FONDSDOC - BEREICH "ERSTELLUNG UND ÜBERARBEITUNG VON RECHTLICH RELEVANTEN TEXTEN"

eingereicht von:

Karin Petter

DIPLOMARBEIT

zur Erlangung des akademischen Grades
Diplom Ingenieur (FH)
(Dipl. Ing. (FH))

Fachhochschule St. Pölten

Studienrichtung: Telekommunikation und Medien

Begutachter:
Grischa Schmiedl

St. Pölten, Herzogenburgerstr. 68, im Mai 2004

Ehrenwörtliche Erklärung

Abstract

The amount of documents necessary to manage funds at Bank Gutmann KAG is ever increasing. The solution proposed is an application which automizes the generation of these documents.

All relevant techniques are explored, and documents are analyzed for their suitability for automation. In order to be suitable, documents must follow a general structure which could be stored by the application. The application then creates individual documents based on these general structures and on specific data. Neccessary techniques are XML, extensible stylesheet langauges (XSL), storage systems and query languages.

Based on the analysis, several possibilities for implementing such an application were found. The most qualified solution is explained in detail:

The general structure is described in XML. XML is used to store these structures because it is more flexible than other techniques. The application turns general structures into specific XML-documents which contain all necessary information to create a PDF as final output.

The transformation of XML-Files into PDF-Format is done by XSLT- und XSL-FO-Processors.

Kurzfassung

Diese Diplomarbeit beschäftigt sich mit der automatisierten Generierung von Dokumenten. Dabei handelt es sich und rechtlich relevante Dokumente die im Zuge der Verwaltung von Fonds in der Bank Gutmann AG erstellt werden müssen.

Es wird untersucht, welche Techniken nötig sind um eine Applikation zu erstellen, die alle Anforderungen erfüllt um eine Dokumentengenerierung durchzuführen.

Im ersten Teil der Arbeit werden verschiedene Techniken die zum Erstellen einer Anwendung, die die oben genannte Dokumente automatisiert erzeugt, erläutert. Solche Techniken sind unter anderen die Verarbeitung von XML, XSL-Transformationen, PDF-Generierung mit FOP und verschiedene Methoden zur Speicherung von strukturierten Daten. Es wird festgestellt, welche Techniken sich am Besten für die speziellen Anforderungen der Applikation eignen.

Anschließend werden, im zweiten Teil, sämtliche Dokumentarten analysiert um festzustellen ob sie sich genug verallgemeinern lassen um mit der Anwendung generiert zu werden. Anschließend werden genauere Vorgaben für die Anwendung, wie notwendige Funktionen, gemacht. Darauf basierend lassen sich verschiedene Umsetzungsvarianten finden.

Abschließend wird, im letzten Teil, auf Basis der am Besten geeigneten Umsetzungsvariante, die Anwendung detailiert geschildert. Die Grundlage der Anwendung sind allgemeine Schemata, die für die Dokumente festgelegt werden können. Die Speicherung dieser Schemata erfolgt als XML-Dateien. XML wurde als Speichermethode gewählt, weil damit eine sehr flexible Darstellung von Dokumentstrukturen möglich ist.
Um ein Dokument zu generieren erzeugt die Anwendung eine XML-Datei aus externen Datenquellen und auf Basis des allgemeinen Schemas. Aus dieser XML-Datei wird mit XSLT- und XSL-FO-Prozessoren ein PDF generiert.

Inhaltsverzeichnis

Ehrenwörtliche Erklärung **ii**

Abstract **iii**

Kurzfassung **iv**

Abbildungsverzeichnis **x**

Tabellenverzeichnis **x**

1 Einleitung **1**
1.1 Erkenntnisgegenstand 1
1.2 Forschungsproblem 1
1.3 Forschungsmethoden 1

2 Definition des Erkenntnisgegenstands **2**

3 Problembenennung **3**
3.1 Was wird als Problem erkannt 3
3.2 Darstellung des Problems 3
3.3 Theoretische Reflexion 4

4 Forschungsleitende Fragestellung **5**
4.1 Strukturdefinitionen der Dokumente 5
4.2 Speicherung fixer Textbausteine und dynamischer FondsDocPlatzhalter . . . 5
4.3 Parallele Änderung mehrerer Dokumente 5
4.4 Erhaltung einer korrekten Grammatik 6

I Theoretischer und Technischer Hintergrund **7**

5 Techniken zur Daten- und Hierarchieabspeicherung **8**

6 (Objekt)Relationale Datenbanken und XML-Datenbanken **9**
6.1 Auswahl einer geeigneten Datenbank 9
6.1.1 Charakter der Daten 9
6.1.2 Art der Anfragen 10
6.1.3 Einsatzzweck 10

6.1.4 Datenintegrität . . . 10
6.1.5 Mehrbenutzerbetrieb . . . 10
6.1.6 Schemaevolution . . . 10

7 XML **11**
7.1 Vorteile von XML . . . 11
7.2 Erstellen einer XML Dokumentes . . . 12
7.2.1 Aufbau eines XML-Dokuments . . . 12
7.2.2 XML-Syntax . . . 12
7.2.3 Well Formed . . . 13
7.2.4 Valid . . . 13
7.2.5 Namespaces . . . 14
7.3 Verwendung von Schemasprachen . . . 14
7.3.1 DTD . . . 14
7.3.2 Elementtypen . . . 14
7.3.3 Attribute . . . 14
7.3.4 Entity-Deklartationen . . . 15
7.3.5 XML Schema . . . 15
7.3.6 Weitere Methoden zur Schemabeschreibung . . . 15
7.3.7 Schematron . . . 16
7.3.8 Relax NG . . . 16
7.3.9 XDR . . . 16

8 XML-Formatierung **17**
8.1 XSLT . . . 17
8.1.1 Aufbau eines XSLT-Stylesheets . . . 18
8.2 XSL-FO . . . 19
8.2.1 Aufbau eines XSL-FO Stylesheets . . . 19
8.2.2 Features von XSL-FO . . . 20
8.3 XPath . . . 21
8.3.1 Achse . . . 21
8.3.2 Knotentest . . . 21
8.3.3 Prädikate . . . 21

9 XML-Verarbeitung mit PHP **23**
9.1 SimpleXML-Erweiterung . . . 23
9.2 Simple API for XML (SAX) . . . 23
9.3 DOM-Erweiterung . . . 23
9.4 EXKURS: DOM . . . 24
9.4.1 Node . . . 24
9.4.2 Element . . . 24
9.4.3 Attribute . . . 24
9.4.4 CharacterData . . . 24

10 Speicherung von XML **25**
10.1 Speicherung als Ganzes . . . 25
10.1.1 Volltextindex . . . 25
10.1.2 Kombinierter Volltext- und Strukturindex . . . 25
10.2 Speicherung der Dokumentstruktur . . . 26
10.2.1 Abbildung der Graphenstruktur . . . 26
10.2.2 Abbildung von DOM-Strukturen . . . 26
10.3 Abbildung der Dokumentstrukturen auf Datenbankstrukturen . . . 26
10.3.1 Automatische Speicherung . . . 26
10.3.2 Mapping Vorschriften . . . 27

11 Auswahl geeigneter Umsetzungstechniken für das Projekt FondsDoc **28**

II Analyse und Konzeption **29**

12 Analyse der zu generierenden Dokumente **30**
12.1 Überblick über die Dokumente . . . 30
12.1.1 Liste der Dokumente . . . 30
12.1.2 Abhängigkeiten zwischen den Dokumenten . . . 31

13 Anforderungen an die Applikation **33**
13.1 Erfassen und Editieren der Dokumente . . . 33
13.1.1 Allgemeine Schemata . . . 33
13.1.2 Absätze . . . 33
13.1.3 Fond-Spezifische Absätze . . . 33
13.1.4 Platzhalter . . . 34
13.1.5 Formatierungsdefinitionen . . . 34
13.1.6 Grafiken . . . 34
13.2 Die Dokumentengenerierung . . . 35
13.2.1 FondsDocSchema, der allgemeine Aufbau des Dokumentes . . . 35
13.2.2 Das spezifische Dokument . . . 36
13.2.3 Das Endprodukt . . . 36

14 Umsetzungsstrategie **39**
14.1 Grundlegende Komponenten . . . 39
14.1.1 Model: FondsDocSchemata . . . 39
14.1.2 View: Darstellung der Schemata im Benutzer Interface . . . 39
14.1.3 Controller: FondsDocSchemas editieren und PDF-Erzeugung . . . 39
14.2 Variante 1: Datenbankbasiert . . . 40
14.2.1 Speicherung der Schemata . . . 40
14.2.2 Editieren der Schemata . . . 40
14.2.3 PDF Generierung . . . 41
14.2.4 Vor- und Nachteile der Variante . . . 41
14.3 Variante 2: Rein XML-Basiert . . . 41
14.3.1 Speicherung der Schemata . . . 41
14.3.2 Editieren der Schemata . . . 41
14.3.3 PDF Generierung . . . 42

14.3.4 Vor- und Nachteile der Variante 42
14.4 Variante 3: XML-Basiert, als Erweiterung der PHPLib 43
14.4.1 Speicherung der Schemata 43
14.4.2 Editieren der Schemata 43
14.4.3 PDF Generierung 43
14.4.4 Vor- und Nachteile der Variante 43
14.5 Variante 4: Oracle XML-Output 43
14.6 Auswahl einer Variante 44
14.6.1 Entscheidende Vorteile 44
14.6.2 Weitere Beweggründe 44

15 Die Entwicklungsumgebung **45**
15.1 Linux 45
15.2 PHP 45
15.3 Verwendung und Erweiterung der PHPLib(intern erstelltes Framework) . . . 45
15.3.1 Aufbau und Konzept der PHPLib 45
15.3.2 Verwendung der PHPLib im Projekt FondsDoc 46
15.4 FondsDB 46
15.5 Browser 46

16 Funktionen des User-Interfaces **47**
16.1 Dokumentstrukturen überarbeiten 47
16.1.1 Absätze 47
16.1.2 FondsDocPlatzhalter 47
16.1.3 FondsDocConstraints 47
16.2 Dokumentstrukturen von Bestehenden Strukturen ableiten 48
16.3 Neue Dokumentstrukturen entwerfen 48

III Aufbau der Anwendung **51**

17 Model **52**
17.1 Die Fonds-Datenbank 52
17.2 Die XML-Dokumente mit den FondsDocSchemata 52

18 Controller **54**
18.1 Erklärungen der Klassen zum Generieren eines XML-Dokumentes 54
18.1.1 XML_Doc 54
18.1.2 XML_Node 55
18.1.3 XML_Attr 55

19 View **58**
19.1 Erklärungen zu den Klassen zum Aufbau des User-Interfaces 58
19.1.1 Formularklassen 58
19.1.2 Formularfeldklassen 58
19.2 User Interface zum Formatieren einzelner Absätze 59
19.2.1 Whizzywig 59
19.2.2 Das Interface von Whizzywig im Detail 59

19.2.3 Anpassungen von Whizzywig an das Projekt FondsDoc 60

20 PDF-Generierung **63**
20.1 Aufbau des XSL-FO Stylesheets 64
20.2 Aufbau des XSLT zur Erzeugung des XSL-FO 65
20.3 Erzeugung des PDFs aus XSL-FO 67

21 Reflexion **69**
21.1 Strukturdefinitionen der Dokumente 69
21.2 Dynamische Textbausteine 69
21.3 Parallele Änderung mehrerer Dokumente 70
21.4 Erhaltung der korrekten Grammatik 70

A Detailbeschreibungen der Dokumente **71**
A.0.1 (1) Depotbankschreiben 72
A.0.2 (2) Sammelaufstellung 74
A.0.3 (3) Sammelurkunde und Beilagebrief 75
A.0.4 (4) Produktbeschreibung 76
A.0.5 (5) FMA Antrag 77
A.0.6 (6) Informationen für Deutschland und Schweiz 78
A.0.7 (7) Bericht Fondsmanager 79
A.0.8 (8) Ertragsrechnung 80
A.0.9 (9) Organe der Ges. u. Depotbank 81
A.0.10 (10) KAG Fondsliste) 82
A.0.11 (11) Verkaufsprospekt 83
A.0.12 (12) Rechenschaftsbericht 84
A.0.13 (13) Halbjahresbericht 85
A.0.14 (14) Vermögensaufstellung 86
A.0.15 (15) Bestätigungsvermerk 87
A.0.16 (16) Erläuterungen zu Steuerliche Behandlung 88
A.0.17 (17) Vereinfachter Prospekt 89
A.0.18 (18) Allgemeine Fondsbestimmungen 90
A.0.19 (19) Besondere Fondsbestimmungen 91
A.0.20 (20) Anhang zu §16 - Börsenliste 92
A.0.21 (21) Steuerliche Behandlung 93
A.0.22 (22) Umlaufbeschluss 94
A.0.23 (23) Delegationsmeldung 95
A.0.24 (24) Anteilsinhaber Info 96
A.0.25 (25) Prüfbericht 97
A.0.26 (26) ÖKB Stammdatenblatt 98
A.0.27 (27) Veröffentlichungen 99
A.0.28 (28) Depotanlageblatt 100

Abbildungsverzeichnis

3.1 Überblick über die Anwendung „FondsDoc“ 4

8.1 Ablauf einer XSLT-Transformation 18
8.2 Erzeugung eines PDFs aus einer XML-Datei mit FOP 19

12.1 Dokumentenbaum Publikumsfonds 32
12.2 Dokumentenbaum Spezialfonds 32

13.1 Grafische Darstellung eines FondsDocSchemas 37
13.2 Schematische Darstellung der Dokumentengenerierung 38

14.1 PDF-Erzeugung mit PDFLib 41
14.2 PDF-Erzeugung mit XSL-FO 42

16.1 Interface 49
16.2 Neuen Absatz einfügen 50
16.3 Neuen Platzhalter einfügen 50
16.4 Neuen Constraint einfügen 50

18.1 Schematische Darstellung des Ablaufes der PDF-Generierung 56
18.2 Modell der für die Anwendung relevanten PHP Klassen 57

19.1 Schematische Darstellung der Änderung des Inhaltes eines Absatzes 61
19.2 Whizzywig Screenshot 62

20.1 Erzeugung eines PDFs aus einer XML-Datei mit FOP 63
20.2 Mit FOP generiertes PDF 68

Tabellenverzeichnis

A.1 Eigenschaften von (1) Depotbankschreiben 72
A.2 Abhängigkeiten von (1) Depotbankschreiben 73
A.3 Eigenschaften von (2) Sammelaufstellung 74
A.4 Abhängigkeiten von (2) Sammelaufstellung 74
A.5 Eigenschaften von (3) Sammelurkunde und Beilagebrief 75
A.6 Abhängigkeiten von (3) Sammelurkunde und Beilagebrief 75
A.7 Eigenschaften von (4) Produktbeschreibung 76
A.8 Abhängigkeiten von (4) Produktbeschreibung 76
A.9 Eigenschaften von (5) FMA Antrag 77
A.10 Abhängigkeiten von (5) FMA Antrag 77
A.11 Eigenschaften von (6) Informationen für Deutschland und Schweiz 78
A.12 Abhängigkeiten von (6) Informationen für Deutschland und Schweiz 78
A.13 Eigenschaften von (7) Bericht Fondsmanager 79
A.14 Abhängigkeiten von (7) Bericht Fondsmanager 79
A.15 Eigenschaften von (8) Ertragsrechnung 80
A.16 Abhängigkeiten von (8) Ertragsrechnung 80
A.17 Eigenschaften von (9) Organe der Ges. u. Depotbank 81
A.18 Abhängigkeiten von (9) Organe der Ges. u. Depotbank 81
A.19 Eigenschaften von (10) KAG Fondsliste 82
A.20 Abhängigkeiten von (10) KAG Fondsliste 82
A.21 Eigenschaften von (11) Verkaufsprospekt 83
A.22 Abhängigkeiten von (11) Verkaufsprospekt 83
A.23 Eigenschaften von (12) Rechenschaftsbericht 84
A.24 Abhängigkeiten von (12) Rechenschaftsbericht 84
A.25 Eigenschaften von (13) Halbjahresbericht 85
A.26 Abhängigkeiten von (13) Halbjahresbericht 85
A.27 Eigenschaften von (14) Vermögensaufstellung 86
A.28 Abhängigkeiten von (14) Vermögensaufstellung 86
A.29 Eigenschaften von (15) Bestätigungsvermerk 87
A.30 Abhängigkeiten von (15) Bestätigungsvermerk 87
A.31 Eigenschaften von (16) Erläuterungen zu Steuerliche Behandlung 88
A.32 Abhängigkeiten von (16) Erläuterungen zu Steuerliche Behandlung 88
A.33 Eigenschaften von (17) Vereinfachter Prospekt 89
A.34 Abhängigkeiten von (17) Vereinfachter Prospekt 89
A.35 (18) Allgemeine Fondsbestimmungen 90
A.36 (18) Allgemeine Fondsbestimmungen 90

A.37 (19) Besondere Fondsbestimmungen 91
A.38 (19) Besondere Fondsbestimmungen 91
A.39 (20) Anhang zu §16 - Börsenliste 92
A.40 (20) Anhang zu §16 - Börsenliste 92
A.41 (21) Steuerliche Behandlung 93
A.42 (21) Steuerliche Behandlung 93
A.43 (22) Umlaufbeschluss 94
A.44 (22) Umlaufbeschluss 94
A.45 (23) Delegationsmeldung 95
A.46 (23) Delegationsmeldung 95
A.47 (24) Anteilsinhaber Info 96
A.48 (24) Anteilsinhaber Info 96
A.49 (25) Prüfbericht 97
A.50 (25) Prüfbericht 97
A.51 (26) ÖKB Stammdatenblatt 98
A.52 (26) ÖKB Stammdatenblatt 98
A.53 (27) Veröffentlichungen 99
A.54 (27) Veröffentlichungen 99
A.55 (28) Depotanlageblatt 100
A.56 (28) Depotanlageblatt 100

Kapitel 1

Einleitung

1.1 Erkenntnisgegenstand

Im Zuge der Verwaltung von Investmentfonds müssen laufend eine Reihe rechtlich relevanter Dokumente erstellt werden. Um das zu automatisieren soll eine Applikation zu Generierung dieser Dokumente erstellt werden.

1.2 Forschungsproblem

Es wird nach einem Konzept zur automatischen Generierung von Dokumenten gesucht. Dazu gilt es festzustellen ob für alle Dokumentarten ein allgemeines Schema gefunden werden kann, das einer Applikation als Basis dient. Ist es möglich die Dokumente ausreichend zu verallgemeinern, so kann eine Applikation konzipiert werden. Es muss dafür eine Methode gefunden werden, wie allgemeine Schemata flexibel abgespeichert werden, sodass sie von den Benutzern der Applikation auch editiert werden können.
Schließlich soll ermittelt werden welche Anforderungen die Applikation erfüllen muss und wie diese realisiert werden.

1.3 Forschungsmethoden

Im Teil I der Arbeit werden sämtliche relevanten Techniken recherchiert und dargestellt, die in zum Erstellen Applikation verwendet werden können. Der Teil II ist eine Analyse sämtlicher Dokumente. Weiters wird hier der Funktionsumfang der Applikation ermittelt. Es werden verschiedene Möglichkeiten für die Umsetzung vorgeschlagen und auf Vor- und Nachteile untersucht. Die Umsetzungsmethode mit den meisten Vorteilen wird schließlich im Teil III detailiert erklärt.

Kapitel 2

Definition des Erkenntnisgegenstands

In der Bank Gutmann KAG werden über 100 Investmentfonds verwaltet. Immer wieder werden neue Fonds aufgelegt und somit steigt die Anzahl der zu verwaltenden Fonds ständig. Für jeden Fond müssen bei der Neuauflage etliche Dokumente erstellt werden: Beispielsweise Verkaufsprospekte oder Rechenschaftsberichte. Weiters müssen für alle bestehenden Fonds diverse Dokumente regelmäßig aktualisiert und neu erstellt werden. Hinzu kommen ständig neue gesetzliche Anforderungen die Änderungen und Ergänzungen in den Dokumenten erfordern.
Bisweilen gibt es keine technische Unterstützung in diesem, mit ca. 30 Prozent des Personalaufwandes sehr großen Bereich der Gutmann Kapitalanlagegesellschaft. Sämtliche Dokumente werden einzeln überarbeitet und ergänzt, was bei der steigenden Anzahl von Fonds zu einer immer größeren Fehlerquelle wird.
Um die Generierung und Überarbeitung der rechtlich relevanten Dokumente überschaubarer und einfacher zu gestalten soll im Zuge das Projektes „FondsDoc“ eine Anwendung erstellt werden mit der sich alle nötigen Arbeitsschritte vereinheitlichen und automatisieren lassen.
Ziel ist es trotz steigender Anzahl an verwalteten Fonds und gesetzlichen Anforderungen Neuanlagen, Berichtswesen (z.B. Rechenschaftsberichte) und Anfragen rascher und effizienter erledigen zu können.
Um dieses Ziel zu erreichen ist eine gründliche Analyse der Dokumente und deren Erstellung nötig. Weiters müssen sämtliche Datenquellen für fondspezifische Informationen und nötige Gesetzestexte analysiert werden. In Zusammenarbeit mit den verantwortlichen Mitarbeitern kann dann ein Konzept zur Umsetzung einer Anwendung ausgearbeitet werden. Diese Anwendung soll in Zukunft die Generierung, Neuerstellung und Überarbeitung aller benötigten Dokumente erleichtern und teilweise sogar vollständig automatisieren. Somit können Fehlerquellen vermieden und Arbeitsschritte erleichtert werden.

Kapitel 3

Problembenennung

3.1 Was wird als Problem erkannt

Der Bereich „Erstellung und Überarbeitung von rechtlich relevanten Texten“ stellt die analytisch größte Herausforderung dar. Hier sollen auf Basis von Fondsstammdaten (in einer DB) und Textvorlagen fertige Dokumente zur Einreichung bzw. Veröffentlichung generiert werden. Bei den erzeugten Dokumenten handelt es sich beispielsweise um Berichte und Reports zur Veröffentlichung aber auch um interne Schriftstücke. Weiters sollen, analog zu einem Versionskontrollsystem für Quellcode, Funktionen wie „diff“ und „merge“ das gleichzeitige Überarbeiten mehrere, ähnlicher Dokumente wesentlich erleichtern (z.B. Anpassung von Fondsbestimmungen nach Gesetzesänderungen). Das Produkt ist im allg. ein juristisch relevanter und geprüfter Text im PDF-Format, zur Einreichung an die Finanzmarktaufsicht und Veröffentlichung.

3.2 Darstellung des Problems

Es soll also ein Konzept zur Generierung von Dokumenten erstellt werden, das folgende Anforderungen erfüllt:

1. Sämtliche Dokumente sollen automatisch generiert werden und ggf. alle oder nur Gruppen mit bestimmten Kriterien (z.B. alle Dokumente mit einem bestimmten Erstellungsdatum) überarbeitet werden können. D.h. für jede Dokumentart soll ein Schema gefunden werden nach dem das Dokument erzeugt werden soll.

2. Dieses Schema setzt sich aus mehreren statischen oder dynamischen Textbausteinen zusammen für die es verschiedenen Datenquellen gibt: Einerseits die Datenbank mit den Fondsdaten, in der sämtliche aktuelle Informationen über die Fonds enthalten sind und andererseits eine Datenquelle aus der die immer gleich bleibenden Texte bezogen werden.

3. Die Schemata müssen veränderbar bleiben, damit nachträgliche Änderungen an den Dokumenten ermöglicht werden.

Folgende Abbildung soll die Aufgabenstellung verdeutlichen:

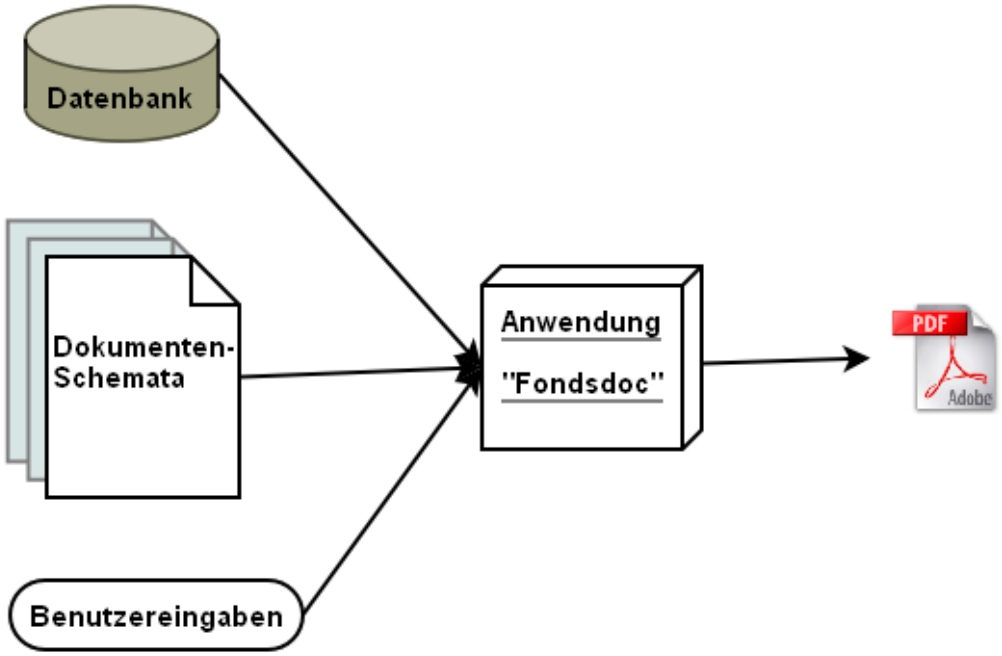

Abbildung 3.1: Überblick über die Anwendung „FondsDoc“

3.3 Theoretische Reflexion

Um all diese Anforderungen zu erfüllen, liegt es nahe sich einer XML-Struktur, eines ER-Models oder einer Kombination daraus zu bedienen, um FondsDocSchemata für die Dokumente zu definieren. Mehrere Ansätze zur Umsetzung werden durchgedacht und nach den Kriterien

- Umsetzungsaufwand
- Wartungsaufwand
- Erweiterbarkeit
- Kosten (Lizenzen für Software)
- Performance

verglichen. Die beste Umsetzungsmöglichkeit wird ausgewählt und zur Weiterentwicklung des Konzeptes verwendet.

Kapitel 4

Forschungsleitende Fragestellung

Hier werden Fragestellungen definiert die im Zuge der Diplomarbeit beantwortet werden. Das Ziel der Arbeit ist, eine Anwendung zu konzipieren mit der die Erstellung sämtlicher Dokumente, die in Zuge der Verwaltung und Neuauflage von Fonds, benötigt werden, möglich ist. Im Vorfeld wurden folgende Probleme erkannt, die es zu lösen gilt:

4.1 Strukturdefinitionen der Dokumente

Die Inhalte sämtlicher Dokumente ergeben sich aus sich ändernden oder fixen Fondstammdaten aus einer Datenbank. Es bestehen zusätzlich Beziehungen zwischen den Dokumenten. Daraus ergibt sich das Forschungsproblem den gesetzlich vorgeschriebenen Aufbau sämtlicher Dokumente genau zu definieren und daraus ein allgemeines FondsDocSchema zur Erzeugung zu ermitteln. Wie muss ein FondsDocSchema aussehen um die oben genanten Funktionen zu Ermöglichen?

4.2 Speicherung fixer Textbausteine und dynamischer FondsDocPlatzhalter

Da die Dokumente nicht nur einmalig erzeugt werden, sondern auch nachträglich geändert bzw. ergänzt werden sollen, gilt es die Dokumentenerzeugung möglichst dynamisch zu gestalten und eventuell eine Art Versionskontrolle zu implementieren. Folgende Forschungsfrage gilt es hier zu beantworten: Wie werden fix vorgegebene Textbausteine und FondsDocPlatzhalter am besten abgelegt um Texte dynamisch erzeugen, erweitern und ändern zu können. (wie z.B. in Datenbanken, als XML, XSL,...)

4.3 Parallele Änderung mehrerer Dokumente

Da häufig (z.B. bei Gesetzesänderungen) eine ganze Reihe von Dokumenten gleichzeitig geändert werden müssen, soll eine simultane Änderung von mehreren Dokumenten ermöglicht werden. Das führt zur Frage: Wie kann eine Änderung oder Ergänzung für eine ganze Reihe von Dokumenten möglichst einfach durchgeführt werden. Kann automatisch ermittelt werden welche Dokumente von einer Änderung betroffen sind?

4.4 Erhaltung einer korrekten Grammatik

Da die erstellten Dokumente offiziellen Charakter haben, an Bankenaufsicht und/oder Kunden versendet werden, ist eine gute und korrekte Formulierung von Bedeutung. Daraus ergibt sich die Frage: Wie kann sichergestellt werden, dass auch bei Änderungen über mehrere Dokumente hinweg die Grammatik korrekt bleibt.

Teil I

Theoretischer und Technischer Hintergrund

Kapitel 5

Techniken zur Daten- und Hierarchieabspeicherung

Um die in den Dokumenten vorkommenden Daten abzuspeichern bieten sich verschiedene Möglichkeiten. Manche bieten mehr Performance und Effizienz bei der Speicherung von großen Datenmengen. Andere eignen sich eher um vorgegebene Strukturen abzuspeichern, zu erweitern und zu rekonstruieren. Im Folgenden sollen benötigte Techniken kurz erläutert werden um einen Überblick über die komplexe Thematik zu vermitteln. Für die gesamte Umsetzung wird es sicher nötig sein, etliche Techniken zu kombinieren. Dabei muss auch immer bedacht werden, ob es geeignete Schnittstellen gibt, die eine möglichst stabile und rasche Interaktion zulassen.
Diese Aufzählung soll keine umfassende Erläuterung sein. Es wäre nicht möglich im Rahmen dieser Diplomarbeit sämtliche Techniken zur Speicherung von Daten und zur Generierung von Dokumenten detailliert zu beschreiben. Vielmehr soll, für ein besseres Verständnis des Konzeptes zur Dokumentengenerierung, ein Überblick über alle Technischen Möglichkeiten gegeben werden. Bei der Beschreibung der einzelnen Technologien werden vor allem für das Projekt FondsDoc wichtige Aspekte herausgearbeitet.

Kapitel 6

(Objekt)Relationale Datenbanken und XML-Datenbanken

Als Basis der Applikation FondsDoc ist eine geeignete Datenbank nötig um die FondsDoc-Schemata, abzuspeichern. Auch ein Abspeichern als Textfiles im Filesystem wäre denkbar, mit Hilfe einer Datenbank ist aber eine Verwaltung von vielen FondsDocSchemata sicherlich einfacher. In die Überlegungen mit einbezogen werden muss sicherlich auch die Entscheidung ob die FondsDocSchemata als ganzes, mit allen fixen Textbestandteilen gespeichert werden sollen, oder ob es sinnvoller ist die einzelnen Bestandteile der Schemas als getrennte Einheiten abzuspeichern.
Es gibt verschiedene Datenbanksysteme die Unterschiedliche Vor- und Nachteile haben. Die Auswahl der richtigen Datenbank ist also wichtig.

6.1 Auswahl einer geeigneten Datenbank

Die gebräuchlichste Methode um verknüpfte Datenstrukturen abzubilden sind sicherlich Relationale Datenbanken. Aber in manchen Fällen können XML-Datenbanken Vorteile bringen. Wichtig für die Entscheidung wie Daten am besten abgespeichert werden ist vor allem die Art der Daten und die Art wie auf diese Daten zugegriffen wird. Folgende Punkte können helfen eine Entscheidung für eine geeignete Datenbank zu treffen:

6.1.1 Charakter der Daten

Dies ist einer der ausschlaggebenden Punkte. Sind die Daten eindeutig und durchgehend strukturiert, ist es sinnvoll ein relationales oder objektrelationales Datenbanksystem zu verwenden.
Liegen Daten aber unstrukturiert vor, sind also wirkliche Dokumente, lassen sie sich in XML-Datenbanken wesentlich leichter Abbilden. unstrukturierte Daten lassen sich häufig als Baumstruktur darstellen. (z.B. Buch: Es besteht aus einzelnen Kapiteln, die wiederum gliedern sich in einzelne Absätze von denen jeder wieder Text, Grafiken oder Tabellen enthalten kann. Auch XML-Dokumente weisen eine Baumstruktur auf. Also lassen sich unstrukturierte Daten oft ohne weitere Umformung als XML darstellen, während eine Speicherung in eine (Objekt-)Relationale Datenbank, ohne Umformung der Daten und dem Erstellen einer, eventuell sehr komplexen, Datenbankarchitektur, nicht möglich ist. Vor allem wenn für Daten kein oder nur

ein grobes Strukturschema angegeben werden kann, ist eine Verwendung von relationalen Datenbanken schlecht möglich, da eine relationale Datenbank immer eine Vorgegebene Anzahl miteinander verknüpften Tabellen hat die für unstrukturierte Daten oft nicht dynamisch genug sein können.

6.1.2 Art der Anfragen

Wenn sich Abfragen oder Update-Operationen gut in SQL formulieren lassen, ist einer relationalen Datenbank der Vorzug zu geben.
Werden aber semistrukturierte oder rekursive Daten verarbeitet, sind native XML-Datenbanken sinnvoller.

6.1.3 Einsatzzweck

Wird XML nur als Zwischenstufe zur Präsentation oder zum Datenaustausch eingesetzt, ist es nicht erforderlich eine XML-Datenbank einzusetzen. Sollen XML-Dateien jedoch dauerhaft gespeichert werden und ist auch ein späterer Zugriff in Form von Abfragen oder Update-Operationen nötig, ist eine XML-Datenbank bestens dafür geeignet.

6.1.4 Datenintegrität

Standardmäßig implementiert keine XML-Datenbank referentielle Integritäten. Werden an Dokumenten Änderungen vorgenommen, ist es erforderlich, dass alle Abhängigkeiten händisch implementiert werden. Kommen solche Änderungen häufig oder in großem Ausmaß vor, ist der Einsatz einer XML-Datenbank nicht sinnvoll.

6.1.5 Mehrbenutzerbetrieb

Fein strukturierte Sperren und dadurch ein gutes Transaktionsverhalten können in (objekt)relationalen Datenbanken besser umgesetzt werden. XML-Datenbanken eignen sich daher schlecht für Applikationen mit Mehrbenutzerbetrieb.

6.1.6 Schemaevolution

Ändert sich das Datenschema, ist eine XML-Datenbank, die ohne Schema auskommt, vorteilhaft. Auch für den Fall dass ein vorgegebenes Schema in der XML-Datenbank besteht, ist es recht einfach, dieses Schema mit einer XSL-Transformation zu ändern.
Eine Änderung des Schemas in (objekt)relationalen Datenbanken gestaltet sich dagegen oft aufwendiger.
[See03, Seite 139ff.]

Kapitel 7

XML

7.1 Vorteile von XML

Es gibt eine Reihe von Gründen warum XML in einer Applikation Anwendung finden kann. Hier einige der für das Projekt FondsDoc relevanten Gründe:

- Bereits bewährte Syntax:
  Die XML Syntax ist mittlerweile weit verbreitet und es gibt eine Reihe von Formatierungsmöglichkeiten dafür die leicht eingebunden werden können.

- Kann als Schnittstelle zu diversen Anwendungen dienen:
  XML hat sich als Schnittstelle zwischen unterschiedlichen Anwendungen bewährt. Der Austausch der erzeugten Dokumente, mit allen darin enthaltenen Informationen, kann somit auch formatunabhängig passieren, was sicher einen großen Vorteil für die Weiterentwicklung der Anwendung in Zukunft bietet.

- Bestehende Möglichkeiten zur Abfrage, Erweiterung (X-PATH, XSLT...):
  Für XML gibt es eine ganze Reihe von Abfragesprachen und Formatierungssprachen, die einen großen Funktionsumfang haben, wodurch eine sehr flexible Anpassung der FondsDocSchemata ermöglicht wird.

- Kann validiert werden:
  XML ist ein Standart und es gibt Tools zu Validierung. Damit kann bei Weiterverarbeitung der erzeugten Daten gewährleistet werden, dass diese eindeutig und fehlerfrei interpretiert werden können.

- Erhaltung des Dokumentenschemas bei der Speicherung:
  Wurde für ein Dokument ein FondsDocSchema erstellt, so kann es mit XML ohne weitere Umformung gespeichert werden. Da eine Umformung sicherlich immer eine Fehlerquelle darstellt ist das ein großer Vorteil.

- Es gibt bereits eine große Anzahl an vorhandenen Tools und Libs um XML zu verarbeiten.

7.2 Erstellen einer XML Dokumentes

XML ist eine Auszeichnungssprache zu Beschreibung von Dokumenten in Form hierarchisch strukturierter Bäume. XML ist ein W3C Standard.

7.2.1 Aufbau eines XML-Dokuments

Wie bereits erwähnt weisen XML-Dokumente eine Baumstruktur auf. Es genau ein Element auf der obersten Ebene geben, darunter können weitere frei definierbare Elemente verschachtelt werden.
Zusätzlich zu dieser Baumstruktur kann optional am Beginn des Dokumentes eine XML-Deklaration angegeben werden. Darin können Zusatzinformationen wie Version (die Versionsnummer der verwendeten XML-Spezifikation nach W3C) und Encoding (die verwendete Zeichenkodierung) angegeben werden.
Ebenfalls optional kann auch eine DTD (siehe Kapitel 7.3.1, Seite 14) angegeben werden um die Struktur des Dokumentes genauer festzulegen.

7.2.2 XML-Syntax

Die XML-Syntax ist an sich relativ einfach. Es gibt verschiedene Teile aus denen sich der XML-Baum zusammensetzt:

Elemente

Die XML-Elemente sind Baumknoten die durch ein, namentlich gleiches, `<Start-Tag>` und ein `</End-Tag>` die einen Inhalt eingrenzen oder einfach ein leeres `<Tag />` ohne Inhalt. Der Inhalt ist einfacher Text in Form einer Zeichenkette.

Attribute

Elemente können Attribute enthalten. Jedes Attribut besteht aus einem Schlüssel-Wert-Paar der `Attributname="Attributwert"`.

Kommentare

Kommentare in einem XML-Dokument werden folgendermaßen notiert: `<!-- Kommentar-Text -->`.

Text

Text kann ganz normal als Zeichenkette in ein XML-Dokument eingefügt werden. Allerdings muss man beachten, dass Sonderzeichen wie `<, > und &` zu Parserfehlern führen. Um diese Sonderzeichen trotzdem verwenden zu können, muss der Text innerhalb eines CDATA-Abschnitts [1] notiert werden. Innerhalb eines solchen Abschnittes werden Sonderzeichen vom Parser nicht interpretiert. Ein CDATA-Abschnitt wird folgendermaßen notiert: `<![CDATA[ Inhalt ]]>`.

Beispiel eines XML-Dokuments:

[1] CDATA steht für Character Data, also Zeichendaten

```
<?xml version="1.0" encoding="UTF-8"?>
<enzyklopädie stand="1.1.2006">
     <titel>Städteverzeichnis</titel>
     <!--Kommentar: ... -->
     <eintrag>
          <stichwort>Wien</stichwort>
          <eintragstext>Wien ist...</eintragstext>
     </eintrag>
     <eintrag>
          <stichwort>Linz</stichwort>
          <eintragstext>
           <![CDATA[Hier können Sonderzeichen notiert werden: ><& ]]>
          </eintragstext>
     </eintrag>
</enzyklopädie>
```

Dieser einfache Aufbau ist die Grundlage für ein XML-Dokument. [Wik06b,]

Weiters gibt es noch einige andere Anforderungen die ein XML-Dokument erfüllen muss damit es sinnvoll in einer Applikation eingesetzt werden kann.

7.2.3 Well Formed

„Well-formed“ ist eine wichtige Eigenschaft eines XML-Dokumentes. Sie sagt aus, dass die Syntax von XML vollständig eingehalten wird. Es muss beispielsweise für jedes öffnende Element (z. B. `"<item>"`) auf gleicher hierarchischer Ebene ein schließendes Element vorhanden sein (z. B. `"</item>"`).

7.2.4 Valid

Entspricht ein XML-Dokument einer DTD bezeichnet man es als „valid“. In einer DTD (siehe Kapitel 7.3.1, Seite 14) wird deklariert welche Elemente und Attribute in einem Dokument erlaubt sind.

Beispiel einer DTD:

```
<!ELEMENT people_list (person*)>
<!ELEMENT person (name, birthdate?, gender?, socialsecuritynumber?)>
<!ELEMENT name (#PCDATA) >
<!ELEMENT birthdate (#PCDATA) >
<!ELEMENT gender (#PCDATA) >
<!ELEMENT socialsecuritynumber (#PCDATA) >
```

[Wik06a,]
Verwendung der DTD erfolgt über die Referenz

```
<!DOCTYPE people_list SYSTEM "example.dtd">
```

am Dokumentanfang.

7.2.5 Namespaces

Da in einem Dokument durchaus auf mehrere DTD verwiesen werden darf, kann es zu Überschneidungen von Attribut- oder auch Elementnamen kommen. Um eine klare Trennung zu schaffen kann es sinnvoll sein so genannte Namespaces einzusetzen. Das sind Deklarationen von Element- und Attributnamen die in einem Dokument verwendet werden dürfen. Sie sind rekursiv gültig, bis ein neuer Namensraum angegeben wird.

7.3 Verwendung von Schemasprachen

Schemata dienen dazu eine Einhaltung von bestimmten Strukturellen Eigenschaften zu gewährleisten. Es gibt einige so genannte Schemasprachen, die dafür entwickelt wurden die Struktur eines XML-Dokumentes zu beschreiben. Am häufigsten werden XML-Schema und DTD verwendet.

7.3.1 DTD

Document Type Definitions wurden vorher schon kurz erläutert. Sie werden zur Beschreibung von XML-Dokumenten eingesetzt. Mit einer DTD lassen sich folgende Eigenschaften eines Dokumentes beschreiben:

7.3.2 Elementtypen

Darin werden ein Element und sein möglicher Inhalt (andere Elemente, Text) definiert. Es gibt Schlüsselworte und Zeichen um verschiedene Reihenfolgen von Elementen zu definieren. Ein Beispiel zur Deklaration des `<html>`-Tags, das zuerst `<head>` und dann `<body>` enthält sieht so aus:

```
<!ELEMENT html (head,body)>
```

DTD ermöglicht die Darstellung von:

- Elementsequenzen,
- Alternativen von Elementen,
- Wiederholungen und
- Kombinationen daraus.

7.3.3 Attribute

Um bestimmte Attribute einem Element zuzuweisen gibt es eine Attributliste. Sie enthält für jedes Element eine Liste mit Namen, Typ und Vorgabewert des Attributes. Beispiel für eine Attributdefinition:

```
<!ATTLIST img
    src    CDATA    #REQUIRED
    alt    CDATA    #IMPLIED
 >
```

7.3.4 Entity-Deklartationen

Eine Entity ist eine Zeichenkette oder ein externes Dokument, das in der DTD oder im XML Dokument benutzt werden kann. Wird folgende Entity definiert

```
<!ENTITY name   "Beispielfond">
```

, so wird im ganzen Dokument das Wort „name“ mit dem String „Beispielfond“ ersetzt.

Es gibt noch eine ganze Reihe weiterer Eigenschaften in DTD mit der sich Dokumente noch genauer beschreiben lassen. DTD ist ein Teil das XML- Standards. In DTD ist eine Unterscheidung zwischen Datentypen (z.B. Namen und Zahlen) nicht möglich.
[BPSM+04]

7.3.5 XML Schema

Auch XML Schema ist ein W3C-Standard und stellt quasi die modernere, umfassendere Methode dar, XML-Dokumente zu beschreiben. Einige wichtige Teile des Standards sollen hier aufgelistet werden:

- Datentypen: Mit XML Schema lassen sich einfache und komplexe Datentypen in XML deklarieren. Zu den einfachen Datentypen zählen z.B. Typen wie bool, string, int. Weiters gibt es komplexe Datentypen wie z.B. verschiedene Datums- und Zeittypen oder Listen. Auch die Definition von neuen Typen ist vorgesehen.

- Struktur: Wichtig für die Struktur von XML Schema sind Element- und Attributdeklarationen. Alle Elemente und Attribute werden deklariert indem ihnen ein Name und ein Typ zugewiesen werden.
  Entweder wird einem Element ein einfacher Datentyp zugewiesen oder eine komplexe Typdefinitionen. Innerhalb derer wird spezifiziert welche Elemente darin zulässig sind und in welcher Reihenfolge bzw. mit welchen Wiederholungen sie vorkommen dürfen. Komplexe Typen sind beliebig verschachtelbar.

- Integritätsbedingungen: Wo rein strukturelle Angaben nicht ausreichen, können Integritätsbedingungen festgelegt werden Es können Schlüssel und Fremdschlüsselbeziehungen oder Eindeutigkeitsbedingungen deklariert werden. Das ist z.B. dann nötig wenn in Dokumenten Querverweise enthalten sind.

- Widerverwendbarkeit und Erweiterbarkeit: XML Schemas können externe Schemas einbinden, so können wieder verwendbare Module erstellt werden.

[KST03, Seite 39ff.]

7.3.6 Weitere Methoden zur Schemabeschreibung

Der Vollständigkeit halber sei erwähnt, dass es noch weitere Möglichkeiten gibt Schemas zu definieren. Sie haben völlig andere Ansätze als DTD oder XML Schema und weisen dadurch auch andere Einsatzstärken und -schwächen auf:

7.3.7 Schematron

Schematron benutzt XPath um den Aufbau eines XML-Dokumentes zu beschreiben. Schematron kann mit DTD, Relax NG oder XML Schema kombiniert werden und ermöglicht es so diese anderen Schemasprachen zu erweitern.

7.3.8 Relax NG

Mit Relax NG können Muster für den Aufbau eins XML-Dokumentes definiert werden. Das besondere dabei ist, dass die Definition selbst auch ein XML-Dokument sein kann. Neben der Definition als XML-Dokument gibt es aber auch eine kompaktere Nicht-XML-Syntax.

7.3.9 XDR

XDR ist eine weitere Schemasprache, wird derzeit aber eher selten verwendet.
[KST03]

Kapitel 8

XML-Formatierung

Da XML eine Sprache zur Beschreibung von strukturierten Dokumenten ist, enthält es keinerlei Informationen zur Darstellung der enthaltenen Informationen. Um XML-Daten in ein geeignetes Ausgabeformat zu bringen, wie z.B. HTML oder PDF, gibt es spezielle Sprachen die dafür entwickelt wurden.
XSL [1] ist so eine Sprachfamilie. Es gibt 3 Sprachen die zu dieser Familie gehören:

- XSLT:
  Eine Sprache zum Transformieren von XML Bäumen
- XSL-FO:
  Dient zum Spezifizieren der Visuellen Ausgabe von XML
- XPath:
  Eine Abfragesprache die von XSLT benutzt wird um auf Elemente in einem XML-Dokument zu referenzieren.

Im Folgenden soll kurz erläutert wofür genau diese Sprachen stehen und wie diese Techniken eingesetzt werden können.

8.1 XSLT

Unter XSL fällt unter anderem XSLT [2]. Damit lassen sich XML-Bäume transformieren (=umwandeln). Der Ablauf ist folgender maßen: Ein XSLT-Stylesheet definiert die Regeln nach der ein XML-Dokument umgewandelt werden soll. Dieses XSLT-Stylesheet und das umzuwandelnde XML-Dokument werden von einem XSLT Prozessor verarbeitet, der dann den transformierten XML-Baum ausgibt.
Anwendungsbeispiel: So kann eine bestehende XML-Datei in ein HTML-Format umgewandelt werden und in einem Browser angezeigt werden. [ABC+01]

Ablauf:

[1] Extensible Stylesheet Language
[2] XSL Transformation

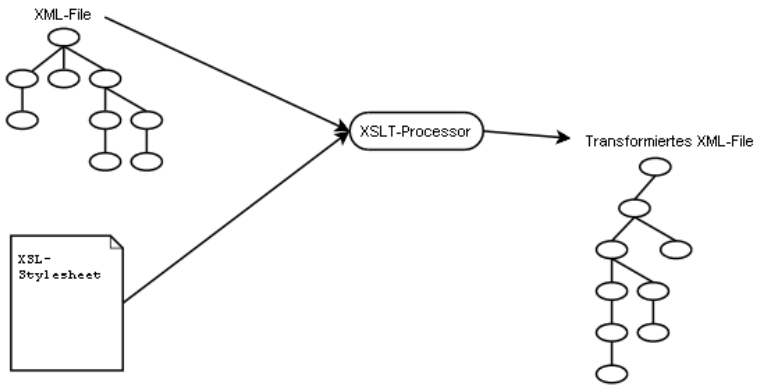

Abbildung 8.1: Ablauf einer XSLT-Transformation

8.1.1 Aufbau eines XSLT-Stylesheets

Ein XSLT-Stylesheet dient, wie oben schon erläutert, zur Umwandlung eines XML-Quellbaumes in einen XML-Zielbaum. Für die Transformation werden eine Reihe von Transformationsregeln geschrieben, die man Templates nennt.

Templates

Jedes Template besitzt einen XPath-Ausdruck, der angibt für welche Knoten des XML-Quellbaumes es angewendet werden soll. Zur Veranschaulichung soll hier ein Beispiel-Template erklärt werden, das alle Elemente des XML-Quellbaumes mit dem Namen „title“ kursiv darstellt:

```
...
<xsl:template match="//title">
   <em>
     <xsl:apply-templates/>
   </em>
 </xsl:template>
 ...
```

Mit `//title`, einem XPath-Ausdruck, wird festgelegt, dass das Template auf alle Elemente des XML-Quellbaumes mit dem Namen „title“ angewendet werden soll. Der Reihe nach wird das Template für all diese Elemente angewendet. `<xsl:apply-templates/>` transformiert anschließend das aktuelle Element mittels sämtlicher dafür angegebenen Regeln im XSLT-Stylesheet.
Im XML-Zielbaum sind dann also alle Elemente mit Namen „title“ von den Tags `<em>` und `</em>` umschlossen und werden somit kursiv dargestellt.

Weitere Syntaxregeln von XSLT

Neben einfachen Templates die sich auf eine Gruppe von Elementen des XML-Quellbaumes beziehen, können mit XSLT auch komplexere Vorgänge wie Sortierungen, Bedingungen und

Schleifen realisiert werden. [Wik06e]

8.2 XSL-FO

Eine weitere XSL-Sprache ist XSL-FO [3]. Darunter kann man sich ein Stylesheet für eine XML-Datei vorstellen das Vorgaben für eine grafische Ausgabe enthält. Zweck ist eine Trennung von Inhalt und Layout, ähnlich wie das bereits weit verbreitet mit CSS [4] für HTML-Seiten gemacht wird. XML-FO ist vor allem darauf spezialisiert PDF-Format auszugeben. Es lassen sich damit Formatierungen wie Seitengröße, Seitenummerierungen, Absätze, Listen, Tabellen, Schriftformate und viele andere Seitenlayoutinformationen festlegen. Der Ablauf um aus einem XML-Baum ein druckfertiges PDF ist folgendermaßen: Zuerst muss das XML-Dokument mit XSLT in ein XSL-FO-Dokument umgewandelt werden, in dem dann alle Layoutinformationen enthalten sind. Dieses wird dann von einem XSL-FO Prozessor in ein fertiges PDF umgewandelt. Abhängig vom Prozessor können statt einem PDF auch andere Formate ausgegeben werden wie z.B. PS, SVG,

Ablauf:

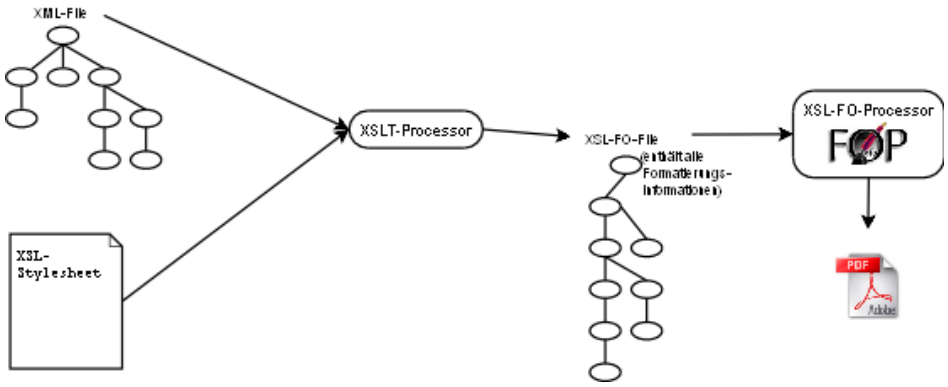

Abbildung 8.2: Erzeugung eines PDFs aus einer XML-Datei mit FOP

[Cla99]

8.2.1 Aufbau eines XSL-FO Stylesheets

Der Aufbau soll anhand eines Beispiels erklärt werden. Folgendes XSL-FO Stylesheet gibt den Text „Hallo Welt!“ auf einer hochformatigen A4 Seite aus:

```
<?xml version="1.0"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
 <fo:layout-master-set>
  <fo:simple-page-master  master-name="A4"
                          page-width="210mm" page-height="297mm">
```

[3] Extensible Stylesheet Language - Formatting Objects
[4] Cascading Style Sheets

```
    <fo:region-body region-name="xsl-region-body"  margin="2cm"/>
   </fo:simple-page-master>
  </fo:layout-master-set>

  <fo:page-sequence  master-reference="A4">
   <fo:flow flow-name="xsl-region-body">
    <fo:block>Hallo Welt!</fo:block>
   </fo:flow>
  </fo:page-sequence>

 </fo:root>
```

[Wik06c]
Der Aufbau eines XSL-FO-Stylesheets folgt immer dem selben Muster. An Anfang steht eine XML-Deklaration. Dann gibt es ein Root-Element, das den Namespace .fo deklariert. Nun folgen die Angaben für eine formatierte Ausgabe. Das Root-Element hat 3 Kind-Elemente: Das „layout-master-set“ und die „page-sequence“ und optional „declarations“.
Im „layout-master-set“ wird das Layout der Seiten in Form eines „simple-page-master“ definiert. In diesem Fall gibt es nur ein „simple-page-master“ für eine Seite im A4-Format. Sie enthält einen Inhaltsbereich, den „region-body“ der einen 2-cm breiten Abstand zum Seitenaußenrand hat. Es ist möglich mehrere Seitenlayouts zu definieren, die dann im Dokument über ihren „master-name“ zugewiesen werden können.
Die „page-sequence“ legt die Reihenfolge fest in der die zuvor im „layout-master-set“ definierten Seiten verwendet werden sollen. Hier wird mit „master-reference=“A4“ natürlich auf die definierte A4-Seite verwiesen. Es folgen jetzt noch ein „flow“ Element, das angibt, dass nun der laufende Inhalt folgt. Um den Inhalt anzuzeigen und zu formatieren muss er noch mindestens von einem „block“-Element eingeschlossen werden. Das dient dazu, dass der Formatierer, wenn der Inhalt des „block“-Elements zu groß ist, den Block auf mehrere Seiten aufteilen kann. Der optionale Bereich „declarations“ ist im Beispiel nicht enthalten. Er würde es erlauben Farbprofile anzulegen.

8.2.2 Features von XSL-FO

XSL-FO unterstützt eine große Zahl von Textformatierungsmöglichkeiten. Folgende können für die Erzeugung von PDFs im Projekt FondsDoc eingesetzt werden:

- Mehrspaltige Seiten
- Seitennummerierung
- Tabellen (Es wird bei sehe langen Tabellen auch der Umbruch der Tabelle über mehrere Seiten unterstützt)
- Listen
- Verschiedene Textauszeichnungsvarianten
- Linien
- Einbinden von Bildern

- Textumbruch und Silbentrennung

8.3 XPath

XPath ist keine Formatierungssprache sondern eine Abfragesprache um bestimmte Knoten in einem XML-Baum zu adressieren. Da XPath essentiell für XSLT-Stylesheets ist (es dient dazu festzulegen für welche Baumknoten ein Template angewendet werden soll), soll es im Folgenden kurz erklärt werden.
Ein XPath Ausdruck setzt sich aus mehreren Lokalisierungsschritten zusammen die mit einem \ getrennt werden. Ein solcher Lokalisierungschritt wird definiert durch eine „Achse“, einen „Knotentest“ und optional von „Prädikaten“.

8.3.1 Achse

Mithilfe von Achsen werden ausgehend von aktuellen Baumknoten ein Teil des XML-Baumes ausgewählt. Beispielsweise werden mit der Achse `child` alle Knoten ausgewählt die dem aktuellen Knoten direkt untergeordnet sind. Es gibt eine ganze Reihe von Achsen: child, parent, self, ancestor...

8.3.2 Knotentest

Mit einem Knotentest lässt sich die Menge an Knoten, die von der Achse ausgewählt worden ist präziser einschränken. Die Notation erfolgt mit `Achse::Knotentest` Zur Veranschaulichung hier ein Beispiel:

```
child::Kapitel
```

Dieser Ausdruck adressiert alle Kindelemente des aktuellen Knotens, jedoch nur dann wenn sie den Namen „Kapitel“ haben.

8.3.3 Prädikate

Prädikate können die Auswahl noch präziser einschränken. Sie werden in `[ ]` an den Knotentest angehängt. Prädikate können entweder ihrerseits einen XPath-Ausdruck enthalten oder es können eine Reihe von Funktionen eingesetzt werden:

- Wahrheitsfunktionen wie `<`, `>`, `=` ...
- Zeichenkettenfunktionen, z.B. `substring()` um auf einen Teilstring zuzugreifen.
- Numerische Funktionen wie +, -, * ... oder
- Knotenmengenfunktionen, z.B. `position()` um zu ermitteln der wievielte Knoten einer Menge adressiert werden soll

Zur Veranschaulichung auch dazu ein Beispiel:

```
child::Kapitel[position() =3]
```

Dieser Ausdruck adressiert genau das dritte Kindelement (Die Nummerierung beginnt bei 1 und nicht wie oft üblich bei 0) des aktuellen Knotens, jedoch nur dann wenn es den Namen „Kapitel“ hat.
[Wik06d]

Kapitel 9

XML-Verarbeitung mit PHP

Da das Projekt FondsDoc mit PHP umgesetzt werden soll, werden hier Möglichkeiten erläutert wie man XML mit PHP verarbeiten kann.

PHP bietet drei verschiedene Möglichkeiten bestehende XML-Dateien auszulesen, zu verarbeiten oder zu erweitern.

9.1 SimpleXML-Erweiterung

Damit kann eine XML-Datei als PHP-Objekt geladen werden. Im Objekt wird der XML-Baum als verschachteltes Array dargestellt. Es gibt damit sowohl Zugriffsmöglichkeiten auf einzelne Elemente, als auch auf Attribute. Auch komplexere Abfragen können gemacht werden, da diese Erweiterung X-Path [1] Abfragen verarbeiten kann. Weiter bietet die SimpleXML-Erweiterung die Möglichkeit Elemente hinzuzufügen und Elemente zu löschen. Besonderer Vorteil dieser Erweiterung ist die rasche Verarbeitungsgeschwindigkeit.

9.2 Simple API for XML (SAX)

Hiermit erfolgt der Zugriff auf XML-Dateien etwas anders. SAX parst das XML File und bei bestimmten Ereignissen (z.B. wenn ein Tag geöffnet oder geschlossen wird) werden editierbare Funktionen aufgerufen. So wird zwar ein sehr rascher Zugriff auf die XML-Daten gewährleistet. Allerdings gibt es keinen Schreibzugriff auf die XML-Datei und Elemente können nur linear verarbeitet werden, da SAX keine XPath Abfragen umsetzen kann.
SAX ist dafür geeignet um XML Eingangsdaten zu verarbeiten.

9.3 DOM-Erweiterung

Hier wird die „Level 2 Core“-Version des DOM-Standards des W3C [HWW05] implementiert. Damit kann auf einzelne Elemente und Attribute in wohlgeformten XML-Dateien und in HTML-Dateien zugegriffen werden. Die Erweiterung ist sehr umfangreich und bietet eine Vielzahl von Methoden und Funktionen für Schreib- und Lesezugriffe, u.A. auch XPath

[1] XPath ist eine Abfragesprache mit der auf bestimmte Elementgruppen in einem XML-Baum verwiesen werden kann.

und XML-Schema, auf XML-Dateien. Beachtet werden muss allerdings dass diese Erweiterung sehr Zeit- und Speicheraufwendig ist. Das könnte zu Problemen führen falls bei einer Anwendung besonders viel Wert auf Performance gelegt wird.

9.4 EXKURS: DOM

Hier werden kurz die wichtigsten Dom-Klassen aufgelistet und erklärt.

9.4.1 Node

Node ist die Basisklasse von der alle anderen Klassen abgeleitet werden. Sie repräsentiert ein allgemeines XML Element. Als allgemeine Klasse kann Node nie direkt in ein XML Dokument eingebunden werden. Es werden nur die Ableitungen verwendet.

9.4.2 Element

Repräsentiert ein XML-Element der Form `<Tag>`. Also ein blankes Tag ohne Attribute und Inhalt.
Die zugehörige PHP-Methode zur Erzeugung eines Elements ist createElement(). Mit der Methode appendElement() lasst sich ein Element an ein anders anhängen.

9.4.3 Attribute

Repräsentiert ein XML-Attribut der Form `<Attributename="Wert">`. Attribute kann an ein Element angehängt werden.
Die zugehörige PHP-Methode zur Erzeugung eines Attributs ist createAttribute(). Mit der Methode appendAttribute() lasst sich ein Attribut einem Element zuweisen.

9.4.4 CharacterData

Wir für Text eingesetzt der auch Sonderzeichen enthalten kann.

Kapitel 10

Speicherung von XML

Hier soll ein kurzer Überblick über die Möglichkeiten XML zu speichern gegeben werden. Dieser Abschnitt richtet sich im Wesentlichen nach der Einteilung in „XML und Datenbanken“ von Klettke und Meyer. [Kle02, S155ff]

10.1 Speicherung als Ganzes

Die einfachste Methode zur Speicherung von XML ist die Dateien als ganzes abzulegen. Das kann entweder direkt im Dateisystem sein oder als CLOBs [1] in einer Datenbank. Wenn auf diese Dateien rasch und effizient zugegriffen werden soll, ist es sinnvoll einen Index auf die Dokumente anzulegen. So können Abfragen beantwortet werden, welche Stichworte oder Stichwortkombinationen sich in den Dokumenten befinden.

10.1.1 Volltextindex

Bei dieser Methode wird zu sämtlichen XML-Dokumenten abgespeichert welche Stichworte sie enthalten. Es liegt dann eine Liste aller Dokumente mit jeweils allen vorkommenden Stichworten vor. Auch eine invertierte Speicherung ist möglich. In diesem Fall sind in der Liste alle Stichworte vermerkt und für jedes Wort eine Aufzählung aller Dokumente in denen es vorkommt.

10.1.2 Kombinierter Volltext- und Strukturindex

Bei einem kombinierten Index wird zusätzlich zu einem Volltexindex (siehe Kapitel 10.1.1, Seite 25) die Struktur des XML-Dokumentes abgespeichert. Dazu werden alle Elemente eines XML-Dokumentes mit Verweisen wo sie sich im Dokument befinden, welche Ebenenordnung sie haben und welches ihr Elternelement ist gespeichert. Dadurch werden komplexere Abfragen möglich, die sich nicht auf das reine Vorkommen bestimmter Wörter im Dokument beschränken.

[1] Character Large Object

10.2 Speicherung der Dokumentstruktur

Bei diesen Verfahren werden sowohl die Dokumentenstruktur, als auch alle Dokumentinhalte abgespeichert.

10.2.1 Abbildung der Graphenstruktur

Zur Abbildung der Graphenstruktur ist ähnlich wie beim Strukturindex jedes Tag mit einer ID versehen. Alle Tags werden in einer Relation gespeichert, jeweils mit ihrer ID, ihrem Elementnamen, ihrem Inhalt, der ID ihres Elternelementes und einer Kind-Nummer, die angibt als wievieltes Kindelement sie innerhalb ihres Elterelementes vorkommen. Die Kind-Nummer ist wichtig zur Erhaltung der Elementreihenfolge bei der Wiederherstellung des XML-Dokumentes. Attribute werden in einer eigenen Relation jeweils mit Name, Wert und ID des Elementes zu dem sie gehören gespeichert.
Da diese Relationen wiederholte Werte enthalten und ziemlich lang sein können, ist ein entsprechendes Normalisieren [2] sinnvoll.

10.2.2 Abbildung von DOM-Strukturen

Eine weitere Möglichkeit die Graphenstruktur abzubilden ist DOM (siehe Kapitel 9.3, Seite 23). Dafür wird die Dom-Klasse Node in einer Relation abgebildet. Jede Node erhält eine ID. Es werden zu jeder ID der Node-Typ (z.B. Element oder Attribut), die ID des Elternknotens und die IDs des vorangehenden und des nachfolgenden Geschwisterknotens gespeichert. Im Großen und Ganzen ähnelt diese Methode der Abbildung der Graphenstruktur. Jedoch hat sie, durch das Verwenden der DOM-Klasse Node, einen Vorteil: Es lassen sich sowohl XML-Abfragesprachen (z.B. XPath) also auch SQL einsetzen.
Ein großer Vorteil des DOM ist, dass hier neben komplexen Abfragen auch Änderungen an den Dokumenten möglich sind.

10.3 Abbildung der Dokumentstrukturen auf Datenbankstrukturen

Bei dieser Variante der Speicherung wird die hierarchische Struktur von XML-Dokumenten auf Strukturinformationen von Datenbanken abgebildet. Dafür eigenen sich besonders objektorientierte [3] oder objektrelationale [4] Datenbanken.

10.3.1 Automatische Speicherung

Die bisher beschriebenen Methoden funktionieren alle Schema-unabhängig. Für die automatische Speicherung muss aber eine genaue Definition der Dokumente, z.B. in Form einer DTD

[2] Unter Normalisieren ist hier die Zerlegung in mehrere Relationen zu verstehen. Dadurch können auch Datentypen realisiert werden.

[3] „Eine objektorientierte Datenbank ist eine Datenbank, deren Inhalt Objekte im Sinn der objektorientierten Programmierung sind.“(http://de.wikipedia.org/wiki/Objektorientiertes_Datenbankmodell)

[4] „Eine Objektrelationale Datenbank (ORDB, ORDBMS) stellt das Bindeglied zwischen klassischen relationalen und objektorientierten Datenbanksystemen dar. Sie kommen überall dort zum Einsatz, wo Mengen von Objekten in Beziehung zu anderen Daten oder Objekten gebracht werden müssen.“(http://de.wikipedia.org/wiki/Objektrelationale_Datenbank)

oder eines XML-Schemas (siehe Kapitel 7.3.5, Seite 15) vorliegen. Anhand dieser DTD kann dann ein Datenbankmodell gebaut werden, das dem Aufbau des XML-Dokumentes entspricht: XML-Elementstrukturen werden zu Klassen, darin enthaltene Elemente ohne weitere Verschachtelung werden zu Attributen dieser Klassen. Enthalten die XML-Strukturen ihrerseits strukturierte Elemente, so sind das dann Referenzen auf Klassen dieser Strukturen usw. Eine Ausführliche Beschreibung dieser Methode findet sich hier: [KST03, Seite 199ff.]

10.3.2 Mapping Vorschriften

Mehr Unabhängigkeit von XML-Struktur und Datenbankaufbau kann durch Mapping Vorschriften erreicht werden. Darin wird spezifiziert welche Informationen aus dem XML-Dokument auf welche Datenbankstrukturen abgebildet wurden. Mapping ist sehr flexibel, erfordert aber einen großen manuellen Aufwand. Oft ist ein Rekonstruieren der XML-Dokumente aus der Datenbank nicht mehr möglich. Für die Syntax der Mapping-Vorschriften wird die Verwendung von XML oder XSLT empfohlen. [Kle02]

Kapitel 11

Auswahl geeigneter Umsetzungstechniken für das Projekt FondsDoc

Da für das Projekt FondsDoc vor allem das Überarbeiten von Dokumenten im Vordergrund steht, eignet sich nur eine Speichermethode die auch Änderungen und Erweiterungen der XML-Struktur zulässt. Das legt nahe eine Speicher- und Zugriffsvariante zu wählen die dem DOM-Modell entspricht.
Aufgrund der Unterschiedlichkeit der verschiedenen Dokumentenschemas macht es allerdings wenig Sinn eine fixe Datenbankstruktur zu entwerfen. Viel eher scheint eine Kombination aus der Speicherung der gesamten XML-Dokumente und einem Zugriff darauf über DOM-Funktionen zielführend.

Teil II

Analyse und Konzeption

Kapitel 12

Analyse der zu generierenden Dokumente

Hier sollen kurz die Ergebnisse der Analyse der Dokumenten erläutern werden. Um festzustellen welche Funktionen eine Anwendung zum erstellen diverser Dokumente haben muss, wurden zuerst sämtliche Dokumente die für einen Fond benötigt werden auf Aufbau und Formatierung untersucht. Es wird zwischen Spezialfonds und Publikumsfonds unterschieden. Der Hauptteil der Spezialfonds ist der Rechenschaftsbericht, der sich aus einigen Dokumenten zusammensetzt. Bei Publikumsfonds gib es zusätzlich noch einen Verkaufsprospekt, der wiederum eine Zusammensetzung mehrerer Dokumente ist. Zusätzlich gibt es für Spezial- und Publikumsfonds noch weitere Dokumente.

Daraus ergibt sich die Anforderung, dass Dokumente ihrerseits ganze Dokumente einbinden können.

Der Aufbau der einzelnen Dokumente folgt Großteils immer einem gleichen Schema, nur einzelne Passagen(wie z.B. der Fondname, Datum und Kennzahlen) müssen individuell angepasst werden. Trotzdem muss die Anwendung flexibel bleiben, da es immer wieder Ausnahmen und Spezialfälle gibt. Beispielsweise scheinen bei manchen Fonds spezielle Absätze auf, bei anderen nicht.

12.1 Überblick über die Dokumente

12.1.1 Liste der Dokumente

Diese Liste beinhaltet alle Dokumente, die im Zuge des Projektes FondsDoc automatisiert erzeugt werden sollen. Die Reihenfolge ist willkürlich. Alle Dokumente sind Anhand ihrer Nummer (Zahl in Klammern) eindeutig identifiziert, wobei Unterarten möglich sind.

- (1) Depotbankschreiben
- (2) Sammelaufstellung
- (3) Sammelurkunde und Beilagebrief
- (4) Produktbeschreibung

- (5) FMA Antrag
- (6) Informationen für Deutschland — Schweiz
- (7) Bericht Fondsmanager
- (8) Ertragsrechnung
- (9) Organe der Ges. u. Depotbank
- (10) KAG Fondsliste
- (11) Verkaufsprospekt
- (12) Rechenschaftsbericht
- (13) Halbjahresbericht
- (14) Vermögensaufstellung
- (15) Bestätigungsvermerk
- (16) Erläuterungen zu Steuerliche Behandlung
- (17) Vereinfachter Prospekt
- (19) Besondere Fondsbestimmungen
- (20) Anhang zu §16 - Börsenliste
- (21) Steuerliche Behandlung
- (22) Umlaufbeschluss
- (23) Delegationsmeldung
- (24) Anteilsinhaber Info
- (25) Prüfbericht
- (26) ÖKB Stammdatenblatt
- (27) Veröffentlichungen
- (28) Depotanlageblatt
- (29) Kenntnisnahme AR Beschluss

12.1.2 Abhängigkeiten zwischen den Dokumenten

Die Nachfolgenden Diagramme zeigen Netzplanartig die Zusammenstellung der Dokumente. Ein Pfeil bedeutet „ist enthalten in“, ist er strichliert, ist das Dokument nicht immer in dem anderen enthalten.
Publikums- und Spezialfonds unterscheiden sich in etlichen Punkten. Der wesentlichste Unterschied: Für Spezialfonds gibt es keine Verkaufsprospekte und Veröffentlichungen, jedoch Informationen für den Anteilsinhaber.
Detailbeschreibungen der Dokumente befinden sich in Anhang A.

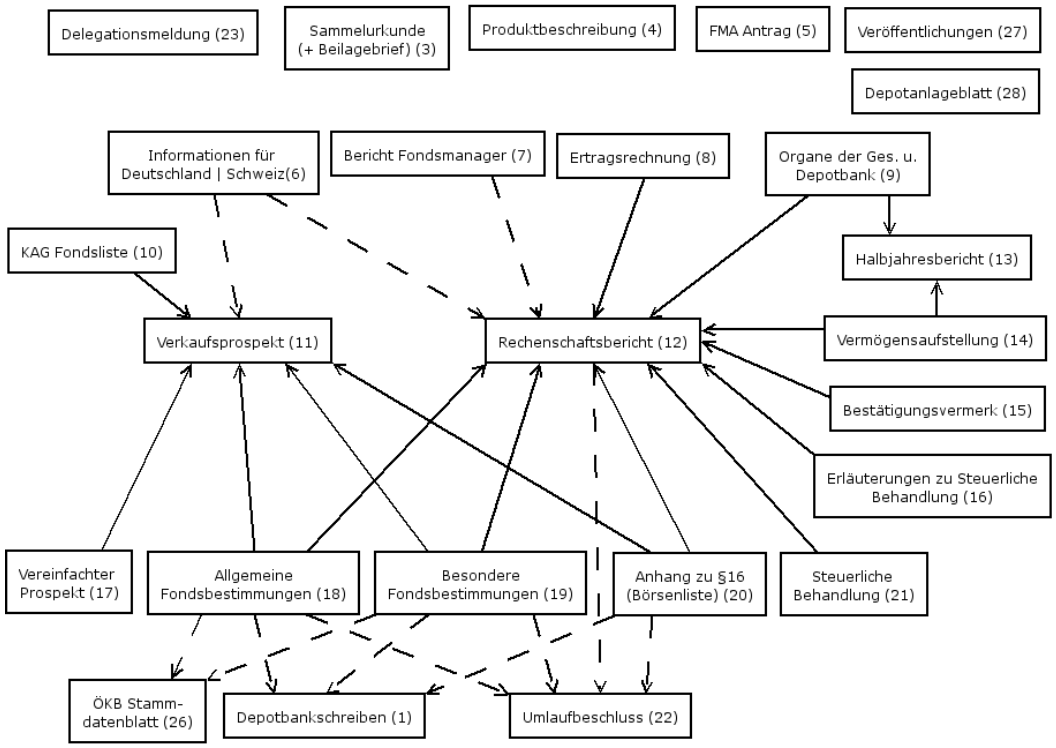

Abbildung 12.1: Dokumentenbaum Publikumsfonds

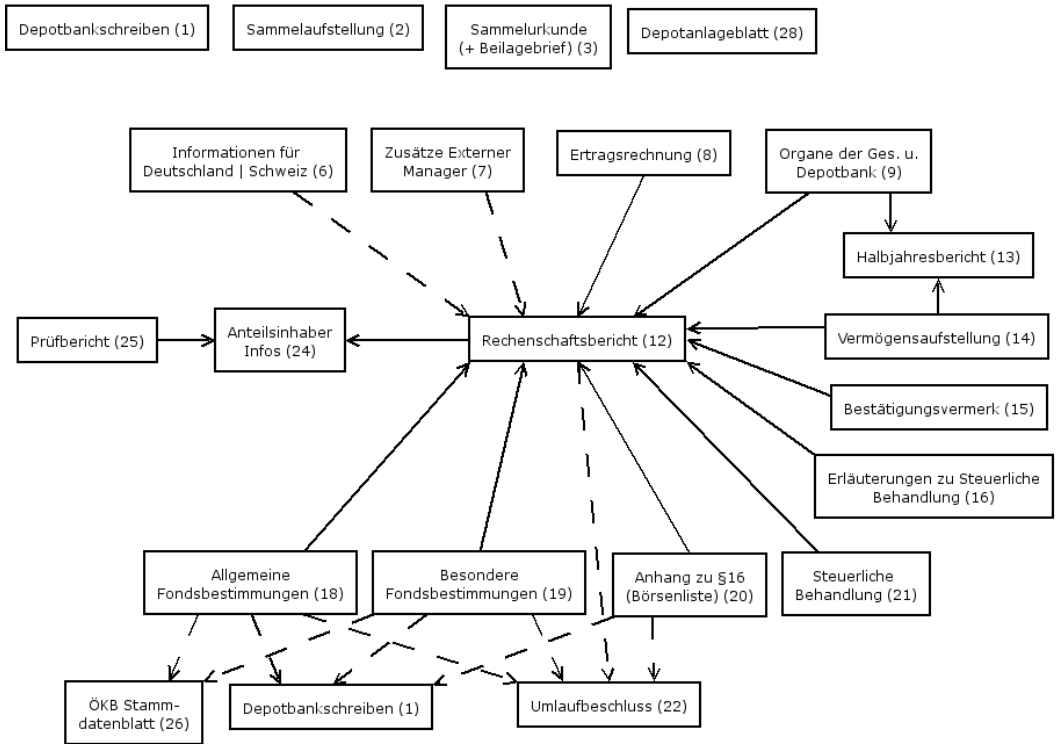

Abbildung 12.2: Dokumentenbaum Spezialfonds

Kapitel 13

Anforderungen an die Applikation

13.1 Erfassen und Editieren der Dokumente

Der erste wichtige Schritt zu einer automatisierten Generierung der oben beschriebenen Dokumente ist das Verallgemeinern der Dokumente. Aus der Analyse sämtlicher Dokumente ergab sich, dass jedes Dokument immer einem ähnlichen Aufbau mit mehr oder weniger vielen fondspezifischen Abweichungen aufweist.

13.1.1 Allgemeine Schemata

Um nun eine Applikation zu entwickeln, welche die oben beschriebenen Dokumente automatisch generiert muss zuerst von den Benutzern für den Aufbau eines jeden Dokumentes ein allgemeines Muster definiert werden. Ein solches Muster wird im Folgenden als „FondsDocSchema“ bezeichnet. Anhand dieses FondsDocSchema kann dann das Dokument für einen spezifischen Fond generiert werden kann. Eine Applikation muss in der Lage sein ein solches FondsDocSchema in irgendeiner Form zu speichern und editierbar zu machen. Ein Beispiel für ein solches FondsDocSchema wird hier in der Abbildung „FondsDocSchema“ dargestellt. Die einzelnen Komponenten die in einem FondsDocSchema enthalten sind werden in den folgenden Kapiteln kurz erklärt.

13.1.2 Absätze

Absätze sind einzelne Textabschnitte mit einem vorgegebenen Inhalt. Absätze können eine Überschrift haben und können ihrerseits wieder Unterabschnitte enthalten. Absätze sind also beliebig verschachtelbar. Daraus folgt dass Dokumente eine Baumstruktur aufweisen.

13.1.3 Fond-Spezifische Absätze

Es kann aber zu den gefundenen FondsDocSchemata auch Ausnahmen geben, denn es folgen nicht alle Dokumente immer exakt demselben Muster. Um für einzelne Fonds individuelle Absätze einzufügen, muss jeder Absatz bestimmten Fonds zugewiesen, bzw. für bestimmte Fonds ausgeschlossen werden können. Das soll durch benutzerdefinierte Einschränkungen, im Folgenden „FondsDocConstraints“ genannt passieren.
Da sich durchaus komplexe Einschränkungen, nach Fondskriterien (z.B. Fondtyp), ergeben können, ist es nicht ausreichend hier Einschränkungen durch Zuweisen einzelner Fond-IDs zu bestimmten Absätzen zu treffen. Es muss durchaus für die Benutzer möglich sein eine

Einschränkung anzugeben, welche die gewünschten Fond-IDs derjenigen Fonds definiert für die ein einzelner Absatz erstellt oder ausgelassen werden soll.

13.1.4 Platzhalter

Die Texte in den Absätzen der Dokumente enthalten aber auch austauschbare Werte. Einfachstes Beispiel hierfür wäre der Fondsname: An sich sind manche Textpassagen in bestimmten Dokumenten völlig gleich, bis auf den Namen des jeweiligen Fonds.
Um die zuerst gefundenen FondsDocSchema also für mehrere Fonds verwenden zu können, muss es eine Art Platzhalter geben, die für austauschbare Werte (Wie z.B. den Fondnamen) eingesetzt werden und erst bei der Generierung des Dokumentes für einen bestimmten Fond durch konkrete Werte ersetzt werden. Diese konkreten Werte finden sich in der Regel in einer Datenbank. Solche Platzhalter werden im Folgenden „FondsDocPlatzhalter“ genannt.

FondsDocPlatzhalter verweisen also, abhängig von der Fond-ID und anderen Parametern (z.B. Datumswerte) auf eine Funktion in der Applikation oder auf ein bestimmtes Feld in der Datenbank.

Fix definierte Platzhalter

Da FondsDocPlatzhalter häufig wieder verwendet werden, macht es durchaus Sinn, die am häufigsten verwendeten fix in die Applikation einzubauen. Der Benutzer soll auf eine Anzahl von FondsDocPlatzhaltern fix zugreifen können, ohne selbst die erforderliche Einschränkung definieren zu müssen.

Anpassbare Platzhalter

Abgesehen davon soll es für die Benutzer möglich sein neue FondsDocPlatzhalter durch Angeben eine Einschränkungsstatements zu definieren.

13.1.5 Formatierungsdefinitionen

Die Applikation soll aber nicht nur die Struktur der Dokumente erfassen, sondern auch Dokumente formatieren. Für die Formatierung der Dokumente sind zweierlei Fälle zu unterscheiden.

1. Allgemeine Formatierungen wie z.B. Absätze, Überschriften verschiedener Ordnungen und andere immer wiederkehrende Formate.

2. Spezielle Formatierungen, die der Benutzer innerhalb eines Textes eingeben kann, wie z.B. Fettschreibung, Farbliche Auszeichnung u.s.w.

13.1.6 Grafiken

Bei manchen Dokumenten müssen auch Grafiken eingefügt werden. Diese sollen auch in Absätze eingefügt werden können.
Es werden nicht nur statische Bilder eingebunden, sonder auch dynamisch erstellte Balkengrafiken (z.B. Risikobewertungen und Charts).

13.2 Die Dokumentengenerierung

Wurde ein Dokument von einem in der Anwendung definiert, d.h.
- es wurde ein FondsDocSchema, mit allen nötigen Absätzen, für das Dokument angelegt
- die Benutzer haben, falls nötig, alle Absätze mit FondsDocConstraints spezifischen Fonds zugewiesen
- alle Platzhalter sind definiert und eingefügt und
- benötigte Grafiken wurden eingebunden

so kann für einen bestimmten Fond ein PDF von einem Dokument generiert werden.
Hier finden sich Überlegungen für ein Konzept wie ein System zur Dokumentengenerierung aufgebaut werden kann. Dabei soll theoretisch behandelt werden welche Schritte nötig sind um aus den spezifischen Informationen für jeden Fond in den FondsDocPlatzhaltern und den nötigen Textpassagen und Grafiken ein fertiges Dokument zu erzeugen, das einem bestimmten Aufbau folgt, der zuvor im FondsDocSchema und festgelegt wurde, zu erzeugen.

Ein fertiges Dokument (als PDF) wird über 3 Zwischenschritte erzeugt. Zuerst gibt es das von den Benutzern festgelegte FondsDocSchema, das nur den allgemeinen Aufbau, wie oben erklärt, definiert. Vor der Ausgabe des fertigen Dokumentes wird daraus in einem Zwischenschritt ein spezifisches Dokument erzeugt, in dem alle dynamischen Teile mit den fondspezifischen Werten ersetzt werden. Aus dem spezifischen Dokument wird dann in einem letzten Schritt das PDF erzeugt. Es werden hier alle drei Schritt genauer erklärt:

13.2.1 FondsDocSchema, der allgemeine Aufbau des Dokumentes

Zuerst muss analysiert werden wie ein Dokument aufgebaut ist. Welche Überschriften, Absätze, Grafiken, usw. es enthält und in welcher Reihenfolge diese vorkommen. Im besten Fall lässt sich ein allgemein gültiger Aufbau finden, dem Dokumente für sämtliche Fonds entsprechen. Ein solcher Aufbau ist das „FondsDocSchema“. Das FondsDocSchema gibt also den Aufbau einer Dokumentenart vor. Wird ein FondsDocSchema verändert, werden auch automatisch alle Dokumente die nach dem FondsDocSchema aufgebaut sind verändert. Um das System möglichst flexibel zu gestalten, wird es nötig sein, verschiedene Versionen eines FondsDocSchema zuzulassen, damit etwaige Änderungen, falls erwünscht nur an bestimmten Dokumenten vorgenommen werden, während andere unverändert bleiben. Das FondsDocSchema setzt sich aus folgenden Teilen zusammen:

- Gleich bleibenden Textbausteinen:
  Das können sowohl Gesetzestexte, als auch andere vorgegebnen Textpassagen sein.

- Automatisch eingefügten Werten:
  Zahlen, Daten, Namen ..usw. aus der FondsDoc-Datenbank [1]

- Automatisch errechnete Werte:
  Diese kommen ebenfalls aus der FondsDoc-Datenbank.

- Grafiken.

[1] Diese Datenbank gibt es bereits. Darin werden für jeden Fond wichtige Daten gespeichert.

- Dynamisch erstellte Tabellen:
  Die Tabellen enthalten ebenfalls Daten aus der FondsDoc-Datenbank
- Dynamisch erstellte Aufzählungen:
  Ebenfalls mit Daten aus der FondsDoc-Datenbank.

13.2.2 Das spezifische Dokument

In einem weiteren Zwischenschritt müssen alle dynamischen Bestandteile für den konkreten Fond ermittelt werden. Es ergibt sich ein dynamisches Dokument aus dem allgemeinen FondsDocSchema. Durch das Angeben der nötigen Parameter werden alle dynamischen Textbausteine auf die spezifischen Werte umgerechnet. Ist das passiert liegen alle Bestandteile vor, die zur Generierung eines fertigen Dokumentes nötig sind. Diese Bestandteile können nun für die Ausgabe weiterverarbeitet werden. Die Daten sollen in einem Format vorliegen, das für verschieden Ausgabezwecke verwendet werden kann. Sei es nun die Umwandlung in ein PDF, die Anzeige am Bildschirm oder den Versand als E-Mail. Primäres Ziel des Projektes FondsDoc ist die Ausgabe der Dokumente als PDF, trotzdem soll die Anwendung flexibel und erweiterbar bleiben und eine Weiterentwicklung ermöglichen.

13.2.3 Das Endprodukt

In diesem Fall ist das Endprodukt ein fertiges PDF mit allen zum betroffenen Fond gehörigen Daten und Textbausteinen. Es stellt somit das Ziel des Projektes FondsDoc dar. Über den Zwischenschritt des spezifischen Dokumentes wird ein fertiges PDF erzeugt. Es ist nicht mehr veränderbar, kann verschickt oder im Filesystem abgelegt werden.

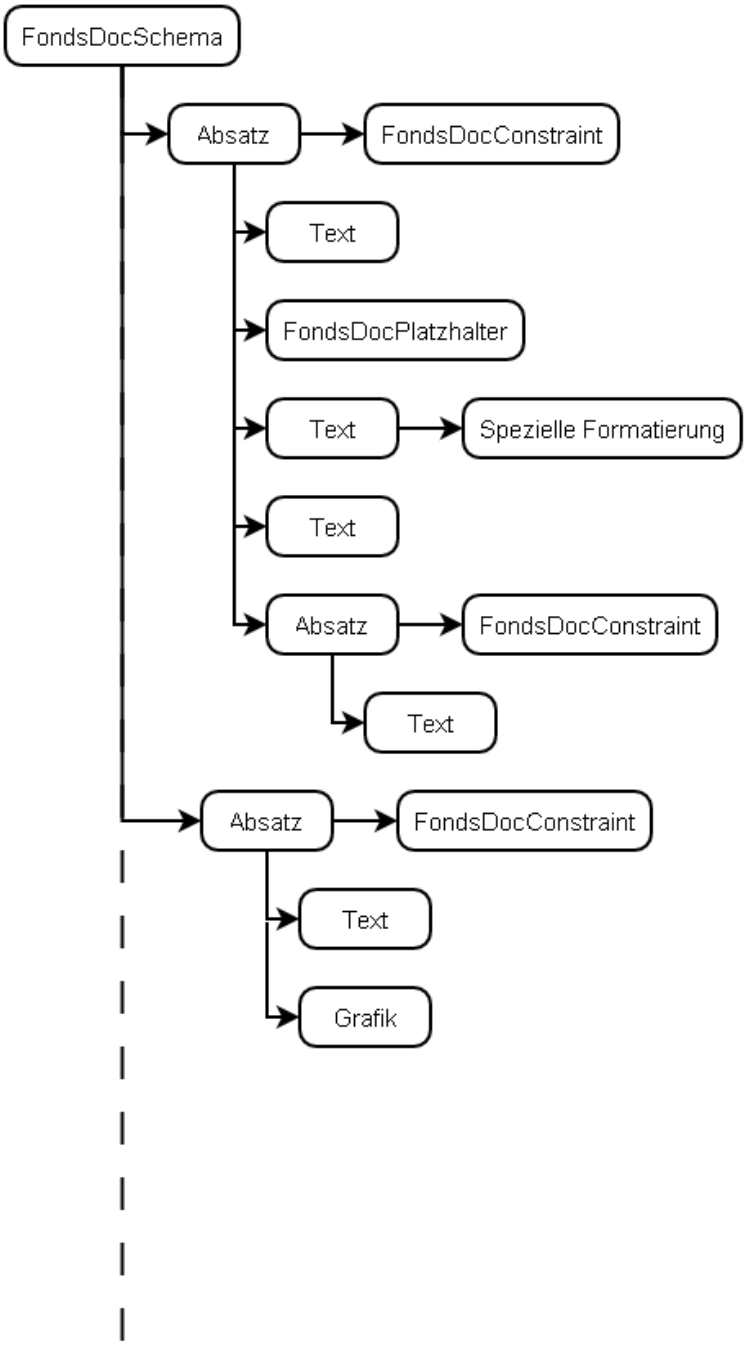

Abbildung 13.1: Grafische Darstellung eines FondsDocSchemas

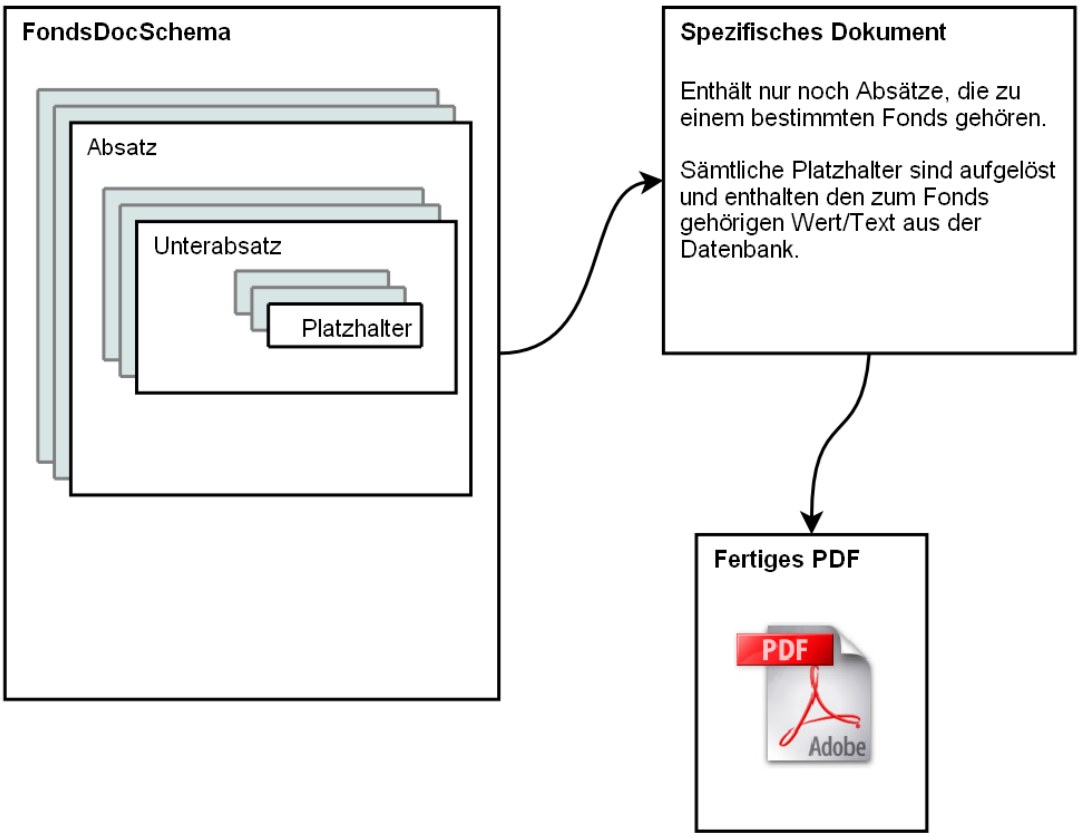

Abbildung 13.2: Schematische Darstellung der Dokumentengenerierung

Kapitel 14

Umsetzungsstrategie

14.1 Grundlegende Komponenten

Der Aufbau der Anwendung soll in drei Schichten nach dem Model-View-Controller Modell unterteilt werden.

14.1.1 Model: FondsDocSchemata

Das Model enthält die Daten die die Anwendung darstellen soll. Im Model selbst sind keine Funktionen vorgesehen die die Daten darstellen oder verändern.

Basis der Anwendung FondsDoc ist immer die Abspeicherung der FondsDocSchemas die die einzelnen Dokumente definieren.

14.1.2 View: Darstellung der Schemata im Benutzer Interface

Die View fragt die Daten aus dem Model ab, stellt sie dar und aktualisiert die Darstellung wenn die Daten verändert wurden. Abhängig von Aufbau der Anwendung kann die View auch Benutzereingaben an den Controller weiterleiten.

Die Benutzer der Anwendung FondsDoc müssen die FondsDocSchemata ansehen können. Dafür müssen diese grafisch in einem Browser dargestellt werden.

14.1.3 Controller: FondsDocSchemas editieren und PDF-Erzeugung

Der Controller ist der zentrale Teil der Anwendung. Er übernimmt Benutzereingaben und wertet sie aus und ändert ggf. auch die Daten aus dem Model.

In dieser Schicht werden die Benutzereingaben verarbeitet und die FondsDocSchemata erzeugt. Neben der Grafischen Darstellung der FondsDocSchemata muss es ein Interface dem User ermöglichen diese FondsDocSchemata zu editieren und neue FondsDocSchemata zu erzeugen. Weiters muss der Benutzer die Generierung der PDFs starten können.
Aus den abgespeicherten allgemeinen Schemata muss ein fertiges PDF erstellt werden, das nur noch Absätze und Werte für den gewünschten Fond enthält.

Fixer Bestandteil der Anwendung ist auf alle Fälle das Benutzer-Interface mit dem die FondsDocSchemata dargestellt werden, also die View. Sie wird mit PHP umgesetzt. Sicherlich lässt sich ein derartiges Interface auch mit anderen Programmiertechniken umsetzen, es wurde hier aber PHP gewählt, weil in der Bank Gutmann AG bereits ein PHP-Framework erstellt wurde, mit dem ein solches Benutzer-Interface erstellt werden kann.
Ebenfalls fix ist die Verwendung einer Datenbank, in der alle fondspezifischen Daten gespeichert werden.
Variationen sind aber bei der Speicherung der FondsDocSchemata (also dem Model) möglich. Hier gilt es festzustellen wie sich FondsDocSchemata am Besten speichern lassen. In Frage kommt eine Speicherung der Schemata als XML-Dateien oder die Entwicklung eines geeigneten ER-Models und die Speicherung in eine Datenbank.
Abhängig davon, in welcher Form die FondsDocSchemata gespeichert werden (als XML oder in eine Datenbank) ergeben sich auch verschieden Möglichkeiten und Probleme für das Editieren, Neuerstellen und Verwalten FondsDocSchemata.
Als Endprodukt soll ein PDF generiert werden, auch dafür gibt es verschiedene Möglichkeiten, die teilweise von der Art der Speicherung der Schemata abhängen.

Es sollen nun kurz einige Varianten durchgedacht werden, wie die Anwendung aufgebaut werden kann. Anschließend wird die am Besten geeignete Variante ausgewählt und weiter umgesetzt.

14.2 Variante 1: Datenbankbasiert

14.2.1 Speicherung der Schemata

Die Textabschnitte werden in eine Datenbank gespeichert. Wie in (siehe Kapitel 10, Seite 25) beschrieben wird zu jedem Abschnitt eine ID und die ID seines Elternelementes gespeichert. Weiters muss bei der Anwendung FondsDoc auch noch gespeichert werden, welche Abschnitte zu welchem Fond gehören.

14.2.2 Editieren der Schemata

Einfügen neuer Abschnitte erfolgt über ein in PHP implementiertes Frontend. Der Controller muss die IDs der Abschnitte so verändern, dass ein Einhalten der richtigen Reihenfolge beim Einfügen neuer Elemente gewährleistet werden kann.
Dazu soll eine Reihe von PHP-Klassen geschrieben werden, die die PHPLib (siehe Kapitel 15.3, Seite 45) erweitern, und diese Aufgaben übernehmen.

Das Einfügen von Platzhaltern erfolgt ebenfalls über das PHP-Frontend. Für die Platzhalter muss eine eigene Syntax überlegt werden, die dann bei der Auswertung mit PHP eindeutig geparst werden kann. Dafür muss entschieden werden, welche Sonderzeichen, bzw. Zeichenketten, in den Dokumenten nicht benötigt werden, und die somit als Eingrenzung der Platzhalter verwendet werden können.

14.2.3 PDF Generierung

Die eigentliche Generierung der Dokumente erfolgt ebenfalls mit PHP. Um das recht umständliche Handling der PDF Erzeugung etwas zu vereinfachen, soll dafür der kommerzielle PDF-Generator PDFLib (siehe http://www.pdflib.com/) angeschafft werden.
Damit kann aus den einzelnen Abschnitten ein fertiges, mehrseitiges, Dokument erstellt werden.
Es gilt zu noch zu überlegen, ob es sinnvoll wäre, vor der PDF-Generierung einen Zwischenschritt einzufügen. Beispielsweise könnte es vorteilhaft sein, aus den einzelnen Abschnitten in der Datenbank zuerst ein XML-File zu erzeugen, aus dem dann erst das PDF generiert wird.

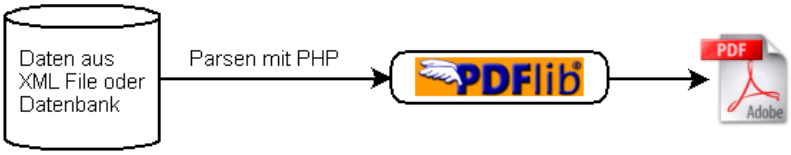

Abbildung 14.1: PDF-Erzeugung mit PDFLib

14.2.4 Vor- und Nachteile der Variante

+

- Weiterverwendung diverser PHPLib-Klassen (siehe Kapitel 15.3, Seite 45 möglich
- Großer Funktionsumfang

–

- Die Anschaffung der kommerziellen PDFLib ist nötig

14.3 Variante 2: Rein XML-Basiert

14.3.1 Speicherung der Schemata

Die Dokumente werden als XML-Dateien abgespeichert. Entweder erfolgt die Speicherung in eine DB (siehe Kapitel 10, Seite 25) oder direkt ins Dateisystem.

14.3.2 Editieren der Schemata

Das Einfügen neuer Abschnitte erfolgt über PHP-XML-Funktionen. Den größten Funktionsumfang bieten die DOM-XML-Funktionen. Da die DOM Funktionen XPath unterstützen, könnten auch komplizierter Vorgänge als neue Abschnitte einfügen oder löschen relativ einfach umgesetzt werden. Die Anwendung bleibt so sehr flexibel und erweiterbar.

Platzhalter können auch mit XML umgesetzt werden. Der Mixed-Type erlaubt das Tags sowohl Fließtext und dazwischen auch erneut XML-Tags vorkommen. Eine Verarbeitung mit XSL ist auch möglich. Das Einfügen der Platzhalter erfolgt über ein PHP-Interface.

Da Platzhalter auf Werte in Datenbanken verweisen, müssen sie sämtliche Informationen enthalten um eine Query zu generieren die den gewünschten Wert, abhängig vom jeweiligen Fond, zurückliefert bzw. ausrechnet. Je nach Fond, wird vor der PDF-Erzeugung noch eine Condition mit der Fond-ID dynamisch hinzugefügt. So werden für jeden Fond die zugehörigen Werte ermittelt.

14.3.3 PDF Generierung

Für die endgültige PDF-Ausgabe erfolgt mit XSL-FO wie im Kapitel 8.2, Seite 19 geschrieben.

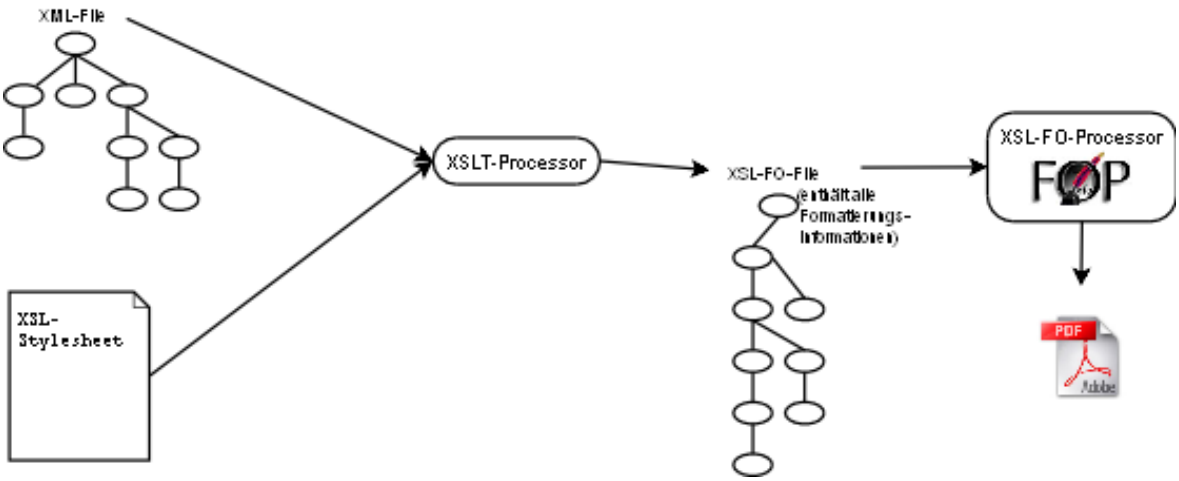

Abbildung 14.2: PDF-Erzeugung mit XSL-FO

14.3.4 Vor- und Nachteile der Variante

\+

- Durch die Verwendung der PHP-XML Funktionen ist die Anwendung sehr flexibel.
- Es muss kein kommerzieller PDF-Generator angeschafft werden, da die Generierung mit XSL-FO erfolgt

–

- Da es noch keine Erfahrungen mit XSL-FO gibt besteht ein gewisses Risiko, dass der Funktionsumfang nicht ausreichend ist.

14.4 Variante 3: XML-Basiert, als Erweiterung der PHPLib

Diese Variante soll die Vorteile der Variante 1 mit den Vorteilen der Variante 2 verknüpfen. Das interne Framework, die PHPLib (siehe Kapitel 15.3, Seite 45), soll hier erweitert und auch als Controller eingesetzt werden.

14.4.1 Speicherung der Schemata

Die FondsDocSchemata werden als XML-Dateien abgespeichert. Entweder in eine DB oder direkt ins Dateisystem(siehe Kapitel 10, Seite 25). Jeder Teil eines FondsdocSchemas (Absätze, Grafiken, FondsDocPlatzhalter...) entspricht einem XML-Element.

14.4.2 Editieren der Schemata

Zum Erzeugen eines XML-Outputs für ein FondsDocSchema werden PHP-Klassen geschrieben. Für jedes XML-Element wird eine Klasse instanziert. Diese Klassen müssen auch Funktionen wie das Einfügen von Platzhalter und das Zuweisen von Absätzen zu bestimmten Fonds übernehmen.
Das Editieren erfolgt über Formulare in einem PHP-Frontend. Möglicherweise können dafür die Formularklassen der PHPLib (siehe Kapitel 15.3, Seite 45) mit relativ wenig Aufwand adaptiert werden.

14.4.3 PDF Generierung

Wie in Variante 2 werden die PDFs mit XSL-FO erzeugt.

14.4.4 Vor- und Nachteile der Variante

\+

- Es muss kein kommerzieller PDF-Generator angeschafft werden, da die Generierung mit XSL-FO erfolgt.
- Diverse PHPLib (siehe Kapitel 15.3, Seite 45) Klassen können weiterverwendet werden.

–

- Da es noch keine Erfahrungen mit XSL-FO gibt besteht ein gewisses Risiko, dass der Funktionsumfang nicht ausreichend ist.
- Da die DOM XML-Funktionen nicht verwendet werden ist die Variante nicht ganz so flexibel wie die Variante 2.

14.5 Variante 4: Oracle XML-Output

Prinzipiell wäre es auch möglich die XML-Funktionen von Oracle für die Generierung von XML-Dokumenten zu verwenden. Anschließend könnte aus diesem XML-Dokument wie in Variante 2 und 3 mit XSL-FO ein PDF erzeugt werden.
Diese Methode zwingt aber dazu alles Oracle-basierend zu entwickeln. Darum ist diese Methode datenbankspezifisch und sie ermöglicht keine Trennung von Datalayer und Businesslayer.

Sie eignet sich daher nicht zur Umsetzung für das Projekt FondsDoc und soll hier nur der Vollständigkeit halber erwähnt werden.

14.6 Auswahl einer Variante

Es wurde entschieden die Variante 3 zu verwenden.

14.6.1 Entscheidende Vorteile

Ausschlaggebend für die Auswahl der Variante 3 war vor allen die Flexibilität die mit der Speicherung der Dokumentstrukturen als XML gegeben ist. Weiters konnte XSL-FO doch mit einem sehr großen Funktionsumfang überzeugen. In Zukunft könnte es durchaus interessant sein, die Dokumente nicht nur als PDF auszugeben, sondern auch aus PS direkt an Drucker zu schicken, bzw. könnten andere Ausgabeformate für Schnittstellen in anderen Anwendungen von Vorteil sein. Falls doch größere Mängel festgestellt werden, wird man ersatzweise auf die PDFLib zurückgreifen.

14.6.2 Weitere Beweggründe

Ein weiterer Beweggrund die Variante 3 zu nehmen, war die Möglichkeit einen Großteil der Anwendung mit abgeleiteten Klassen der PHPLib (siehe Kapitel 15.3, Seite 45) zu schreiben. So fügt sich diese Anwendung bestens in die einheitliche Linie ein, die mit der PHPLib geschaffen wurde.

Kapitel 15

Die Entwicklungsumgebung

15.1 Linux

Die gesamte Anwendung wird auf einem Linux Server laufen.

15.2 PHP

Wie sämtliche andere interne Anwendungen soll die Anwendung mit PHP programmiert werden. Das ist wichtig, da man sich intern geeinigt hat sämtliche Anwendungen mit PHP zu erstellen. Durch diese einheitliche Linie soll die Wartung, Erweiterung und gegebenenfalls die Verknüpfung der Anwendungen vereinfacht werden. Die derzeitige Entwicklung erfolgt mit der PHP-Version 5.1.

15.3 Verwendung und Erweiterung der PHPLib(intern erstelltes Framework)

Die PHPLib ist ein speziell für Eigenentwicklungen erstelltes Framework, das sämtliche PHP Klassen zum Erstellen von Anwendungen nach dem Model-View-Controller Paradigma enthält. Diese Klassen werden laufend weiterentwickelt und erweitert. Ziel ist es, eine einheitliche Linie zu schaffen, auf der diverse Anwendungen aufgebaut werden können. Somit sollen immer wieder kehrende Problemstellungen Automatisiert werden. Die einheitliche Verwendung der gleichen Klassen vermeidet weiters das Auftreten von typischen Fehlern und beschleunigt den Entwicklungsprozess.

15.3.1 Aufbau und Konzept der PHPLib

Klassen für die View

Alle Klassen die einen für den User (oder ggf. eine andere Schnittstelle) sichtbaren Output erzeugen werden von der Klasse XObj abgeleitet. XObj erzeugt einen XML-String der den logischen Aufbau des Outputs, sei es ein Formular, eine Menüstruktur oder eine Tabelle mit Werten aus einer Datenbank, wiedergibt. Dieser XML-String kann dann weiterverarbeitet werden. Er wird z.B. für die Anzeige im Browser wird dieser String mit XSLT-Stylesheets in HTML umgewandelt.

Von XObj abgeleitet sind z.B. die Klassen Form und FormField. Mit ihnen wird ein HTML-Formular dargestellt. Die Klasse Form liest auch Benutzereingaben zur Weiterverarbeitung ein.

Klassen für den Controller

Wichtige Klassen wie DBConnector und DBQuery automatisieren und vereinheitlichen den Aufbau von Datenbankverbindungen und die Erstellung von Datenbankabfragen zu verschiedenen Datenbanken (u.A. Postgres und Oracle). Sie sind nicht von XObj abgeleitet, da sie nie direkt Output erzeugen. Stattdessen gibt es Klassen mit geeigneten Schnittstellen, die die Darstellung, wie oben erläutert, übernehmen.
Ein besonderer Vorteil der PHPLib ist das Interface „Transactioneer“. Es ermöglicht Transaktionalität über den Durchlauf einer Webanwendung. Zu Beginn der Anwendung wird das Kommando 'begin' gesetzt, am Ende 'commit'. Nur wenn keine Fehler aufgetreten sind, werden Änderungen in Datenbanken auch wirklich vorgenommen. So kann eine Konsistenz der Daten gewährleistet werden. Die Transaktioneerklasse ermögliche auch Transaktionen über mehrere Datenbankverbindungen zu verschiedenen Datenbanken.
Für den Lese- oder Schreibzugriff auf Dateien gibt es spezielle File-Storage-Klassen.

Model

Das Model ist Teil der jeweiligen Anwendung in im Framework nicht definiert. Wie oben bereits erwähnt gibt es aber eine Reihe von Controller-Klassen für Datenbankanbindungen und Dateizugriffe.

Das Konzept für die PHPLib wurde von Hannes Kolar 2005/2006 entwickelt. Derzeit wird die PHPLib in der Bank Gutmann für diverse interne Anwendungen erfolgreich eingesetzt.

15.3.2 Verwendung der PHPLib im Projekt FondsDoc

Für das Projekt FondsDoc weiterverwendet werden die Klassen Form und FormField. Sie werden so abgeleitet, dass sie die FondsDocSchemata zum Erstellen der Dokumente darstellen und editierbar machen.
Weiters werden sämtliche Klassen zum Aufbau von Datenbankverbindungen und zum Erstellen von Datenbankabfragen verwendet.

15.4 FondsDB

Zur Speicherung aller Fonds-relevanten Daten gibt es eine Postgres-Datenbank.

15.5 Browser

Es sollte nach Möglichkeit Wert auf Ausgabe von kompatiblem HTML gelegt werden, um Probleme bei einem eventuellen Umstieg auf einen anderen Browser zu vermeiden.

Kapitel 16

Funktionen des User-Interfaces

Hier soll kurz aufgelistet werden, welche Funktionen das User-Interface umfassen muss und wie es aussehen soll.

16.1 Dokumentstrukturen überarbeiten

Bestehende Dokumentstrukturen müssen laufend überarbeitet und aktualisiert werden können.

16.1.1 Absätze

Einerseits sollen neue Absätze hinzugefügt und bestehende gelöscht werden können. Andererseits soll es auch möglich sein bestehende Absätze zu editieren. Darunter fällt nicht nur vorhandenen Text zu ändern, sonder auch zu formatieren. Es soll möglich sein Fettschreibung, Kursivschreibung, verschiedene farbliche Auszeichnungen und eine Auswahl von vorformatierten Formatvorlagen auf beliebige Teile des Textes anzuwenden.
Bei den Absätzen muss noch zwischen Kapitelabsätze und reinen Textabschnitten unterschieden werden. Textabschnitte können nur Formatierten Text und Platzhalter enthalten. Kapitel enthalten weiters eine Überschrift, Unterabsätze (die ihrerseits wieder Kapitelabsätze sein können), sowie Regeln zur Nummerierung der Unterabsätze.

16.1.2 FondsDocPlatzhalter

Platzhalter stehen für die Fondspezifischen Texte und Werte im Text die aus der Fonds-Datenbank bezogen werden. Das Einfügen dieser Platzhalter ist eine der wesentlichsten Funktionen der Anwendung. Platzhalter können in jedem Absatz an bestimmten Positionen um Text eingefügt werden. Bestehende Platzhalter müssen editiert und gelöscht werden können. Beim Einfügen und Editieren ist zu beachten, dass es sowohl vordefinierte Platzhalter als auch völlig neu erstellbare Platzhalter gibt. Bei den neu erstellbaren muss der Benutzer ein SQL-Statement angeben, das den gewünschten Wert zurückgibt. Ein genaues Prüfen der Benutzereingabe ist hier nötig.

16.1.3 FondsDocConstraints

Ein genaues Zuordnen einzelner Absätze zu bestimmten Fonds muss ermöglicht werden. Dazu können zu jedem Absatz FondsDocConstraints angegeben werden. Jeder FondsDocConstraint

setzt sich zusammen aus einem Datenbankfeld, einem Operator und einem Wert. Dadurch können Absätze einem einzelnen Fond, einer Fondsgruppe oder allen Fonds zugeordnet werden. Hier ist es im Interface besonders wichtig dem Benutzer einen Überblick zu geben, welche Absätze für welchen Fond ausgegeben werden. Es wird sicher nötig sein abhängig von den Fond-IDs alle Absätze die nicht generiert werden ein- bzw. auszublenden.

16.2 Dokumentstrukturen von Bestehenden Strukturen ableiten

Damit die einzelnen FondsDocSchemata überschaubar bleiben, wird es hin und wieder sinnvoll sein für eine Gruppe von Fonds ein neues FondsDocSchemata zu erstellen. Um zu vermeiden, dass alles neu angelegt werden muss, kann der Benutzer ein gesamtes FondsDocSchemata kopieren und anschließend anpassen.

16.3 Neue Dokumentstrukturen entwerfen

Falls noch keine Bestehenden FondsDocSchemata vorhanden sind, die angepasst werden können, muss ein neues FondsDocSchemata erstellt werden. Dazu wird ein leeres Grundgerüst erzeugt, dem der Benutzer dann Absätze hinzufügen kann.

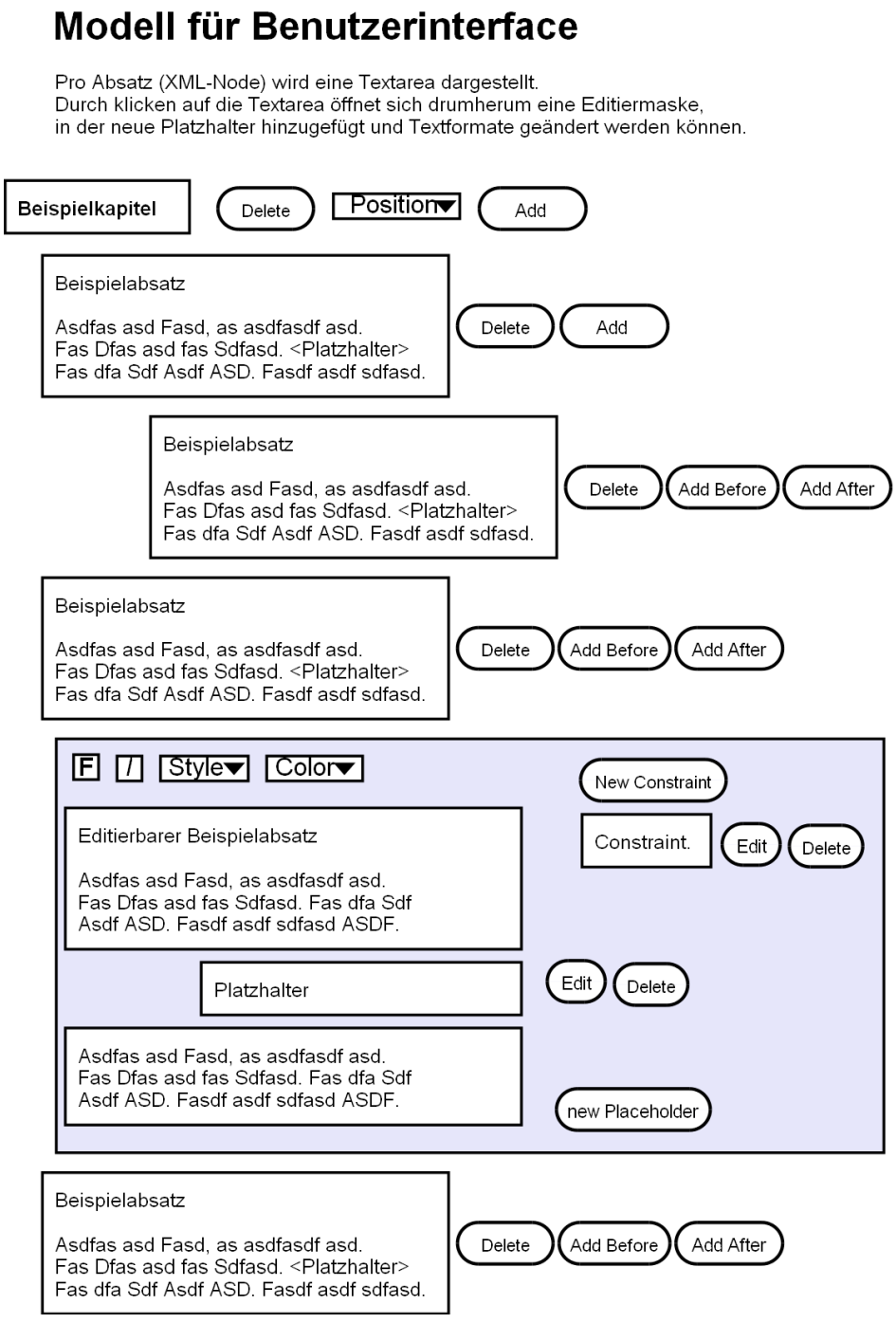

Abbildung 16.1: Interface

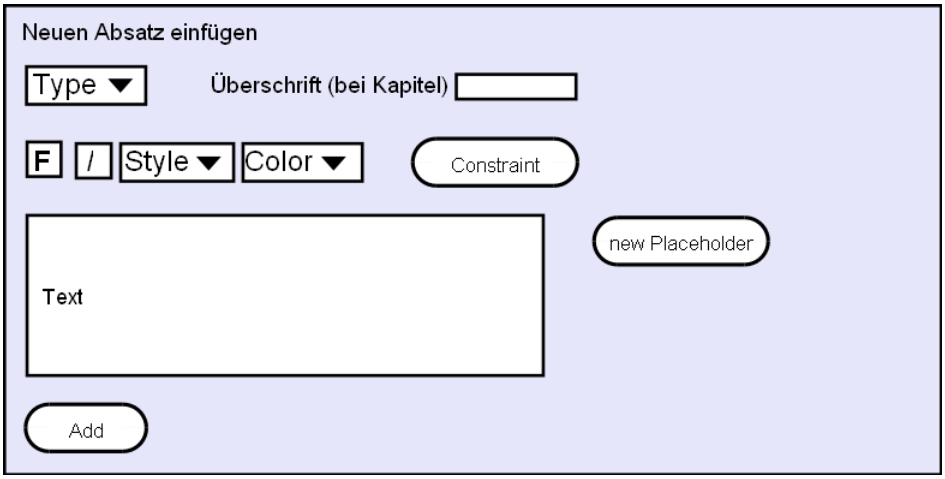

Abbildung 16.2: Neuen Absatz einfügen

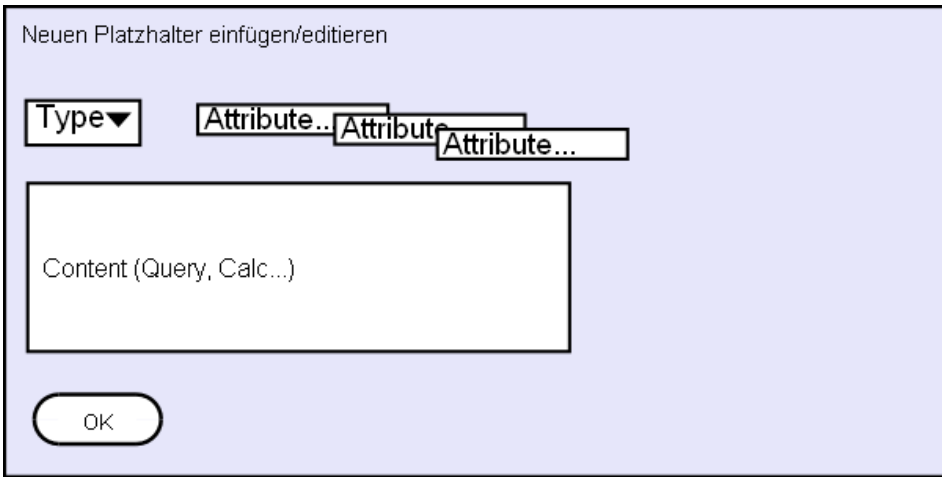

Abbildung 16.3: Neuen Platzhalter einfügen

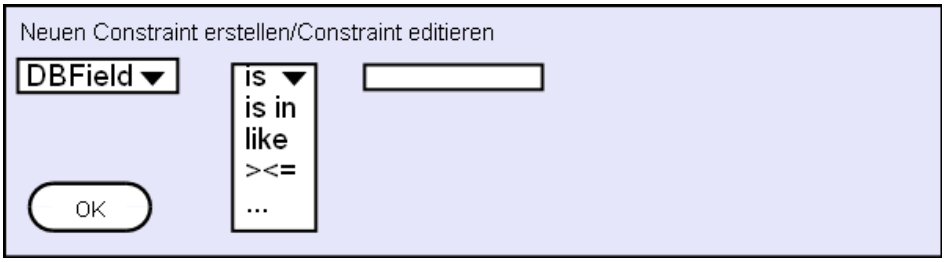

Abbildung 16.4: Neuen Constraint einfügen

Teil III

Aufbau der Anwendung

Kapitel 17

Model

Das Model von FondsDoc setzt sich auch 2 Teilen zusammen: Die Fonds-Datenbank und die XML-Dokumente mit den FondsDocSchemata.

17.1 Die Fonds-Datenbank

Sie liefert alle Daten zu einem Fond. Sie fungiert als Datenquelle der Platzhalter.

17.2 Die XML-Dokumente mit den FondsDocSchemata

Sie bilden den Kern der Anwendung, da sie alle Informationen enthalten, die benötigt werden um ein fertiges Dokument zu erzeugen. Der Benutzer kann über das Benutzer-Interface auf diese XML-Dokumente zugreifen, sie editieren und verwalten.
Diese XML-Dokumente werden selbst auch in einer Postgres Datenbank als Text gespeichert. Das bietet diverse Vorteile als sie einfach so im Dateisystem abzuspeichern:
Die Datenbank kann leichter gesichert werden: Es wurden bereits Backuproutinen dafür erstellt. Falls nötig können auch alle Änderungen mitprotokolliert werden, was einer Art Versionskontrolle entspricht. Methoden für eine solche Versionskontrolle wurden bereits implementiert und können verwendet werden. Weiters können auf die Datenbank Zugriffsrechte vergeben werden. So wird verhindert, dass die Dokumente absichtlich oder unabsichtlich verändert werden. Ein Ändern der XML-Dokumente ohne die in der Anwendung implementierten Regeln und Fehlerüberprüfungen, würde die Dokumente unbrauchbar machen, da die Dokumente einem ganz bestimmten Aufbau haben müssen damit sie korrekt zu einem PDF weiterverarbeitet werden können.

Hier ein Beispiel für ein FondsDocSchema in XML:

```
<fondsdocschema name="Beispieldokument">
    <Kapitel Header="Beispielüberschrift" Constraint="NULL" >
        <Absatz Header="Beispielüberschrift" Constraint="FondID < 10">
            text text text text text text text text text
            text text <Platzhalter name="Fondname" /> text
            text text text text text text text text text
        </Absatz>
```

```
        <Absatz>
            text text text text text text text text text
            <Grafik Typ="statisch" Quelle="/img/Beispiel.jpg" />
            text text text text text text text text text
        </Absatz>
    </Kapitel>
    <Kapitel>
        <Kapitel Constraint="NULL">
            <Absatz Header="Beispielüberschrift" Constraint="NULL">
                text text text text text text text text text
                text text <Platzhalter name="Fondname" /> text
                text text text text text text text text text
            </Absatz>
            <Absatz Constraint="NULL">
                text text text text text text text text text
                text text text text text text text text text
                text text text text text text text text text
            </Absatz>
        </Kapitel>
        <Absatz Constraint="NULL">
            text text text text text text text text text
            text text text text text text text text text
            text text text text text text text text text
        </Absatz>
    </Kapitel>
</fondsdocschema>
```

Kapitel 18

Controller

Der Controller erfüllt mehrere Aufgaben, die hier im Detail erklärt werden sollen. Den Aufgaben entsprechend werden einige PHP-Klassen erstellt, welche die PHPLib (siehe Kapitel 15.3, Seite 45) erweitert. Sie dienen dazu die XML-Dokumente zu erzeugen und zu editieren, aus denen später die fertigen PDFs erzeugt werden.
Im Controller die passiert Umwandlung der allgemeinen XML-Dokumente zu fertigen PDFs. Dazu wird das FondsDocSchema zuerst geladen und für jedes XML-Element wird eine Klasse instanziert, welche die weitere Verarbeitung übernimmt. So werden die Constraints und Platzhalter aller Absätze verarbeitet und aufgelöst. Hier passieren alle Abfragen gegen die Fonds-Datenbank. Als Ergebnis liegt ein neues XML-Dokument vor: Es enthält nur noch die Absätze die zu einem Fonds gehören und anstelle sämtliche Platzhalter stehen nun die Daten des jeweiligen Fonds.
Dieses XML-Dokument kann jetzt mit passenden XSLT-Stylesheets in ein XSL-FO Dokument umgewandelt werden. Dann enthält es alle Style-Informationen die ein XSL-FO-Prozessor benötigt um es in ein PDF umzuwandeln.

18.1 Erklärungen der Klassen zum Generieren eines XML-Dokumentes

Die 3 Hauptklassen des Modells sind XML_Doc, XML_Node und XMl_Attr, die dazu dienen ein XML-Dokument aufzubauen.

18.1.1 XML_Doc

Diese Klasse repräsentiert die Generierung eines vollständigen XML-Dokumentes. Ihr wird ein XML-Dokument übergeben und das Root-Element zugewiesen, von dem ausgehend rekursiv der gesamte XML-Output erzeugt wird. Dazu wird für jedes XML-Element eine XML_Node-Klasse instanziert.
Zur Generierung eines fertigen fondspezifischen XML-Dokumentes ist es weiters notwendig alle Platzhalter und Constraints aufzulösen. Das passiert in der Methode buildReferences(FondID), die sozusagen den Kern der Anwendung darstellt. Wir diese Methode, mit einer bestimmten FondID aufgerufen so sorgt sie dafür, dass alle XML_Node Klassen die Constraints oder Platzhalter beinhalten Abfragen den die Datenbank schicken und so statt den allgemeinen Definitionen konkrete Werte für den gewünschten Fond zurückgeben.

18.1.2 XML_Node

Wie der Name schon sagt, wird mit dieser Klasse ein einzelnes XML-Tag gebaut. Alle nötigen Funktionen wie:

- Name und Inhalt des Tags ändern,
- Constraints setzen,
- Das Tag an bestehende Tags anhängen (Methode add_subnodes(), bzw. append())
- Das Tag löschen (Methode delete()),
- Vordefinierte Style-Attribute anhängen (Methode set_Style()) und
- Platzhalter in den Text einfügen (Methode add_Placeholder())

werden hier implementiert.
Von der Basisklasse abgeleitet werden diverse Subklassen, die Spezialfälle darstellen. Beispielsweise SectionNode (Diese Klasse repräsentiert das XML-Tag `<Absatz>`) und ChapterNode (Repräsentiert das XML-Tag `<Kapitel>`)die selbst ein Absatz ist, der wiederum Unterabsätze enthalten kann. Um diesen Unterschied zwischen einfachen Absätzen und Absätzen mit Unterabsätzen auch im XML zu verdeutlichen gibt es die Aufteilung in `<Kapitel>` und `<Absatz>`. Grund der Trennung ist ein effektiveres Verarbeiten, da so bei einem `<Kapitel>` schon von Anfang an klar ist, dass es weitere, verschachtelte Absätze enthält.
Weiters gibt es XML_PlaceholderNode und XML_QueryPlaceholderNode, die die Platzhalterfunktion übernehmen und entweder vordefinierte Platzhaltertypen (XML_PlaceholderNode) oder eine SQL-Query für einen neuen Platzhalter enthalten(XML_QueryPlaceholderNode).
Je nach Bedarf können von XML-Node noch weitere Klassen abgeleitet werden die dann andere XML-Elemente repräsentieren.

18.1.3 XML_Attr

Ein einfaches XML-Attribut mit einem Namen und einem Wert wird mit dieser Klasse dargestellt. Davon werden, wie schon bei der XML_Node, Ableitungen für Spezialfälle gemacht. Zum Beispiel XML_StyleAttr, ein Attribut das entweder einen vordefinierten Stil oder weitere Informationen zur Formatierung enthält.

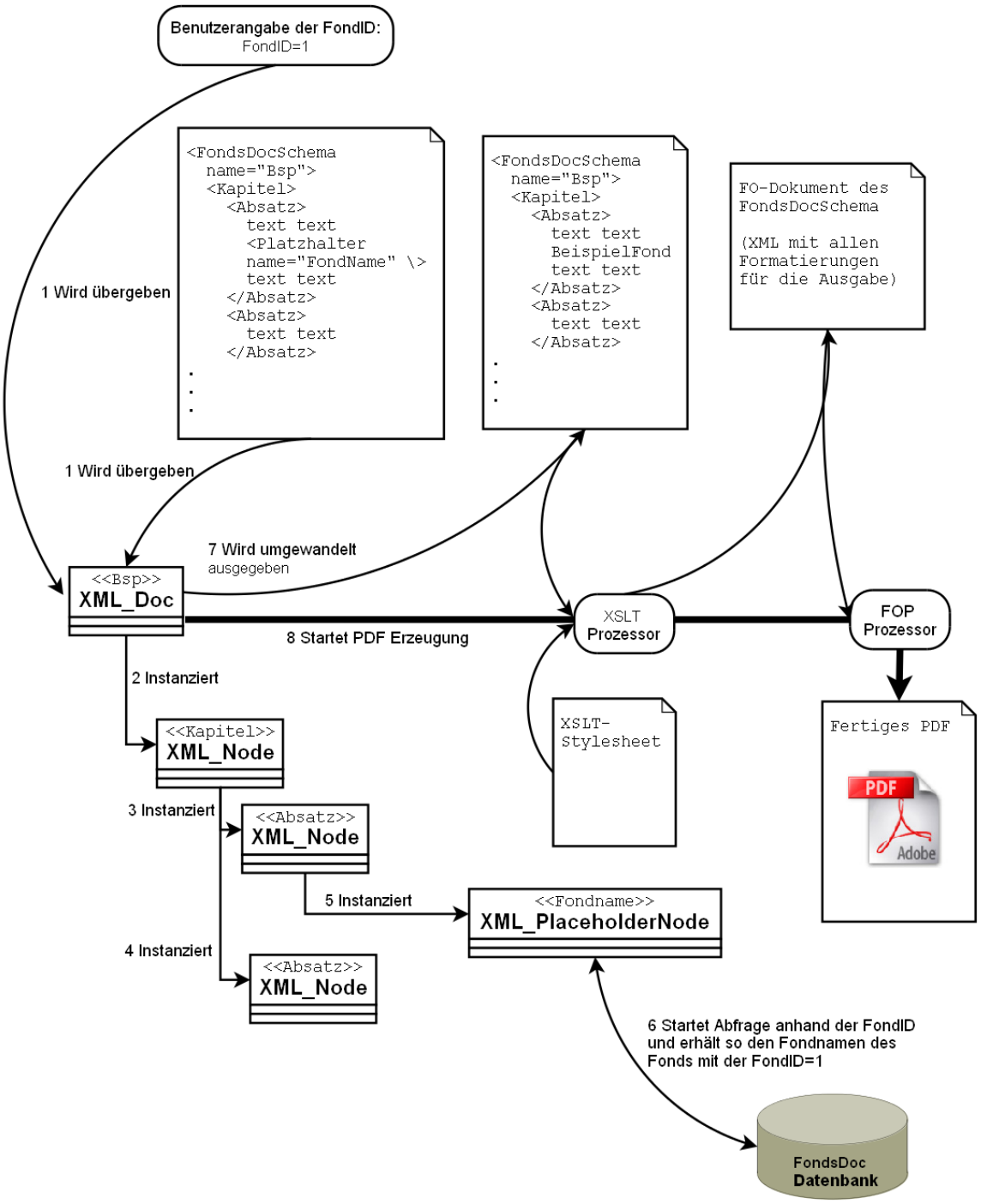

Abbildung 18.1: Die Grafik veranschaulicht wie aus einem FondsDocSchema ein PDF erzeugt wird:
1: Die Klasse XML_Doc wird mit FondID und FondsDocSchema aufgerufen.
2-4: Die Klasse XML_Doc instanziert für jeden Absatz und für jedes Kapitel eine XML_Node. XML_Node-Klassen instanzieren ihrerseits wieder XML_Node für alle Unterelemente, bis es für alle Elemente im XML-Baum eine XML_Node gibt.
5: Für den Platzhalter wird eine XML_PlaceholderNode erstellt.
6: Die XML_PlaceholderNode enthält eine Methode mit der eine Datenbankabfrage gestartet wird um so den Fondnamen für die FondID 1 zu erhalten.
7: Nun kann die Klasse XML_Doc ein fondspezifisches XML-Dokument ausgeben.
8: Die PDF Erzeugung passiert in zwei Schritten: Zuerst wird aus dem fondspezifischen XML-Dokument und einem XSLT-Stylesheet von XSLT Prozessor ein FO-Dokument generiert. Anschließend verarbeitet der FOP-Prozessor das FO-Dokument zu einem PDF.

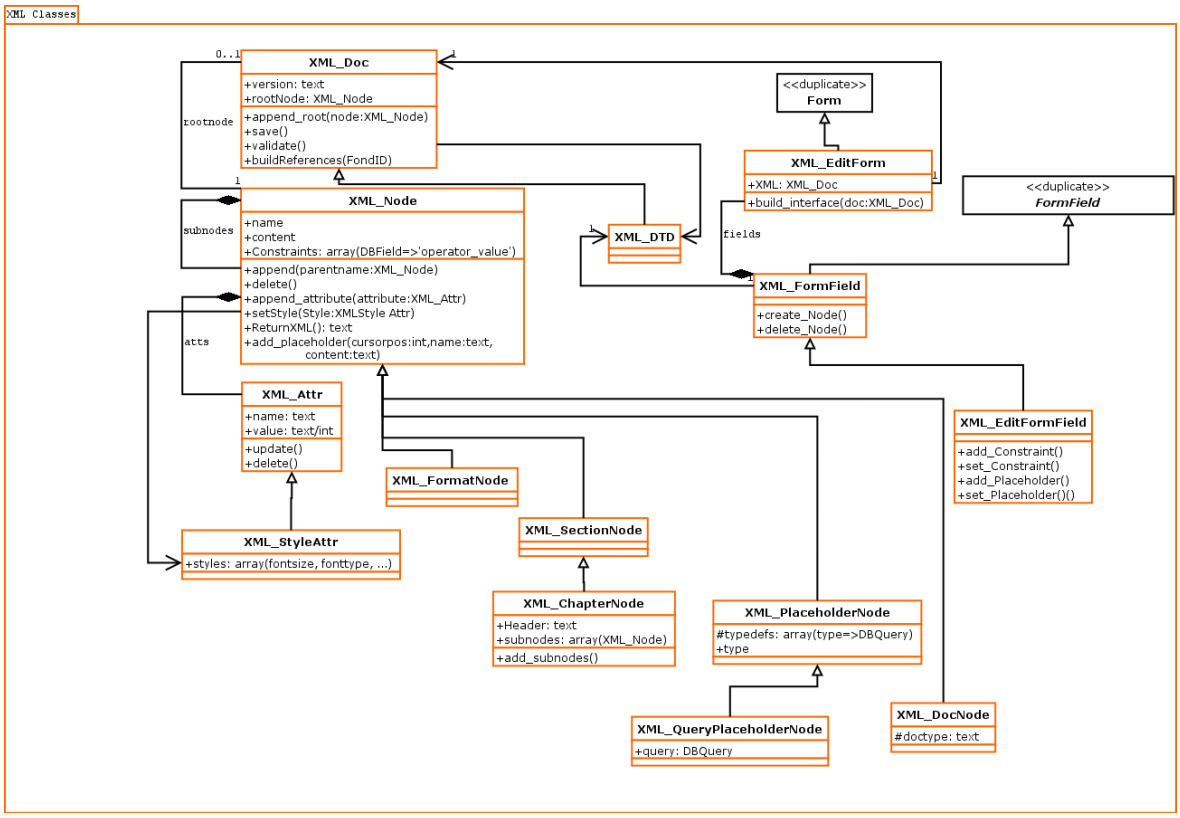

Abbildung 18.2: Modell der für die Anwendung relevanten PHP Klassen

Kapitel 19

View

Streng genommen dient die View nur zum Darstellen der Daten aus dem Model. Da aber in der Anwendung die dargestellten FondsDocSchemata auch gleich editiert werden können sind Darstellung und Benutzereingaben eng verbunden und sollen für ein besseres Verständnis auch als ein Schritt erklärt werden.

19.1 Erklärungen zu den Klassen zum Aufbau des User-Interfaces

19.1.1 Formularklassen

Neben den Klassen zum Aufbau des XML-Dokumentes sind Klassen nötig, die das User Interface aufbauen. Diese werden von den PHPLib Klassen abgeleitet, erzeugen ihrerseits XML, das mit einem XSLT-Stylesheet umgewandelt ein fertiges HTML-Formular zum editieren eines XML_Docs.

Von der PHPLib-Klasse Form abgeleitet ist XML_Edit_Form, von der PHPLib-Klasse FormField abgeleitet ist XML_FormField. Anhand eines übergebenen XML_Docs baut das XML_EditForm für jede XML_Node ein XML_FormField. So wird das XML_Doc für die Benutzer/Innen visualisiert und editierbar gemacht.

19.1.2 Formularfeldklassen

Jedes generierte Formularfeld repräsentiert also eine Node des XML-Dokumentes. Während die Klasse XML_FormField nur den Inhalt der Node richtig formatiert darstellt und Basisfunktionen wie Nodes löschen oder neue hinzufügen, muss es auch eine Möglichkeit geben einzelne Nodes detailliert zu editieren. Das geschieht folgendermaßen:
Sobald der User eine bestimmte Node anklickt wird von der, von XML_FormField abgeleiteten Klasse XML_EditFormField ein erweiterter Eingabebereich geöffnet, in dem Constraints, Platzhalter und Formatierungen gesetzt werden können.

19.2 User Interface zum Formatieren einzelner Absätze

Um die Anwendung möglichst benutzerfreundlich zu gestalten, soll das Editieren der einzelnen Absätze im Großen und Ganzen einem WYSIWYG [1]Texteditor entsprechen. Vorgenommene Textformatierungen sollen möglichst sofort sichtbar gemacht werden, Platzhalter sollen als Grafische Elemente im Fließtext visualisiert werden, um das Aussehen des fertigen Dokumentes so gut als möglich darzustellen. Formatierungen werden aber auch über XML-Tags und Attribute festgelegt. Die Benutzer sollen aber nicht selbst Teile des Textes in XML formatieren, das wäre eine zu große Fehlerquelle und außerdem nicht komfortabel. Deshalb soll ein Editor, mit dem Texte direkt im Browser formatiert werden können, verwendet werden.

Da es bereits eine Vielzahl von WYSIWYG Editoren gibt, ein Großteil auf Javascript basierend, liegt es nahe nicht selbst die gesamte Funktionalität selbst neu umzusetzen, sondern einen bestehenden Editor anzupassen und in das Projekt mit einzubinden.

19.2.1 Whizzywig

Es wurden dazu etliche Editoren verglichen, am Besten geeignet erwies sich „Whizzywig“ (siehe http://unverse.net/), ein freier, javascriptbasierender Editor. Ausschlaggebend für die Wahl waren vor allem ein möglichst einfach einzubindender Code und ein korrekter XHTML Output für die Weiterverarbeitung mit XSLT. Whizzywig entsprach all diesen Anforderungen und kann leicht mit dem XML_EditFormField kombiniert werden. Ein weiterer Vorteil ist, dass Whizzywig das Einfügen von statischen Grafiken unterstützt.

19.2.2 Das Interface von Whizzywig im Detail

Es sollen folgende Funktionen von Whizzywig verwendet werden:

- Textauszeichnung Fett
- Textauszeichnung Kursiv
- Textauszeichnung Unterstrichen
- Text rechtsbündig, linksbündig oder zentriert ausrichten
- Text einrücken/ausrücken
- Farbiger Text
- Farbig hinterlegter Text
- Grafiken einfügen
- Schriftgröße angeben

[1]Abkürzung für „What You See Is What You Get“. Damit ist ein Editor gemeint der während der Bearbeitung eines Dokumentes am Bildschirm genauso anzeigt, wie es dann auch bei der Ausgabe z.B. auf einem Drucker aussieht.

19.2.3 Anpassungen von Whizzywig an das Projekt FondsDoc

Damit der Editor wirklich eingesetzt werden kann, sind einige Anpassungen und Erweiterungen nötig.

Platzhalter

Der Editor muss um die Funktionalität erweitert werden, FondsDocPlatzhalter in den Text einzufügen. Dazu soll in den Fließtext auf Knopfdruck der gewünschte FondsDocPlatzhalter (in der Form`<Platzhalter name="..."/>`) eingefügt werden. Anstellt des XML-Codes soll aber eine Platzhalter-Symbol-Grafik angezeigt werden.

Vordefinierte Styles

Neben den verwendeten Textauszeichnungen soll der Editor auch über eine Auswahl an vordefinierten Styles verfügen. Dazu muss Whizzywig um einige Schaltflächen erweitert werden, die dann die gewünschten Style-Tags in den Text einfügen.

Dynamische Grafiken

Da nicht nur statische Grafiken, sondern auch dynamisch erstellte Balkengrafiken benötigt werden, müssen über Whizzywig auch XML-Tags zur Definition dieser Balkengrafiken eingefügt werden können. Sie sollen ebenfalls mit Symbol-Grafiken veranschaulicht werden.

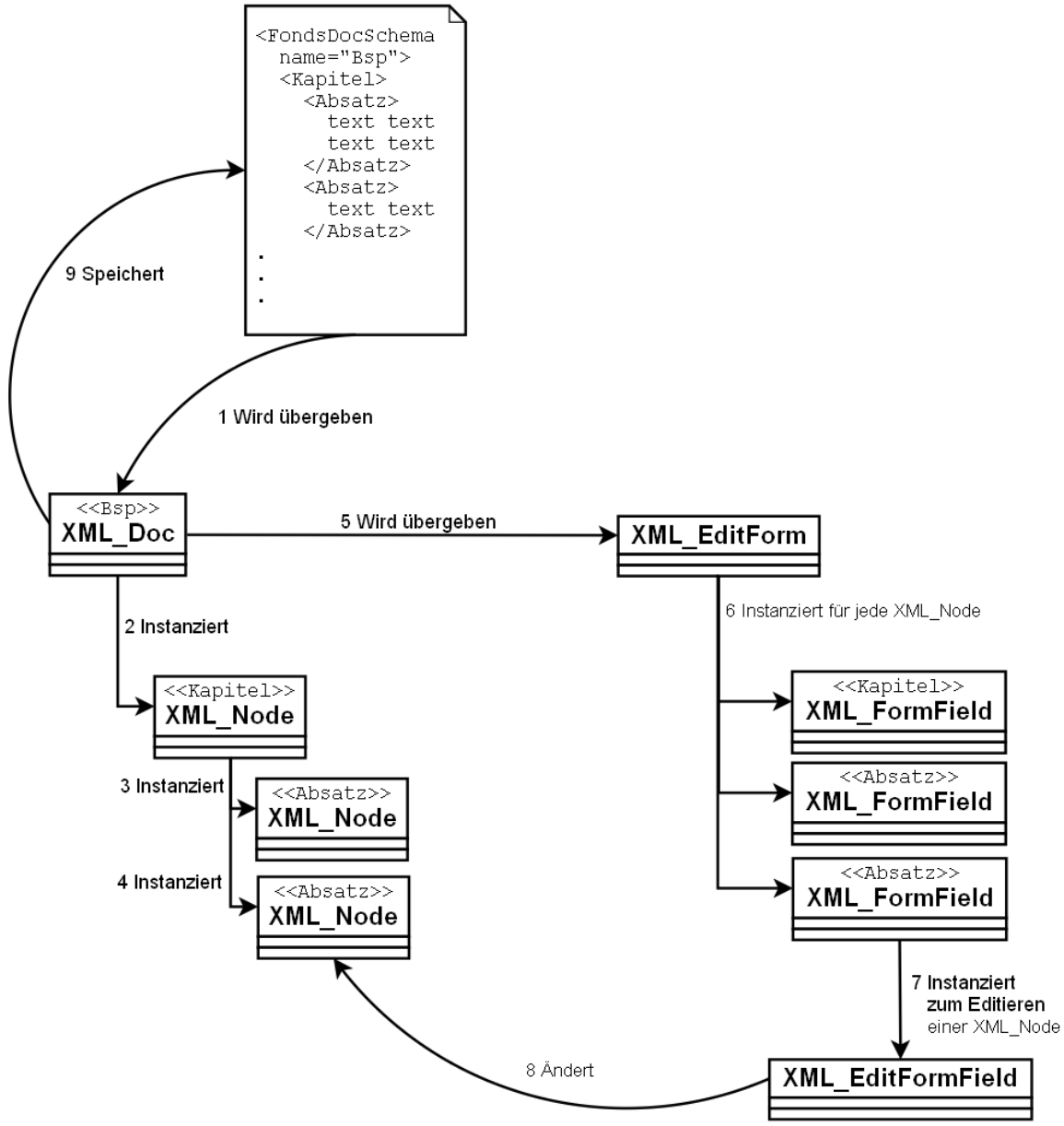

Abbildung 19.1: Hier wird Schritt für Schritt Dargestellt wie ein Absatz in einem FondsDocSchema geändert werden kann:
1: Das zu ändernde FondsDocSchema wird an eine XML_Doc-Klasse übergeben.
2-4: Die XML_Doc-Klasse instanziert für jedes Element im XML-Baum des FondsDocSchemas eine XML_Node-Klasse.
5: Es wird ein XML_EditForm mit der XML_Doc-Klasse aufgerufen
6: Das XML_EditForm erstellt für jede XML_Node-Klasse ein XML_FormField, das den Text für die Benutzer darstellt.
7: Soll ein Text editiert werden, wird dazu ein XML_EditFormField instanziert, das Benutzereingaben übernimmt und den geänderten Text an die XML_Node übergibt.
8: XML_Doc speichert das geänderte FondsDocSchema wieder als XML-Datei.

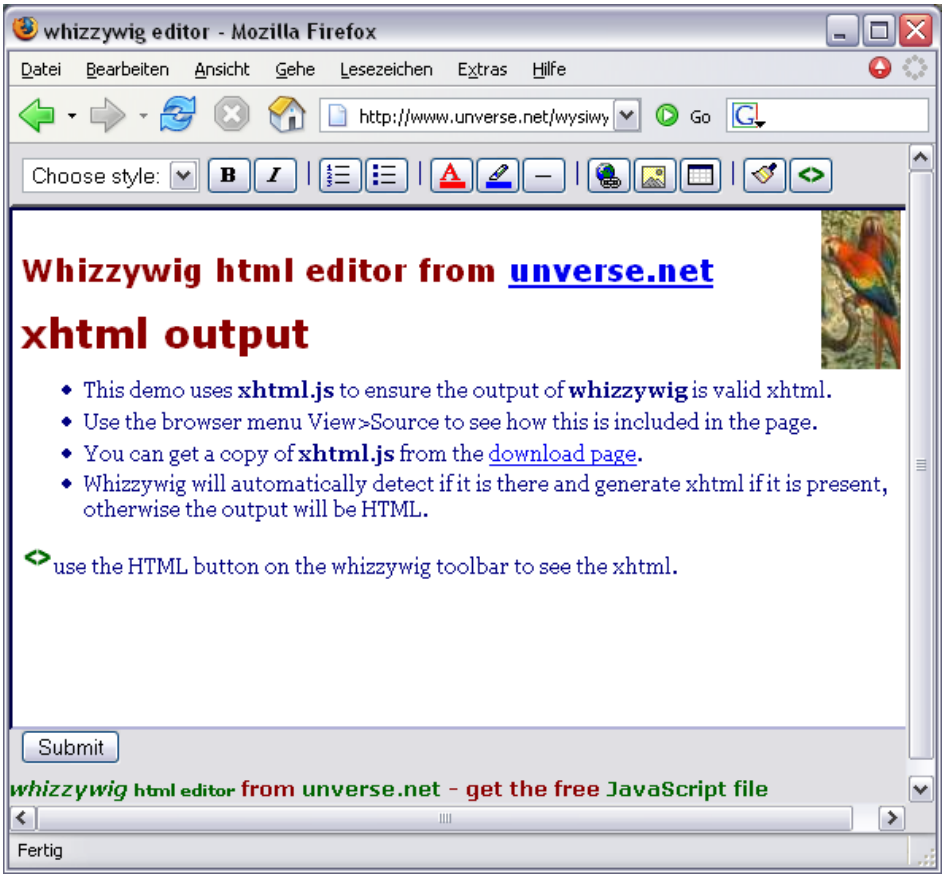

Abbildung 19.2: Whizzywig Screenshot

Kapitel 20

PDF-Generierung

Die PDF Generierung mit XSL-FO geschieht nach dem Schema, das bereits in Kapitel 8.2, Seite 19 erklärt wurde

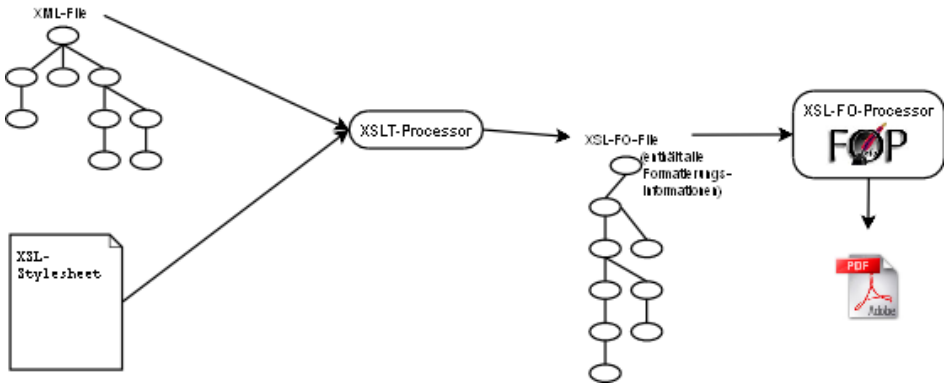

Abbildung 20.1: Erzeugung eines PDFs aus einer XML-Datei mit FOP

Hier soll dieser Ablauf anhand eines Beispiels erklärt werden. Ausgangsdokument ist folgendes XML-File:

```
<?xml version="1.0" encoding="UTF-8"?>
<fondsdocschema name="Beispieldokument">
<Kapitel Header="Kapitelüberschrift">

    <Absatz Header="Absatzüberschrift " >
        Beispieltext im Absatz 1. Beispieltext im Absatz 1.
        Beispieltext im Absatz 1.Beispieltext im Absatz 1.
        Beispieltext im Absatz 1.Beispieltext im Absatz 1.
        Beispieltext im Absatz 1.
    </Absatz>
    <Absatz Header="Absatzüberschrift">
        Beispieltext im Absatz 2.  Beispieltext im Absatz 2.
        Beispieltext im Absatz 2.  Beispieltext im Absatz 2.
    </Absatz>
</Kapitel>
```

```
</fondsdocschema>
```

20.1 Aufbau des XSL-FO Stylesheets

Das XML-File soll so umgewandelt werden, das es vom XSL-FO Prozessor (in diesem fall „Apache FOP“) verarbeitet werden kann. Dazu muss es in ein .fo Dokument folgender Art umgewandelt werden:

```
<?xml version="1.0" encoding="utf-8"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">

    <fo:layout-master-set>
        <fo:simple-page-master master-name="A4"
            page-width="297mm"
            page-height="210mm">
            <fo:region-body region-name="xsl-region-body"
                margin="2cm"
                column-count="2"/>
            <fo:region-before extent="1cm" padding-left="1cm"/>
            <fo:region-after extent="1cm" padding-left="1cm"/>
            <fo:region-start extent="2cm" padding-left="1cm"/>
            <fo:region-end extent="2cm" padding-left="1cm"/>
        </fo:simple-page-master>
    </fo:layout-master-set>

    <fo:page-sequence master-reference="A4">
         <fo:static-content flow-name="xsl-region-after">
            <fo:block>Seite: <fo:page-number/></fo:block>
         </fo:static-content>

        <fo:flow flow-name="xsl-region-body">
           <fo:block font-size="18pt" font-weight="bold">
               Beispieldokument
           </fo:block>
           <fo:block font-size="14pt" font-weight="bold">
               Kapitelüberschrift
           </fo:block>
           <fo:block font-size="12pt" font-weight="bold">
               Absatzüberschrift
           </fo:block>
           <fo:block font-size="10pt">
                Beispieltext im Absatz 1. Beispieltext im Absatz 1.
                Beispieltext im Absatz 1. Beispieltext im Absatz 1.
                Beispieltext im Absatz 1. Beispieltext im Absatz 1.
                Beispieltext im Absatz 1. Beispieltext im Absatz 1.
            </fo:block>
           <fo:block font-size="12pt" font-weight="bold">
```

```
            Absatzüberschrift
        </fo:block>
        <fo:block font-size="10pt">
            Beispieltext im Absatz 2. Beispieltext im Absatz 2.
            Beispieltext im Absatz 2. Beispieltext im Absatz 2.
        </fo:block>
    </fo:flow>
</fo:page-sequence>

</fo:root>
```

20.2 Aufbau des XSLT zur Erzeugung des XSL-FO

Die Umwandlung von XML zu einem FO-Dokument passiert mithilfe einer XSLT-Transformation. Folgendes Stylesheet wird auf das XML-Dokument angewendet:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">

    <xsl:template match="fondsdocschema">
        <fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">

            <fo:layout-master-set>
                <fo:simple-page-master master-name="A4"
                    page-width="297mm"
                    page-height="210mm">
                    <fo:region-body region-name="xsl-region-body"
                        margin="2cm"
                        column-count="2"/>
                    <fo:region-before extent="1cm" padding-left="1cm"/>
                    <fo:region-after extent="1cm" padding-left="1cm"/>
                    <fo:region-start extent="2cm" padding-left="1cm"/>
                    <fo:region-end extent="2cm" padding-left="1cm"/>
                </fo:simple-page-master>
            </fo:layout-master-set>

            <fo:page-sequence master-reference="A4">
                <fo:static-content flow-name="xsl-region-after">
                    <fo:block>Seite: <fo:page-number/></fo:block>
                </fo:static-content>
                <fo:flow flow-name="xsl-region-body">
                    <fo:block xmlns:fo="http://www.w3.org/1999/XSL/Format"
                        font-size="18pt"
                        font-weight="bold">
                        <xsl:value-of select="@name"/>
                    </fo:block>
```

```
                    <xsl:apply-templates/>
                </fo:flow>
            </fo:page-sequence>

        </fo:root>
    </xsl:template>

    <xsl:template match="Absatz">
        <fo:block xmlns:fo="http://www.w3.org/1999/XSL/Format"
            font-size="12pt"
            font-weight="bold">
            <xsl:value-of select="@Header"/>
        </fo:block>
        <fo:block xmlns:fo="http://www.w3.org/1999/XSL/Format"
            font-size="10pt">
            <xsl:value-of select="."/>
        </fo:block>
    </xsl:template>


    <xsl:template match="Kapitel">
        <fo:block xmlns:fo="http://www.w3.org/1999/XSL/Format"
            font-size="14pt"
            font-weight="bold">
            <xsl:value-of select="@Header"/>
        </fo:block>
        <xsl:apply-templates/>
    </xsl:template>

</xsl:stylesheet>
```

Nach der Transformation ist das XML zu dem gewünschten .fo Dokument geworden:

```
<?xml version="1.0" encoding="utf-8"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">

    <fo:layout-master-set>
        <fo:simple-page-master master-name="A4"
            page-width="297mm"
            page-height="210mm">
            <fo:region-body region-name="xsl-region-body"
                margin="2cm"
                column-count="2"/>
            <fo:region-before extent="1cm" padding-left="1cm"/>
            <fo:region-after extent="1cm" padding-left="1cm"/>
            <fo:region-start extent="2cm" padding-left="1cm"/>
            <fo:region-end extent="2cm" padding-left="1cm"/>
        </fo:simple-page-master>
```

```
    </fo:layout-master-set>

    <fo:page-sequence master-reference="A4">
         <fo:static-content flow-name="xsl-region-after">
            <fo:block>Seite: <fo:page-number/></fo:block>
        </fo:static-content>

        <fo:flow flow-name="xsl-region-body">
           <fo:block font-size="18pt" font-weight="bold">
               Beispieldokument
           </fo:block>
           <fo:block font-size="14pt" font-weight="bold">
               Kapitelüberschrift
           </fo:block>
           <fo:block font-size="12pt" font-weight="bold">
               Absatzüberschrift
           </fo:block>
           <fo:block font-size="10pt">
                Beispieltext im Absatz 1. Beispieltext im Absatz 1.
                Beispieltext im Absatz 1. Beispieltext im Absatz 1.
                Beispieltext im Absatz 1. Beispieltext im Absatz 1.
                Beispieltext im Absatz 1. Beispieltext im Absatz 1.
            </fo:block>
           <fo:block font-size="12pt" font-weight="bold">
               Absatzüberschrift
           </fo:block>
           <fo:block font-size="10pt">
                Beispieltext im Absatz 2. Beispieltext im Absatz 2.
                Beispieltext im Absatz 2. Beispieltext im Absatz 2.
            </fo:block>
        </fo:flow>
    </fo:page-sequence>

</fo:root>
```

20.3 Erzeugung des PDFs aus XSL-FO

Das .fo Dokument kann nun vom XSL-FO Prozessor verarbeitet werden und er gibt ein PDF aus.

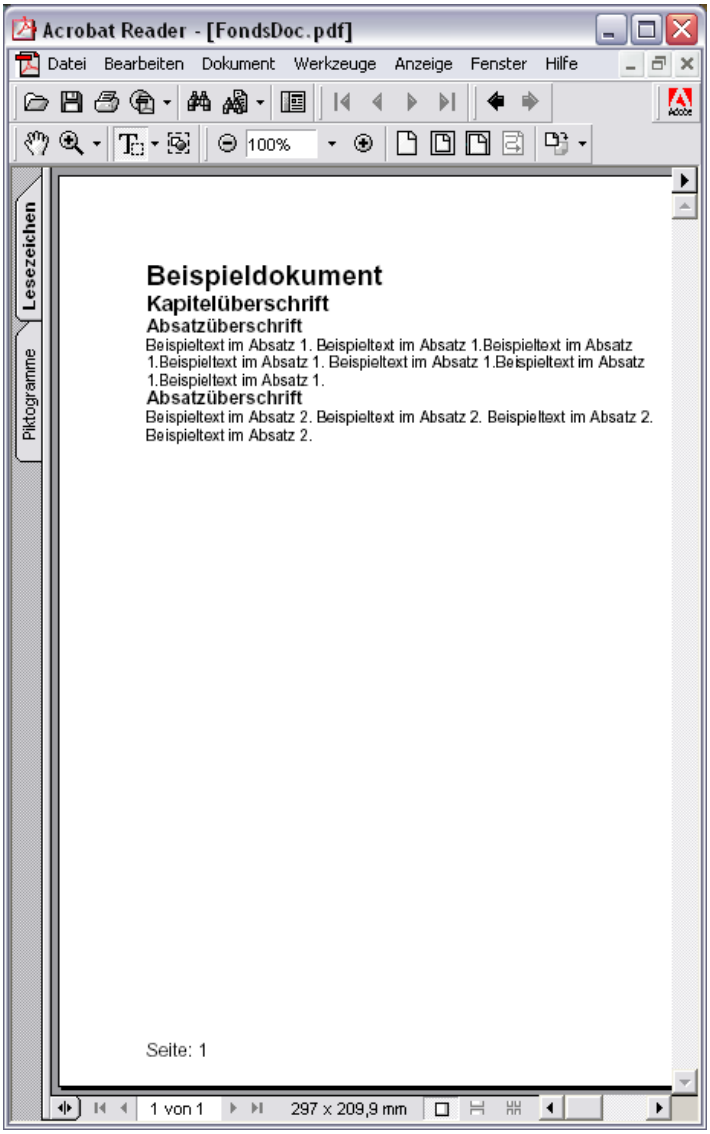

Abbildung 20.2: Mit FOP generiertes PDF

Kapitel 21

Reflexion

Es konnte ein funktionierendes Konzept zur dynamischen Generierung fondsrelevanter Dokumente erstellt werden.
Das Projekt kann mit diesem Konzept vollständig umgesetzt werden.

Abschließend sollen hier die forschungsleitenden Fragestellungen beantwortet werden:

21.1 Strukturdefinitionen der Dokumente

Die Analyse der Dokumente ergab, dass es genügend Übereinstimmungen zwischen den Dokumenten gibt, um allgemeine Schemas für alle Dokumentarten zu definieren. Allerdings folgen nicht alle Dokumente starr einem fix vorgegebenen Schema. Es gibt durchaus einzelne Dokumente die in einem oder mehreren Punkten vom Schema abweichen können, da sie zusätzliche Absätze oder Textpassagen enthalten, bzw. einzelne Absätze nicht benötigen. Deswegen wurden s.g. Constraints eingeführt (siehe Kapitel 16.1.3, Seite 47) mit deren Hilfe Absätze, auf ihr Vorkommen hin, für bestimmte Fonds eingeschränkt werden können.
Die Verwendung von XML erwies sich als geeignet alle Informationen abzuspeichern, die nötig sind um den Aufbau der Dokumente ausreichend zu beschreiben und in der Anwendung weiter zu verarbeiten.
Es wurde mit dem Modell der Klassen XML_Doc, XML_Node und XML_Attr eine Anwendung gestaltet, mit der sämtliche Operationen durchführbar sind, um alle fondrelevanten Dokumente zu generieren.

21.2 Dynamische Textbausteine

Eines der Hauptprobleme bei der Gestaltung der Anwendung, war das Vorkommen Dynamischer Textbausteine innerhalb von Absätzen. Alle diese Platzhalter verweisen auf ein oder mehrere Felder in einer Datenbank. Gelöst wurde das Problem indem die einzelnen Textbausteine als XML-Tags definiert werden. Zu diesem Zweck wurde eine Ableitung der Klasse XML_Node geschrieben, die diese Tags mit allen benötigten Informationen erstellt und auch die SQL-Abfragen zur Auswertung der Tags bereitstellt.
Es stellte sich heraus, dass ein Großteil der Platzhalter wiederholt vorkommt. Somit wurde eine Auswahl von Platzhalter fix implementiert. Diese können vom User direkt eingefügt werden, ein Angeben der benötigten Datenbankfelder oder eines SQL-Statements ist nicht

erforderlich. Zusätzliche dynamische Platzhalter, in denen die Benutzer ein SQL-Statement angeben können stellen eine zu große Fehlerquelle dar und könnten ein Sicherheitsrisiko sein. Sie werden daher nicht, wie ursprüngliche konzipiert (siehe Kapitel 13.1.4) eingebaut.

21.3 Parallele Änderung mehrerer Dokumente

Da für die Dokumente allgemeine Schemas definiert werden können, ist eine Änderung, von der mehrere Dokumente betroffen sind möglich. Wie oben beschreiben können Absätze über Constraints bestimmten Fonds zugewiesen werden, so kann ein Schema beliebig komplex erweitert werden.
Durch diese Art der Dokumentenverwaltung ist es auch nicht nötig wie Anfangs angedacht eine Versionskontrolle zu implementieren. Sollten Absätze für einige Fonds überarbeitet werden, während andere Fonds diese Absätze noch in der ursprünglichen Version benötigen, können ohne Probleme mehrere Fassungen eines Absatzes mit unterschiedlichen Constraints angelegt werden.

21.4 Erhaltung der korrekten Grammatik

Die Erhaltung einer korrekten Grammatik erwies sich nicht als Problem. Platzhalter die dynamisch in den Text eingefügt werden, bestehen nur aus einzelnen Werten wie z.B. Namen oder Datumsangaben. Dadurch ergibt sich nie die Notwendigkeit die Satzgrammatik, abhängig von den eingefügten Werten, zu ändern.

Anhang A

Detailbeschreibungen der Dokumente

Hier werden die zu erstellenden Dokumente analysiert. Es werden folgende Eigenschaften ermittelt:

- Fondspezifisch: Enthält das Dokument Fondspezifische Daten oder ist es für alle Fonds gleich?
- Erstellungszeitpunkt: Wann wird das Dokument erzeugt?
- Regelmäßige Neuerstellung: Wird das Dokument im bestimmten Abständen neu erstellt?
- Regelmäßige Aktualisierung: Enthält das Dokument Daten die regelmäßig aktualisiert werden müssen?
- Wird versendet an: Wird das Dokument versendet? Wohin?
- Altes Dateiformat: In welchen Format wurde das Dokument bisher erzeugt?
- Spezial-/Publikumsfond: Gibt es das Dokument für Spezialfonds für Publikumsfonds oder für beide?
- Unterarten: Gibt es Unterarten des Dokuments?
- CI Relevant: Muss das Dokument dem Design der Corporate Identity der Bank Gutmann entsprechen?
- Signiert von: Wird das Dokument signiert?
- Sprachen: In welcher Sprache ist das Dokument verfasst?
- Verantwortlichkeit: Wer ist für die Erstellung und die Richtigkeit das Dokuments verantwortlich?

Weiters werden Abhängigkeiten zwischen den Dokumenten und Abhängigkeiten der Generierung eines Dokumentes von bestimmten Voraussetzungen (z.B. Genehmigungen) ermittelt:

- Enthaltene Dokumente: Enthält das Dokument andere Dokumente?
- Notwendig für Dokumente: Ist das Dokument in anderen Dokumenten enthalten?
- Voraussetzung für Erstellung: Gibt es bestimmte Anforderungen die zur Erstellung des Dokuments nötig sind?
- Voraussetzung für Veröffentlichung: Gibt es bestimmte Anforderungen die zur Veröffentlichung des Dokuments nötig sind?

A.0.1 (1) Depotbankschreiben

Bestätigung der Depotbank der Akzeptanz der Fondsbestimmungen. Ändert sich wenn Fondsbestimmungen geändert werden oder ein Fonds neu aufgelegt wird.

Eigenschaften

Wichtigste Parameter des Dokumententyps

Fondsspezifisch	Ja. U.u. mehrere Fonds zusammengefasst.
Erstellungszeitpunkt	Fondsstart und Änderung Fondsbestimmungen
Regelmäßige Neuerstellung	Nein
Regelmäßige Aktualisierung	Nein
Wird versendet an	-
Altes Dateiformat	Word
Produzent/Lieferant	KAG/Bank
Spezial-/Publikumsfonds	Beide
Unterarten	(1.1) Fondsgründung (1.2) Änderung Fondsbestimmungen
CI Relevant	Nein
Signiert von	Depotbank PPA
Sprachen (in Summe)	Deutsch
Verantwortlichkeit	Rechtsabteilung

Tabelle A.1: Eigenschaften von (1) Depotbankschreiben

Abhängigkeiten

Beziehungen der Dokumente untereinander

Enthaltene Dokumente	(18) Allgemeine Fondsbestimmungen
	(19) Besondere Fondsbestimmungen
	(20) Anhang zu §16 - Börseliste
Notwendig für Dokumente	-
Voraussetzungen für Erstellung	-
Voraussetzungen für Veröffentlichung	-

Tabelle A.2: Abhängigkeiten von (1) Depotbankschreiben

A.0.2 (2) Sammelaufstellung

Auflistung aller im abgelaufenen Halbjahr neu gegründeten oder aufgelösten Spezialfonds.

Eigenschaften

Wichtigste Parameter des Dokumententyps

Fondsspezifisch	Ja
Erstellungszeitpunkt	Kalenderhalbjahr
Regelmäßige Neuerstellung	Ja
Regelmäßige Aktualisierung	Nein
Wird versendet an	FMA
Altes Dateiformat	-
Produzent/Lieferant	Depotbank (Pucher)
Spezial-/Publikumsfonds	S
Unterarten	Keine
CI Relevant	Nein
Signiert von	Depotbank PPA
Sprachen (in Summe)	Deutsch
Verantwortlichkeit	-

Tabelle A.3: Eigenschaften von (2) Sammelaufstellung

Abhängigkeiten

Beziehungen der Dokumente untereinander

Enthaltene Dokumente	-
Notwendig für Dokumente	-
Voraussetzungen für Erstellung	-
Voraussetzungen für Veröffentlichung	-

Tabelle A.4: Abhängigkeiten von (2) Sammelaufstellung

A.0.3 (3) Sammelurkunde und Beilagebrief

Eigenschaften

Wichtigste Parameter des Dokumententyps

Fondsspezifisch	Ja
Erstellungszeitpunkt	Fondsstart
Regelmäßige Neuerstellung	Nein
Regelmäßige Aktualisierung	Nein
Wird versendet an	OeKB
Altes Dateiformat	-
Produzent/Lieferant	KAG
Spezial-/Publikumsfonds	Beide
Unterarten	Keine
CI Relevant	Nein
Signiert von	KAG Vorstand, KAG Aufsichtsrat Vorsitzender, Depotbank PPA
Sprachen (in Summe)	Deutsch
Verantwortlichkeit	Rechtsabteilung

Tabelle A.5: Eigenschaften von (3) Sammelurkunde und Beilagebrief

Abhängigkeiten

Beziehungen der Dokumente untereinander

Enthaltene Dokumente	-
Notwendig für Dokumente	-
Voraussetzungen für Erstellung	-
Voraussetzungen für Veröffentlichung	-

Tabelle A.6: Abhängigkeiten von (3) Sammelurkunde und Beilagebrief

A.0.4 (4) Produktbeschreibung

Einseitige Zusammenstellung der wichtigsten Angaben, eine Beschreibung des Fonds und Fondmanagements.

Eigenschaften

Wichtigste Parameter des Dokumententyps

Fondsspezifisch	Ja
Erstellungszeitpunkt	Fondsstart
Regelmäßige Neuerstellung	Nein
Regelmäßige Aktualisierung	Bei Änderung der Basisdaten.
Wird versendet an	PM-Web [1]
Altes Dateiformat	-
Produzent/Lieferant	KAG
Spezial-/Publikumsfonds	P
Unterarten	Keine
CI Relevant	Ja
Signiert von	-
Sprachen (in Summe)	Deutsch
Verantwortlichkeit	Fondsmanager

Tabelle A.7: Eigenschaften von (4) Produktbeschreibung

Abhängigkeiten

Beziehungen der Dokumente untereinander

Enthaltene Dokumente	-
Notwendig für Dokumente	-
Voraussetzungen für Erstellung	-
Voraussetzungen für Veröffentlichung	-

Tabelle A.8: Abhängigkeiten von (4) Produktbeschreibung

A.0.5 (5) FMA Antrag

Anträge aller Art bezüglich Fondsgründung, FB Änderung und EU Pass bei FMA.

Eigenschaften

Wichtigste Parameter des Dokumententyps

Fondsspezifisch	Ja
Erstellungszeitpunkt	Fondsstart und Änderung Fondsbestimmungen
Regelmäßige Neuerstellung	Nein
Regelmäßige Aktualisierung	Nein
Wird versendet an	FMA
Altes Dateiformat	Word
Produzent/Lieferant	KAG
Spezial-/Publikumsfonds	P
Unterarten	(5.1) Fondsgründung (5.2) FB Änderung (5.3) EU Pass
CI Relevant	Nein
Signiert von	-
Sprachen (in Summe)	Deutsch
Verantwortlichkeit	Rechtsabteilung

Tabelle A.9: Eigenschaften von (5) FMA Antrag

Abhängigkeiten

Beziehungen der Dokumente untereinander

Enthaltene Dokumente	(18, 19, 20) Fondsbestimmungen Bei (5.3): (17) Vereinfachter Prospekt
Notwendig für Dokumente	-
Voraussetzungen für Erstellung	-
Voraussetzungen für Veröffentlichung	-

Tabelle A.10: Abhängigkeiten von (5) FMA Antrag

A.0.6 (6) Informationen für Deutschland und Schweiz

Einseitige Zusammenstellung der wichtigsten Angaben, eine Beschreibung des Fonds und Fondmanagements.

Eigenschaften

Wichtigste Parameter des Dokumententyps

Fondsspezifisch	Ja. Für die meisten Fonds identisch.
Erstellungszeitpunkt	Fondsstart
Regelmäßige Neuerstellung	Nein
Regelmäßige Aktualisierung	Nein
Wird versendet an	-
Altes Dateiformat	Word
Produzent/Lieferant	KAG
Spezial-/Publikumsfonds	Beide
Unterarten	(6.1) Schweiz, (6.2) Deutschland
CI Relevant	Ja
Signiert von	-
Sprachen (in Summe)	Deutsch
Verantwortlichkeit	Rechtsabteilung

Tabelle A.11: Eigenschaften von (6) Informationen für Deutschland und Schweiz

Abhängigkeiten

Beziehungen der Dokumente untereinander

Enthaltene Dokumente	-
Notwendig für Dokumente	(11) Verkaufsprospekt (12) Rechenschaftsbericht
Voraussetzungen für Erstellung	Verträge mit Partnern in betreffenden Ländern. Vertriebszulassung in DE und CH ??
Voraussetzungen für Veröffentlichung	-

Tabelle A.12: Abhängigkeiten von (6) Informationen für Deutschland und Schweiz

A.0.7 (7) Bericht Fondsmanager

Kommentare zur Marktlage, dem Fonds oder Fondsmanagement verfasst vom Fondsmanager.

Eigenschaften

Wichtigste Parameter des Dokumententyps

Fondsspezifisch	Ja.
Erstellungszeitpunkt	Jahresabschluss
Regelmäßige Neuerstellung	Ja
Regelmäßige Aktualisierung	Nein
Wird versendet an	-
Altes Dateiformat	-
Produzent/Lieferant	Fondsmanager
Spezial-/Publikumsfonds	Beide
Unterarten	Keine
CI Relevant	Ja
Signiert von	-
Sprachen (in Summe)	Deutsch und jeweilige Sprache des Fondsmanagers (wird aber nicht verwendet)
Verantwortlichkeit	Fondsmanager

Tabelle A.13: Eigenschaften von (7) Bericht Fondsmanager

Abhängigkeiten

Beziehungen der Dokumente untereinander

Enthaltene Dokumente	-
Notwendig für Dokumente	(12) Rechenschaftsbericht(nicht essenziell)
Voraussetzungen für Erstellung	-
Voraussetzungen für Veröffentlichung	-

Tabelle A.14: Abhängigkeiten von (7) Bericht Fondsmanager

A.0.8 (8) Ertragsrechnung

Aufstellung der Erträge und Aufwendungen des Fonds in Buchhaltungsform.

Eigenschaften

Wichtigste Parameter des Dokumententyps

Fondsspezifisch	Ja
Erstellungszeitpunkt	Jahresabschluss
Regelmäßige Neuerstellung	Ja
Regelmäßige Aktualisierung	Nein
Wird versendet an	Wirtschaftsprüfer
Altes Dateiformat	Excel
Produzent/Lieferant	Fondsbuchhaltung
Spezial-/Publikumsfonds	Beide
Unterarten	Keine
CI Relevant	Ja
Signiert von	-
Sprachen (in Summe)	Deutsch
Verantwortlichkeit	

Tabelle A.15: Eigenschaften von (8) Ertragsrechnung

Abhängigkeiten

Beziehungen der Dokumente untereinander

Enthaltene Dokumente	
Notwendig für Dokumente	(12) Rechenschaftsbericht
Voraussetzungen für Erstellung	Jahresabschluss in PFS, Ausschüttungsbetrag.
Voraussetzungen für Veröffentlichung	Wirtschaftsprüfer Bestätigung

Tabelle A.16: Abhängigkeiten von (8) Ertragsrechnung

A.0.9 (9) Organe der Ges. u. Depotbank

Auflistung von handelnden Personen in KAG, Depotbank und Fondsmanagement

Eigenschaften

Wichtigste Parameter des Dokumententyps

Fondsspezifisch	Ja (teilweise)
Erstellungszeitpunkt	Fondsstart
Regelmäßige Neuerstellung	Nein
Regelmäßige Aktualisierung	Bei Änderung
Wird versendet an	-
Altes Dateiformat	-
Produzent/Lieferant	KAG
Spezial-/Publikumsfonds	Beide
Unterarten	Keine
CI Relevant	Ja
Signiert von	-
Sprachen (in Summe)	Deutsch
Verantwortlichkeit	Rechtsabteilung

Tabelle A.17: Eigenschaften von (9) Organe der Ges. u. Depotbank

Abhängigkeiten

Beziehungen der Dokumente untereinander

Enthaltene Dokumente	-
Notwendig für Dokumente	(12) Rechenschaftsbericht (13) Halbjahresbericht
Voraussetzungen für Erstellung	Verträge über Management und Depotbank.
Voraussetzungen für Veröffentlichung	-

Tabelle A.18: Abhängigkeiten von (9) Organe der Ges. u. Depotbank

A.0.10 (10) KAG Fondsliste)

Auflistung aller von der KAG verwalteten Fonds.

Eigenschaften

Wichtigste Parameter des Dokumententyps

Fondsspezifisch	Nein
Erstellungszeitpunkt	Nach Bedarf
Regelmäßige Neuerstellung	Nein
Regelmäßige Aktualisierung	Bei Änderung
Wird versendet an	-
Altes Dateiformat	-
Produzent/Lieferant	KAG
Spezial-/Publikumsfonds	-
Unterarten	Keine
CI Relevant	Ja
Signiert von	-
Sprachen (in Summe)	Deutsch
Verantwortlichkeit	Rechtsabteilung

Tabelle A.19: Eigenschaften von (10) KAG Fondsliste

Abhängigkeiten

Beziehungen der Dokumente untereinander

Enthaltene Dokumente	-
Notwendig für Dokumente	(11) Verkaufsprospekt
Voraussetzungen für Erstellung	-
Voraussetzungen für Veröffentlichung	-

Tabelle A.20: Abhängigkeiten von (10) KAG Fondsliste

A.0.11 (11) Verkaufsprospekt

Eigenschaften

Wichtigste Parameter des Dokumententyps

Fondsspezifisch	Ja
Erstellungszeitpunkt	Fondsstart
Regelmäßige Neuerstellung	Nein
Regelmäßige Aktualisierung	Bei Änderung
Wird versendet an	PM-Web
Altes Dateiformat	Word
Produzent/Lieferant	KAG
Spezial-/Publikumsfonds	P
Unterarten	Keine
CI Relevant	Ja
Signiert von	-
Sprachen (in Summe)	Deutsch
Verantwortlichkeit	Rechtsabteilung

Tabelle A.21: Eigenschaften von (11) Verkaufsprospekt

Abhängigkeiten

Beziehungen der Dokumente untereinander

Enthaltene Dokumente	(6) Informationen für Deutschland (10) KAG Fondsliste (17) Vereinfachter Prospekt (18) Allg. Fondsbestimmungen (19) Bes. Fondsbestimmungen (20) Anhang zu §16 (Börsenliste)
Notwendig für Dokumente	-
Voraussetzungen für Erstellung	Genehmigung Fondsbestimmung ISIN von OeKB
Voraussetzungen für Veröffentlichung	-

Tabelle A.22: Abhängigkeiten von (11) Verkaufsprospekt

A.0.12 (12) Rechenschaftsbericht

Reporting der KAG an Anteilsinhaber. Offizielles Dokument zur Veranlagung von Anlegern.

Eigenschaften

Wichtigste Parameter des Dokumententyps

Fondsspezifisch	Ja
Erstellungszeitpunkt	Wirtschaftsjahresende
Regelmäßige Neuerstellung	Ja
Regelmäßige Aktualisierung	Nein
Wird versendet an	Wirtschaftsprüfer: OeKB, Spezialfonds: Anteilsinhaber
Altes Dateiformat	Word, Excel
Produzent/Lieferant	KAG
Spezial-/Publikumsfonds	Beide
Unterarten	Keine
CI Relevant	Ja
Signiert von	KAG Vorstand
Sprachen (in Summe)	Deutsch
Verantwortlichkeit	-

Tabelle A.23: Eigenschaften von (12) Rechenschaftsbericht

Abhängigkeiten

Beziehungen der Dokumente untereinander

Enthaltene Dokumente	(6) Informationen für Deutschland (7) Bericht Fondsmanager (optional) (8) Ertragsrechnung (9) Organe des Ges u. Depotbank (14) Vermögensaufstellung (15) Bestätigungsvermerk (16) Erläuterungen zu Steuerliche Behandlung (18) Allg. Fondsbestimmungen (19) Bes. Fondsbestimmungen (20) Anhang zu §16 (21) Steuerliche Behandlung
Notwendig für Dokumente	-
Voraussetzungen für Erstellung	-
Voraussetzungen für Veröffentlichung	OK durch Wirtschaftsprüfer

Tabelle A.24: Abhängigkeiten von (12) Rechenschaftsbericht

A.0.13 (13) Halbjahresbericht

Reporting der KAG an Anteilsinhaber. Offizielles Dokument zur Veranlagung von Anlegern.

Eigenschaften

Wichtigste Parameter des Dokumententyps

Fondsspezifisch	Ja
Erstellungszeitpunkt	Wirtschaftshalbjahr
Regelmäßige Neuerstellung	Ja
Regelmäßige Aktualisierung	Nein
Wird versendet an	PM-Web
Altes Dateiformat	-
Produzent/Lieferant	KAG
Spezial-/Publikumsfonds	Beide
Unterarten	Keine
CI Relevant	Ja
Signiert von	KAG Vorstand
Sprachen (in Summe)	Deutsch
Verantwortlichkeit	-

Tabelle A.25: Eigenschaften von (13) Halbjahresbericht

Abhängigkeiten

Beziehungen der Dokumente untereinander

Enthaltene Dokumente	(9) Organe des Ges u. Depotbank (14) Vermögensaufstellung
Notwendig für Dokumente	-
Voraussetzungen für Erstellung	-
Voraussetzungen für Veröffentlichung	-

Tabelle A.26: Abhängigkeiten von (13) Halbjahresbericht

A.0.14 (14) Vermögensaufstellung

Listet alle Wertpapierpositionen, Finanzterminkontrakte und Konten des Fonds unterteilt in verschiedene Kategorien auf. Über etliche Zwischensummen scheint am Ende das Fondsvolumen zum Stichtag auf.

Eigenschaften

Wichtigste Parameter des Dokumententyps

Fondsspezifisch	Ja
Erstellungszeitpunkt	Wirtschaftshalbjahr und Wirtschaftsjahresende
Regelmäßige Neuerstellung	Ja
Regelmäßige Aktualisierung	Nein
Wird versendet an	KAG, Wirtschaftsprüfer
Altes Dateiformat	RTF
Produzent/Lieferant	FoBu
Spezial-/Publikumsfonds	Beide
Unterarten	Keine
CI Relevant	Ja
Signiert von	-
Sprachen (in Summe)	Deutsch
Verantwortlichkeit	Wirtschaftsprüfer

Tabelle A.27: Eigenschaften von (14) Vermögensaufstellung

Abhängigkeiten

Beziehungen der Dokumente untereinander

Enthaltene Dokumente	-
Notwendig für Dokumente	(12) Rechenschaftsbericht (13) Halbjahresbericht
Voraussetzungen für Erstellung	-
Voraussetzungen für Veröffentlichung	-

Tabelle A.28: Abhängigkeiten von (14) Vermögensaufstellung

A.0.15 (15) Bestätigungsvermerk

Bestätigung der Richtigkeit eines Rechenschaftsberichtes durch den Wirtschaftsprüfer.

Eigenschaften

Wichtigste Parameter des Dokumententyps

Fondsspezifisch	Ja
Erstellungszeitpunkt	Wirtschaftsjahresende
Regelmäßige Neuerstellung	Ja
Regelmäßige Aktualisierung	Nein
Wird versendet an	-
Altes Dateiformat	-
Produzent/Lieferant	Wirtschaftsprüfer/KAG
Spezial-/Publikumsfonds	Beide
Unterarten	Keine
CI Relevant	Ja
Signiert von	Wirtschaftsprüfer
Sprachen (in Summe)	Deutsch
Verantwortlichkeit	-

Tabelle A.29: Eigenschaften von (15) Bestätigungsvermerk

Abhängigkeiten

Beziehungen der Dokumente untereinander

Enthaltene Dokumente	-
Notwendig für Dokumente	(12) Rechenschaftsbericht
Voraussetzungen für Erstellung	Erfolgte Prüfung des Rechenschaftsberichtes.
Voraussetzungen für Veröffentlichung	-

Tabelle A.30: Abhängigkeiten von (15) Bestätigungsvermerk

A.0.16 (16) Erläuterungen zu Steuerliche Behandlung

Erklärende Worte zur Steuerlichen Behandlung.

Eigenschaften

Wichtigste Parameter des Dokumententyps

Fondsspezifisch	Nein
Erstellungszeitpunkt	-
Regelmäßige Neuerstellung	Nein
Regelmäßige Aktualisierung	Nur bei Änderungsbedarf
Wird versendet an	-
Altes Dateiformat	-
Produzent/Lieferant	KAG
Spezial-/Publikumsfonds	Beide
Unterarten	Keine
CI Relevant	Ja
Signiert von	-
Sprachen (in Summe)	Deutsch
Verantwortlichkeit	Wirtschaftsprüfer

Tabelle A.31: Eigenschaften von (16) Erläuterungen zu Steuerliche Behandlung

Abhängigkeiten

Beziehungen der Dokumente untereinander

Enthaltene Dokumente	-
Notwendig für Dokumente	(12) Rechenschaftsbericht
Voraussetzungen für Erstellung	-
Voraussetzungen für Veröffentlichung	-

Tabelle A.32: Abhängigkeiten von (16) Erläuterungen zu Steuerliche Behandlung

A.0.17 (17) Vereinfachter Prospekt

Eigenschaften

Wichtigste Parameter des Dokumententyps

Fondsspezifisch	Ja
Erstellungszeitpunkt	Fondsstart
Regelmäßige Neuerstellung	Nein
Regelmäßige Aktualisierung	Kalender- und Rechnungsjahresende
Wird versendet an	PM-Web
Altes Dateiformat	Word
Produzent/Lieferant	KAG
Spezial-/Publikumsfonds	P
Unterarten	Keine
CI Relevant	Ja
Signiert von	-
Sprachen (in Summe)	Deutsch
Verantwortlichkeit	Rechtsabteilung

Tabelle A.33: Eigenschaften von (17) Vereinfachter Prospekt

Abhängigkeiten

Beziehungen der Dokumente untereinander

Enthaltene Dokumente	-
Notwendig für Dokumente	(11) Verkaufsprospekt (5.3) FMA Antrag - EU Pass
Voraussetzungen für Erstellung	-
Voraussetzungen für Veröffentlichung	Genehmigung der Fondsbestimmungen

Tabelle A.34: Abhängigkeiten von (17) Vereinfachter Prospekt

A.0.18 (18) Allgemeine Fondsbestimmungen

Grundlegende Regelungen für Fonds der KAG. Genehmigungspflichtig von FMA und Aufsichtsrat mittels Umlaufbeschluss.

Eigenschaften

Wichtigste Parameter des Dokumententyps

Fondsspezifisch	Nein, aber Fondstyp: Spezial-/Puplikumsfonds und 20 oder 20a
Erstellungszeitpunkt	-
Regelmäßige Neuerstellung	Nein
Regelmäßige Aktualisierung	Bei Bedarf - z.B. Gesetzesänderungen
Wird versendet an	-
Altes Dateiformat	Word
Produzent/Lieferant	KAG
Spezial-/Publikumsfonds	Beide
Unterarten	Keine
CI Relevant	Ja
Signiert von	-
Sprachen (in Summe)	Deutsch
Verantwortlichkeit	Rechtsabteilung

Tabelle A.35: (18) Allgemeine Fondsbestimmungen

Abhängigkeiten

Beziehungen der Dokumente untereinander

Enthaltene Dokumente	-
Notwendig für Dokumente	(11) Verkaufsprospekt (12) Rechenschaftsbericht (22) Umlaufbeschluss
Voraussetzungen für Erstellung	-
Voraussetzungen für Veröffentlichung	Genehmigung

Tabelle A.36: (18) Allgemeine Fondsbestimmungen

A.0.19 (19) Besondere Fondsbestimmungen

Herzstück der formalen Fondsregelung. Genehmigungspflichtig von FMA.

Eigenschaften

Wichtigste Parameter des Dokumententyps

Fondsspezifisch	Ja
Erstellungszeitpunkt	Fondsstart
Regelmäßige Neuerstellung	Nein
Regelmäßige Aktualisierung	Bei Änderungen
Wird versendet an	PM-Web
Altes Dateiformat	Word
Produzent/Lieferant	KAG
Spezial-/Publikumsfonds	Beide
Unterarten	Keine
CI Relevant	Ja
Signiert von	-
Sprachen (in Summe)	Deutsch
Verantwortlichkeit	Rechtsabteilung

Tabelle A.37: (19) Besondere Fondsbestimmungen

Abhängigkeiten

Beziehungen der Dokumente untereinander

Enthaltene Dokumente	-
Notwendig für Dokumente	(11) Verkaufsprospekt (12) Rechenschaftsbericht (22) Umlaufbeschluss
Voraussetzungen für Erstellung	-
Voraussetzungen für Veröffentlichung	-

Tabelle A.38: (19) Besondere Fondsbestimmungen

A.0.20 (20) Anhang zu §16 - Börsenliste

Listet alle Börsen und geregelte Märkte auf an denen der Fonds handeln darf.

Eigenschaften

Wichtigste Parameter des Dokumententyps

Fondsspezifisch	Nein
Erstellungszeitpunkt	-
Regelmäßige Neuerstellung	Nein
Regelmäßige Aktualisierung	Ja
Wird versendet an	-
Altes Dateiformat	Word
Produzent/Lieferant	KAG
Spezial-/Publikumsfonds	Beide
Unterarten	Keine
CI Relevant	Ja
Signiert von	-
Sprachen (in Summe)	Deutsch
Verantwortlichkeit	Rechtsabteilung

Tabelle A.39: (20) Anhang zu §16 - Börsenliste

Abhängigkeiten

Beziehungen der Dokumente untereinander

Enthaltene Dokumente	-
Notwendig für Dokumente	(1) Depotbankschreiben (12) Rechenschaftsbericht (22) Umlaufbeschluss
Voraussetzungen für Erstellung	-
Voraussetzungen für Veröffentlichung	-

Tabelle A.40: (20) Anhang zu §16 - Börsenliste

A.0.21 (21) Steuerliche Behandlung

Erklärung der steuerlichen Komponenten einer Ausschüttung. Notwendig für Veranlagung von Investoren.

Eigenschaften

Wichtigste Parameter des Dokumententyps

Fondsspezifisch	Ja
Erstellungszeitpunkt	Wirtschaftsjahresende
Regelmäßige Neuerstellung	Ja
Regelmäßige Aktualisierung	Nein
Wird versendet an	-
Altes Dateiformat	-
Produzent/Lieferant	FoBu
Spezial-/Publikumsfonds	Beide
Unterarten	Keine
CI Relevant	Ja
Signiert von	-
Sprachen (in Summe)	Deutsch
Verantwortlichkeit	Wirtschaftsprüfer

Tabelle A.41: (21) Steuerliche Behandlung

Abhängigkeiten

Beziehungen der Dokumente untereinander

Enthaltene Dokumente	-
Notwendig für Dokumente	(12) Rechenschaftsbericht
Voraussetzungen für Erstellung	Festlegung der Ausschüttung
Voraussetzungen für Veröffentlichung	OK des Wirtschaftsprüfers

Tabelle A.42: (21) Steuerliche Behandlung

A.0.22 (22) Umlaufbeschluss

Ein Umlaufbeschluss dient zur Einholung der Zustimmung oder Kenntnisnahme des Aufsichtrates der KAG. Dieser muss von allen Aufsichtsräten einzeln unterzeichnet werden. Es können die unterschiedlichsten Entscheidungen sein, hier die wichtigsten:

- Neuauflage eines Fonds
- Änderung von Fondsbestimmungen
- Kenntnisnahme der Rechenschaftsberichte

Eigenschaften

Wichtigste Parameter des Dokumententyps

Fondsspezifisch	Ja, oft auch Gruppen von Fonds
Erstellungszeitpunkt	Nach Bedarf
Regelmäßige Neuerstellung	Nein
Regelmäßige Aktualisierung	Nein
Wird versendet an	Aufsichtsräte, FMA
Altes Dateiformat	Word
Produzent/Lieferant	KAG
Spezial-/Publikumsfonds	Beide
Unterarten	(22.1) Fondsgründung (22.2) Änderung Fondsbestimmungen
CI Relevant	Nein
Signiert von	KAG Vorstand, KAG Aufsichtsrat, FMA
Sprachen (in Summe)	Deutsch
Verantwortlichkeit	Rechtsabteilung

Tabelle A.43: (22) Umlaufbeschluss

Abhängigkeiten

Beziehungen der Dokumente untereinander

Enthaltene Dokumente	Variabel, häufig: (12) Rechenschaftsbericht (18) Allg. Fondsbestimmungen (19) Bes. Fondsbestimmungen (20) Anhang zu §16
Notwendig für Dokumente	-
Voraussetzungen für Erstellung	-
Voraussetzungen für Veröffentlichung	-

Tabelle A.44: (22) Umlaufbeschluss

A.0.23 (23) Delegationsmeldung

Zeigt der FMA die Delegation eines Fondsmanagements an. Besteht aus einem Begleitbrief und der eigentlichen Delegationsmeldung.

Eigenschaften

Wichtigste Parameter des Dokumententyps

Fondsspezifisch	Ja
Erstellungszeitpunkt	Fondsstart
Regelmäßige Neuerstellung	Bei Änderung Fondsmanagement.
Regelmäßige Aktualisierung	Nein
Wird versendet an	FMA
Altes Dateiformat	Word
Produzent/Lieferant	KAG
Spezial-/Publikumsfonds	Beide
Unterarten	(23.1) Spezialfonds (23.2) Publikumfonds
CI Relevant	Nein
Signiert von	KAG Vorstand
Sprachen (in Summe)	Deutsch
Verantwortlichkeit	Rechtsabteilung

Tabelle A.45: (23) Delegationsmeldung

Abhängigkeiten

Beziehungen der Dokumente untereinander

Enthaltene Dokumente	-
Notwendig für Dokumente	-
Voraussetzungen für Erstellung	Delegationsvertrag
Voraussetzungen für Veröffentlichung	-

Tabelle A.46: (23) Delegationsmeldung

A.0.24 (24) Anteilsinhaber Info

Informationen an die Anteilsinhaber - insbesondere Rechenschaftsberichte.

Eigenschaften

Wichtigste Parameter des Dokumententyps

Fondsspezifisch	Ja, auch für mehrere Fonds
Erstellungszeitpunkt	Wirtschaftsjahresende
Regelmäßige Neuerstellung	Ja
Regelmäßige Aktualisierung	Nein
Wird versendet an	Anteilsinhaber
Altes Dateiformat	-
Produzent/Lieferant	KAG
Spezial-/Publikumsfonds	S
Unterarten	Keine
CI Relevant	Ja
Signiert von	KAG 2x ppa oder Vorstand
Sprachen (in Summe)	Deutsch
Verantwortlichkeit	-

Tabelle A.47: (24) Anteilsinhaber Info

Abhängigkeiten

Beziehungen der Dokumente untereinander

Enthaltene Dokumente	(12) Rechenschaftsbericht (25) Prüfbericht
Notwendig für Dokumente	-
Voraussetzungen für Erstellung	-
Voraussetzungen für Veröffentlichung	-

Tabelle A.48: (24) Anteilsinhaber Info

A.0.25 (25) Prüfbericht

Bericht des Wirtschaftsprüfers über den Rechenschaftsbericht; beinhaltet den Rechenschaftsbericht. Gebundene Form mit Originalunterschrift des Prüfers. Exemplar mit ebenfalls Originalunterschriften des KAG Vorstandes geht an die FMA.

Eigenschaften

Wichtigste Parameter des Dokumententyps

Fondsspezifisch	Ja
Erstellungszeitpunkt	Wirtschaftsjahresende
Regelmäßige Neuerstellung	Ja
Regelmäßige Aktualisierung	Nein
Wird versendet an	FMA
Altes Dateiformat	Gebunden/Papier
Produzent/Lieferant	Wirtschaftsprüfer
Spezial-/Publikumsfonds	Beide
Unterarten	Keine
CI Relevant	Ja
Signiert von	Wirtschaftsprüfer
Sprachen (in Summe)	Deutsch
Verantwortlichkeit	Wirtschaftsprüfer

Tabelle A.49: (25) Prüfbericht

Abhängigkeiten

Beziehungen der Dokumente untereinander

Enthaltene Dokumente	-
Notwendig für Dokumente	(24) Anteilsinhaber Info
Voraussetzungen für Erstellung	OK für Rechenschaftsbericht durch Wirtschaftsprüfer
Voraussetzungen für Veröffentlichung	-

Tabelle A.50: (25) Prüfbericht

A.0.26 (26) ÖKB Stammdatenblatt

Stammdaten des Fonds auf einen standardisierten Formular, zur Vergabe einer ISIN für einen neuen Fonds.

Eigenschaften

Wichtigste Parameter des Dokumententyps

Fondsspezifisch	Ja
Erstellungszeitpunkt	Fondsstart
Regelmäßige Neuerstellung	Nein
Regelmäßige Aktualisierung	Bei Änderung Stammdaten
Wird versendet an	OeKB
Altes Dateiformat	Excel (von OeKB)
Produzent/Lieferant	KAG
Spezial-/Publikumsfonds	Beide
Unterarten	Keine
CI Relevant	Nein
Signiert von	-
Sprachen (in Summe)	Deutsch
Verantwortlichkeit	Wirtschaftsprüfer

Tabelle A.51: (26) ÖKB Stammdatenblatt

Abhängigkeiten

Beziehungen der Dokumente untereinander

Enthaltene Dokumente	-
Notwendig für Dokumente	-
Voraussetzungen für Erstellung	-
Voraussetzungen für Veröffentlichung	-

Tabelle A.52: (26) ÖKB Stammdatenblatt

A.0.27 (27) Veröffentlichungen

Immer wenn die KAG etwas offiziell publik machen muss wird eine Veröffentlichung erstellt. Der Inhalt ist relativ vielfältig.

Eigenschaften

Wichtigste Parameter des Dokumententyps

Fondsspezifisch	Ja , manchmal auch mehrere Fonds
Erstellungszeitpunkt	Nach Bedarf
Regelmäßige Neuerstellung	Nein
Regelmäßige Aktualisierung	Nein
Wird versendet an	Veröffentlichung
Altes Dateiformat	Word
Produzent/Lieferant	KAG
Spezial-/Publikumsfonds	P
Unterarten	Ja, je nach Grund der Veröffentlichung
CI Relevant	Ja
Signiert von	-
Sprachen (in Summe)	Deutsch
Verantwortlichkeit	Rechtsabteilung

Tabelle A.53: (27) Veröffentlichungen

Abhängigkeiten

Beziehungen der Dokumente untereinander

Enthaltene Dokumente	-
Notwendig für Dokumente	-
Voraussetzungen für Erstellung	FMA Bescheid über Neugründung oder Änderung Fondsbestimmungen.
Voraussetzungen für Veröffentlichung	-

Tabelle A.54: (27) Veröffentlichungen

A.0.28 (28) Depotanlageblatt

Internes Formular für die Anlage eines Depots und Konten für einen neuen Fonds (Neuauflage und Übernahme). Es enthält alle notwendigen Informationen für die Anlage.

Eigenschaften

Wichtigste Parameter des Dokumententyps

Fondsspezifisch	Ja
Erstellungszeitpunkt	Fondsstart
Regelmäßige Neuerstellung	Nein
Regelmäßige Aktualisierung	Nein
Wird versendet an	Depotbank
Altes Dateiformat	Word
Produzent/Lieferant	Bank (Formular)/KAG (Inhalte)
Spezial-/Publikumsfonds	Beide
Unterarten	Keine
CI Relevant	Nein
Signiert von	?
Sprachen (in Summe)	Deutsch
Verantwortlichkeit	Wertpapier-Abwicklung (Rechtsabteilung)

Tabelle A.55: (28) Depotanlageblatt

Abhängigkeiten

Beziehungen der Dokumente untereinander

Enthaltene Dokumente	-
Notwendig für Dokumente	-
Voraussetzungen für Erstellung	-
Voraussetzungen für Veröffentlichung	-

Tabelle A.56: (28) Depotanlageblatt

Literaturverzeichnis

[ABC+01] Adler, Sharon ; Bergl, Anders ; Caruso, Jeff ; Deach, Stephen ; Graham, Tony ; Grosso, Paul ; Gutentag, Eduardo ; Milowski, Alex ; Parnell, Scott ; Richman, Jeremy ; Zilles, Steve: *Extensible Stylesheet Language (XSL) Version 1.0*. http://www.w3.org/TR/xsl, 2001 `http://www.w3.org/TR/xsl`

[BPSM+04] Bray, Tim ; Paoli, Jean ; Sperberg-McQueen, C. M. ; Maler, Eve ; Yergeau, François: *Extensible Markup Language (XML) 1.0 (Third Edition)*. http://www.w3.org/TR/REC-xml/, 2004 `http://www.w3.org/TR/REC-xml/`

[Cla99] Clark, James: *XSL Transformations (XSLT)Version 1.0*. http://www.w3.org/TR/xslt, 1999 `http://www.w3.org/TR/xslt`

[HWW05] Hégaret, Philippe L. ; Whitmer, Ray ; Woods, Lauren: *Document Object Model (DOM)*. http://www.w3.org/DOM/, 2005 `http://www.w3.org/DOM/`

[Kle02] Klettke, Meike: *XML & Datenbanken - Konzepte, Sprachen und Systeme*. dpunkt-Verlag, 2002. – ISBN 3898641481

[KST03] Kazakos, Wassilios ; Schmidt, Andreas ; Tomczyk, Peter: *Datenbanken und XML*. Springer-Verlag GmbH, 2003. – ISBN 354041956X

[See03] Seemann, Michael: *Native XML-Datenbanken im Praxiseinsatz*. Software & Support, 2003. – ISBN 3935042353

[Wik06a] Wikipedia: *Dokumenttypdefinition — Wikipedia, Die freie Enzyklopädie*. `http://de.wikipedia.org/w/index.php?title=Dokumenttypdefinition&oldid=20084250`, 2006. – [Online; Stand 25. August 2006]

[Wik06b] Wikipedia: *Extensible Markup Language — Wikipedia, Die freie Enzyklopädie*. `http://de.wikipedia.org/w/index.php?title=Extensible_Markup_Language&oldid=20498028`, 2006. – [Online; Stand 25. August 2006]

[Wik06c] Wikipedia: *Extensible Stylesheet Language Formatting Objects — Wikipedia, Die freie Enzyklopädie*. `http://de.wikipedia.org/w/index.php?title=Extensible_Stylesheet_Language_\%E2\%80\%93_Formatting_Objects&oldid=20279752`, 2006. – [Online; Stand 25. August 2006]

[Wik06d] Wikipedia: *XPath — Wikipedia, Die freie Enzyklopädie*. `http://de.wikipedia.org/w/index.php?title=XPath&oldid=20279264`, 2006. – [Online; Stand 28. August 2006]

[Wik06e] WIKIPEDIA: *XSL Transformation — Wikipedia, Die freie Enzyklopädie*. `http://de.wikipedia.org/w/index.php?title=XSL_Transformation&oldid=20444041`, 2006. – [Online; Stand 25. August 2006]