Abschlussbericht

Bachelorarbeit

Advanced Writing App

Thomas Mayer

0925406

1. Projektbeschreibung

1.1 Allgemeines

Advanced Writing App ist ein spezielles Tool, welches dem Benutzer beim Erstellen wissenschaftlicher Arbeiten (im Prototyp ausschließlich Seminararbeiten) unterstützt. Dies wird einerseits durch eine vordefinierte Gliederung der Arbeit erzielt. Andererseits erhält der User Informationen zu einzelnen Passagen und wird so durch den Schreibprozess geleitet. Ein weiterer Aspekt im Zuge dieser Vereinfachung ist die interne Referenzenverwaltung, die eine sehr einfache und effektive Handhabung der verwendeten wissenschaftlichen Quellen (im Prototyp 3 Typen von Quellen) ermöglicht. Der Benutzer kann sich somit ganz auf den eigentlichen Prozess des Schreibens (welcher ohnehin selbst eigenen Kriterien genügen muss) konzentrieren ohne sich mit der formalen Strukturierung wissenschaftlicher Texte auseinander setzen zu müssen. Das Programm bietet die Möglichkeit erstellte Arbeiten als *LaTeX*-Dateien zu exportieren. Die exportierten Dateien können als Grundlage für weitere Modifizierungen dienen, oder direkt kompiliert und damit in ein PDF-Format übergeführt werden (Aufgabe der *LaTeX*-Distribution bzw. des -Compilers).

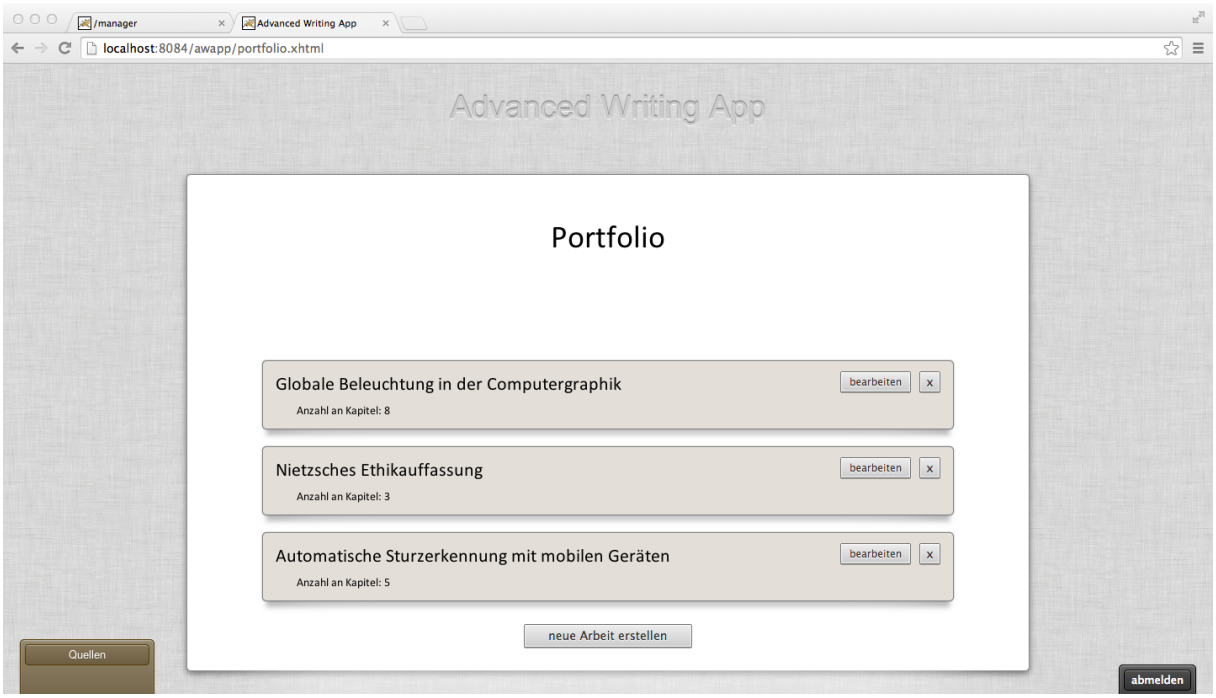

Abbildung 1: User-Interface (Auflistung der einzelnen Arbeiten)

1.2 Anwendungsfälle:

Gerade zu Beginn eines Studiums – bzw. einer akademischen Laufbahn – sind die Studierenden oft mit den Konventionen, Notationen und formalen Kriterien – die zum verfassen wissenschaftlicher Arbeiten notwendig sind – hilflos überfordert. Dies liegt im Allgemeinen daran, dass ein wissenschaftlicher Text einer logischen, nachvollziehbaren und argumentativen Struktur verpflichtet ist. Es wird somit ein erheblicher Einarbeitungsaufwand erforderlich.

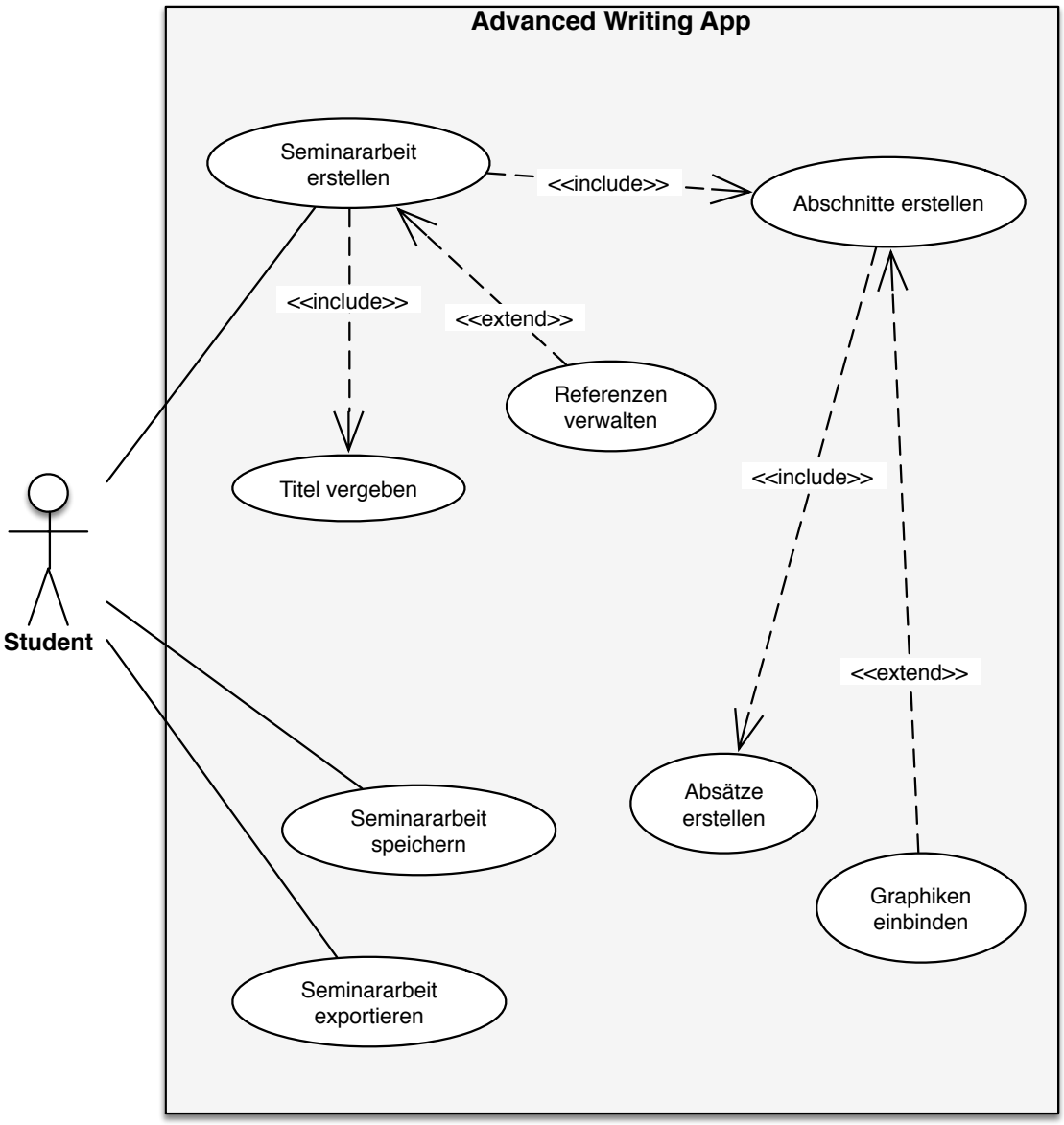

Abbildung 2: Usecase-Diagramm

Die Software lässt den Benutzer auf einfachste Weise brauchbare Seminararbeiten auf schnellem Weg zu erstellen. Der generierte *LaTeX*-Output kann nachträglich beliebig verändert, ausgebaut, sowie erweitert werden.

1.3. Aufbau:

Wissenschaftliche Texte sind üblicherweise in Ebenen aufgebaut und verzweigen sich in die Tiefe. Formal entspricht dies der Gliederung in Haupt- und Unterkapitel.

Um diese Aspekte in der Applikation zu berücksichtigen, findet programmintern eine Unterteilung in Dokument-, sowie Abschnittebene statt.

Dokumentenebene:

Aufgabenbereich: Festlegung der Grundinformationen + Gliederung der Arbeit (Festlegung der Struktur durch Abschnitte/Kapitel)

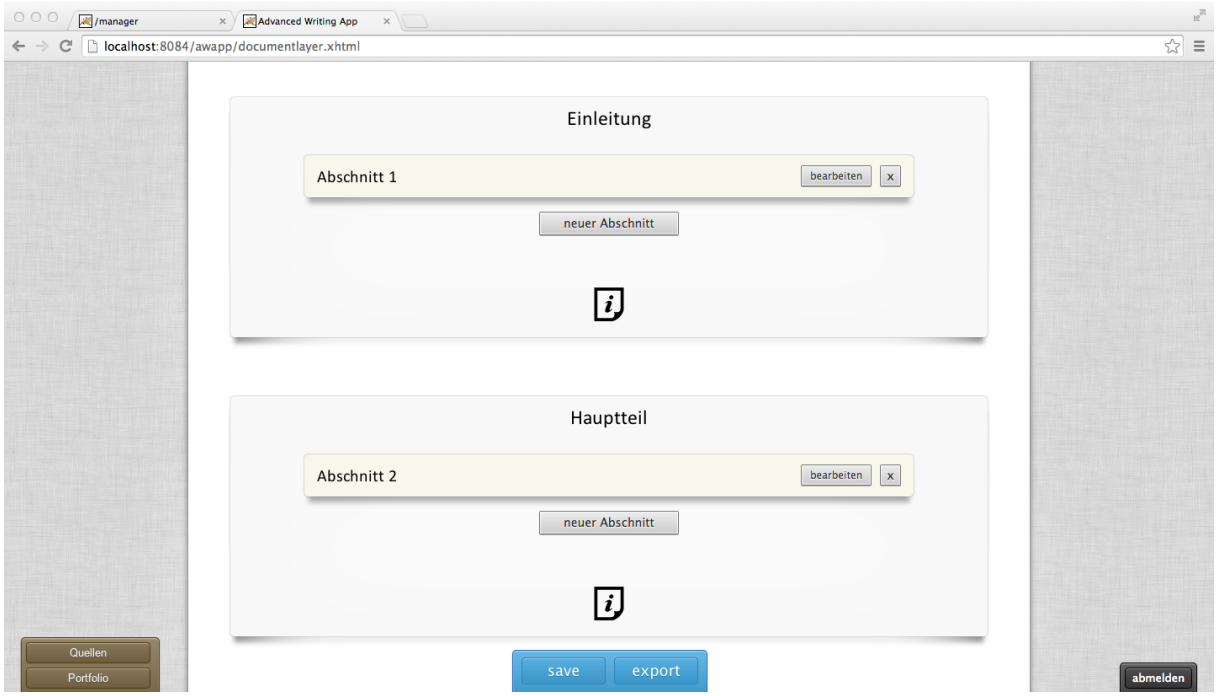

Abbildung 3: Dokumentenebene

Abschnittebene:

Aufgabenbereich: Schreiben des Textes (eigentliche Texterstellung) + Gliederung durch einzelne Absätze (Paragraphen)

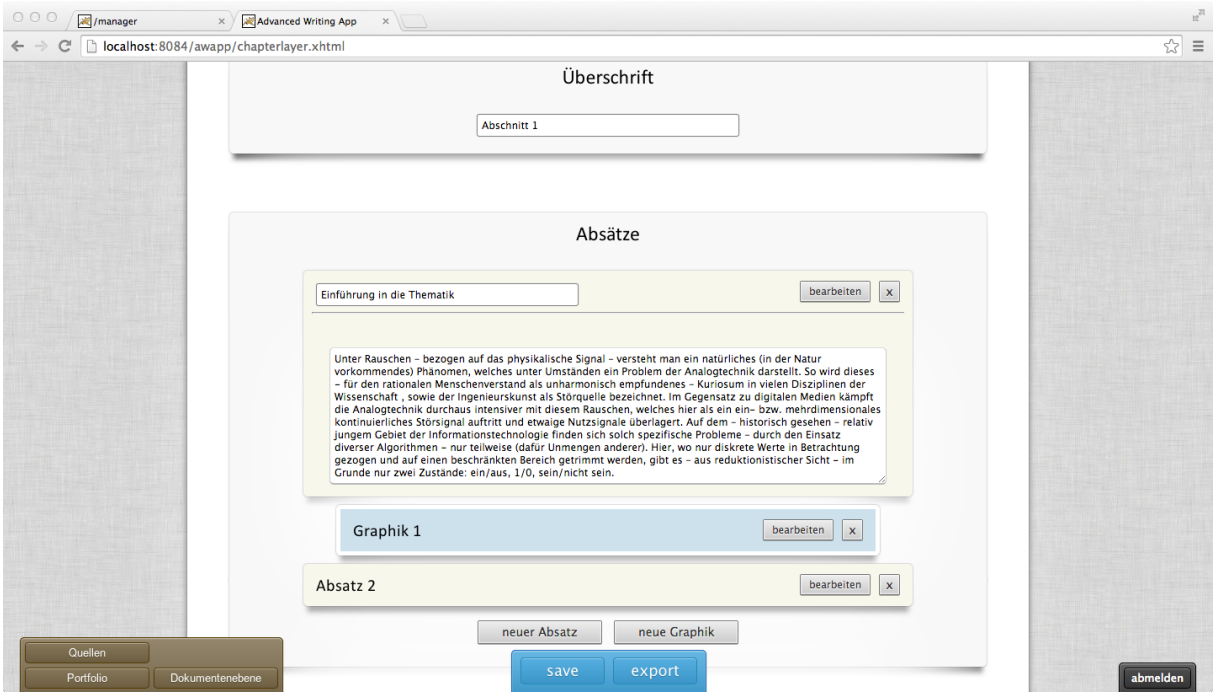

Abbildung 4: Abschnittebene

1.4 Referenzenverwaltung

Die Applikation bietet dem Benutzer die Möglichkeit verwendete Quellen zu erstellen, zu verwalten und diese schließlich in die eigene Arbeit zu integrieren (um daraus zu zitieren).

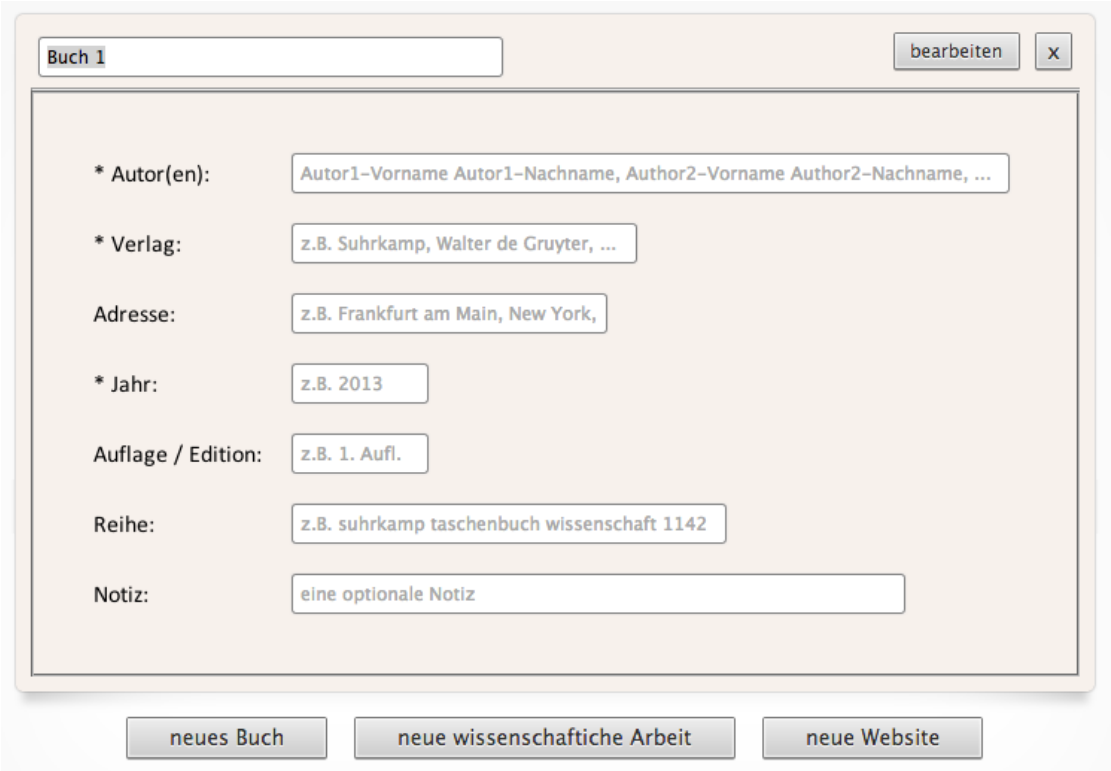

Abbildung 5: Hinzufügen eines wissenschaftlichen Buches

1.5. Weitere Merkmale und Features

. Internationalisierung (i18n)

. unterstützte Bildformate: JPG, PNG

. *BibTeX* - Unterstützung

. *HTML5* Drag & Drop

. *GNU GPL*

. *Ajax* - Funktionalität

2. Implementierung und technische Details

2.1 Architektur

Beim Projekt handelt es sich um eine Webapplikation, welche mit Hilfe des JSF-Frameworks (*JavaServer Faces*) realisiert wurde. Die Version 2.2 ermöglicht erstmals eine einfache Kombination von clientseitiger *HTML5*-Funktionalität und den serverseitigen JSF-Technologien. Somit kann ein Großteil der UI-Logik auf den Client ausgelagert werden.

Frontend: *HTML5* + *JavaScript* + *CSS* (u.a. *CSS3*)

Backend: MVC-Pattern + *JSF* + *Spring* + *Hibernate* + *MySQL*

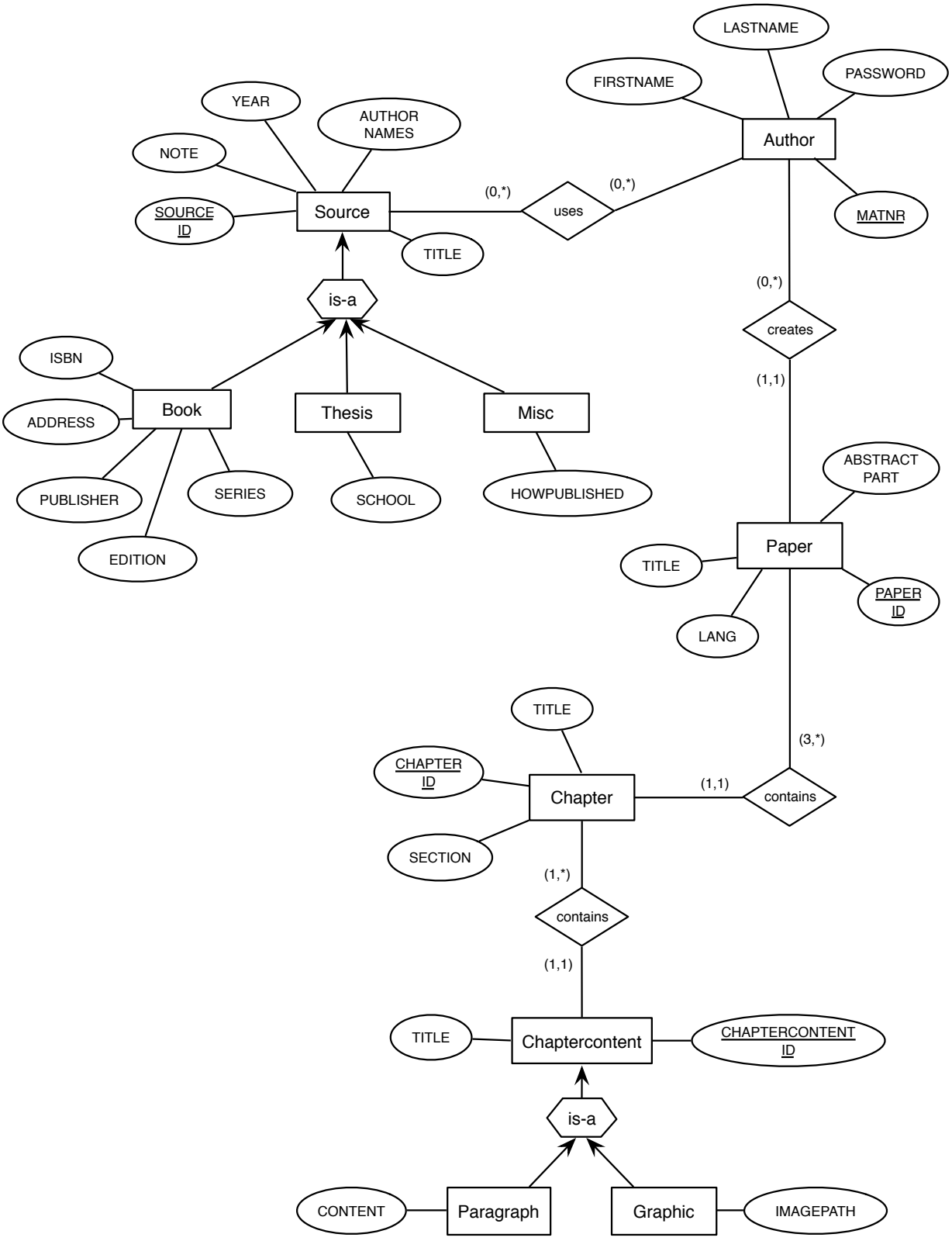

Abbildung 6: ER-Diagramm des Models

2.2 Deployment / Inbetriebnahme

Weiters handelt es sich bei der Webapplikation um ein *Maven*-Projekt, welches als WAR-File (Web application Archive) deployed wird. Somit kann die - in *Java* geschriebene - Webanwendung mit Hilfe eines Servletcontainers ausgeführt werden. Dies geschieht unter Einsatz des Open Source Webservers und Webcontainers *Tomcat*.

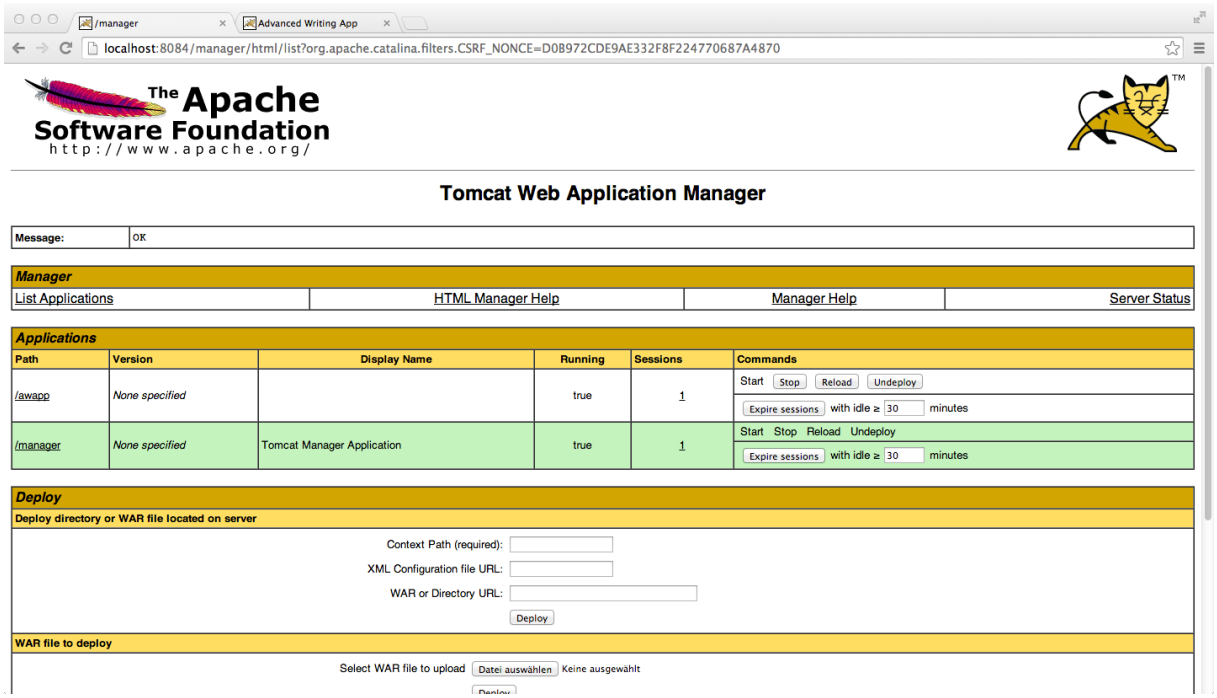

Abbildung 7: Auflistung der verfügbaren Web-Applikationen im Tomcat Web Application Manager

Für die persistente Speicherung der Daten wird eine *MySQL*-Datenbank benötigt. Die erforderlichen Informationen bzw. Zugangsdaten befinden sich in der *Spring*-Konfigurationsdatei (spring-config.xml), wo die Datenquelle als *Spring*-bean definiert ist:

```
<bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource"
      p:driverClassName="com.mysql.jdbc.Driver" p:url="jdbc:mysql://localhost/awapp_db"
      p:username="root" p:password="LF6doNkj" p:initialSize="5" p:maxActive="10">
</bean>
```

Abbildung 8: Ausschnitt aus spring-confing.xml

2.3 Verwendete Libraries

Folgende Libraries stellen sozusagen die Eckpfeiler der Anwendung dar bzw. sind wesentlich für deren Funktionalität. Eine vollständige Auflistung der vom *Maven*-Projekt verwendeten Bibliotheken (inkl. Versionen) findet sich in der pom.xml-Datei.

- *Apache MyFaces* — *MyFaces* ist eine Open Source Implementierung der JSF-Spezifikation.
  Konfigurationsfiles: faces-config.xml, web.xml

- *Hibernate/JPA* — *Hibernate* ist eine Implementierung der JPA-Spezifikation und wird in der Applikation für ORM verwendet.
  Konfigurationsfiles: persistence.xml

- *Spring* — *Spring* ist ein quelloffenes Framework für die *Java*-Plattform, welches als Hauptaufgabe die Vereinfachung der Projektentwicklung hat. In der Applikation wird *Spring* für die allgemeine „Verdrahtung" der einzelnen Komponenten herangezogen. Die sogenannten „beans" werden einerseits direkt in der spring-config.xml-Datei deklariert, andererseits durch Annotationen im Quellcode.
  Konfigurationsfiles: spring-config.xml

- *Apache Velocity* — Bei *Velocity* handelt es sich um ein sogenanntes Templating-Tool, das Platzhalter in Textdateien durch dynamische Inhalte ersetzt. *Advanced Writing App* verwendet generell 2 unterschiedliche Templates, also Vorlagen oder Schablonen, die im Ordner WEB-INF/velocity" zu finden sind:
  - template_essay_de.vm (*LaTeX*-Vorlage für Seminararbeit)
  - bibtex.vm (*BibTeX*-Vorlage für Quellenverzeichnis)

- *Modernizr* — *JavaScript* - Bibliothek

- *jQuery* — *JavaScript* - Bibliothek

2.2 Bekannte Probleme, unvollständige Features und Bugs

Titelerzeugung:

Im vorliegenden Prototyp ist es nicht möglich den Titel der Arbeit – z.B. durch die Vergabe von Schlüsselwörtern – automatisch generieren/erzeugen zu lassen. Die graphische Benutzeroberfläche dazu ist bereits implementiert.

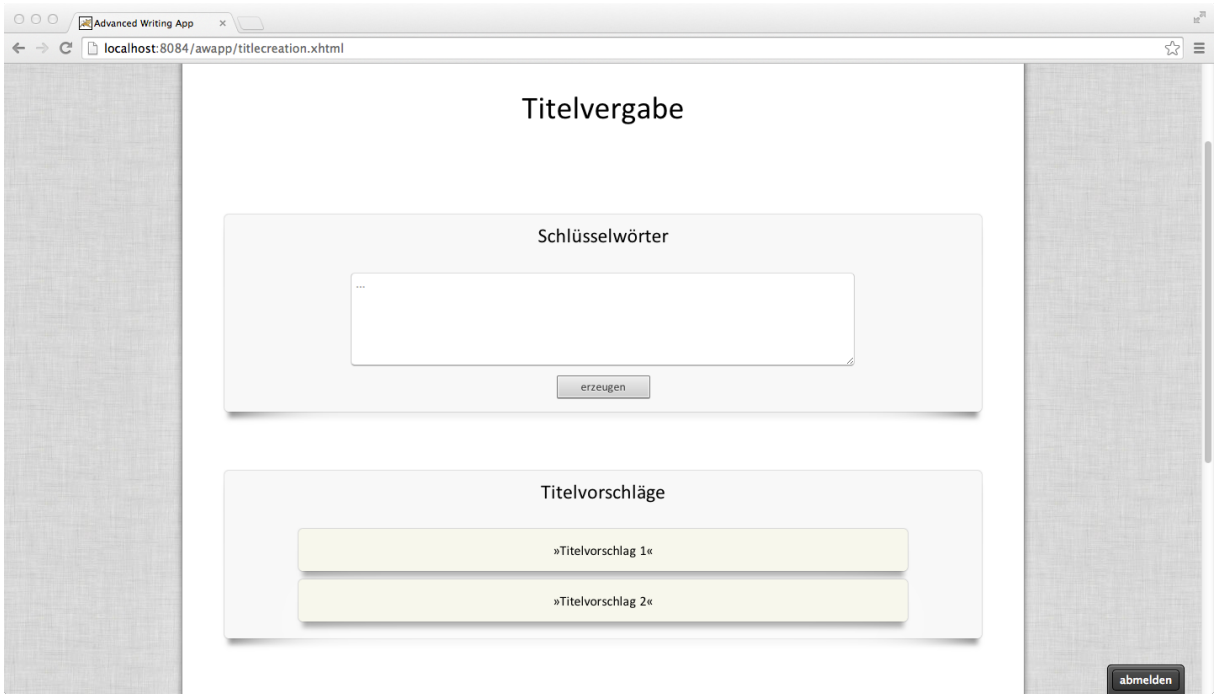

Abbildung 9: Titelerzeugung

Hier kann man die Idee dieser Titelerzeugung bereits graphisch nachvollziehen. Nach Vergabe der Keywords und Betätigung des „erzeugen“-Buttons soll eine Auflistung der Titelvorschläge in Listenform erscheinen. Klickt man auf einen dieser vorgeschlagenen Einträge, wird dieser als finaler Titel in ein Textfeld eingefügt. Der Text des finalen Titels kann wie gewöhnlich manuell bearbeitet werden bevor die Änderungen komplett übernommen werden. Die Textarea zur Eingabe der Schlüsselwörter ist im Prototyp noch disabled.

Probleme mit Graphik-Upload:

Auf der Abschnittebene kann der Benutzer neben Absätzen, also Paragraphen, auch Graphiken im JPG- oder PNG-Format hinzufügen. Hier gibt es nach wie vor Probleme mit dem eigentlichen Upload und dem Speichern der Bilder in der DB (vgl. ImageServlet).

Kein Responsive Design:

Da es sich bei der Web-Anwendung um eine Schreibapplikation handelt, wurde dieser Aspekt nicht berücksichtigt. Der Grund dafür ist, dass sich die Bedienung durch virtuelle Tastaturen auf mobilen Geräten als umständlich gestaltet bzw. nur bedingt möglich ist. Im Prototyp werden derzeit absolute Einheuten verwendet.

2.4 Potentiale für Weiterentwicklungen

. Um mehrere Templates (z.B. Bachelor-, Masterarbeit, etc.) erweitern

. Titelvergabe / -erzeugung einbetten