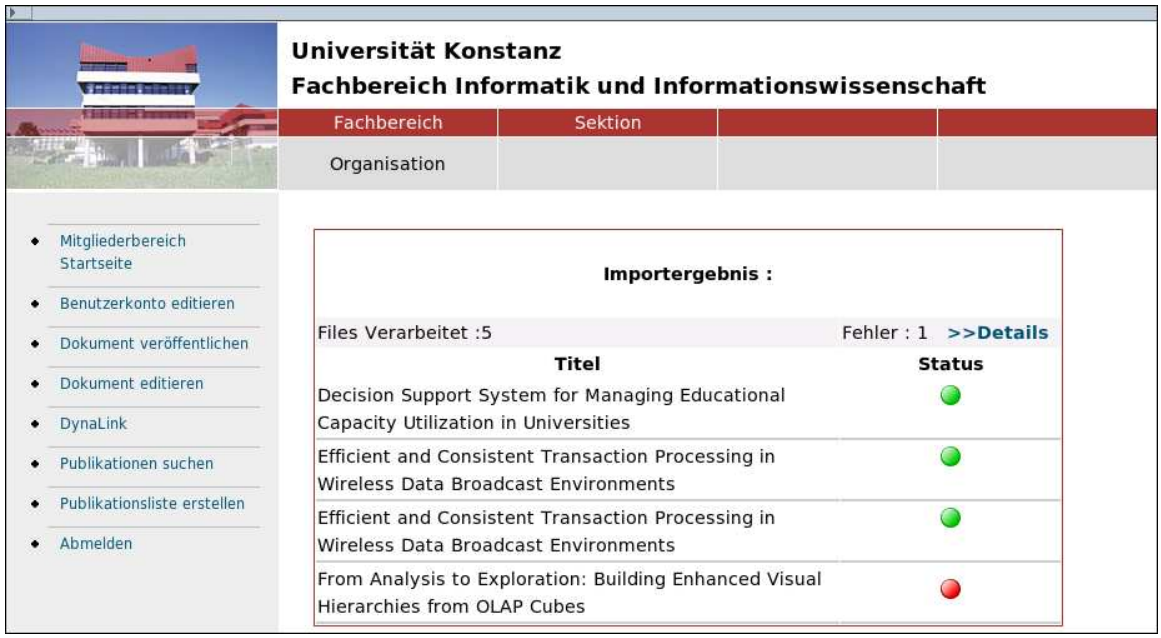


Abbildung 4.11: Präsentation der Import-Ergebnisse

4.8 Suchfunktion

Die Möglichkeit, den Publikationsbestand eines Publikationsverwaltungsystems nach bestimmten Fachbegriffen zu durchsuchen und diejenigen Dokumente zu extrahieren, welche die gesuchten Terme beinhalten, zählt zu den essentiellen Funktionen eines jeden Publikationsverwaltungssystems. Die wesentlichen Grundanforderungen an eine solche Funktionalität sind die Vollständigkeit und die Korrektheit der Suchergebnisse, ferner eine intuitive Suchoberfläche gefolgt von einer übersichtlichen Präsentation der Suchergebnisse. In diesem Abschnitt soll die Implementierung der Suchfunktion in Anlehnung an die genannten Kernpunkte beschrieben werden.

4.8.1 Die Suchoberfläche

Die Suchoberfläche der vorliegenden Anwendung als Schnittstelle für die Benutzerinteraktion im Suchprozess zusammen mit den dort verfügbaren Suchoptionen, ist in Abbildung 4.12 dargestellt. Die Implementierung der Suchfunktion orientiert sich an dem in Kapitel 2 beschriebenen Booleschen Modell. Dabei können die Suchanfragen nach zutreffenden Publikationen aus den Feldern Titel, Autor und Erscheinungsjahr sowie mehreren Schlüsselwörtern konstruiert werden. In Anlehnung an das Boolesche Modell können die einzelnen Suchbegriffe mit den logischen Operatoren `AND` und `OR` verknüpft werden. Dabei erlauben die Schlüsselwortfelder für die Suche nach koplexeren Begriffen Mehrfacheinträge, (zum Beispiel Data Warehouse). In solchen Fällen sind die einzelnen Suchbegriffe mit einem logischen „UND “miteinander verknüpft. Das Titelfeld erlaubt

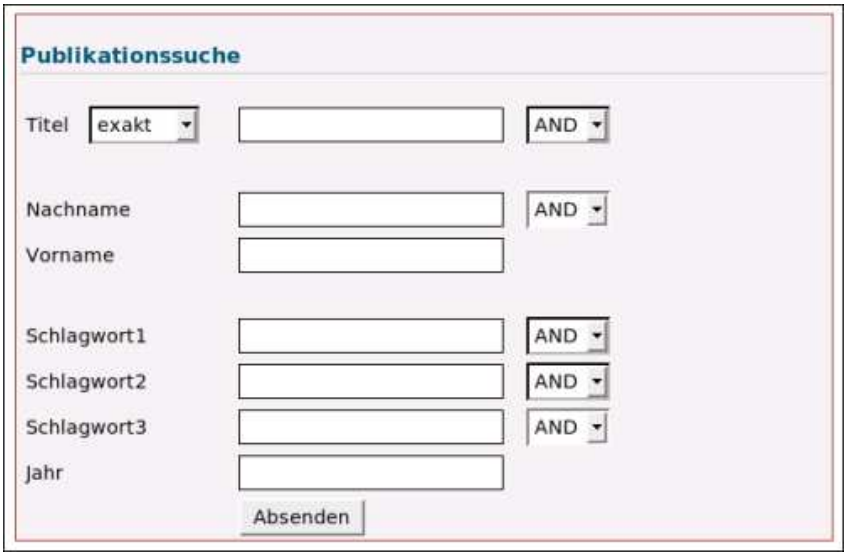


Abbildung 4.12: Die Suchoberfläche

eine Phrasensuche, wobei hier die Auswahl zwischen einem exakten Wortlaut (exakter Titel) oder einer „enthalten in“- Relation besteht.

4.8.2 Konstruktion der Suchanfragen

Eine fehlerfreie Konstruktion der Suchanfragen ist maßgebend für die Generierung einer vollständigen und korrekten Ergebnismenge. Die Anfragekonstruktion dieser Anwendung verläuft in zwei Schritten. Der erste Schrit dient der Erkennung tatsächlich mit Benutzereingaben versehenen Suchfelder. Diese werden im zweiten Schrit dynamisch zu einer SQL Anfrage zusammengesetzt. Der Vorgang besteht darin, zunächst für jedes mit Benutzereingaben versehenes Feld eine Suchanfrage zu generieren, um diese dann im letzen Schritt in eine komplexe Suchanfrage zu transformieren. Die Ergebnisse der Teilanfragen können dabei als Teilmengen verstanden werden, welche mittels Mengenoperatoren in eine komplexe Ergebnismenge überführt werden können. Der logische Operator `AND` als Verküpfung der einzelner Suchfelder wird dabei als Schnittmenge mit dem `INTERSECT`-Operator realisiert, da die Dokumente, welche Suchbegriff 1 und Suchbegriff 2 enthalten, in der Schnittmenge der einzelnen Teilergebnisse liegen müssen. Analog zu dieser Beobachtung erfolgt die Übersetzung der `OR` Verknüpfung in den Vereinigungsoperator `UNION`. Bei der Konstruktion von Teilanfragen wird die Direktive „enthalten in“aus dem Suchfeld Titel mit Hilfe des `ILIKE` -Operators der Anfragesprache SQL realisiert. Somit können beliebige Zeichenketten innerhalb von Texteinträgen ungeachtet der Groß−/Kleinschreibung erkannt werden. Die exakte Titelsuche sowie die Autorensuche und die Suche nach dem Erscheinungsjahr ist mittels des Gleichheitsoperators implementiert.
Die Konstruktion der Suchanfragen, welche als Suchkriterium Schlüsselwörter beinhal-

ten, lehnt sich im Wesentlichen an die Funktionalitäten des integrierten Tsearch2-Moduls an. Dabei werden die gesuchten Schlüsselwörter mittels der Funktion `to_tsquery` den gleichen Transformationen unterzogen, wie es während der Indexierung von Texten der Fall ist. Da die Suche nach Schlüsselwörtern mittels der Tsearch2 Funktionen lediglich auf dem exakten Matching basieren, kann nur auf diese Weise eine korrekte Treffermenge sichergestellt werden. Im Gegensatz zum Indexierungsvorgang, bei welchem die Indexterme u.a. aus dem Titel- und Abstraktfeld gewonnen werden, stehen bei der Anfragenkonstruktion in der Regel weniger Begriffe zur Verfügung. Diese Tatsache führt zu dem Problem der Spracherkennung für die Definition der Transformationsregeln der Suchterme. Aufgrund weniger vorliegender Suchterme ist eine eindeutige Spracherkennung wie sie im Abschnit über das Indexieren der Dokumente beschrieben wurde, nur selten möglich. Um dennoch eine vollständige und korrekte Ergebnismenge zu erhalten, besteht hierbei der Ansatz darin, die vom Benutzer übergebenen Terme sowohl nach den englischen, als auch nach den Transformationsregeln für die deutsche Sprache zu verarbeiten. Dabei entstehen zwei Anfragen, welche mit dem `UNION`-Operator vereinigt werden. Somit werden alle ermittelten Treffer in die Ergebnismenge übernommen. Eine Beispielanfrage als Resultat der beschriebenen Transformationsschritte ist in Abbildung 4.13 gegeben.

```
SELECT distinct pub_id,title,year,www_path,authors,abstract FROM publication_overview
        WHERE title ILIKE '%Analysis%'
INTERSECT
SELECT pub_id,title,year,www_path,authors,abstract FROM publication_overview
        WHERE (keyidx @@ to_tsquery('default', 'Data')
                OR  absidx @@ to_tsquery('default', 'Data')
                OR titleidx @@ to_tsquery('default', 'Data'))
UNION
SELECT pub_id,title,year,www_path,authors,abstract FROM publication_overview
        WHERE (keyidx @@ to_tsquery('default_german', 'Data')
                OR  absidx @@ to_tsquery('default_german', 'Data')
                OR titleidx @@ to_tsquery('default_german', 'Data'))
INTERSECT
SELECT pub_id,title,year,www_path,authors,abstract FROM publication_overview
        WHERE (keyidx @@ to_tsquery('default', 'Challenge')
                OR  absidx @@ to_tsquery('default', 'Challenge')
                OR titleidx @@ to_tsquery('default', 'Challenge'))
UNION
SELECT pub_id,title,year,www_path,authors,abstract FROM publication_overview
        WHERE (keyidx @@ to_tsquery('default_german', 'Challenge')
                OR  absidx @@ to_tsquery('default_german', 'Challenge')
                OR titleidx @@ to_tsquery('default_german', 'Challenge'))
INTERSECT
SELECT pub_id,title,year,www_path,authors,abstract FROM publication_overview
        WHERE (keyidx @@ to_tsquery('default', 'Visual')
                OR  absidx @@ to_tsquery('default', 'Visual')
                OR titleidx @@ to_tsquery('default', 'Visual'))
UNION
SELECT pub_id,title,year,www_path,authors,abstract FROM publication_overview
        WHERE (keyidx @@ to_tsquery('default_german', 'Visual')
                OR  absidx @@ to_tsquery('default_german', 'Visual')
                OR titleidx @@ to_tsquery('default_german', 'Visual'));
```

Abbildung 4.13: Beispiel einer Suchanfrage

4.8.3 Präsentation der Suchergebnisse

Der letzte Schrit des Suchprozesses besteht in der Präsentation der Suchergebnisse. Nachdem die konstruierte Suchanfrage auf der Datenbank ausgeführt wurde, steht eine Ergebnismenge zur Verfügung, deren Größe je nach Umfang der vom Benutzer durchgeführten Einschränkungen der Suchattribute und der Anzahl zutreffender Dokumente im Publikationssystem variiert. Eine übersichtliche Präsentation der Suchergebnisse erleichtert die Selektion von Publikationen, welche für den Benutzer von Interesse sind, erheblich. Die Ergebnisaufbereitung der Anwendung ist in Abbildung 4.14 illustriret. Neben der

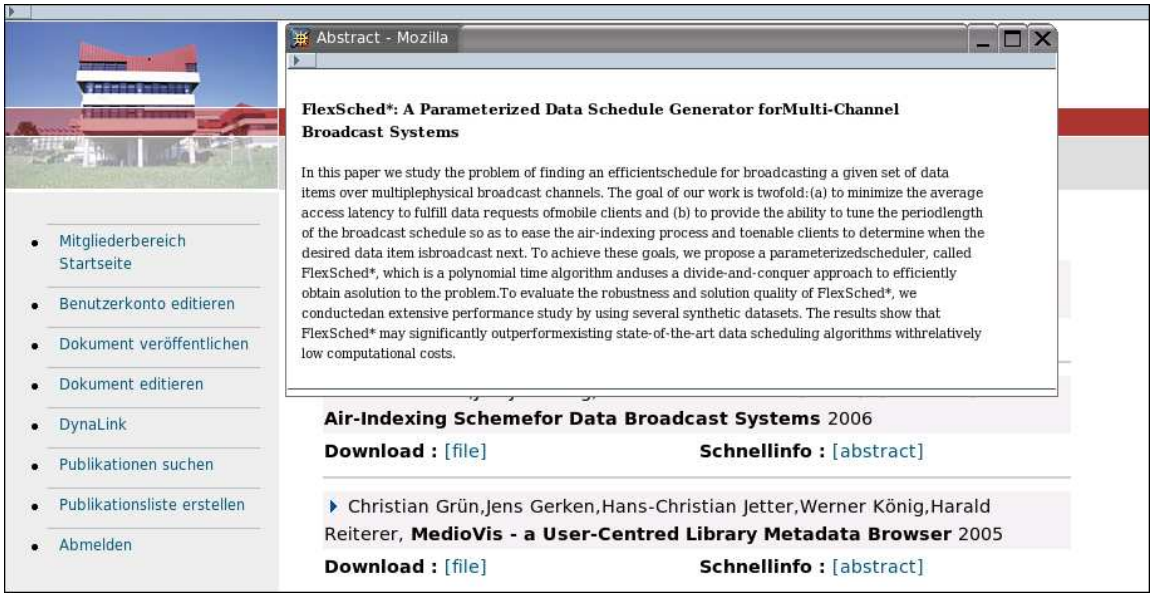


Abbildung 4.14: Präsentation der Suchergebnisse

reinen Auflistung gefundener Einträge bietet die Übersicht, sofern vom Benutzer im Publikationsprozess eingegeben, die Möglichkeit, die Zusammenfassungen der einzelnen Publikationen einzusehen. Diese werden nach Auswahl der entsprechend gekennzeichneten Links in einem separaten Fenster angezeigt. Wurde ferner im Publikationsprozess das Dokument selbst an das System übergeben oder eine Referenz bereits auf anderen Rechnern in digitaler Form verfügbare Dokumente in Form einer validen URL angegeben, so besteht hier die Möglichkeit das Dokument selbst online einzusehen oder für weitere Zwecke auf den eigenen Rechner zu laden.

4.9 Editieren von Einträgen

Trotz sorgfältiger Interaktion bleibt eine bestimmte Wahrscheinlichkeit von Fehleingaben durch den Benutzer während der digitalen Veröffentlichung von Publikationen bestehen. Entgegen den Ansätzen vieler Systeme zu Verwaltung von Publikationen, welche die

Übergabe von Dokumenten an das Verwaltungssystem als final betrachten und anschließend keine Veränderungen erlauben, verfolgt diese Anwendung einen anderen Ansatz. Schon in der Ausrichtungsdisskussion wurde verdeutlicht, dass die vorliegende Anwendung primär der praktischen Verwaltung von Publikationsbeständen durch die Wissensproduzenten dienen soll. Folglich ist auch für den Fall einer Fehleingabe eine nachträgliche Korrektur vorgesehen und erlaubt. Dabei erstrecken sich die Editiermöglichkeiten über alle Publikationsattribute, welche bereits im Publikationsprozess erfasst werden, bis hin zum Austausch der Dokumentdateien. Auch in der Editierfunktion schlägt sich die hierarchische Organisationsstruktur der Relation Lehrstuhl wissenschaftlicher Mitarbeiter nieder. So ist in der Verwaltungsöberfläche der Lehrstühle die Möglichkeit der Korrektur aller dem Lehrstuhl zugeordneter Publikationen verfügbar, während die einzelnen Mitglieder lediglich einen Zugriff auf die von ihnen selbst übergebenen Publikationen erhalten. Beiden Fällen gemeinsam ist jedoch der Korrekturvorgang. Für die Selektion der zu editierenden Veröffentlichung steht zu Beginn eine vereinfachte Suchoberfläche zur Verfügung. Hierbei stehen die Suchoptionen Titel, Publikationsart und Erscheinungsjahr zur Auswahl. Aufgrund der eingeschränkten Parameterauswahl wurde die eigentliche Suchfunktionalität in Form einer Datenbankfunktion gekapselt. Diese besteht in einer Vorauswahl der zu Benutzer / Lehrstuhl zugeordneten Publikationsmenge im ersten und der Anwendung übergebener Suchparameter im zweiten Schritt. Das Resultat ist eine nach den Suchkriterien generierte Publikationsliste, welche zunächst nur die notwendigsten Attribute der Publikation darstellt. Erst nach Auswahl der zu editierenden Publikation erfolgt eine ausführliche Anfrage, welche alle verfügbaren Attribute, die zu einer Publikation im System existieren, aus der Datenbank extrahiert. Die so erhaltenen Informationen über die zu editierende Publikation werden zur weiteren Verarbeitung in ein vordefiniertes Java-Objekt abgelegt. Aus diesem können in den nachfolgenden Schritten vorausgefüllte Formularseiten generiert werden. Benutzer können dort beliebige Einträge korrigieren. Die Benutzerführung orientiert sich dabei im Wesentlichen an der schon aus dem Veröffentlichungsvorgang mittels der Formaluarfunktion bekannten Prozedur. Nach erfolgreichem Schreibvorgang in der Datenbank sind die Änderungen an allen entsprechenden Stellen im System sichtbar. Dies betrifft primär die Publikationslisten der betroffenen Mitglieder und der zugehörigen Lehrstühle sowie die Publikationslisten, welche öffentlich zugänglich sind und stets in Echtzeit aktualisiert werden[4].

4.10 Export von Publikationslisten

Internetauftritte der Lehrstühle und dazugehöriger wissenschaftlicher Mitarbeiter gehören in der heutigen Zeit zum absoluten Standard. Ebenso erwartet wird die Präsentation der Forschungsgebiete und der erzielten Ergebnisse auf solchen Webseiten, nicht

[4]Die primäre Rolle der öffentlich zugänglichen Publikationslisten in der Anwendung selbst liegt in der Aufbereitung der Publikationsbestände für die Indexierung durch Suchmaschinen. Somit werden die bestehenden Publikationen für ein breites Publikum auffindbar

zuletzt auch in Form veröffentlichter wissenschaftlicher Abhandlungen. Sehr häufig ist dabei die manuelle Pflege solcher Internetauftritte, insbesondere der Publikationslisten, anzutreffen. Dabei bietet es sich geradezu an, ein bestehendes System zur Verwaltung von Publikationen auch zu diesem Zweck einzusetzen. Ein aktuelles Repositorium von wissenschaftlichen Veröffentlichungen ist in solch einem System ebenso gegeben wie die Möglichkeit, mit verhältnismäßig geringem Aufwand die bestehenden Publikationslisten in eigene Webpräsenzen einzubinden. Für diesen Fall wurde innerhalb der vorliegenden Anwendung eine Reihe von Funktionen konzipiert und implementiert, welche genau diese Aufgaben bedeutend erleichtern sollen.
Registrierten Benutzern und Lehrstühlen stehen zwei Optionen für die Generierung und Einbindung von Publikationslisten in eigene Webpräsenzen zur Auswahl. Es sind sowohl Listenexporte, welche von der grafischen Oberfläche durch den Benutzer ausgelöst werden können möglich, als auch die automatische Einbindung in die eigene Webpräsenz über eine geeignete Schnittstelle. Ungeachtet der Exportmethode ist die Vorbereitung der Publikationsdaten jedoch beiden Ansätzen gemein. Bei Betrachtung des im Kapitel 3 beschriebenen Datenbankmodells wird deutlich, dass die Kollektion aller Publikationen zu einem bestimmten Benutzer oder Lehrstuhl mit einem nicht unerheblichen Berechnungsaufwand verbunden ist. Zudem ist eine entsprechende Formatierung der präsentierten Daten erforderlich. Bei Publikationslisten handelt es sich im Allgemeinen um einen Ausschnit des gesamten Datenbestandes, der einem bestimmten Nutzer präsentiert wird. Aus Konsistenzgründen ist es allerdings unabdingbar, eventuelle Änderungen in der Datenbasis auch in diesem Ausschnitt zu aktualisieren. Dies kann mit dem Konzept der Views (*Sichten*) realisiert werden. Dabei stellen Views im Gegensatz zu den Basisrelationen virtuelle Tabellen dar, welche anhand einer Sichtendefinition als Abbildungvorschrift zur Anfragezeit aus einer oder mehreren Tabellen abgeleitet werden. Diese separate Ableitung zur Laufzeit für jede Anfrage stellt auch eines der hauptsächlichen Effizienzprobleme im Umgang mit Sichten dar. Eine relativ komplexe Ableitungsregel der Publikationslisten ist zu erwarten. Um eine effiziente Bearbeitung solcher Anfragen zu unterstützen, wurde in der Implementierung ein aus dem Bereich der *Data Warehouses* bekanntes und häufig eingesetztes Konzept übernommen, nämlich das der *materialisierten Sichten*. Hierbei werden die nötigen und häufig verwendeten Sichten initial berechnet und als eigenständige Relation abgespeichert. Somit entfällt die aufwendige Neuberechnug für jeden einzelnen Aufruf. Dieser Vorteil wird durch den Aufwand der Implementierung entsprechender Aktualisierungsroutinen und ihre tatsächliche Laufzeit bei vorliegender Änderung der Basisdaten relativiert. Hierbei ist zu beachten, dass die erwartete Anzahl der Anfragen an das System, welche nach bestimmten Publikationslisten verlangen, erheblich höher ist als die der Datenänderungen, etwa in Form von Neupublikationen. Diese Tatsache befürwortet den Einsatz einer solchen Lösung. Die Aktualisierungsmethoden sind dazu als fester Bestandteil der Routinen für das Schreiben der Publikationsinformationen im Veröffentlichungsprozess integriert. Eine Neuberechnung findet demnach jeweils bei der Hinzunahme einer neuen Veröffentlichungen statt. Im Zuge dieser Neuberechnung erfolgt gleichzeitig auch die entsprechende Formatierung in alle verfügbaren Formate. Somit kann der Selektionsaufwand bedeutend reduziert werden.

4.10.1 Manueller Listenexport

Die Benutzeroberfläche für den manuellen Export bietet umfangreiche Auswahlmöglichkeiten zur Gestaltung und Sortierung der Publikationslisten. Zur Auswahl stehen hier die Formate HTML, XML oder das BibTex-Format. Die Sortierung der Ergebnisse kann dabei nach der Publikationsart oder dem Veröffentlichungsjahr vorgenommen werden. Die Anfrage- und Bearbeitungslogik ist für jede dieser Kombinationen in einer eigenen Methode der Klasse `DbRw` gekapselt. Nach Ausführung einer der Selektionsmethoden werden die gesammelten Publikationsdaten der ausgewählten Formatierung entsprechend zunächst in einer Kontrollansicht dargeboten. Durch Betätigen eines dazu gekennzeichneten Links erfolgt die Ausgabe des zugehörigen Quellcodes in einem separaten Fenster. Dieser kann nun für weitere Verarbeitungszwecke (wie etwa weitere Formatierung nach besonderen Vorgaben) auf den Rechner der Benutzer übertragen werden. Eine Beispielausgabe der Listenfunktion ist in Abbildung 4.15 gegeben.

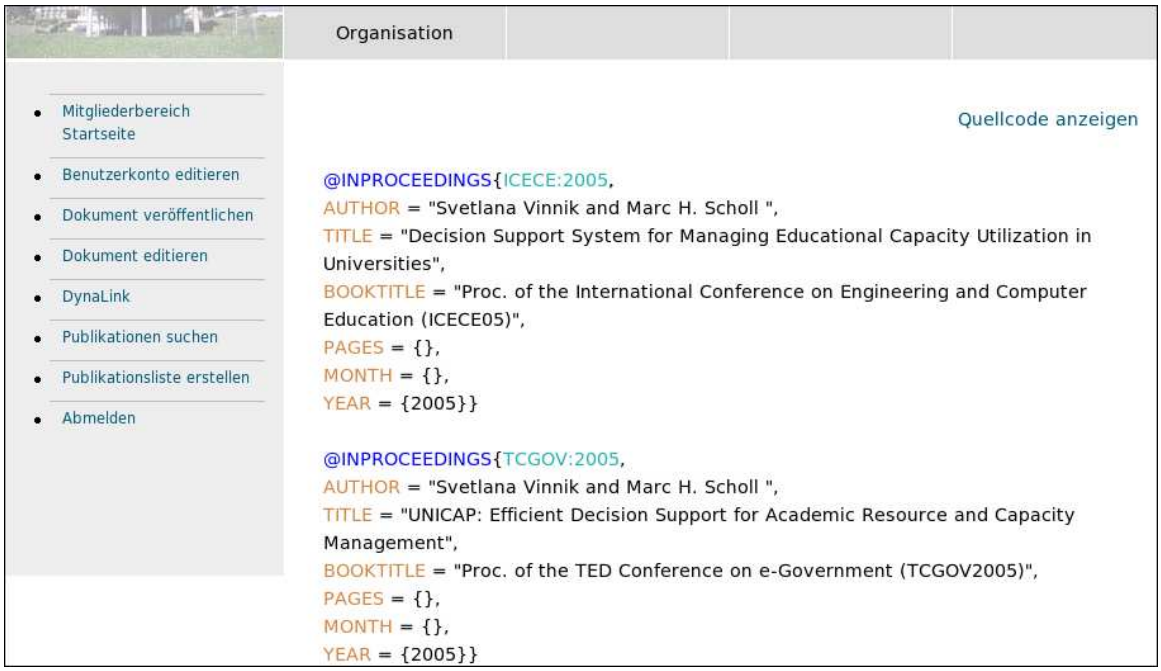


Abbildung 4.15: Präsentation der Publikationslisten

4.10.2 Automatischer Listenexport: die Dynalink-Funktion

Eine weitere Möglichkeit, die Publikationslisten der eigenen Webpräsenzen mit Hilfe des Publikationsverwaltungssystems stets aktuell zu halten, ist die automatische Listengenerierung. Über eine geeignete Schnittstelle können dabei vordefinierte Anfragen ausgeführt und die Ergebnisse, welche in Form von HTML-Code zurückgeliefert werden, in Echtzeit im Internetauftritt angezeigt werden. Ein wesentlicher Vorteil ist hierbei die

ständige Aktualisierung des Publikationsbestands durch die Anwendung. Alle vom Benutzer veröffentlichten Publikationen sowie die Dokumente, welche von Koautoren in das System eingebracht werden, sind sofort und ohne jegliche Codeänderungen in der Webpräsenz selbst sichtbar. Dabei ist der initiale Aufwand der Einbindung dieser Funktion minimal. Registrierten Benutzern steht die Option DynaLink zur Verfügung. Über eine Benutzeroberfläche kann dabei die gewünschte Sortierung der zu erzeugenden Publikationsliste gewählt werden, entweder nach Veröffentlichungsjahr oder nach Publikationsart. Darauf hin wird vom System eine personalisierte URL für den entfernten Aufruf der Exportmethode generiert. Dies kann vom Benutzer in den Quellcode der eigenen Webpräsenz, etwa in Form eines `<IFRAME>`-Tags, auf einfache Weise integriert werden. Ab diesem Moment sind stets aktuelle Publikationslisten verfügbar.
Diese entfernten Aufrufe werden ebenso wie die gewöhnlichen Benutzerinteraktionen vom Server behandelt. Aus dem übergebenen Aufruf erzeugt das Servlet des Tomcat-Webservers ein Request-Objekt. Dieses wird zusammen mit den Aufrufparametern vom Struts-Framework an die entsprechende Action-Klasse, welche die eigentliche Verarbeitungslogik kapselt, weitergeleitet. Die so gewonnenen Ergebnisse, die in HTML Formatierte Publikationsliste, wird als Response-Objekt an den aufrufenden Client zurückgegeben.

4.11 Validierung von Benutzereingaben

Auch wenn die Applikationslogik über Konstrukte für den Umgang mit Fehleingaben verfügt, ist es für einen reibungsloseren Programmablauf sinnvoll, darauf zu achten, dass die Anwendung vollständige und weitgehend korrekte Eingaben der Benutzer erhält. Dazu zählen die lexikalische, syntaktische und semantische Validierung der Benutzereingaben. Hierbei dient die lexikalische Validierung der Überprüfung von Eingabetypen, die syntaktische Validierung prüft die korrekte Abfolge der eingegebenen Daten. Als Beispiel kann hier die Validierung von Datumseingaben oder die der Emailadressen angegeben werden. Die semantische Validierung zuletzt ist stark abhänging von der Anwendungslogik. Zu ihren Überprüfungen zählt etwa die Kontrolle der Datumsfelder, ein Veröffentlichungsdatum in der Zukunft ist zum Beispiel nicht zulässig. Das Struts-Framework bietet dazu Funktionalitäten an, um diese Validierung entsprechend zu integrieren. Die Struts *ActionForms* ermöglichen durch die Implementierung der *validate*()−Methode alle drei Validierungsformen. Darin werden wiederum die entsprechenden Validierungsmethoden der *Validator*-Klasse mit den Feldeingaben als Übergabeparameter aufgerufen. Das Ergebnis dieser Prüfungen sind stets Boolesche Werte. Für die Implementierung der Methoden zur lexikalischen und syntaktischen Validierung bietet sich der Einsatz von regulären Ausdrücken an.

Kapitel 5

Zusammenfassung und Ausblick

In der vorliegenden Arbeit wurde die Konzeption und Implementierung einer Anwendung zur Verwaltung wissenschftlicher Publikationen beschrieben. Dabei wurde im Gegensatz zu den bestehenden Implementierungen der konzeptionelle Schwerpunkt auf den Benutzerkreis der Wissensproduzenten, die wissenschaftlichen Mitarbeiter, gelegt.
Einer Disskusion grundlegender und für die weitere Ausarbeitung notwendiger Fachbegriffe folgend, brachte die Anforderungsanalyse eine Reihe von Funktionalitäten hervor, welche von dem ausgewählten Benutzerkreis als unverzichtbare Arbeitshilfen im praktischen Umgang mit wissenschaftlichen Publikationen empfunden wurden. Bisher vermisste Funktionen zur Beschleunigung des Veröffentlichungsprozesses, wie es die Verarbeitung einzelner oder in Listenform vorhandener BibTex−Einträge sind, eine Möglichkeit die hierarchische Struktur zwischen den einzelnen Mitarbeitern und dem Lehrstuhl als Organisationseinheit abzubilden, ferner eine komfortable Generierung und der Export von Publikationslisten zur Einbindung in eigene Webpräsenzen wurden dabei als hauptsächliche Differenzierungsmerkmale zu bestehenden Applikationen genannt. Diese stellen zugleich die bedeutenden Mehrwerte für die Benutzer dar.
Angelehnt an die so erfassten Kernpunkte wurde des weiteren unter Berücksichtigung allgemeiner Qualitätsanforderungen an die Entwicklung von Softwareanwendungen eine geeignete Systemarchitektur entworfen und ihre wesentlichen Bestandteile näher erläutert. Das anschließend konziperte Datenmodell folgte gleichermaßen den so ermittelten Vorgaben. Im Implementierungskapitel wurden nach der Beschreibung allgemeiner Anwendungsmodule wie Datenbankanbindung und Suchfunktion die Anwendungsfunktionen mit den jeweiligen Problemstellungen und Lösungsansätzen erläutert.

Eine zukünftige Arbeit könnte sich auf die Weiterentwicklung bestehender Elemente konzentrieren. Die Erweiterung der Suchfunktion um die Suchmöglichkeit nach den Zitierungsschlüsseln sowie eine Navigationsmöglichkeit im Publikationsbestand nach Autoren oder Themengebieten stellen sicherlich notwendige Erweiterungen dar. Eine breitere Auswahl an Exportformaten und Layouts von Publikationslisten innerhalb der DynaLink Funktion, etwa in Form von den benutzern frei definierbaren `css` Dateien, würden einen weiteren Mehrwert für die Benutzer bedeuten.

Anhang A

Das Datenbankmodell

Das Ergebniss der im Kapitel 3 beschriebenen Datenbank-Entwurfsprozesse ist in den folgenden zwei Abbildungen dargestellt. Dabei illustriert die Abbildung A.1 das gesamte *Entity-Relationship Model* der Datenbank-Konzipierung. Das aus der Transformation resultierende Relationale Modell ist in Form eines Datenbankschemas in A.2 dargestellt.

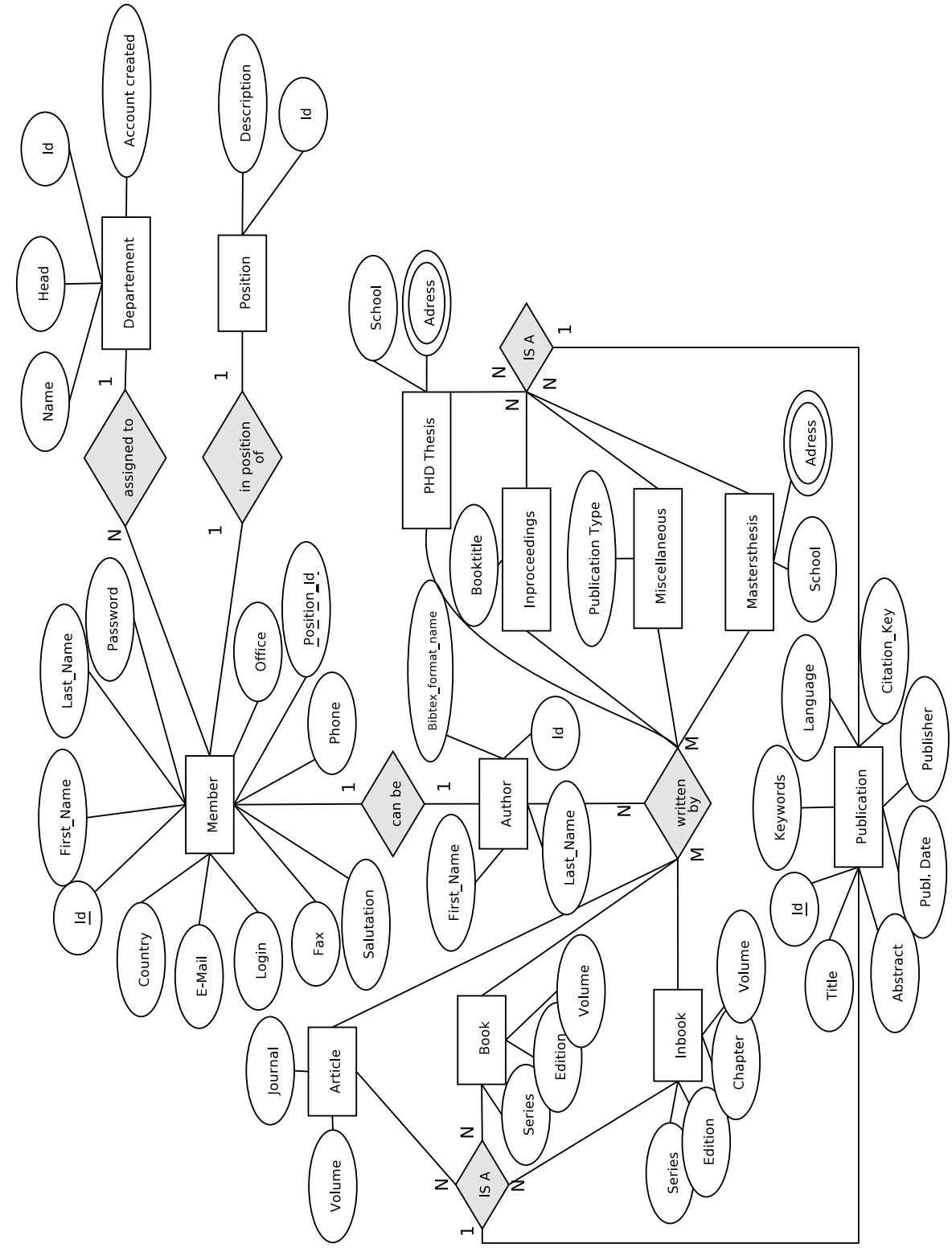


Abbildung A.1: Das Entity Relationship Modell

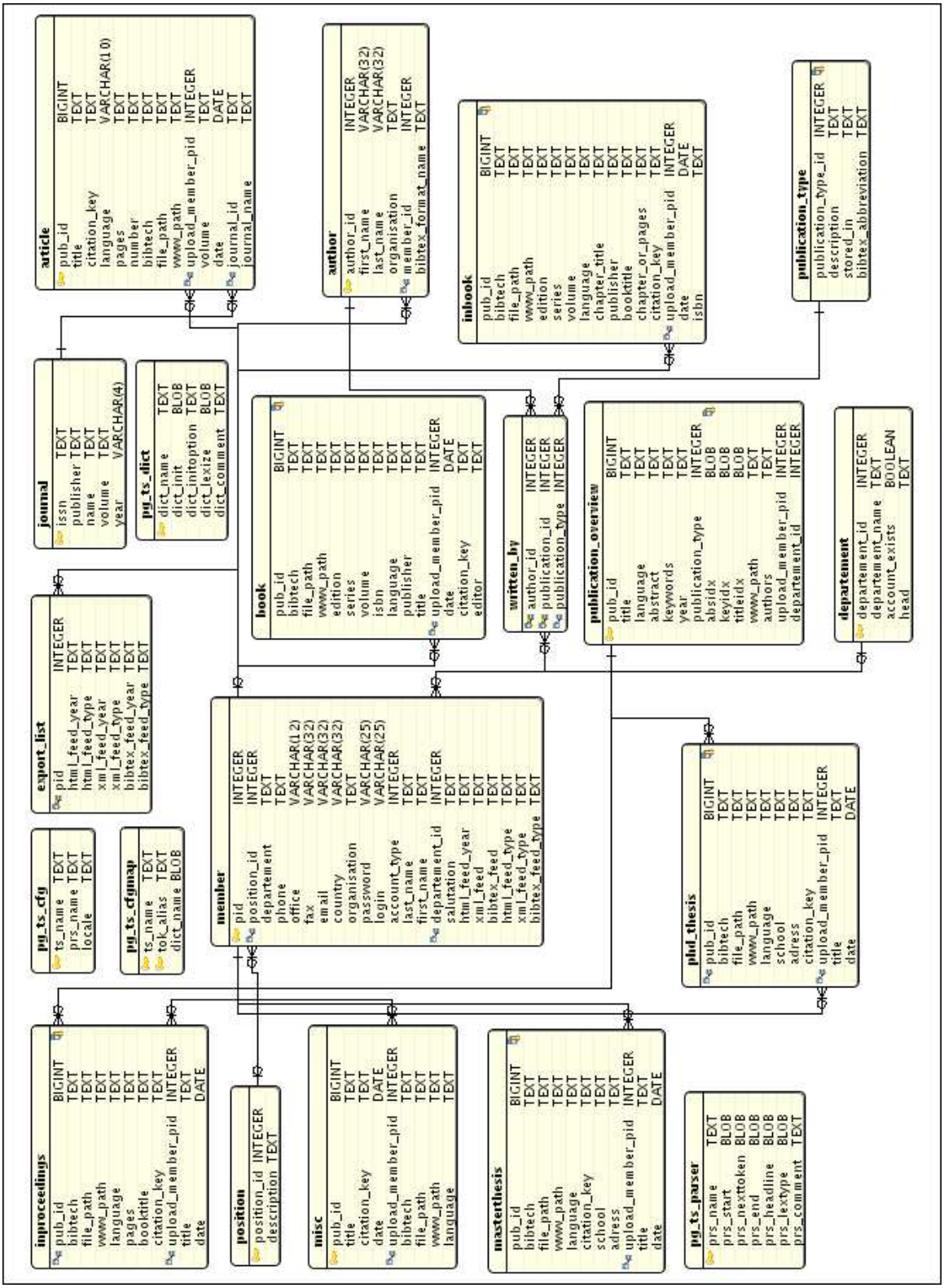


Abbildung A.2: Das Datenbankschema

Literaturverzeichnis

[AH00] G. Saake A. Heuer. *Datenbanken, Konzepte und Sprachen.* mitp-Verlag, Bonn, 2000.

[Aig] Aigaion : `http://aigaion.sourceforge.net/`; Letzter Zugriff am 25 September 2006.

[AK04] A. Eickler A. Kemper. *Datenbanksysteme.* Oldenbourg Verlag, München, 2004.

[Bek] B. Bekavac. Skript zum Kurs Information Retrieval,Skript WS 2001/2002. `http://www.inf-wiss.uni-konstanz.de/CURR/winter0102/IR/ir_script_ws01.pdf`. Letzter Zugriff am 25 September 2006.

[Biba] Biblioscape : `http://www.biblioscape.com/`. Letzter Zugriff am 25 September 2006.

[bibb] bibparse : `http://www.math.utah.edu/pub/bibparse/`. Letzter Zugriff am 25 September 2006.

[bibc] bibclean : `http://www.math.utah.edu/pub/bibclean/`. Letzter Zugriff am 25 September 2006.

[bibd] bib2xml : `http://www-plan.cs.colorado.edu/henkel/stuff/bib2xml/`. Letzter Zugriff am 25 September 2006.

[ces] Center for Science and Technology Studies Bern:`http://www.cest.ch/de/`; Letzter Zugriff am 25 September 2006.

[Deh03] W. Dehnhardt. *Java und Datenbanken.* Carl Hanser Verlag, München, 2003.

[End] Endnote : `http://www.endnote.com/`. Letzter Zugriff am 25 September 2006.

[Fer03] R. Ferber. *Information Retrieval.* dpunkt.verlag GmbH, Heidelberg, 2003.

[FH03] Philip Schaffhauser Felix Hauser. *Database-driven XML-enabled Bibliography Management System*. 2003. `http://dret.net/netdret/docs/da-ws2002-hauser-schaffhauser.pdf`; Letzter Zugriff am 25 September 2006.

[GS83] Michael J. McGill Gerard Salton. *Information Retrieval – Grundlegendes für Informationswissenschaftler*. McGraw-Hill Book Company GmbH Konstanz, Hamburg, 1983.

[HBMW05] S. Haiges, A. Bien, M. May, and B. Woehrlin. *Struts Java Framework für Webanwendungen*. Software und Support Verlag GmbH, Frankfurt, 2005.

[JDB] JDBC API : `http://java.sun.com/j2se/1.5.0/docs/guide/jdbc/`. Letzter Zugriff am 25 September 2006.

[JMH95] Avi Pfeffer Joseph M. Hellerstein, Jeffrey F. Naughton. Generalized search trees for database systems. In *Proc. 21st Int'l Conf. on Very Large Data Bases*, pages 562–573, 1995.

[Kop] Konstanzer Online-Publikations-System : `http://www.ub.uni-konstanz.de/kops/`; Letzter Zugriff am 25 September 2006.

[Kuh77] Rainer Kuhlen. *Experimentelle Morphologie in der Informationswissenschaft*. Verlag Dokumentation, München, 1977.

[Kuh95] Rainer Kuhlen. *Informationsmarkt: Chancen und Risiken der Kommerzialisierung von Wissen*. Universitätsverlag Konstanz, Konstanz, 1995.

[MVC] Java BluePrints. Model-View-Controller. `http://java.sun.com/blueprints/patterns/MVC-detailed.html` ; Letzter Zugriff am 25 September 2006.

[Opu] OPUS : `http://elib.uni-stuttgart.de/opus/`; Letzter Zugriff am 25 September 2006.

[OSW] OSWD : `http://www.bsz-bw.de/cgi-bin/oswd-suche.pl`; Letzter Zugriff am 25 September 2006.

[Php] phpbiblio : `http://sourceforge.net/projects/phpbiblio`. Letzter Zugriff am 25 September 2006.

[Por] M. F. Porter. An algorithm for suffix stripping. `http://telemat.det.unifi.it/book/2001/wchange/download/stem_porter.html`; Letzter Zugriff am 25 September 2006.

[Pos] PostgreSQL : `http://www.postgresql.org/`; Letzter Zugriff am 25 September 2006.

[Pro] Procite : http://www.procite.com/. Letzter Zugriff am 25 September 2006.

[Rij79] C. J. Van Rijsbergen. *Infromation Retrieval*. BUTTER WORKS, London, 1979.

[Sal68] Gerard Salton. *Automatic information organization and retrieval*. McGraw-Hill, New York, 1968.

[Sch04] Markus Schneider. *Implementierungskonzepte für Datenbanksysteme*. Springer-Verlag, Berlin, 2004.

[Sno] Snowball stemming algorithm. http://www.snowball.tartarus.org/; Letzter Zugriff am 25 September 2006.

[Str] Apache Software Foundation. Struts Framework.http://struts.apache.org/; Letzter Zugriff am 25 September 2006.

[Tom] Apache Tomcat : http://tomcat.apache.org/; Letzter Zugriff am 25 September 2006.

[Tse] Tsearch2 : http://www.sai.msu.su/~megera/postgres/gist/tsearch/V2/. Letzter Zugriff am 25 September 2006.

[Vos00] G. Vossen. *Datenbankmodelle, Datenbanksprachen und Datenbankmanagementsysteme*. Oldenbourg Verlag, München, 2000.

Selbstständigkeitserklärung

Hiermit erkläre ich, daß ich die vorliegende Arbeit selbstständig und nur mit erlaubten Hilfsmitteln angefertigt habe.

Konstanz, 26. September 2006

Siniša Avramović