*pub"*). Die Zuweisung der Zugriffslisten zu einem Mandanten erfolgt über die ID. Die ID eines Mandanten in der relationalen Datenbank entspricht der ID der jeweiligen Zugriffsliste in der XML-Datenbank. In Beispiel 6.1. *Zugriffsliste für Nutzer* ist exemplarisch eine entsprechende XML-Datei für die Konfiguration in Abbildung 6.1. *Konfiguration der Nutzergruppen* abgebildet.

### Beispiel 6.1. Zugriffsliste für Nutzer

```
<useraccess id="1" ino:docname="useraccess/1"
            xmlns:ino="http://namespaces.softwareag.com/tamino/response2">
   <admin id="admin">
      <access permit="mp@bitset.de" />
   </admin>
   <management id="management">
      <access permit="stephan@stephan-wiesner.de" />
      <access permit="bonin@fhnon.de" />
   </management>
   <employee id="employee">
      <access permit="*@fhnon.de" />
   </employee>
   <dependant id="dependant">
      <access permit="*@rzserv2.fhnon.de" />
   </dependant>
</useraccess>
```

Neben individuellen Zugriffslisten für einen Mandanten können auch Dokumentenkategorien für jeden Mandanten gesondert verwaltet werden (vgl. L01.2). Für diese existiert eine weitere Liste, welche konsequenterweise ebenfalls in dem Tamino XML Server gespeichert wird. In dieser Liste werden alle Kategorien verwaltet, die einem Dokument zur Veröffentlichung unter diesem Mandanten zugewiesen werden können. Im Bereich einer Hochschule wären beispielsweise Kategorien wie Vorlesungsskripte, Übungen oder Mitschriften sinnvoll, im Bereich eines Unternehmens zum Beispiel Geschäftsberichte oder Produktdokumentationen.

Das Ändern von Zugriffs- oder Kategorielisten ist Nutzern der Gruppe admin vorbehalten. Das Erstellen, Löschen und Ändern von Mandanten (dies betrifft die Daten, welche in der relationalen Datenbank gespeichert sind) hingegen ist nur von dem System-Administrator durchführbar.

Zur Verwaltung von Mandanten dient die Klasse `ClientAdministration`, ein Mandant wird dabei durch die Klasse `Client` abgebildet. Zugriffs- und Kategorielisten werden durch die Klassen `AccessList` und `ClassificationList` abgebildet.

## 6.2. Benutzermanagement

Zum Suchen nach und Abrufen von Dokumenten kann ein Nutzer anonym bleiben, er muss dem System nicht bekannt sein und sich daher auch nicht anmelden. Um jedoch Dokumente im DigiPub-System veröffentlichen zu können, muss sich ein Nutzer an dem System anmelden und einen Mandanten wählen, unter dem er sein Dokument veröffentlichen möchte. Die

Berechtigung für die Veröffentlichung unter dem gewählten Mandanten wird, wie in Abschnitt 6.1 beschrieben, anhand der Emailadresse überprüft. Kann ein Nutzer keiner der in Tabelle 6.1. *Nutzergruppen eines Mandanten* aufgelisteten Gruppen zugeordnet werden, so ist für ihn das Veröffentlichen von Dokumenten in diesem Mandanten nicht möglich.

Zur Registrierung eines Nutzers ist die Eingabe eines Loginnamens und einer Emailadresse zwingend erforderlich, beide müssen zudem in dem DigiPub-System eindeutig sein. Grund hierfür ist die Überprüfung der Emailadresse: nach einer erfolgreichen Registrierung wird ein vom DigiPub-DMS generiertes Passwort an diese Adresse verschickt, welches zusammen mit dem Loginnamen zum Anmelden am System benötigt wird. Ein Nutzer kann sich also erst anmelden, wenn er sein Passwort per Email erhalten hat. Des Weiteren kann er sich nur einmal für jede Emailadresse registrieren und nicht ein und dieselbe Adresse für verschiedene „alias“-Namen benutzen. Weitere Angaben sind im Allgemeinen nicht zwingend erforderlich (zu weiteren Nutzerdaten siehe Abbildung 5.3. *Relationale Datenbank "digipub"*, Entität „user“). Soll ein Nutzer einem Mandanten als Verantwortlicher zugewiesen werden, so stellt dieser eine Ausnahme dar. In diesem Fall ist die zusätzliche Angabe des Vor- und Nachnamens nötig.

Einem Nutzer wird standardmäßig bei der Registrierung die Rolle Benutzer zugewiesen (vgl. Tabelle 5.2. *Nutzerrollen*). Die Rolle des System- oder Inhaltsadministrators ist aus Sicherheitsgründen nicht über die Benutzungsoberfläche des DigiPub-DMS möglich, sondern nur direkt in der Datenbank einstellbar.

Für Nutzer, die kostenpflichtige Dokumente veröffentlichen möchten, sind die Angaben zu einer Bankverbindung und einer Adresse erforderlich. Diese Daten werden genutzt, um dem Nutzer einen möglichen Erlös für diese Dokumente zukommen zu lassen.

Ein Nutzer kann seine Daten selbständig innerhalb der Anwendung ändern, davon ausgenommen ist das Ändern der Emailadresse, des Loginnamens und der Nutzerrolle. Für den Fall, dass ein Nutzer sein Passwort vergessen hat, kann er dieses durch die Eingabe seines Loginnamens an die für diesen Namen registrierte Emailadresse senden lassen.

Das Passwort wird aus Sicherheitsgründen vor dem Speichern in der Datenbank nach dem *Password-Based Cryptography Standard* [62] verschlüsselt. Zur Generierung eines Passwortes dient die Klasse `PasswordGenerator`, die Verschlüsselung wird durch die Klasse `PasswordEncryptor` vorgenommen. Die Möglichkeit des Emailversandes ist mit Hilfe der JavaMail-API implementiert und wird durch die Klasse `Mailer` abgehandelt.

Zum Verwalten von Benutzern dient die Klasse `UserAdministration`, wobei ein Benutzer durch die Klasse `User` abgebildet wird.

[62] Siehe http://www.rsasecurity.com/rsalabs/pkcs/pkcs-5/index.html (Zugriff: 11.01.2003)

## 6.3. Dokumentenmanagement

Dieser Abschnitt beschreibt den Ablauf vom Einstellen von Dokumenten in das System bis zum Abruf sowie die Aktionen, die vom DigiPub-System hierbei ausgeführt werden müssen. Anhand von XML-Quellcodebeispielen wird der systeminterne Datenverarbeitungsprozess exemplarisch dargestellt.

### 6.3.1. Einstellen von Dokumenten

Das DigiPub-DMS erlaubt das Veröffentlichen von elektronischen Dokumenten jeglichen Formates (vgl. L02). Hierbei wird zwischen zwei Arten von Dokumenten unterschieden: Zum Einen können DocBook-Dokumente veröffentlicht werden, die noch keinerlei Layout enthalten (vgl. L02.1), zum Anderen auch jede andere Art von elektronischen Dokumenten, welche als Binärdokumente bezeichnet werden (vgl. L02.2). In beiden Fällen muss das Einstellen der Dokumente über das Webinterface möglich sein und es müssen XML-Metadaten in dem Tamino XML Server hinterlegt werden, um Dokumente anhand dieser Daten im System abrufen zu können. Weiterhin erfordert die Einbindung eines Sicherheitssystems für PDF-Dokumente und die eines Zahlungssystems die Speicherung der hierfür relevanten Daten in der relationalen Datenbank. In Abbildung 6.2. *Dokumente einstellen - Übersicht* ist der allgemeine Ablauf, der für beide Fälle gleichermaßen gilt, illustriert. Eine genaue Erläuterung der nötigen Aktionen findet sich in den jeweiligen Abschnitten, Abschnitt 6.3.1.1 und Abschnitt 6.3.1.2.

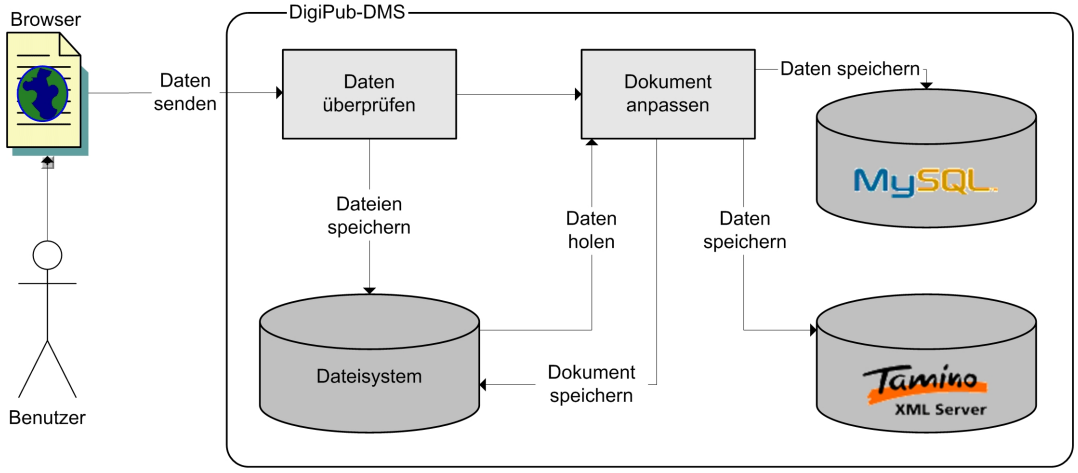


**Abbildung 6.2. Dokumente einstellen - Übersicht**

#### 6.3.1.1. DocBook-Dokumente einstellen

Das DigiPub-DMS ist speziell konzipiert für das Publizieren von Dokumenten, die in dem XML-Format DocBook erstellt worden sind. Diese Dokumente werden komplett in dem Tamino XML Server gespeichert, um eine Volltextdurchsuchbarkeit (vgl. L03.2) und die Erstellung virtueller Dokumente (vgl. L04) zu gewährleisten. DocBook unterstützt die Erstellung von Dokumenten verschiedener Typen, so beispielsweise eines Buches (book), eines Artikels

(article) oder auch nur eines Kapitels (chapter). Das DigiPub-DMS erlaubt jedoch nur die Speicherung von DocBook-Dokumenten des Dokumententyps book (vgl. Abbildung 5.4. *XML-Datenbank "digipub"*, Collection „book“). Diese Entscheidung wurde getroffen, um sich im Rahmen des DPS-Projektes auf diesen Dokumententyp konzentrieren zu können.

Ein komplettes DocBook-Dokument ist nicht beschränkt auf XML-Inhalt. Es kann weitere Komponenten wie beispielsweise Bilder enthalten, die innerhalb des XML-Inhaltes referenziert werden. Auf diese referenzierten Komponenten muss bei der Generierung eines PDF-Dokumentes zugegriffen werden können, weshalb alle Dateien, die in dem DocBook-Dokument referenziert werden, gemeinsam mit diesem in das System geladen werden müssen. Eine Ausnahme stellen Dateien dar, die über eine absolute URI im Internet referenziert werden. Für alle Dateien, die zusammen mit dem XML-Dokument in das System geladen werden, erfolgt ebenfalls eine Hinterlegung in dem Tamino XML Server.

#### 6.3.1.1.1. Dokumentenupload

Ein Nutzer, dem entsprechende Rechte für einen Mandanten zugewiesen sind (vgl. Tabelle 6.1. *Nutzergruppen eines Mandanten*), kann innerhalb der Anwendung sein Dokument über ein Formular in das DigiPub-DMS einspielen. Akzeptiert wird hierbei ein einzelnes XML-Dokument oder auch eine gepackte ZIP-Datei, falls weitere Komponenten eingebunden sind. Diese ZIP-Datei darf jedoch nur eine einzige XML-Datei enthalten, damit das eigentliche Dokument eindeutig identifiziert werden kann. In beiden Fällen wird die XML-Datei anhand ihrer Dateiendung (*.xml*) identifiziert.

Im DPS-Projekt wurde entschieden, ein separates Sicherheitskonzept für DocBook-Dokumente zu implementieren. Dadurch wird einem Autor die Möglichkeit gegeben, beim Einstellen eines Dokumentes direkt im System eigene Sicherheitseinstellungen vorzunehmen, welche in das auszuliefernde PDF-Dokument integriert werden (siehe Abschnitt 1.3). In Abbildung 6.3. *DocBook-Dokumente einstellen - Formular* ist das Formular zum Einstellen von DocBook-Dokumenten abgebildet. Sicherheitsaspekte, die erst bei der Erstellung eines PDF-Dokumentes zum Tragen kommen, sind darin durch einen *Pfeil* gekennzeichnet.

Der anzugebende *Dokumentenname* in Abbildung 6.3. *DocBook-Dokumente einstellen - Formular* dient zur eindeutigen Identifizierung eines Dokumentes in dem DigiPub-DMS. Er ist somit zwingend erforderlich und muss im System eindeutig sein. Die Veröffentlichung von Dokumenten mit ISBN ist nur dem Inhalts-Administrator oder System-Administrator gestattet (vgl. Tabelle 5.2. *Nutzerrollen*). Nur ein Nutzer mit einer dieser Rollen erhält beim Einstellen von DocBook-Dokumenten die in Abbildung 6.3. *DocBook-Dokumente einstellen - Formular* abgebildete Eingabemöglichkeit zur Festlegung einer ISBN für das Dokument (vgl. L08). Die Veröffentlichung von ISBN-Dokumenten ist bisher nur bei Nutzung des DocBook-Formates vorgesehen, da für diese Formate ein ISBN Buchland-Strichcode automatisch generiert werden kann.

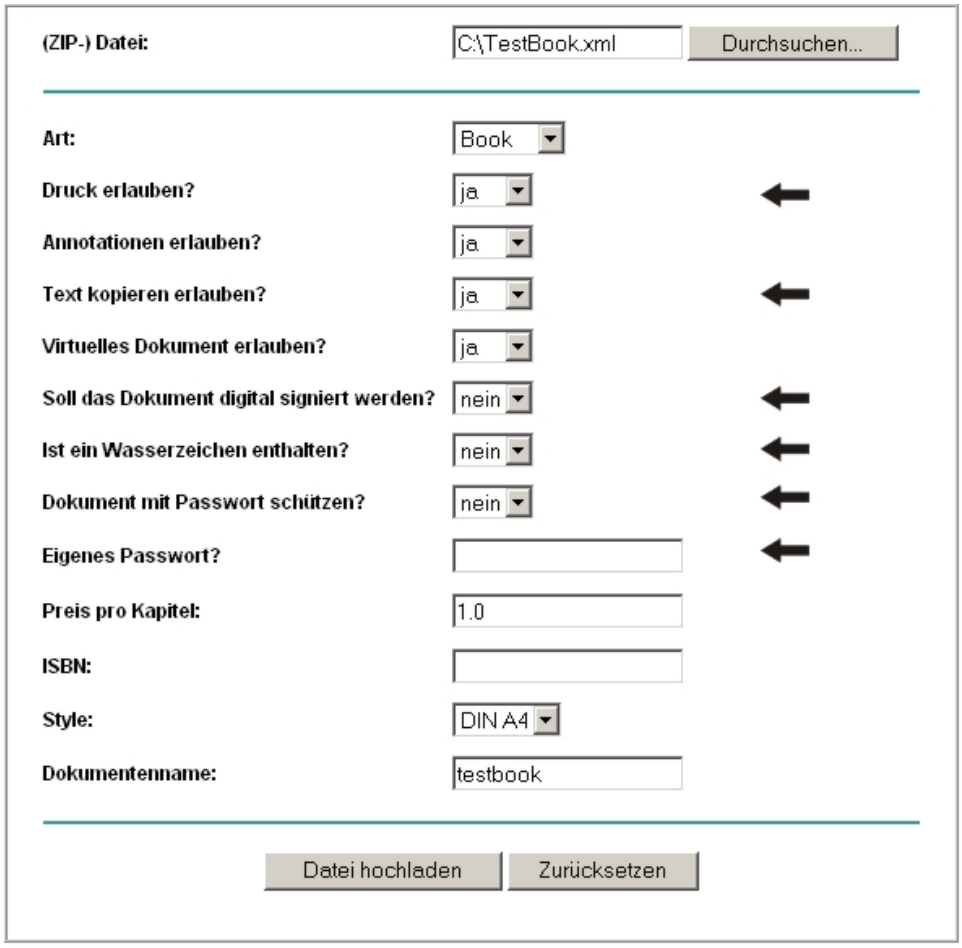


**Abbildung 6.3. DocBook-Dokumente einstellen - Formular**

Anhand der Auswahl eines *Styles* hat der Nutzer die Möglichkeit, zwischen verschiedenen Formaten für sein Dokument zu wählen. Im Rahmen des DPS-Projektes wurden zwei unterschiedliche Möglichkeiten implementiert, welche sich hauptsächlich im Papierformat und der Schriftart unterscheiden. Eine Erweiterung auf mehrere Formatvorlagen ist durch eine entsprechende Konfiguration einpflegbar (siehe Abschnitt 5.8).

Bei der Auswahl der jeweiligen Einstellungen werden vom DigiPub-DMS Abhängigkeiten zwischen den einzelnen Wahlmöglichkeiten überprüft, da nicht jede Kombination erlaubt wird. Die verschiedenen Überprüfungen sind im Folgenden erläutert:

- Ist ein eigenes Passwort gesetzt, darf es nicht erlaubt sein, aus Teilen dieses Dokumentes ein virtuelles Dokument zu erstellen. Das DigiPub-System könnte im Fall eines virtuellen Dokumentes mit Bestandteilen aus zwei oder mehreren Dokumenten, in denen ein benutzerdefiniertes Passwort gesetzt ist, keine eindeutige Zuweisung für ein Passwort erstellen.

- Ebenfalls ist es nicht erlaubt, einen Preis von größer 0 für ein Dokument zu verlangen, in dem ein eigenes Passwort gesetzt wird. Ein eigenes Passwort ist sinnvoll, um den Leserbereich zu beschränken, es wird einem Nutzer aber nicht vom DigiPub-System mitgeteilt. Wird ein kostenpflichtiges Dokument von einem Nutzer angefragt, so muss dieser auch

Zugriff auf das Dokument erlangen, was in diesem Fall nicht gewährleistet werden könnte.

- Eine digitale Signatur kann nur für Dokumente vergeben werden, die nicht mehr verändert werden, da diese auf dem Dokumenteninhalt beruht. Aus diesem Grund ist diese Einstellung nur wählbar, wenn das Einfügen von Annotationen verboten und das Dokument nicht mit einem Passwort versehen wird (dieses wird im Nachhinein in das Dokument generiert und würde den Inhalt ebenfalls ändern).

Bei einer unzulässigen Auswahl wird eine entsprechende Meldung an den Nutzer ausgegeben. Die Überprüfung der Abhängigkeiten ist mit Hilfe von JavaScript implementiert, um direkt beim Nutzer stattfinden zu können. Hierdurch entfällt eine Übertragung der Daten (inklusive der gewählten Datei) bei fehlerhaften Angaben. Eine Überprüfung auf dem Server durch eine `ActionForm`-Klasse (siehe Abschnitt 4.4) würde erst nach einer Übertragung aller vom Nutzer gesendeten Daten stattfinden können. Da der Dokumentenname zur eindeutigen Referenzierung des Dokumentes genutzt wird und zusammen mit dem Schemanamen (vgl. Abschnitt 5.7.2) als ID dient, werden für die Wahl des Namens nur Zeichenkombinationen akzeptiert, die ausschließlich Buchstaben und Zahlen enthalten. So kann dieser Name als Wert für ein id-Attribut genutzt werden.

Die Implementation zur Überprüfung der einzelnen Abhängigkeiten ist direkt in der JSP-Datei zur Anzeige des Uploadformulars integriert. Zum Upload der Datei in das DigiPub-DMS und einem eventuellem Entpacken einer ZIP-Datei dient die Klasse `FileUploadHandler`.

#### 6.3.1.1.2. Dokumentenupdate

Bevor die XML-Dokumente in der Datenbank gespeichert werden können, müssen diese vom System überarbeitet werden. In gültigen DocBook-Dokumenten müssen Attribute vom Typ *ID* (vgl. „DTD“) im gesamten Dokument eindeutig sein. Dies wird für normale Dokumente beim Speichern in dem Tamino XML Server automatisch überprüft, eine Speicherung von ungültigen Dokumenten wird abgelehnt. Die Eindeutigkeit dieser Attribute ist allerdings ebenfalls für virtuelle Dokumente nötig (vgl. Abschnitt 6.3.3), was nicht zwingend gegeben ist, da ein und derselbe Wert für ein id-Attribut in unterschiedlichen Dokumenten auftauchen kann und somit bei einer neuen Zusammenstellung auch in dem virtuellen Dokument vorhanden wäre.

Außerdem ermöglicht das DigiPub-DMS das Annotieren von Dokumenten direkt im Dokument. Der zu annotierende Abschnitt muss eindeutig referenzierbar sein. Aus diesen beiden Gründen wird bereits beim Einstellen der Dokumente dafür gesorgt, dass alle id-Attribute in der gesamten Collection *book* eindeutig sind. Zudem wird für eine spätere mögliche Annotation für jedes chapter und section Element, welches kein id-Attribut enthält, ein eigenes solches Attribut generiert (eine genaue Beschreibung zur Implementation dieses Konzeptes ist in Abschnitt 6.4 erläutert). Da in DocBook-Dokumenten interne Referenzen über Referenzierun-

gen von id-Attributen geregelt sind, müssen ebenfalls alle Attribute des Typs *IDREF* und *IDREFS* auf die Referenzierung der neuen id-Werte geändert werden.

Weiterhin müssen relative Verweise auf externe Dateien, die keine absolute URI referenzieren, umgestellt werden. Bei diesen Dateien, die wie vorab beschrieben beim Einstellen der Dokumente mit in dem System gespeichert werden müssen, werden die Verweise auf ein zentrales Servlet, `GetBinaryDataServlet`, umgestellt. Diese dient als Schnittstelle zwischen der XML-Datenbank, speziell der *binary*-Collection, und dem FOP-Prozess (Näheres siehe „Dokumentenspeicherung und PDF-Erzeugung“).

Enthält das Dokument eine ISBN-Angabe, so wird diese entfernt, da bei der Vergabe einer ISBN besondere Sorgfalt vorliegen sollte. Zum Veröffentlichen solcher Dokumente berechtigte Personen (siehe Tabelle 5.2. *Nutzerrollen*) erhalten, wie in Abbildung 6.3. *DocBook-Dokumente einstellen - Formular* aufgezeigt, ein separates Feld zur Angabe einer ISBN. Diese wird dann als Inhalt eines isbn Elementes in dem bookinfo Teil des Dokumentes hinterlegt. Mit diesem Konzept wird gleichzeitig die Veröffentlichung eines ISBN-Dokumentes durch unberechtigte Nutzer unterbunden.

Zur Ermöglichung einer hierarchischen Anzeige der Kapitel bei einer späteren Suchabfrage müssen alle Titel dieser Kapitel sowie alle ID's zur Referenzierung für ein mögliches virtuelles Dokument bekannt sein. Um hier eine möglichst performante Abfrage zu erreichen, wird bereits im Vorfeld in jedes title-Element eines Kapitels (chapter) oder Abschnittes (section) die ID des jeweiligen Elementes als Attribut eingefügt. Des Weiteren wird ebenfalls die ID des Oberkapitels oder -abschnittes mit als Attribut eingefügt, um den hierarchischen Baum aufbauen zu können. Damit hierbei keine bereits vorhandenen Attribute des Dokumentes überschrieben werden, wurden zu diesem Zweck eigene Attribute definiert: parent_id enthält die ID des Oberelementes, chapter_id bzw. section_id die ID des entsprechenden Kapitels oder Abschnittes. Durch dieses Konzept kann die Datenmenge für ein Suchergebnis sehr gering gehalten werden, ein kompletter Abruf aller Kapitel entfällt. Ein Beispiel für ein solches Ergebnis kann in Beispiel 6.5. *Suchergebnis in XML* eingesehen werden.

Neben diesen nötigen Änderungen der DocBook-Datei werden weitere Metainformationen in das Dokument eingefügt, die vom DigiPub-DMS zur späteren Verarbeitung benötigt werden. Hierzu dienen unter anderem die vom Nutzer in das Formular zum Dokumentenupload eingegebenen Daten (siehe Abbildung 6.3. *DocBook-Dokumente einstellen - Formular*). Eingefügt werden diese in das DocBook-Element bookinfo, da dieses für Metainformationen über das Dokument dient[63]. In allen Fällen erfolgt die Speicherung der Informationen in einem Attribut, da auf diese leichter zugegriffen werden kann[64]. Des Weiteren wurde jedem Elementnamen der Begriff *digipub* vorangestellt, um möglichen Namenskonflikten bei einer eventuellen Erweiterung der DocBook-Dokumentendefinition aus dem Weg zu gehen. Die vom DigiPub-DMS erstellten Metainformationen sind in Beispiel 6.2. *Metainformationen* erläutert.

[63] Siehe [Walsh1999], S. 166

[64] Vgl. [Klettke2002], S. 145

### Beispiel 6.2. Metainformationen

```
<bookinfo ino:id="1" ino:docname="book/testbook"
          xmlns:ino="http://namespaces.softwareag.com/tamino/response2">
  <title>Testdokument</title>
  <author>
    <firstname>Marko</firstname>
    <surname>Petersen</surname>
    <email>mp@bitset.de</email>
  </author>
  <keywordset>
    <keyword>DigiPub</keyword>
    <keyword>XML</keyword>
  </keywordset>
  <abstract>
    <para>eine kleine Zusammenfassung</para>
  </abstract>
  <title>Testdokument</title>
  <digipubinfo>
    <digipubclient value="Testverlag"/>                              ❶
    <digipubuser value="3"/>                                         ❷
    <digipubuploadtime value="03.01.2003 12:12:56"/>                 ❸
    <digipublevel value="admin"/>                                    ❹
    <digipubclassification value="Book"/>                            ❺

    <digipubprice value="1.0"/>                                      ❻
    <digipubchapter value="1"/>

    <digipubaccessible value="true"/>                                ❼
    <digipubannotation value="true"/>
    <digipubvirtual value="true"/>

    <digipubfileextension value="DigiPub PDF"/>                      ❽
    <digipubstyle value="/usr/local/docbook/stylesheet/fo.xsl"/>     ❾
  </digipubinfo>
</bookinfo>
```

Anmerkung zu Beispiel 6.2. *Metainformationen*: Leerzeilen innerhalb des Quelltextes wurden nachträglich zur Gliederung von Blöcken eingefügt, um Elemente mit ähnlicher Funktionalität in einem Abschnitt zu beschreiben.

❶ Name des Mandanten, in dem das Dokument veröffentlicht ist

❷ ID des Nutzers, der das Dokument veröffentlicht hat

❸ Systemzeit, zu der das Dokument veröffentlicht wurde

❹ Nutzergruppe des Nutzers (❷) für den entsprechenden Mandanten(❶)

❺ Kategorie des Dokumentes

❻ Preis für ein Kapitel (digipubprice) sowie Kapitelanzahl des Dokumentes (digipubchapter) zur Berechnung des Dokumentenpreises

❼ Merkmale, ob das Dokument zugreifbar oder gesperrt (digipubaccessible) ist, annotierbar (digipubannotation) ist, und ob Erstellen virtueller Dokumente aus Teilen des Dokumentes erlaubt ist (digipubvirtual)

❽ Dateiendung des Dokumentes (für DocBook-Dokumente wird hier der Eintrag „DigiPub PDF“ verwendet)

❾ Pfad zu dem zu verwendenden XSL-Stylesheet im System

Zur Speicherung der Metainformationen ist das Element bookinfo demnach zwingend erforderlich, allerdings nicht in der Dokumentendefinition des Elementes book (vgl. Beispiel 6.7. *Valides DocBook-Dokument*). Aus diesem Grund wird das Element bookinfo automatisch eingefügt, falls es fehlen sollte. Des Weiteren wird in dem Fall, dass kein title Element in den Metainformationen enthalten ist, versucht, alle title Elemente, die direkte Unterknoten von book sind, in den Metadatenbereich des Elementes bookinfo zu kopieren. Diese Angaben werden vom DigiPub-DMS bei einer Suchanfrage zum Darstellen des Ergebnisses genutzt (vgl. Abschnitt 6.3.2).

Der gesamte Updateprozess des XML-Dokumentes wird durch eine XSLT-Transformation durchgeführt. Dies bietet den Vorteil, dass bei möglichen Erweiterungen der DocBook-Dokumentendefinition Änderungen in dem XSL-Stylesheet (`DocUpdate.xsl`) vollzogen werden können und nicht in den Java-Quelltext der Anwendung eingegriffen werden muss.

#### 6.3.1.1.3. Dokumentenspeicherung und PDF-Erzeugung

Die Speicherung des DocBook-Dokumentes erfolgt im Tamino XML Server. Das XML-Dokument wird in der Collection *book* hinterlegt, wobei eine Überprüfung auf Gültigkeit des Dokumentes automatisch erfolgt. Alle weiteren zu dem Dokument gehörigen Dateien wie Bilder werden in der Collection *binary* hinterlegt. Metadaten zur Weiterverarbeitung für das Preis- und Sicherheitskonzept werden in der relationalen Datenbank abgelegt (Tabelle *document*), wobei für DocBook-Dokumente der Wert für *is_binary* auf „false“ gesetzt wird.

Eine spätere Auslieferung der Dokumente erfolgt im PDF-Format, welches durch eine XSLT-Transformation und den anschließenden Einsatz von FOP erzeugt wird. Durch den Einsatz von FOP 0.20.5 zur Erzeugung der PDF-Datei wird bereits direkt nach der Speicherung des Dokumentes eine Transformation in dieses Format vorgenommen. Dies ist erforderlich, da es durchaus möglich ist, ein gültiges DocBook-Dokument zu erstellen, welches sich nicht erfolgreich zu einem PDF-Dokument transformieren lässt. Dieser Fall muss bereits beim Einstellen des Dokumentes überprüft werden, damit keine fehlerhaften PDF-Dokumente in dem System gespeichert werden. Tritt bei der PDF-Erzeugung ein Fehler auf, werden die entsprechenden Daten wieder aus der Datenbank entfernt und eine Meldung für den Nutzer, der dieses Dokument einspielen wollte, angezeigt.

Zum Beispiel ist ein Dokument, welches lediglich aus einem leeren Element book besteht, ein gültiges DocBook-Dokument. Im Rahmen des Umformprozesses (vgl. „Dokumentenupdate“) wird diesem Element ein bookinfo Element hinzugefügt. *FOP* benötigt zur Erzeugung eines PDF-Dokumentes bei der Verwendung der aktuellen DocBook-Stylesheets jedoch weitere Elemente, da auf jeden Fall eine Seitensequenz für den eigentlichen Inhalt des Dokumentes erstellt wird. Ohne mindestens ein Kapitel (chapter), ein Glossar (glossar) oder ein anderes erlaubtes Element, aus dem Teile für diesen Inhalt generiert werden, schlägt die Transformation mit *FOP* fehl.

Bei der Transformation in ein PDF-Dokument werden die dem XML-Dokument zugehörigen Dateien wie Bilder eingebunden. Hierzu benötigt FOP einen Zugriff auf die Dateien und deren jeweiligen MIME-Typen. Da sich diese Dateien in dem Tamino XML Server befinden, können sie nur mit erfolgreicher Authentifizierung abgerufen werden. FOP besitzt leider noch keine Möglichkeit, Authentifizierungsdaten an einen Server zu übermitteln. Dies wurde bereits im Rahmen des AWÖ-Projektes im Sommersemester 2002 festgestellt, ein entsprechender Fehlerreport ist im *Apache Bug Database* unter der Nummer 9880 einzusehen[65]. Zum Abruf von Binärdateien aus dem Tamino XML Server dient ein Servlet, welches die Authentifizierung übernimmt, die Dateien anfragt, anhand der Dateiendung einen MIME-Typen für die Antwort bestimmt und die angefragte Datei an den FOP-Prozess sendet. Die Feststellung des MIME-Type der Datei wird zwar von Tamino unterstützt, nicht jedoch von der eingesetzten XML:DB-API [66], weshalb ein eigener Mechanismus für diesen Zweck implementiert wurde.

Bei Dokumenten mit einer ISBN wird diese auf der Titelseite vermerkt. Zudem wird ein entsprechender ISBN Buchland-Strichcode erzeugt. Für diese Zwecke wurden Stylesheets der Firma Render-X genutzt[67].

Erst nach einer erfolgreichen Erstellung des PDF-Dokumentes ist sichergestellt, dass das eingestellte DocBook-Dokument vom DigiPub-DMS erfolgreich verarbeitet werden kann. Dieser Transformationsprozess ist sehr speicher- und prozessorlastig. Das gesamte DocBook-Dokument muss genau wie das zu verwendende XSL-Stylesheet in den Hauptspeicher geladen werden. Des Weiteren beansprucht dieser Prozess einen nicht unerheblichen Zeitaufwand.

In Beispiel 6.3. *PDF-Erzeugung* ist ein Teil der FOP-Informationsausgabe für die Erzeugung einer PDF-Datei aus einem 300 KB großen DocBook-Dokument dargestellt. Diese Angabe wurde am 12.01.2003 unter Microsoft Windows XP auf einem AMD Athlon 1800 XP+ mit 512 MB Arbeitsspeicher erzeugt. Die Performanceangabe hat lediglich beispielhaften Charakter, die Geschwindigkeit beim Transformationsprozess hängt u.a. von der Einbindung von Grafiken sowie deren Größe und der Nutzung von Referenzen in dem Dokument ab (vgl. http://xml.apache.org/fop/faq.html, Zugriff: 12.01.2003).

### Beispiel 6.3. PDF-Erzeugung

```
[DEBUG] Total time used: 17015ms
[DEBUG] Pages rendererd: 88
[DEBUG] Avg render time: 193ms/page
```

[65] Erreichbar unter http://nagoya.apache.org/bugzilla/show_bug.cgi?id=9880 (Zugriff: 12.01.2003)

[66] Ein Antrag zur Integration entsprechender Methoden wurde an die XML:DB-Mailingliste gesendet, eine Diskussion hierzu ist im Tamino-Entwicklerforum erreichbar (siehe http://forums.tamino.com/3/OpenTopic?a=tpc&s=153292895&f=7192946476&m=6302968286, Zugriff: 12.01.2003)

[67] Erreichbar unter http://www.renderx.com/barcodes.html (Zugriff: 18.01.2003)

Um einem wiederholten Transformationsprozess vorzubeugen, werden die erstellten PDF-Dokumente im Dateisystem gespeichert. Bei einem Abruf eines DocBook-Dokumentes wird dieses gespeicherte PDF zurückgeliefert und eine erneute Transformation ist nicht erforderlich.

Der Uploadprozess in die XML-Datenbank erfolgt durch die Klasse `DocumentUpload`, welche ebenfalls Methoden zum Aufruf des beschriebenen XSLT-Prozesses und zur Speicherung der Metadaten in der relationalen Datenbank enthält. Zur FOP-Transformation benutzt diese Klasse die Klasse `ResultTransformer`. Zur Implementation des Zahlungssystems ist es nötig, dass die gespeicherten Dateien den Preis im Dateinamen enthalten, welcher exemplarisch für einen Preis von 2 Euro und einen Upload durch den Benutzer mit ID 4 folgendermaßen aufgebaut ist: `0020_document.id_4.pdf`. Die Formatierung des Preises wird durch die Klasse `PriceLinkManager` vorgenommen, welche von Stephan Wiesner implementiert wurde.

Zur Verarbeitung von Bildern beim FOP-Prozess auf einer Linux-Umgebung wird eine grafische Oberfläche (ein X-Server) benötigt, da das von FOP verwendete Batik-Framework diese zum Rendern kontaktiert[68]. Die Nutzung des JDK 1.4.1 ermöglicht durch den Parameter *`Djava.awt.headless=true`* eine virtuelle X-Server-Umgebung im Speicher des Systems. Dieser Parameter wurde in die Startdatei des Tomcat-Webservers (`catalina.sh`) für dessen Aufruf mit eingebunden, sodass kein aktiver X-Server laufen muss.

### 6.3.1.2. Binärdokumente einstellen

Um für Binärdokumente eine Suchmöglichkeit anzubieten (vgl. L03.1) und eine gemeinsame Schnittstelle mit DocBook-Dokumenten für den Dokumentenabruf zu erreichen, werden für Binärdokumente vom DigiPub-DMS DocBook-Dokumente mit Metadaten erstellt und in dem Tamino XML Server gespeichert. Zur Spezifizierung dieser Metadaten enthält das DigiPub-DMS ein Eingabeformular, in dem der Titel des Dokumentes angegeben werden muss und weitere Angaben wie Suchbegriffe, Informationen über den Autor und eine Zusammenfassung des Dokumentes angegeben werden können. Die Werte für Autorenangaben sind standardmäßig mit den Nutzerdaten des einstellenden Nutzers belegt, lassen sich jedoch ändern. Angaben zu Art, Preis und dem Dokumentennamen sind äquivalent zu dem Einstellen von DocBook-Dokumenten (vgl. Abschnitt 6.3.1.1). Im Gegensatz zu dem Einstellen von DocBook-Dokumenten wird eine ZIP-Datei in diesem Fall nicht entpackt.

[68] Siehe http://xml.apache.org/fop/faq.html (Zugriff: 22.01.2003)

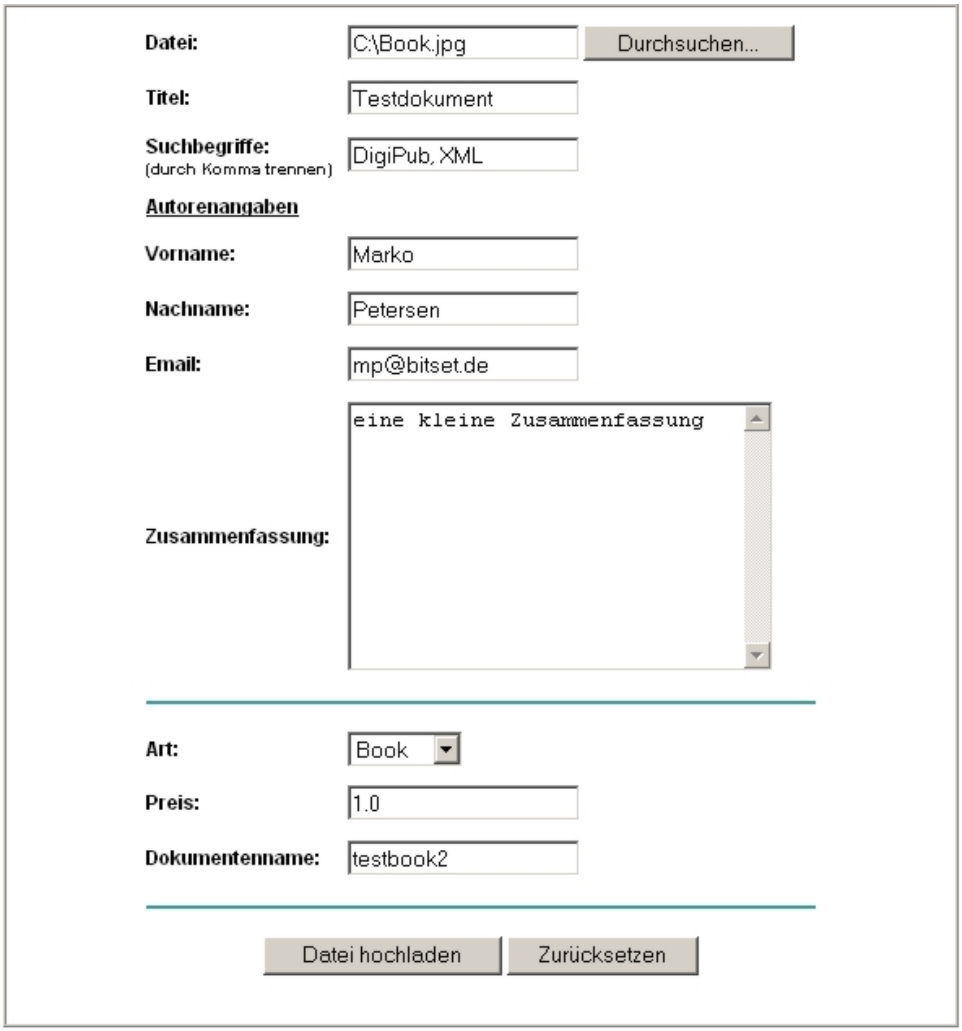


**Abbildung 6.4. Binäre Dokumente einstellen - Formular**

Das erstellte DocBook-Dokument besteht aus einem book-Element, welches lediglich ein bookinfo-Element mit den entsprechenden Metainformationen enthält. Der Inhalt eines bookinfo-Elementes, wie es für die in Abbildung 6.4. *Binäre Dokumente einstellen - Formular* angegebenen Daten erstellt wird, ist äquivalent aufgebaut zur Darstellung in Beispiel 6.2. *Metainformationen*. Für das Element digipubchapter wird hierbei ein Standardwert für @value von 1 eingefügt. Die @value-Attribute für die Elemente digipubannotation und digipubvirtual werden standardmäßig auf „false“ gesetzt, da für Binärdokumente keine Möglichkeit zum Annotieren oder Erstellen virtueller Dokumente gegeben ist. Das Attribut @value des Elementes digipubfileextension wird mit der Dateiendung der eingespielten Datei belegt, um auch Informationen über den Dateitypen in den Metadaten zu erhalten. Die Generierung der Metainformationen in dem DocBook-Dokument geschieht, genau wie in „Dokumentenupdate“ beschrieben, mit Hilfe desselben XSL-Stylesheets.

Um eine äquivalente Behandlung der unterschiedlichen Dokumentenarten bei der Auslieferung der Dokumente zu erhalten, werden Binärdokumente genau wie erstellte PDF-Dokumente im Dateisystem gespeichert. Eine Speicherung in einer Datenbank ist hier nicht nötig, da diese Dokumente in dem DigiPub-DMS unveränderbar und nicht durchsuchbar sind.

Für die Implementation des Zahlungssystems (siehe Abschnitt 1.3) werden auch hier die benötigten Daten in der relationalen Datenbank gespeichert. Sicherheitsaspekte, die lediglich für vom DigiPub-DMS erstellte PDF-Dateien relevant sind, werden hierbei von der Datenbank auf einen Standardwert gesetzt, wobei der Wert für den Eintrag *is_binary* auf „true“ gesetzt wird.

Auch das Einstellen von Binärdokumenten wird durch die Klasse `DocumentUpload` gehandhabt, die Erstellung des Metadokumentes ist mit Hilfe der Java DOM-API realisiert.

## 6.3.2. Dokumentensuche

Das DigiPub-DMS bietet einen Suchmechanismus zum Durchsuchen der Tamino Datenbank an (vgl. L03). Hierfür wird eine XPath-Query (vgl. „XPath“) anhand der vom Nutzer zur Suche eingegebenen Daten erstellt. Das Ergebnis der Suchabfrage ist ein XML-Dokument, welches durch einen XSLT-Prozess zu einem Suchergebnis aufbereitet wird.

### 6.3.2.1. Suchabfrage

Um den Suchmechanismus komfortabel zu gestalten, wurde neben der einfachen Angabe von Metadaten (wie Bereich oder Autor) auch die Möglichkeit zur Angabe von Suchoptionen (wie *UND* oder *ODER*) für die einzugebenden Suchwörter implementiert (vgl. Abbildung 6.5. *Dokumentensuche - Formular*). Außerdem erhält der Nutzer die Möglichkeit, entweder nur die Metadaten eines Dokumentes (*Stichwortsuche*) oder aber den gesamten Text (*Volltextsuche*) zu durchsuchen. Soll ein Wort nicht enthalten sein, so ist ihm das Zeichen „-“ voranzustellen. Die Problematik bei der Verwendung dieser verschiedenen Suchoptionen besteht in der internen Abbildung auf eine XPath-Anweisung. Die für die in Abbildung 6.5. *Dokumentensuche - Formular* abgebildeten Suchoptionen generierte XPath-Query ist exemplarisch in Beispiel 6.4. *XPath-Query* dargestellt.

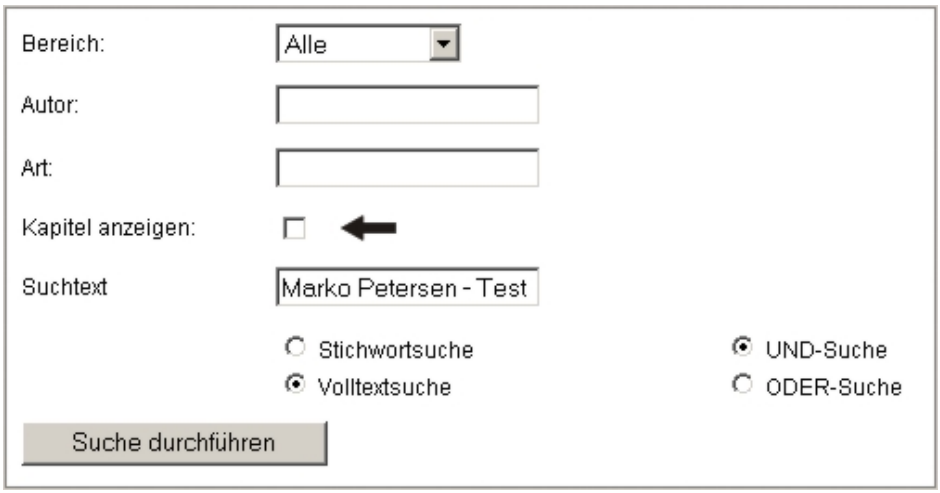


**Abbildung 6.5. Dokumentensuche - Formular**

**Beispiel 6.4. XPath-Query**

```
//bookinfo[(//* ~= 'Marko' and //* ~= 'Petersen' and not(//* ~= 'Test')) and
            not(//digipubaccessible/@value = 'false')]
```

Zur Bildung einer validen XPath-Anweisung wird der Suchtext in einzelne Worte zerlegt und mit den weiteren angegebenen Suchoptionen zusammengesetzt. Durch die Angabe von „not(//digipubaccessible/@value = 'false')“ werden nur Dokumente gefunden, die nicht gesperrt worden sind (vgl. Abschnitt 6.3.5). Es wird immer eine Anfrage nach bookinfo-Elementen gestartet, da diese auf Grund der enthaltenen Metadaten für den Aufbau des Suchergebnisses genutzt werden.

Bei Anwahl der Option *Kapitel anzeigen* (vgl. L03.3) wird neben der Anfrage nach bookinfo-Elementen ebenfalls eine Anfrage nach allen Titeln (title) der Kapitel (chapter) und Abschnitte (section) gestartet, in denen die Suchabfrage einen Treffer erzielt, sowie nach allen Kapiteln oder Abschnitten, in denen die Suchabfrage einen direkten Treffer erzielt. Durch die Baumstruktur von XML enthält jedes Kapitel einen Treffer, in dem ein Unterkapitel einen Treffer enthält. Direkter Treffer bedeutet, dass der Abschnitt selbst das Suchkriterium enthält und nicht ein in dem Abschnitt enthaltener Unterabschnitt. Diese beiden weiteren Abfragen sind nötig, um die Baumstruktur der einzelnen Abschnitte zu erhalten und einen direkten Treffer zu markieren.

## 6.3.2.2. Aufbereitung des Suchergebnisses

Als Ergebnis einer Anfrage bei der Tamino Datenbank wird ein XML-Dokument zurückgeliefert. Die Transformation des Ergebnisses in eine aussagekräftige Darstellung wird aufgrund der Kontextabhängigkeit der verschiedenen Elemente mit Hilfe einer XSLT-Transformation in eine JSP-Seite eingebunden. Das Ergebnis der Suchanfrage wird in der Session des Nutzers gespeichert, um ein schnelles Blättern in den Suchergebnissen zu gewährleisten, ohne die Suche erneut durchführen zu müssen. Die Nutzung des DOM zur Speicherung in der Session hat sich als sehr speicheraufwändig herausgestellt, daher wird ein `String` zusammengesetzt und für diese Zwecke verwendet. Ein Beispiel für das Ergebnis eine Suchanfrage im System, bei dem ebenfalls Kapitel angezeigt werden, ist in Beispiel 6.5. *Suchergebnis in XML* dargestellt.

### Beispiel 6.5. Suchergebnis in XML

```
<?xml version="1.0"?>
<book>
   <bookinfo ino:docname="book/testbook1" ino:id="5" lang="de"
             xmlns:ino="http://namespaces.softwareag.com/tamino/response2">
      <keywordset>
            <keyword>docbook</keyword>
      </keywordset>
      <author>
         <personname>
            <firstname>Marko</firstname>
            <surname>Petersen</surname>
         </personname>
      </author>
      <title lang="de">Testdokument</title>
      <digipubinfo>
         <digipubclient value="Testverlag"/>
         <digipubuser value="105"/>
         <digipubuploadtime value="18.01.2003"/>
         <digipublevel value="admin"/>
         <digipubclassification value="Book"/>
         <digipubprice value="1.0"/>
         <digipubchapter value="1"/>
         <digipubaccessible value="true"/>
         <digipubannotation value="true"/>
         <digipubvirtual value="true"/>
         <digipubfileextension value="DigiPub PDF"/>
         <digipubstyle value="/usr/local/docbook/stylesheet/fo.xsl"/>
      </digipubinfo>
   </bookinfo>
   <bookinfo ino:docname="book/rtzurtzu" ino:id="17"
             xmlns:ino="http://namespaces.softwareag.com/tamino/response2">
      <title>Testbild</title>
      <author>
         <firstname>Marko</firstname>
         <surname>Petersen</surname>
         <email>lg002237@rzserv2.fhnon.de</email>
      </author>
      <keywordset>
         <keyword>Suchwort</keyword>
      </keywordset>
      <digipubinfo>
         <digipubclient value="Testverlag"/>
         <digipubuser value="106"/>
         <digipubuploadtime value="21.01.2003"/>
         <digipublevel value="dependant"/>
         <digipubclassification value="Misc"/>
         <digipubprice value="0.0"/>
         <digipubchapter value="1"/>
         <digipubaccessible value="true"/>
         <digipubannotation value="false"/>
         <digipubvirtual value="false"/>
         <digipubfileextension value="JPG"/>
         <digipubstyle value=""/>
      </digipubinfo>
   </bookinfo>
   <title chapter_id="testbook1_N1001F" id="testbook1_N10021"
          ino:docname="book/testbook1" ino:id="5"
          parent_id="testbook1_N10001"
          xmlns:ino="http://namespaces.softwareag.com/tamino/response2"
          >Beipiel</title>
   <title id="testbook1_N10029" ino:docname="book/testbook1" ino:id="5"
         parent_id="testbook1_N1001F" section_id="testbook1_N10027"
         xmlns:ino="http://namespaces.softwareag.com/tamino/response2"
         >Beispiel 2</title>
   <section id="testbook1_N10027" ino:docname="book/testbook1" ino:id="5"
            xmlns:ino="http://namespaces.softwareag.com/tamino/response2"/>
</book>
```

Das Suchergebnis wird anhand des Stylesheets `DocSuche.xsl` zusammengesetzt. Hierbei werden sowohl sprachabhängige und anwendungsspezifische als auch sessionabhängige Werte als Parameter von der einbindenden JSP-Datei `docresult.jsp` übergeben. Für jedes gefundene Dokument existiert ein bookinfo-Element im Suchergebnis, welches zur Darstellung der allgemeinen Dokumenteninformationen genutzt wird. Der Link zum *Dokument sperren* wird hierbei nur generiert, wenn der anfragende Nutzer das Dokument selbst eingestellt oder die Rolle des Administrators innehat. Dem System-Administrator wird zusätzlich ein Link zum *Dokument löschen* angeboten. Der Preis des Dokumentes wird aus der angegebenen Kapitelanzahl und dem Kapitelpreis errechnet. Die Ausgabe des Suchergebnisses für Beispiel 6.5. *Suchergebnis in XML* ist in Abbildung 6.6. *Suchergebnis* dargestellt. Die farbliche Unterscheidung der verschiedenen Ergebnisse dient zur Identifikation des Nutzerlevels des Nutzers, der das Dokument eingestellt hat, in dem entsprechenden Mandanten.

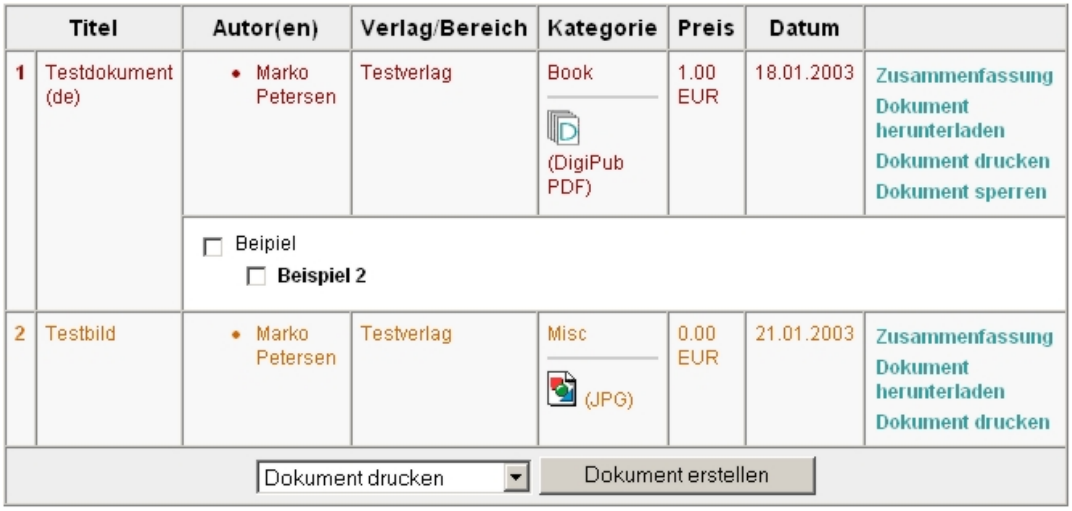


**Abbildung 6.6. Suchergebnis**

In Abbildung 6.6. *Suchergebnis* repräsentiert *Beispiel 2* einen direkten Treffer. Die Zuweisung der einzelnen Kapitel und Abschnitte zueinander sowie zu den bookinfo-Elementen geschieht über die eingefügten Attribute chapter_id, section_id und parent_id (siehe „Dokumentenupdate“). Die ID des jeweiligen Abschnittes wird hierbei als Wert für die Checkbox im HTML-Formular genutzt, um das Erstellen virtueller Dokumente zu ermöglichen. Anhand des value-Attributes des digipubfileextension-Elementes wird ein passendes Icon für die Dateiart des Dokumentes gewählt. Neben einigen selbst erstellten Symbolen wurden hierzu die vom Apache Webserver zur Verfügung gestellten und frei nutzbaren „Public Domain Icons“ verwendet.

## 6.3.3. Virtuelle Dokumente

Das DigiPub-DMS ermöglicht das Erstellen virtueller Dokumente aus der Anzeige eines Suchergebnisses (vgl. L04). Der Nutzer hat freie Wahl bei der Zusammenstellung der einzelnen Abschnitte für ein Dokument sowie die Wahl, das Dokument in gedruckter oder aber in elek-

tronischer Form zu erhalten (siehe Abbildung 6.6. *Suchergebnis*). Das Konzept für virtuelle Dokumente wurde bereits im LIVE-System entwickelt, musste jedoch für das DigiPub-DMS komplett neu umgesetzt werden.

Dokumente im LIVE-System basieren auf einer eigen entwickelten DTD, welche explizit auf die Erstellung von virtuellen Dokumenten abgestimmt wurde[69]. Sie besteht aus drei Hauptteilen: Ein Teil für Metaangaben (Element header), ein Teil für den Inhalt (Element content) und ein Teil für globale Referenzierungsziele im Dokument. Virtuelle Dokumente basieren auf dem Element chapter, welches rekursiv definiert ist und zwingend ein id-Attribut zur Referenzierung benötigt. Dieses ist das einzige Element, das direkt als Unterelement des Inhaltsteils content vorkommen kann. Eine eindeutige Referenzierung und Platzierung in einem neuen Dokument ist somit eindeutig gegeben, ebenfalls sind alle Metaangaben und globale Referenzen leicht erreichbar.

Die DocBook-DTD hingegen ist erheblich komplexer und bietet einem Autor weit mehr Freiheiten; ein Beispiel kann exemplarisch in Beispiel 6.6. *DocBook Element: chapter* eingesehen werden. Die Rekursivität von Kapiteln ist nicht gegeben, für diese Zwecke kann jedoch das Element section genutzt werden, welches dann ab der zweiten Ebene (also unterhalb eines Kapitels) rekursiv verwendet werden kann. Neben section bietet DocBook auch Elemente zur exakten Beschreibung der Abschnittstiefe (z.B. sect1), diese werden jedoch bei der Aufbereitung des Suchergebnisses und somit ebenfalls bei virtuellen Dokumenten nicht berücksichtigt. Ein section-Element kann jedoch nicht auf derselben Ebene vorkommen wie ein chapter-Element. Deshalb muss zum Beispiel für ein virtuelles Dokument, für das nur ein einziges section-Element gewählt wurde, für dieses Element ebenfalls ein chapter-Element als Elternelement zur Verfügung gestellt werden. Letzteres wurde allerdings nicht vom Nutzer ausgewählt. Um dieser Problematik vorzubeugen, werden alle gewählten section-Elemente, für die kein Kapitel gewählt wurde, in chapter-Elemente umtransformiert. Dies ist möglich, da das section-Element eine Teilmenge des Elementes chapter enthalten kann und das einzige weitere sich unterscheidende Element das Element chapterinfo für ein Kapitel und sectioninfo für einen Abschnitt ist. Da deren Struktur absolut identisch definiert ist, wird auch dieses Element angepasst. Dieser Anpassungsvorgang wird durch eine XSLT-Transformation mithilfe des Stylesheets `VirtualUpdate.xsl` vollzogen.

Um weitere Informationen zu den Dokumenten zu erhalten, aus denen die Kapitel stammen, wird eine Datenbankabfrage durchgeführt, die neben den einzelnen gewählten Kapiteln und Abschnitten ebenfalls das Element bookinfo für Metainformationen und die Elemente index, glossary und bibliography für weitere Dokumenteninformationen mitliefert. Hierdurch können Informationen zum gesamten Dokument mit in das virtuelle Dokument generiert werden. Sollten in dem neuen Dokument Referenzen auf fehlende Passagen, z.B. nicht gewählte Kapitel des Dokumentes, enthalten sein, so werden diese bei der Transformation in ein PDF-Dokument mit Fragezeichen ersetzt, da sie nicht auflösbar sind.

Zur Transformation in ein neues DocBook-Dokument kann die Komplexität der DTD jedoch genutzt werden: Hierfür dient ein set-Element, welches für jedes angefragte Dokument mit

[69] Die LIVE-DTD ist erreichbar unter http://live.fhnon.de:8080/xmldb/xml_stuff/liveskripte.dtd (Zugriff: 10.01.2003)

Kapiteln und Abschnitten ein book-Element mit den Informationen enthält. Dieses neue DocBook-Dokument kann nun wieder wie ein normales Dokument mit Hilfe von FOP und den DocBook XSL-Stylesheets zu einem PDF-Dokument gewandelt werden. Nach der Transformation wird die PDF-Datei im Dateisystem abgelegt, da vor der Auslieferung erst überprüft werden muss, ob es sich um ein kostenpflichtiges Dokument handelt.

### 6.3.4. Dokumentenabruf

Das DigiPub-System ermöglicht die Auslieferung von binären Dokumenten nur in elektronischer Form, die von DocBook-Dokumenten nur im PDF-Format oder in gedruckter Form. Durch die Verwendung verschiedener XSL-Stylesheets wäre eine Auslieferung von DocBook-Dokumenten in weiteren Formaten wie zum Beispiel HTML möglich. Im Rahmen des DPS-Projektes wurde jedoch aufgrund der Unvereinbarkeit mit dem Sicherheitskonzept auf diese Formate verzichtet. HTML-Dokumente beispielsweise sind Textdateien, für die Sicherheitsaspekte wie die Sperrung der Druckfunktion oder aber das Verbot des Kopierens von Text nicht ausreichend implementiert werden können. Ein Versuch, diese Funktionalitäten für HTML-Inhalte zu sperren und den Quelltext so zu verschlüsseln, dass er für einen Nutzer unlesbar ist, kann bedingt erreicht werden. Der Entschlüsselungsalgorithmus hingegen muss vom Browser ausgeführt und somit auch abgerufen werden können. Er muss frei zugänglich sein und ist dadurch von einem versierten Programmierer wieder entschlüsselbar. Eine Implementation eines solchen Verschlüsselungsverfahrens wurde ebenfalls während der Entwicklung des DigiPub-DMS implementiert, allerdings aufgrund der unzureichenden Sicherheit wieder ausgegliedert[70].

Um für kostenpflichtige Dokumente und Druckkosten einen Preis zu berechnen, muss die Auslieferung der Dokumente im DigiPub-System über das Zahlungssystem erfolgen. Als Schnittstelle für die Auslieferung werden die nötigen Informationen für das jeweilige Dokument in einem `DocumentInformation`-Objekt gespeichert. Für virtuelle Dokumente wird für jedes Dokument, aus dem Teile enthalten sind, ein dementsprechendes Objekt erstellt, welches in diesem Fall ebenfalls Angaben über die jeweils eingebundenen Kapitel und Abschnitte enthält. Diese Objekte werden in einem `Vector` hinterlegt und zusammen mit einem `String`, der den Pfad zu der abzurufenden Datei enthält, als Attribute in den `HttpServletRequest` gesetzt und an eine JSP-Seite weitergereicht. Anhand dieser Angaben kann eine Entscheidung getroffen werden, ob das Dokument zahlungspflichtig ist oder nicht. Für das weitere Vorgehen wird auf die Diplomarbeit von Stephan Wiesner verwiesen.

Neben dem Abruf kompletter Dokumente ist der Abruf einer Zusammenfassung über das Dokument möglich. Hierzu wird mithilfe der DocBook XSL-Stylesheets ein HTML-Dokument generiert, welches neben der Zusammenfassung das Inhaltsverzeichnis enthält. Abrufbar ist die Zusammenfassung ebenfalls im Suchergebnis (siehe Abbildung 6.6. *Suchergebnis*).

[70] Ein Webinterface zum Testen des Verschlüsselungsalgorithmusses ist erreichbar unter http://bitset.dyndns.org:8080/html-encoder/index.jsp (Zugriff: 18.01.2003)

### 6.3.5. Dokumentenverwaltung

Das Löschen von Dokumenten aus dem System kann unter Umständen zu einer rechtlichen Frage werden. Ein Dokument, welches Annotationen enthält, kann nicht nur geistiges Eigentum des Autors, sondern auch eines anderen Nutzers enthalten. Aus diesem Grund wurde die Funktionalität zum Löschen von Dokumenten auf Benutzer mit der Rolle System-Administrator oder Inhalts-Administrator beschränkt. Mindestens diese Personen müssen diese Möglichkeit erhalten. Um dennoch die Erreichbarkeit eines Dokumentes beeinflussen zu können, erhalten Nutzer die Möglichkeit, ein Dokument für eine Suchabfrage zu sperren. Diese Dokumente sind dann nur für die Nutzer zugänglich, von denen sie in das System eingestellt worden sind oder welche die Rolle System-Administrator oder Inhalts-Administrator innehaben.

## 6.4. Annotationen

Als besonderes Feature bietet das DigiPub-DMS die Möglichkeit, Dokumente im System zu annotieren (vgl. L06). In diesem Kapitel wird beschrieben, wie dieser Aspekt realisiert ist und welche Problemstellungen hierbei auftraten.

Durch die Möglichkeit, Dokumente im System annotieren zu können, wird eine Interaktionsmöglichkeit zwischen dem Autor und dem Leser sowie den Lesern untereinander geschaffen. Ein Beispiel für eine solche Annotationsmöglichkeit ist unter Abschnitt 3.5 zu finden, das Zope Book. In dem Zope Book ist es möglich, eine Annotation in das vorhandene Dokument einzufügen. Nachteilig ist bei diesem Konzept jedoch, dass die Annotationen nur in der als HTML konvertierten Onlineversion des Dokumentes auftauchen. Die zum Download angebotene PDF-Version des Dokumentes enthält diese Annotationen nicht.

Das DigiPub-DMS bietet die Möglichkeit, Dokumente, die nach der DocBook-DTD erstellt wurden und in denen die Möglichkeit des Annotierens beim Einstellen in das System erlaubt wurde (siehe Abschnitt 6.3.1), direkt auf Kapitel- (chapter-Element) oder Abschnittsebene (section-Element) zu annotieren oder auch eine Annotation zu einer bestehenden Annotation vorzunehmen. Diese Dokumente erhalten beim Einstellen in das System einen Link, der auf das DigiPub-DMS verweist und dort einem angemeldeten Benutzer über ein Formular die Möglichkeit gibt, seine Anmerkungen einzutragen. Der annotierte Text wird hierbei direkt als XML-Knoten in das entsprechende Dokument eingefügt und das Dokument in der XML-Datenbank aktualisiert. Dies bietet den Vorteil, dass die Annotation direkt als Dokumenteninhalt angesehen wird. Sie wird somit bei zukünftigen Suchabfragen oder Transformationen mit berücksichtigt.

Nach erfolgreicher Annotation wird das dem Dokument zugehörige PDF-Dokument neu erstellt, sodass auch dort die Annotation enthalten ist. Abbildung 6.7. *Annotation vornehmen* veranschaulicht den Vorgang dieser Aktion.

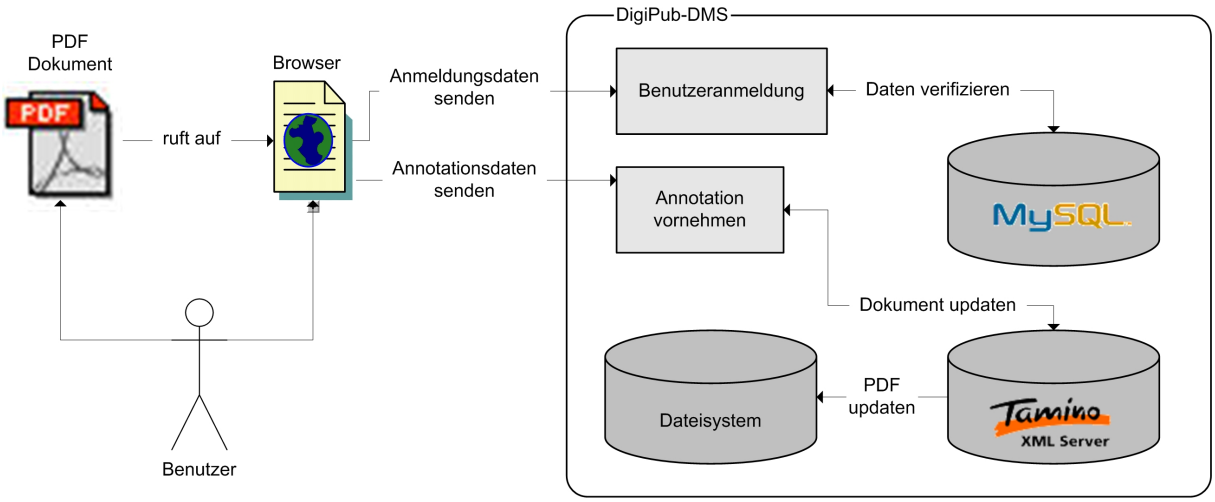


**Abbildung 6.7. Annotation vornehmen**

Das annotierte Dokument muss bei diesem Vorgang ein valides DocBook-Dokument bleiben. Zum gegenwärtigen Zeitpunkt existiert kein auf Annotationen spezialisiertes Element in DocBook, eine entsprechende Anfrage existiert aber bereits im „DocBook Open Repository“, Request ID 574880 vom 27.06.2002 [71]. Als Ersatz für dieses fehlende Element wurde als Wurzelelement für Annotationen das para-Element gewählt. Dieses kann rekursiv verschachtelt werden und eignet sich somit zum Annotieren von bestehenden Annotationen. Annotationen können dadurch theoretisch nur für Elemente vorgenommen werden, die ein para-Element enthalten können. Im DPS-Projekt wurde jedoch entschieden, Annotationen nur auf Kapitel- und Abschnittsebene zuzulassen. Hierdurch wird der Fluss des eigentlichen Inhaltes durch neu eingefügte Annotationen nicht auseinander gerissen. Die Elemente chapter und section können ein para-Element enthalten.

Als Einstieg für Annotationen muss zunächst ein para-Element in das Dokument eingefügt werden, um eine Referenz auf den zu annotierenden Abschnitt zu bilden. Gleichzeitig muss ein Link in das Dokument eingefügt werden, der auf das DigiPub-DMS verweist. Dieses Element soll vorzugsweise nach dem Inhalt eines Kapitels oder Abschnittes angezeigt werden, jedoch vor Beginn eines Unterabschnittes. Die exakte Platzierung dieses Elementes in einem Kapitel oder einem Abschnitt wird jedoch durch die Komplexität der DocBook-DTD erschwert, da sie dem Autor großen gestalterischen Freiraum bietet.

Die Problematik wird im Folgenden exemplarisch anhand des chapter-Elementes erläutert, trifft aber ebenso auf das section-Element zu. Der nachfolgende Quelltext ist ein Auszug aus der DocBook-DTD und definiert das chapter-Element.

[71] Siehe http://sourceforge.net/tracker/index.php?func=detail&aid=574880&group_id=21935&atid=384107 (Zugriff 16.12.2002)