JOHANNES KEPLER
UNIVERSITÄT LINZ | JKU

Technisch-Naturwissenschaftliche
Fakultät

OpenCodex

Erzeugung individualisierter Gesetzbücher aus den Daten des RIS

MASTERARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Masterstudium

Informatik

Eingereicht von:
Josef K. Schaitl

Angefertigt am:
Institut für Informationsverarbeitung und Mikroprozessortechnik (FIM)

Betreuung:
Assoz.Prof. Mag. Dipl.-Ing. Dr. Michael Sonntag

Linz, Juli 2013

Zusammenfassung

Sammlungen österreichischer Gesetze erscheinen bisher, aufgelegt von verschiedenen Verlagen, in Buchform. Die Sammlungen werden zwar redaktionell betreut, was eine hohe Qualität bezüglich Inhalt und Layout garantiert, sie sind jedoch nicht individualisierbar, nicht tagesaktuell und teuer. Teuer vor allem, wenn man bedenkt, dass die Ausgangsdaten – die Gesetzestexte des konsolidierten Bundesrechts – frei verfügbar sind.

In der vorliegenden Arbeit wird die Anwendung OpenCodex beschrieben. Sie ermöglicht es, individualisierte Gesetzbücher über eine Web-Oberfläche zusammenzustellen und zu erzeugen. Der Benutzer kann die zu integrierenden Gesetze, deren Umfang sowie das Gültigkeitsdatum wählen. Nach einem kurzen Erstellungsprozess erhält man eine druckfertige PDF-Datei. Die Anwendung bietet darüber hinaus die Möglichkeit, zwei Fassungen eines Gesetzes zu vergleichen und somit einen direkten Überblick über die Gesetzesänderungen in einem definierbaren Zeitraum zu erhalten.

Zu Beginn der Arbeit werden Grundlagen zur Struktur des Rechtsinformationssystems des Bundes erklärt, sowie ein Überblick über die bestehenden Lösungen gegeben. Der zweite Teil der Arbeit ist eine Beschreibung von Technologien, auf welche die Anwendung zurückgreift. Insbesondere das Datenformat XML, in dem die Ausgangsdaten vorliegen und das Textsatzsystem LaTeX, das zum Setzen der Gesetzessammlungen dient, werden genauer erläutert. Es folgt eine detaillierte Betrachtung der Implementierung. Sie gliedert sich in ein Back-End, das die Sammlungs- und Vergleichserstellung durchführt, und ein Front-End, welches die Benutzerinteraktion ermöglicht. Den Abschluss der Arbeit bildet ein Ausblick auf mögliche Erweiterungen der Anwendung.

Abstract

Currently collections of Austrian laws are printed by different publishers as books. These collections are editorially maintained and thus guarantee high quality concerning both content and typography. However, the collections are neither customizable nor up-to-the-minute and they are expensive, especially when one takes into account that the base data is freely available.
This thesis describes the application OpenCodex. It provides a web platform that enables users to create personalized legal books. The user may choose the laws that are to be included, their effective date and their range. Once the build process has completed, the user is provided with a ready to print PDF-file. Additionally the application provides the opportunity to show the changes in a law between two effective dates, giving the user a direct overview of changes in a law.
The thesis starts with basics on the structure of the federal legal information system and an overview of existing solutions. This is followed by a description of technologies used by the application. The data format XML – the format of the raw data – and the typesetting system LaTeX– used for typesetting the collections – are described in detail. The next chapter gives an in-depth view on the implementation of the application. The program is split into a back-end that creates the collections and the version comparisons, and a front-end that enables user interaction. The thesis is concluded by a perspective on possibilities to extend the application.

Inhaltsverzeichnis

Quellcodeverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einführung

„Der Staat ist also Sache des Volkes. . . “
Cicero, *De re publica I, 39* [Cic50]

Was Cicero bereits um 50 v. Chr. über das Verhältnis von Staat und Volk schrieb, ist in einer heutigen Demokratie zweifellos weiterhin gültig. Bezieht man das Zitat auf das Verhältnis der Gesetzgebung – sowie der daraus resultierenden Gesetzestexte – und der Bürger[1], welche davon betroffen sind, so wird rasch klar, dass Letzteren freier Zugang zu Ersteren gewährt werden soll. Moderne Telekommunikationswege bieten den Gesetzgebungsorganen hierzu eine schnelle und kostengünstige Möglichkeit. Die Republik Österreich nimmt mit dem Rechtsinformationssystem des Bundes (RIS)[2], insbesondere aber mit der Onlineveröffentlichung der authentischen Gesetzgebung seit 2004, eine europäische Vorreiterrolle ein.

1.1 Motivation

Die heute gängige Praxis, auf gedruckte Gesetzestexte zurückzugreifen, basiert auf der Verwendung von Gesetzessammlungen, welche von Verlagen herausgegeben werden. Besonders die Reihe *KODEX* des Verlages *LexisNexis* und die Reihe *Große Gesetzausgaben* des Verlages *Manz* sind hierbei hervorzuheben. Eine von einem Verlag herausgegebene Gesetzessammlung bietet zwar den Vorteil einer redaktionellen Betreuung, was mit anzunehmender hoher Qualität bzgl. Inhalt und Layout einhergeht, hat jedoch auch eine Reihe von Nachteilen:

Wenig bis keine Individualität Eine vorgefertigte Gesetzessammlung enthält meist nicht nur die gewünschten Gesetze, sondern auch viele nicht benötigte. Umgekehrt ist es möglich, dass man mehrere solcher Bücher braucht, um alle nötigen Gesetze vorliegen zu haben.

[1] Aufgrund der besseren Lesbarkeit wird in der vorliegenden Arbeit bei der Verwendung personenbezogener Bezeichnungen (z. B. Bürger, Benutzer) auf die explizite Anführung der jeweiligen weiblichen Form verzichtet. Angesprochen sind dabei selbstverständlich immer beide Geschlechter.
[2] Siehe `http://ris.bka.gv.at`, (22.07.2013) und 1.3.1.

Hohe Kosten Der Preis von Gesetzessammlungen unterscheidet sich bei etwa gleichem Seitenumfang zum Teil sehr stark (aktuell ca. 20 - 100 €). Insbesondere seltener benötigte und deshalb weniger auflagenstarke Gesetzessammlungen sind teuer. Jedoch auch bei günstigen Einzelsammlungen sind die laufenden Kosten hoch, da in Bereichen wie z. B. dem Steuerrecht aufgrund häufiger Gesetzesänderungen oft aktualisierte Auflagen erscheinen.

Geringe Aktualität Verlage, welche Sammlungen von Gesetzen anbieten, aktualisieren diese zwar meist bei großen und wichtigen Gesetzesänderungen, jedoch ist immer mit Verzögerungen zu rechnen, bzw. werden kleine Änderungen eventuell nicht sofort berücksichtigt.

Fehlende Vergleichsmöglichkeiten Mit Hilfe von Gesetzessammlungen ist es nicht möglich, die konkreten Änderungen eines bestimmten Gesetzes festzustellen. Das Parlament bietet bei vorgeschlagenen Gesetzesänderungen zwar Vergleichsdokumente an[3], diese erlauben jedoch immer nur den Vergleich der aktuellen mit der vorgeschlagenen (welche sich bis zum Beschluss oft noch ändert) Fassung eines Gesetzes. Eine individuelle Vergleichsmöglichkeit zwischen den Fassungen beliebiger Daten ist nicht möglich. Ferner werden die beiden Textversionen lediglich gegenübergestellt, eine direkte Hervorhebung der Änderungen in *einem* Text wird nicht angeboten.

Gedenkt ein Benutzer eine individualisierte Gesetzessammlung anzulegen, so bleibt ihm lediglich die Möglichkeit, Einzeldokumente oder gesamte Rechtsvorschriften aus dem Browser oder über vorgefertigte Portable Document Format (PDF)-Dateien aus dem RIS auszudrucken. Der Bedienungskomfort, die Einstellmöglichkeiten, Zusatzfunktionen wie ein Inhalts- und Schlagwortverzeichnis und die typografische Qualität des Ergebnisses lassen wohl bei dieser manuellen Lösung zu wünschen übrig.

1.2 Aufgabenstellung

Es soll ein Programm geschrieben werden, welches das Erstellen einer individualisierten Gesetzessammlung und deren Export in ein druckfertiges PDF-Dokument erlaubt. Die Bedienung des Programms für den Benutzer (Front-End) erfolgt über einen Webbrowser, die eigentliche Programmlogik selbst (Back-End) läuft auf einem Server.

[3] Siehe z. B. Unterpunkt „Gesetzesvorschläge in Bearbeitung“ auf `http://www.parlament.gv.at`, (22.07.2013)

1.2.1 Gesetzbuch

Der Benutzer kann Sammlungen von verfügbaren Gesetzen und Vorschriften erstellen. Die verfügbaren Rechtsvorschriften[4] sind aus der Anwendung „Bundesrecht konsolidiert“ im RIS (siehe 1.3). Eine Sammlung kann sowohl ganze Gesetze, also alle Paragraphen/Artikel/Anlagen, als auch nur Ausschnitte einer Rechtsvorschrift enthalten. Ein Zugriff auf alte Fassungen von Gesetzestexten ist – sofern verfügbar – ebenfalls möglich. Die einzelnen Elemente einer Sammlung können beliebig sortiert werden. Nach Angabe einer Bezeichnung der Gesetzessammlung und Anpassung des Deckblattes, sowie der Auswahl des zu erzeugenden Papierformates, kann das Programm aus einer Sammlung ein Buch in Form eines PDF-Dokumentes erstellen, welches dann heruntergeladen werden kann. Es ist ferner möglich, eine Zusammenstellung von Gesetzen zu speichern, um später wieder darauf zugreifen zu können.

1.2.2 Versionsvergleich

Man kann zwei Versionen eines gewählten Gesetzes vergleichen. Eine „Version“ ist die Fassung eines bestimmten Gesetzes zu einem bestimmten Gültigkeitsdatum. Hierzu wird zuerst über eine Suchmaske das Gesetz ausgewählt und dann werden die beiden zu vergleichenden Gültigkeitsdaten gewählt. Der Benutzer kann den Umfang des Gesetzes einschränken. Hinzugefügte bzw. gelöschte Paragrafen werden im Vergleichsdokument entsprechend gekennzeichnet; Änderungen innerhalb eines Paragrafen werden je nach Umfang bzw. je nach Benutzereinstellung farblich hervorgehoben oder die alte und die neue Version nebeneinander gegenübergestellt.

1.3 Das Rechtsinformationssystem des Bundes – RIS

Das RIS bildet die Datenquelle der in dieser Arbeit vorgestellten Software. Deshalb wird ein kurzer Abriss zu dessen Entstehen präsentiert, bevor näher auf die verwendete RIS-Anwendung *Bundesrecht konsolidiert* und der Struktur der Dokumente darin eingegangen wird. Weiters wird kurz erläutert, was „Open Government Data“ beinhaltet und wie das *Bundesrecht konsolidiert* darin enthalten ist.

[4] In weiterer Folge werden die Begriffe „Gesetz“, „Rechtsvorschrift“ und „Rechtsnorm“ äquivalent behandelt. Gemeint ist damit, obwohl es sich hierbei im juristischen Sinn nicht unbedingt um das gleiche handelt, jegliche Form von Rechtsnorm (z. B. Gesetz, Verordnung, Staatsvertrag, internationales Übereinkommen, etc.). Die Unterscheidung wird ignoriert, da sie für die vorliegende Arbeit keinerlei Unterschied macht.

1.3.1 Geschichte

Nach Svoboda, Manak und Weinguny [SMW94, S. 15] ist als erste österreichische Rechtsdatenbank das „EDV-Versuchsprojekt Verfassungsrecht“ im Jahre 1971 anzusehen. Hellwig [Hel04] zitiert in seinem Aufsatz aus einem Bericht des Bundeskanzleramtes aus dem Jahr 1973; diesem Bericht sei „[. . .] eine gehörige Skepsis herauszulesen, ob und wie ein Rechtsinformationssystem geschaffen werden kann und ob der Aufwand überhaupt gerechtfertigt ist“.

Es kann also aus heutiger Sicht von Glück gesprochen werden, dass in den 1980er Jahren die erste Version des RIS, so Lachmayer und Stöger [LS04], als „Abfallprodukt“ bzw. „Zufallsprodukt“ zweier unterschiedlicher Sachverhalte entstand: Zum Einen wurden im Zuge der Digitalisierung des Buchdrucks Gesetze auf digitale Datenträger erfasst, welche man dann ebenso für eine Datenbank „zweckentfremden“ konnte. Zum Anderen standen mit dem Aufbau des zentralen Ausweichsystems des Bundes (ZAS)[5] beträchtliche ungenutzte Rechenkapazitäten für ein Rechtsdatenbanksystem frei. Am 7. Oktober 1986 beschloss der Ministerrat den „Aufbau eines umfassenden Rechtsinformationssystems“ [Hel04, S. 131]. Der Zugang zu dem auf Großrechnern beheimateten System, welcher nicht nur Spezialisten, sondern auch geschultem Personal aller Ministerien gewährt wurde, wurde über das Protokoll der IBM 3270-Terminals sichergestellt. In den folgenden Jahren öffnete sich das RIS immer weiter. Zunächst wurde neben den Terminalzugängen allen Mitarbeitern der Ministerien der Zugang über das Behörden-Intranet gewährt. Seit 1997 steht das RIS nach eigener Aussage[6] der allgemeinen Bevölkerung über das Internet zur Verfügung.

Am 1. Jänner 2004 trat das *Bundesgesetz über das Bundesgesetzblatt 2004 (BGBlG)* in Kraft. In § 7 Abs. 1 wird das RIS als offizielle Quelle des Bundesgesetzblattes genannt:

> „Die im Bundesgesetzblatt zu verlautbarenden Rechtsvorschriften sind im Internet unter der Adresse **www.ris.bka.gv.at** zur Abfrage bereit zu halten.“

In [Par03, S. 176 f.] bezeichnet Dr. Ulrike Baumgartner-Gabitzer, Abgeordnete des Nationalrates, diesen Wechsel von der papiergebundenen zur elektronischen Veröffentlichung des Bundesgesetzblattes als „Meilenstein im Kundmachungswesen“.

Die Integrität der authentischen Gesetzgebungsdaten wird über den Einsatz elektronischer Signaturen sichergestellt.

[5] Heute würde man es wohl als „Backup-Rechenzentrum“ bezeichnen.
[6] Siehe `http://www.ris.bka.gv.at/UI/Info.aspx`, (22.07.2013)

Beim Studium der Literatur zur Geschichte des RIS ist eine interessante Entwicklung festzustellen. Zunächst wurde der Datenbestand zentral auf Großrechnern gehalten und die Terminals dienten lediglich der Anfrage und der Darstellung. In den 1990er Jahren, als durch das Medium CD-ROM die Verteilung größerer Datenmengen einfach und populär wurde, ereignete sich eine gewisse Abkehr der zentralen Datenhaltung und es wurden lokale, schnell verfügbare Gesamtkopien des Datenbestandes angestrebt. Mit der Einführung des öffentlichen Zugangs zum RIS über das Internet setzte jedoch ein Gegentrend ein: Die eigentlichen Daten sind wieder zentral gespeichert, das Terminal zum Abfragen und Anzeigen wurde lediglich durch den Browser ersetzt. Bei der Erstellung von OpenCodex wurden allerdings wieder Mängel am zentralisierten System, insbesondere was die Abfragegeschwindigkeit betrifft, bemerkt, sodass durch das in 3.2.2 vorgestellte Caching wieder nach und nach eine lokale Kopie der Daten erstellt wird.

1.3.2 Das *Bundesrecht konsolidiert*

Das Bundeskanzleramt, Betreiber des RIS, definiert hierzu folgendes:

> „[...] das österreichische Bundesrecht in konsolidierter Fassung, wobei Konsolidierung bedeutet, dass in einer Rechtsvorschrift sämtliche später kundgemachten Änderungen und Berichtigungen eingearbeitet wurden. Diese Dokumente dienen lediglich der Information, sind also rechtlich unverbindlich.“ [Öst13, *Bundesrecht konsolidiert*]

Trotz der rechtlichen Unverbindlichkeit ist die konsolidierte Fassung die wohl am häufigsten genutzte, da sie schnellen Zugriff auf die aktuell (oder zu einem festgelegten Zeitpunkt) gültige Fassung eines Gesetzes bietet. Die Suchmaske zum *Bundesrecht konsolidiert*[7] bietet neben einer Stichwortsuche den direkten Zugriff über den Titel oder die Abkürzung eines Gesetzes. Die weiteren Suchparameter zeigt Abbildung 1, ein Screenshot der Suchmaske.

1.3.2.1 Struktur

Das *Bundesrecht konsolidiert* besteht aus einer Vielzahl von Einzeldokumenten. Ein Dokument ist hierbei ein Paragraf, ein Artikel oder eine Anlage einer Rechtsvorschrift. [Öst13] Jedes Dokument hat eine eindeutige Dokumentnummer der Form `NORxxxxxxxx` (`x` = [0-9]).

[7] Siehe `http://www.ris.bka.gv.at/Bundesrecht`, (22.07.2013)

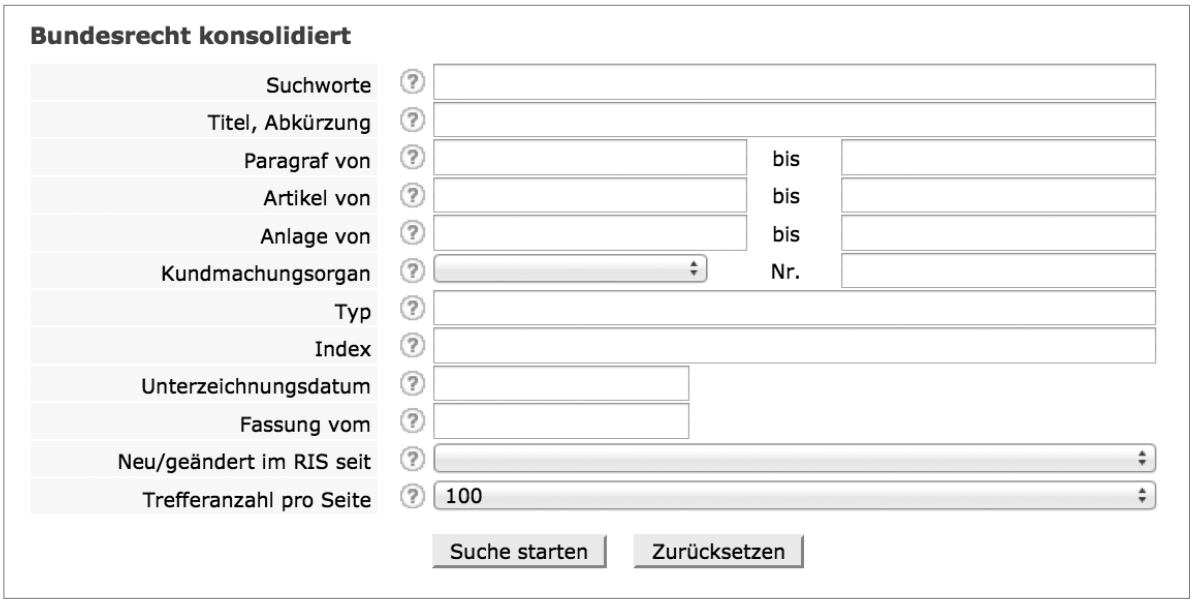


Abbildung 1: Die Suchmaske des *Bundesrecht konsolidiert* im RIS.

Ein Dokument bleibt, wenn es einmal angelegt wurde, immer gleich. Bei einer Gesetzesänderung wird auf ein neues Dokument mit dem geänderten Inhalt verwiesen, das alte bleibt zu Dokumentationszwecken bestehen. Erst durch diese Tatsache wird ein lokales Caching, wie unter 3.2.2 beschrieben, möglich. Ein Gesetz im Bundesrecht ist also eine Liste von Einzeldokumenten, die jeweils einen eigenen Gültigkeitszeitraum besitzen.

1.3.2.2 Ein Einzeldokument und der § 0

Jedes Einzeldokument enthält neben dem eigentlichen Inhalt einige Felder mit Metadaten. Für die vorliegende Arbeit sind hierbei folgende besonders relevant:

§/Artikel/ Anlage Dieses Feld enthält die Gliederungseinheit, die das Dokument innerhalb des Gesetzes repräsentiert, in Textform. Es wurden im Zuge dieser Arbeit sieben mögliche Formate identifiziert; Details hierzu siehe 3.2.3.

Index Enthält die Identifikationsnummern der Haupt- bzw. Untergruppen, sowie der zugehörigen Bezeichnung im Index des Bundesrechts[8]. Häufig enthält dieses Feld nur ein Element, es ist jedoch auch eine Liste mit mehreren Einträgen möglich, etwa wenn eine Rechtsnorm verschiedene Gruppen betrifft.

Schlagworte Enthält eine durch Beistrich getrennte Liste von Schlagworten, auf welche sich das Dokument bezieht. Damit ist es in weiterer Folge

[8] Siehe `http://www.ris.bka.gv.at/UI/Bund/Bundesnormen/IndexBundesrecht.aspx`, (22.07.2013)

möglich, ein Schlagwortverzeichnis zu einzelnen Gesetzen zu erstellen (siehe 3.2.4.2).

Kurz-/Langtitel Enthält die gesetzliche Kurz- bzw. Langform des Titels eines Gesetzes. Existiert kein gesetzlicher Kurztitel, so wird die gebräuchliche Kurzbezeichnung verwendet.

Abkürzung (optional) Nur wenn dieses Feld bei einer Rechtsvorschrift gesetzt ist, existiert eine offizielle[9] Abkürzung des Gesetzesnamens (z. B. ABGB für *Allgemeines Bürgerliches Gesetzbuch*). Eine Suche nach diesen Abkürzungen im RIS, und somit auch in OpenCodex, ist möglich.

Gesetzesnummer Die Gesetzesnummer ist eine achtstellige Nummer, welche ein Gesetz eindeutig identifiziert. Über diese Gesetzesnummer wird die Zuordnung getroffen, dass ein Paragraf/Artikel zu einem bestimmten Gesetz gehört.

Beachte Dieses Feld enthält Informationen, die im Zusammenhang mit dem Dokument zu beachten sind. Dies ist z. B. eine Besonderheit der Gültigkeit eines Paragrafen/Artikels oder die Information, dass es sich bei dem Inhalt des Dokumentes um eine Verfassungsbestimmung handelt.

Präambel/ Promulgationsklausel (nur § 0) Die Promulgationsklausel ist die Eingangsformel eines Gesetzes. Sie enthält meist den Namen des gesetzgebenden Organs; „Der Nationalrat hat beschlossen:“ ist eine gängige Formulierung bei Bundesgesetzen. Darüber hinaus enthält die Präambel mancher Gesetze ein tabellarisches Inhaltsverzeichnis.

Jede Rechtsnorm im *Bundesrecht konsolidiert* umfasst, unabhängig davon, ob sie in Paragrafen oder Artikel untergliedert ist, immer einen § 0. Dieser „Informationsparagraf“ ist nicht Teil des eigentlichen Gesetzes und enthält neben der eben erwähnten Präambel bzw. Promulgationsklausel oft weitere Anmerkungen, Schlagworte zur gesamten Rechtsvorschrift, Referenzen auf Gesetzesänderungen, Hinweise zu Übersetzungen, etc. und ist nicht Teil der eigentlichen Rechtsnorm. In OpenCodex kann gewählt werden, ob dieser Paragraf in einer Sammlung integriert sein soll oder nicht.

[9] In von Verlagen veröffentlichten Gesetzessammlungen werden oft Abkürzungen verwendet, die keinen offiziellen Charakter haben. Diese führen bei einer Suche im RIS zu keinem oder einem falschen Ergebnis.

1.3.3 Open Government Data

Am 13. Juli 2011 wurde vom österreichischen Bundeskanzleramt und einigen der Landeshauptstädte die „Cooperation Open Government Data Österreich“ gegründet. Ziel war und ist es, eine Plattform zu betreiben, unter der die Metadaten von in Österreich frei verfügbaren Verwaltungsdaten zusammengefasst werden. [BM13] Hierbei wird insbesondere auf die Open Data Prinzipien Wert gelegt. Diese wurden im *White Paper der Cooperation OGD Austria* [Eib+12] festgehalten. Die für diese Arbeit wichtigsten Prinzipien daraus sind:

Vollständigkeit Nur wenn Zugriff auf den gesamten Datenbestand des *Bundesrecht konsolidiert* gewährleistet ist, gibt eine Anwendung zum Erzeugen von Gesetzbüchern überhaupt Sinn.

Zeitnahe Zurverfügungstellung Hierdurch ist gewährleistet, dass die verarbeiteten Daten aktuell sind und aktuelle Gesetzesänderungen bei generierten Sammlungen und Vergleichen berücksichtigt werden können.

Maschinenlesbar Erst dadurch ist es sinnvoll möglich, die abgerufenen Daten automatisiert und strukturiert weiterzuverarbeiten.

Dauerhaftigkeit Eine dauerhafte und archivierende bzw. versionierende Datenstruktur erlaubt das lokale Zwischenspeichern von Daten.

Lizenzierung/ Nutzungskosten Die Daten werden unter der Creative Commons Lizenz *CC BY 3.0 AT*[10] zur Nutzung kostenfrei zur Verfügung gestellt. Deshalb kann OpenCodex kostenlos genutzt werden.

Das *Bundesrecht konsolidiert* ist auf `data.gv.at` als SOAP-Webservice[11] verfügbar (Details zu Webservices siehe 2.2). Dieser Webservice hat zwei definierte Schnittstellen, welche es, im Vergleich zur auf menschliche Bedienung ausgelegten Webseite des RIS, ermöglichen, dass ein Programm strukturiert Daten aus dem RIS abruft. Die Schnittstelle `request` beantwortet Suchanfragen; die möglichen Suchparameter sind der Online-Suchmaske (Abbildung 1) sehr ähnlich. Die Schnittstelle `getDocument` antwortet auf die Anfrage nach einer Dokumentnummer mit dem Inhalt des Dokumentes und dessen Metadaten in Form einer eXtensible Markup Language (XML)-Datei.

[10] Siehe `http://creativecommons.org/licenses/by/3.0/at/deed.de`, (22.07.2013)

[11] Im Deutschen ist die zusammengesetzte Schreibweise üblich, im Englischen hingegen schreibt man *web service*.

1.4 Bestehende Lösungen

Neben den oben erwähnten gedruckten Sammlungen von Gesetzestexten und den Vergleichsdokumenten des Parlaments wurden folgende Lösungen gefunden, welche jedoch nicht ausreichend den Anforderungen entsprechen:

1.4.1 RIS2PDF

Die im Rahmen eines Projektpraktikums von Markus Haudum erstellte Web-Anwendung *RIS2PDF* erlaubt es dem Benutzer, einzelne Dokumente aus dem RIS zu einer Sammlung (also einem Buch) zusammenzufassen und als druckbares PDF auszugeben. Es werden hierbei nicht nur Gesetze aus dem *Bundesrecht konsolidiert* unterstützt, sondern alle im RIS als Einzeldokument verfügbaren Rechtsvorschriften und Entscheidungen. Zu beachten ist hierbei jedoch, dass ein RIS-Einzeldokument immer nur ein einzelner Paragraph bzw. Artikel einer Rechtsvorschrift ist. Das Hinzufügen eines gesamten Gesetzes zu einer Sammlung auf einmal ist nicht möglich.

Da zum Zeitpunkt der Erstellung des Programms das Portal `data.gv.at`, welches offene Datensätze und Dienste (darunter auch das „Bundesrecht konsolidiert“ des RIS, siehe 1.3.3) anbietet, noch nicht existierte, wurden die Daten nicht über den nunmehr angebotenen Webservice abgefragt, sondern über die Webseite des RIS.

1.4.2 Gesetzbuch24.de

Der Dienst der Firma *RICHARD BOORBERG VERLAG GmbH & Co KG*, welcher auf der Seite `gesetzbuch24.de` angeboten wird, erlaubt es „aus über 8000 Gesetzen und Vorschriften des Europa-, Bundes- und Landesrechts eine topaktuelle Gesetzessammlung ganz nach Ihren Wünschen in Büchern zusammen[zu]stellen und online deren Druck [zu] bestellen“ [Ges13]. Die Seite bietet dabei neben dem Europarecht lediglich Zugriff auf das *deutsche* Bundes- und Landesrecht, auf österreichische Rechtsquellen kann nicht zugegriffen werden. Es ist ferner über die Seite nur möglich, kostenpflichtig eine fertig gedruckte und gebundene Gesetzessammlung zu erhalten, ein kostenloser Zugriff auf das zugrunde liegende Dokument ist nicht möglich.

1.4.3 RIS:App

Die Firma *right2innovation® BY WASS GmbH* bietet mit der Anwendung *RIS:App*[12] eine mobile Lösung für Android und iOS an. Das BKA stellt die Anwendung folgendermaßen vor[13]:

> „Die RIS:App bietet Ihnen die Möglichkeit, in der konsolidierten Fassung des Bundes- und Landesrechts zu suchen. Ferner werden auch eine Favoritenfunktionalität und eine Pushbenachrichtigung bei Novellierungen angeboten.“

Die Anwendung stellt allerdings Gesetze nicht zusammenhängend dar, sondern bietet lediglich Zugriff auf einzelne Paragrafen bzw. Artikel. Der Zugriff auf die Landesrechte erfolgt nicht über den angebotenen Webservice, sondern über eine nicht frei verfügbare Schnittstelle (siehe hierzu C.1.2 und C.2.2). Diese kann deshalb nicht in OpenCodex verwendet werden.

1.5 Neuer Lösungsansatz

Mit der Präsentation des Open Government Data (OGD) Portals `data.gv.at` im April 2012 und der Veröffentlichung einer Datenschnittstelle zur RIS-Anwendung „Bundesrecht konsolidiert“[14] wurden neue Möglichkeiten geschaffen, Gesetzestexte automatisiert in elektronischer Form zu verarbeiten. Das in der vorliegenden Arbeit vorgestellte Programm OpenCodex nutzt diese Schnittstelle, um das RIS zu durchsuchen und Dokumente (siehe 1.3.2.1) daraus abzurufen. Das Programm kommuniziert über den bereitgestellten Webservice mit dem Dokumentenserver. Die abgerufenen Dokumente liegen in einem definierten XML-Format vor und werden automatisch mit Hilfe des Textsatzsystems LaTeX in ein druckfertiges PDF übergeführt. Um den Dokumentenabruf zu beschleunigen werden Dokumente lokal zwischengespeichert (siehe 3.2.2).

[12] Siehe `http://www.right2innovation.com/risapp`, (22.07.2013)
[13] Siehe `http://www.ris.bka.gv.at/UI/RISApp.aspx`, (22.07.2013)
[14] Siehe `http://www.data.gv.at/datensatz/?id=31430a9f-c8ba-4654-ab68-c9c3dff0361b`, (22.07.2013)

2 Verwendete Technologien und Bibliotheken

OpenCodex verwendet eine Reihe von etablierten Technologien. Das Programm selbst ist in der Programmiersprache *Java*, Version 7, geschrieben. Die Dokumente aus dem RIS werden im *XML*-Format bereitgestellt, das Bereitstellen selbst erfolgt über einen *Webservice*. Die XML-Dokumente werden von einem Parser in *LaTeX* umgewandelt; der generierte LaTeX-Code stützt sich dabei auf verschiedene *Pakete*. Die Benutzeroberfläche (Graphical User Interface, GUI) funktioniert unter Zuhilfenahme des Framework-Standards *JavaServer Faces*. Es kommen hierbei verschiedene Komponenten aus der GUI-Komponentenbibliothek *PrimeFaces* zur Anwendung.

2.1 XML

Die aktuell gültige Definition von XML wird vom World Wide Web Consortium (W3C)[15] in der Empfehlung `REC-xml-20081126`, „*Extensible Markup Language (XML) 1.0 (Fifth Edition)*“ [Bra+08] festgehalten.

2.1.1 Entstehung

1996 wurde vom W3C eine Arbeitsgruppe unter der Leitung von Jon Bosak gegründet. Ziel war es, eine auf der Standard Generalized Markup Language (SGML) basierende, aber wesentlich vereinfachte Sprache zum strukturierten Speichern von Daten in Form von Textdateien zu erstellen. Es wurden hierzu im November 1996 im ersten Arbeitsentwurf (*working draft*) [BSM96, Kapitel 1.1] folgende Designziele definiert:

[15] Das W3C definiert sich selbst unter `http://www.w3.org/Consortium/mission`, (22.07.2013) folgendermaßen: “The W3C mission is to lead the World Wide Web to its full potential by developing protocols and guidelines that ensure the long-term growth of the Web.”

1. XML soll unkompliziert über das Internet nutzbar sein.
2. XML soll eine große Bandbreite von Anwendungen unterstützen.
3. XML soll mit SGML kompatibel sein.
4. Es soll einfach sein, Programme zu schreiben, die XML-Dokumente verarbeiten.
5. Die Anzahl optionaler Funktionen soll so gering wie möglich gehalten werden, idealerweise Null.
6. XML-Dokumente sollen menschenlesbar und hinreichend klar sein.
7. Das Design von XML soll schnell bereitgestellt werden.
8. Das Design von XML soll formal und prägnant sein.
9. XML-Dokumente sollen leicht zu erzeugen sein.
10. Knappheit im XML-Markup ist von geringer Bedeutung.

Bereits im Februar 1998 war die vom W3C begutachtete und bestätigte Version 1.0 der XML-Definition fertig. An dieser Definition hat sich seitdem wenig geändert; lediglich mit der fünften und aktuellen Edition wurden Beschränkungen bezüglich erlaubter Zeichen aufgehoben. Im Februar 2004 wurde die erste Ausgabe von XML 1.1 veröffentlicht. Diese hat jedoch in der Praxis wenig Bedeutung und es wird weiterhin die Verwendung der Version 1.0 empfohlen. [Har04, Kapitel 3]

2.1.2 Namensräume

XML-Element- und Attributnamen können in sogenannte Namensräume (engl. *namespaces*) gefasst werden, sodass es in einem einzelnen XML-Dokument möglich ist, verschiedene XML-Formate zu verwenden, ohne dass Überschneidungen zu befürchten sind. Namensräume werden in [Bra+09] beschrieben und spezifiziert. Der Standard-Namensraum wird als `xmlns`-Attribut eines Elementes – meist werden Namensräume für das gesamte Dokument im Wurzel-Element definiert – angegeben. Weitere, benannte, Namensräume werden über die Attribute `xmlns:nsname` angegeben, die darin definierten Elemente/Attribute werden dann über `nsname:elementname` bzw. `nsname:attributname` angesprochen. Will man z. B. in einem XHTML-Dokument auch die Komponentenbibliothek *PrimeFaces* verwenden, so kann man folgenden Wurzelknoten verwenden:

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:p="http://primefaces.org/ui">
```

XHTML-Tags können dann wie gewohnt verwendet werden, also etwa `<head>` oder `<br/>`. Um Primefaces-Komponenten einzubinden verwendet man dann z. B. `<p:layout>` oder `<p:spacer/>`.

2.1.3 DTD

Mit der Spezifikation von XML-Dokumenten selbst wird auch die *Document Type Definition* spezifiziert, die es ermöglicht, die Struktur eines XML-Dokumentes festzulegen. Struktur bedeutet in diesem Zusammenhang, welche Elemente innerhalb welcher anderen Elemente auftreten dürfen und welche Attribute ein Element haben darf oder muss. Dieser Form von Schemadefinition wird vorgeworfen, dass sie selbst nicht in XML-Syntax geschrieben wird, keine Namensräume und keine verschiedenen Datentypen, insbesondere Zahlen- und Datumsformate, unterstützt; deshalb wird sie an dieser Stelle nur der Vollständigkeit halber genannt, jedoch nicht weiter ausgeführt.

2.1.4 XML Schema

Aus den soeben genannten Vorwürfen entstand der Bedarf einer neuen Beschreibungssprache. In [MM99, Kapitel 5] werden an diese neue Sprache unter anderem folgende Anforderungen gestellt:

> "The XML schema language shall be:
>
> 1. more expressive than XML DTDs;
> 2. expressed in XML;
> 3. self-describing;
> 4. [...]"
>
> "The XML schema language must define:
>
> 1. mechanisms for constraining document structure (namespaces, elements, attributes) and content (datatypes, entities, notations);
> 2. mechanisms to enable inheritance for element, attribute, and datatype definitions;
> 3. [...]"

Aus diesen Forderungen entstand XML Schema, welches in Form einer W3C-Empfehlung im Mai 2001 veröffentlicht wurde. Diese wurde, von diversen Fehler bereinigt, als zweite Ausgabe im Oktober 2004 neu aufgelegt und in folgenden drei Dokumenten veröffentlicht: [FW04; Tho+04; BM04]. Die im April 2012 veröffentlichte Version 1.1 der Schema Definition wird momentan nicht weitläufig unterstützt[16] und ist deshalb von geringer Bedeutung.

[16] Die Java 7 API Dokumentation beschreibt etwa, dass ein `javax.xml.validation.SchemaFactory`-Objekt nur XML-Schema 1.0 unterstützen muss.

Ein XML Schema ist ein wohlgeformtes und valides[17] XML-Dokument, welches den Aufbau und die Struktur von XML-Dokumenten beschreibt, die diesem Schema nach valide sind. Die Dateiendung `.xsd` steht für *XML Schema Definition*.

XML Schema ist aufgebaut als Definition von Datentypen. Es gibt 19 vordefinierte, „einfache“ Ur-Datentypen (engl. *primitive datatypes*) wie etwa `string`, `boolean` oder `decimal`. Diese können in einem Schema direkt benützt werden, um XML-Elemente oder Attribute zu beschreiben. Sie können dabei je nach Typ mit verschiedenen Beschränkungen versehen werden. Bei Zeichenketten sind das etwa die minimale bzw. maximale Länge oder ein regulärer Ausdruck, dem die Zeichenkette entsprechen muss; Zahlen lassen sich z. B. durch Minimal- und Maximalwerte beschränken.

Aus diesen einfachen Datentypen kann man komplexe Typen erzeugen, indem man primitive und weitere komplexe Datentypen zu Sequenzen oder Auswahlen zusammenfasst. Die Bestandteile eines komplexen Datentyps können wieder mit verschiedenen Beschränkungen versehen werden. Listing 1 zeigt einen Ausschnitt aus der XSD-Datei, welche gespeicherte OpenCodex-Sammlungen validiert.

```
1  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2  <xs:schema version="1.0" xmlns:xs="http://www.w3.org/2001/XMLSchema">
3    <xs:element name="opencodex" type="book"/>
4    <xs:complexType name="book">
5      <xs:sequence>
6        <xs:element name="chapter" type="chapter" minOccurs="0" maxOccurs="
       unbounded"/>
7      </xs:sequence>
8      <xs:attribute name="fontSize" type="xs:string" use="required"/>
9      <xs:attribute name="paperFormat" type="xs:string" use="required"/>
10     <xs:attribute name="subTitle" type="xs:string"/>
11   ...
12   </xs:complexType>
13   ...
14 </xs:schema>
```

Listing 1: Ein Ausschnitt aus dem „OpenCodex-Sammlung“ XML Schema.

Das Beispieldokument definiert zunächst als Wurzel ein Element von Typ `book`. Dieses Element wird dann als komplexer, neuer Typ definiert. Ein `book`-Knoten enthält eine beliebige Anzahl von Elementen vom Typ `chapter` sowie die obligaten Attribute `fontSize` und `paperFormat`. Das Attribut `subTitle` muss nicht vorhanden sein. Weitere Attri-

[17] Es existiert unter `http://www.w3.org/2001/XMLSchema.xsd`, (22.07.2013) ein XML Schema mit dem XML Schemata selbst validiert werden können.

bute sowie die Definition eines `chapter`-Elements sind nicht in der obigen Darstellung sichtbar.

Das Beispiel zeigt auch, dass eine XSD-Datei ein normales XML-Dokument ist, welches den Namensraum `http://www.w3.org/2001/XMLSchema` unter dem Präfix „`xs:`“ verwendet.

2.2 Webservices

Das W3C definiert in [Boo+04, Kapitel 1.4] einen Webservice folgendermaßen:

> “A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.”

Ein Webservice erlaubt es also einem Programm, über ein Netzwerk, meist das Internet, strukturierte Daten von einem Server abzufragen.

2.2.1 WSDL

Die Webservice Description Language (WSDL) ist eine vom W3C spezifizierte Sprache zum Beschreiben von Webservices. Ein WSDL-Dokument ist eine XML-Datei, welche verschiedene Elemente eines Webservices beschreibt. Die aktuelle Version der WSDL-Spezifikation ist 2.0, die für OpenCodex verwendete WSDL-Datei ist jedoch im alten Format 1.1 (definiert unter [Chr+01]) gehalten, weshalb darauf näher eingegangen wird.

Ein WSDL-File beschreibt einen Webservice mittels der folgenden sechs Hauptelemente:

types legen mittels einer geeigneten Sprache (meist XSD) fest, mit welchen Datentypen in den *messages* zu rechnen ist.

message besteht aus mehreren Teilen, welche jeweils auf einen festgelegten Typ verweisen und bilden die definierten Nachrichten.

portType definiert eine Reihe von benannten Operationen. Eine Operation legt fest, welche Eingangs- und Ausgangsnachricht zu erwarten ist. Sie entsprechen der Schnittstellendefinition im engeren Sinn.

binding definiert das Protokoll und das Datenformat für je einen *portType*. Als Protokoll kommt meist SOAP zum Einsatz, siehe auch 2.2.2.

port legt die Adresse fest, unter der ein *binding* erreichbar ist.

service gliedert *ports* logisch.

Ein Programm, das aus einer WSDL-Datei Programmcode generiert (z. B. JAX-WS, siehe 3.2.1), weiß damit *welche Anfrage* (portType) mit *welchem Inhalt* (message) an *welche Adresse* (port) über *welches Protokoll* (binding) zu senden ist und mit *welcher Antwort* (portType) zu rechnen ist.

Listing 2 ist ein Ausschnitt aus der WSDL-Datei des RIS-OGD Webservice, welche unter der URL `http://data.bka.gv.at/RIS/OGDService.wsdl` zu finden ist. Der Ausschnitt zeigt die Definition der Ein- und Ausgabenachrichten der drei Schnittstellen `request`, `getDocument` und `version`.

```
<wsdl:portType name="OGDServiceSoap">
  <wsdl:operation name="request">
    <wsdl:input message="tns:requestSoapIn"/>
    <wsdl:output message="tns:requestSoapOut"/>
  </wsdl:operation>
  <wsdl:operation name="getDocument">
    <wsdl:input message="tns:getDocumentSoapIn"/>
    <wsdl:output message="tns:getDocumentSoapOut"/>
  </wsdl:operation>
  <wsdl:operation name="version">
    <wsdl:input message="tns:versionSoapIn"/>
    <wsdl:output message="tns:versionSoapOut"/>
  </wsdl:operation>
</wsdl:portType>
```

Listing 2: Ausschnitt aus der WSDL-Datei des RIS-OGD Webservice.

2.2.2 SOAP

SOAP, ursprünglich Abkürzung für *Simple Object Access Protocol*, seit Version 1.2 [ML07; Gud+07; Mor+07] als eigenständiger Begriff, ist eine Protokolldefinition des W3C, mit der Informationen per XML in einer dezentralisierten Netzwerkumgebung ausgetauscht werden können. Es wäre über den Einsatz sogenannter *Bindings* möglich, als darunter liegende Transportschicht verschiedene Protokolle zu verwenden, wohl am häufigsten verwendet und vom W3C selbst spezifiziert ist jedoch das Binding an HTTP. Da in OpenCodex die Verwendung von SOAP nur implizit durch automatisch ge-

nerierten Code geschieht, wird auf dessen Funktionsweise an dieser Stelle nicht weiter eingegangen. Es sei auf die oben genannten Quellen der Protokollspezifikation verwiesen.

2.3 TEX und LATEX

Das Textsatzsystem TEX[18] ist – meist unter Verwendung der Macrosammlung LATEX[18] – trotz seines Alters der Quasi-Standard im Satz (natur-)wissenschaftlicher Publikationen. Der Erfolg geht auf seine Überlegenheit bzgl. Zuverlässigkeit, Qualität, Geschwindigkeit und der Tatsache, dass die Verwendung gratis ist, zurück. Seine Fähigkeit, aus einer Klartextdatei ein typographisch hochwertiges Dokument (oft, aber nicht nur) im PDF-Format zu generieren, erscheint für die Verwendung von OpenCodex geradezu ideal.

2.3.1 Geschichte

Als Donald E. Knuth im Jahr 1976 eine überarbeitete Ausgabe des zweiten Bandes seines Standardwerks *The Art of Computer Programming* [Knu68; Knu69; Knu73; Knu11] veröffentlichen wollte, weigerte sich sein Verlag aus Kostengründen, das Buch in der alten „Monotype“-Technologie[19] zu setzen. Knuth war mit dem Ergebnis mehr als unzufrieden:

> “I didn’t know what to do. I had spent 15 years writing those books, but if they were going to look awful I didn’t want to write any more.” [Knu98, S. 5]

Als der Informatiker wenig später von der Möglichkeit des Digitaldruckes erfuhr, sah er einen Ausweg aus dem Dilemma. Er beschloss, ein Programm zu schreiben, das druckfertige Dokumente liefert, die seinen (hohen) Anforderungen genügen. Knuth musste rasch feststellen, dass das Projekt weit umfangreicher sein würde und weit länger dauern würde als zuerst gedacht. Die Form der Lettern, welche im Monotype-Druck verwendet wurden, lagen natürlich nicht digital vor, und der Versuch diese zu digitalisieren, lieferte für den Perfektionisten Knuth kein befriedigendes Ergebnis. Also schuf er zuerst das Programm METAFONT, welches die Erstellung von Schriftzeichen per Beschreibungssprache ermöglicht. [Knu98, Kapitel 1: *Digital Typography*]

[18] Es ist nur jeweils diese Schreibweise korrekt; bei nichtproportionalen Schriften sollte man TeX/LaTeX schreiben, nicht TEX/LATEX.

[19] Eine Setzmaschine, welche ab dem späten 19. Jahrhundert zum Einsatz kam und in den 1960er Jahren durch den Fotosatz abgelöst wurde.

In dem im Mai 1977 erstellten Berichtsentwurf `TEXDR.AFT`[20] [Knu98, Kapitel 24] beschreibt KNUTH die prinzipielle Intention und Arbeitsweise des Textsatzsystems TEX und geht auch auf dessen Aussprache ein. TEX sei, so KNUTH, abgeleitet von „technology“ dessen erste drei Buchstaben im griechischen Wortstamm $\tau\epsilon\chi$ lauten. Das χ in TEX sei deshalb auszusprechen wie in dem deutschen Wort „ach“ und nicht wie ein „k“ oder ein „cks“. Die von KNUTH hier an den Tag gelegte Kleinlichkeit soll seiner eigenen Aussage nach verdeutlichen, dass TEX ebenso auf Kleinigkeiten wert legt und deshalb Dokumente bester Qualität liefert. [Knu84, Kapitel 1: *The Name of the Game*]

Mitte der 1980er Jahre wollte der Mathematiker LESLIE LAMPORT unter Zuhilfenahme von TEX ein Buch schreiben, empfand aber die bisher dafür veröffentlichten Macro-Sammlungen als unzureichend. Er schrieb also selbst eine Reihe von Macros für TEX, um dessen Nutzung zu vereinfachen und anzupassen. Diese Macro-Sammlung wurde rasch unter dem Namen LATEX (Lamport-TEX) bekannt. [LZ00; Lam13] LATEX entwickelte sich aufgrund der einfacheren Nutzung schnell zum Quasi-Standard der TEX-Nutzung. Die aktuelle Version ist LATEX 2_ε.

2.3.2 Funktionsweise

LATEX unterscheidet sich in der Verwendung stark von Textverarbeitungsprogrammen, bei dem die Anzeige des Programms dem resultierenden Dokument entspricht. Bei Programmen wie dem Open-/LibreOffice Writer oder Microsoft Word gilt das Prinzip WYSIWYG (What You See Is What You Get), d. h. es wird im Programm das aktuell zu bearbeitende Dokument (fast) so angezeigt, wie das fertige Dokument aussehen wird. Eine Überschrift etwa wird in Fettdruck und größerer Schrift dargestellt. Bei der Verwendung von LATEX hingegen *be*schreibt man den Text, man weist also einem Text z. B. die Eigenschaft „Kapitelüberschrift“ zu. Dieses Beschreiben des gewünschten Textes erfolgt in Form einer reinen Textdatei ohne optische Textauszeichnung. Dieser Quelltext wird dann von dem LATEX-Compiler in das gewünschte Ausgabeformat, heutzutage meist PDF, umgewandelt.

Listing 3 zeigt den Quelltext eines Beispieldokumentes, welches einige der Möglichkeiten von LATEX demonstrieren soll. Die einzelnen Zeilen haben hierbei folgende Bedeutung:

`1` gibt den Dokumenttypen an, in diesem Fall ein Artikel aus dem KOMA-Script (siehe 2.3.3.1); die Schriftgröße `11pt` wird der Dokumentenklasse als Parameter übergeben. Generell gilt in LATEX, dass benötigte Parameter – in diesem Fall der

[20] KNUTHS damaliger Computer konnte nach eigener Aussage einen Dateinamen wie `TEX.DRAFT` nicht verarbeiten.

```
1  \documentclass[11pt]{scrartcl}
2  \usepackage[T1]{fontenc}
3  \usepackage{textalpha}
4  \pagestyle{empty}
5  \begin{document}
6  \section{\emph{Hello, World!} in different languages}
7  This little example should illustrate some of the capabilities of \LaTeX{}.
8  \subsection{German}
9  Hallo, sch{\"o}ne Welt!
10 % examples taken from http://www.roesler-ac.de/wolfram/hello.htm
11 \subsection{Greek}
12 \TextGreek{Geia sou k\'osme!}
13 \section{\LaTeX{} can do math very well}
14 This (probably senseless) formula shows some of \LaTeX's math-features:
15 \[\sum_{i=6}^{12} x_i = \int_0^\pi \mathrm{e}^{-\sqrt[3]{y}}\,\mathrm{d}y+
16 \frac{1+x}{\frac{x^2}{2y}}\]
17 \end{document}
```

Listing 3: Ein LATEX-Beispieldokument.

Name der Dokumentenklasse – in geschwungenen Klammern angegeben werden und optionale Parameter – häufig Abweichungen von Default-Werten – in eckige Klammern gesetzt werden.

2 deklariert das Schriftencoding und weist LATEX an, statt den veralteten OT1 Schriften die moderneren T1 Schriften zu verwenden. Dies ist u. A. für eine bessere Unterstützung von Umlauten nötig.

3 bindet das Paket `textalpha` ein, welches die einfache Eingabe von griechischen Texten ermöglicht.

4 definiert den Seitenstil als *leer*, also ohne Kopf- und Fußzeile.

5 gibt an, dass die Präambel des Dokumentes zu Ende ist und das eigentliche Dokument beginnt.

6 definiert einen neuen Abschnitt. Im fertigen Dokument wird dieser dann in einer anderen Schriftart, größer und fett gesetzt, sowie ein vordefinierter Abstand darunter und darüber frei gelassen. Es wird ausserdem eine automatische Abschnittsnummerierung eingefügt. Der von `\emph{` und `}` umschlossene Text wird zusätzlich hervorgehoben (engl. „*emph*asize“). Diese Hervorhebung ist standardmässig Kursivdruck.

7 ist normaler Fließtext. Der Befehl `\LaTeX{}` gibt an, dass an dieser Textstelle der LATEX-Schriftzug gesetzt werden soll.

8 erzeugt einen neuen Unterabschnitt mit der Bezeichnung „German“. Auch hier wird wieder die Überschriften-Schrift, eine größere Schriftart und Fettdruck verwendet und ein weiterer (Unter-)Zähler an die Abschnittszählung angehängt.

9 ist der Text dieses Unterabschnitts. Das deutsche „ö“ wird durch die umschreibende Zeichenkombination `{\"o}` oder vereinfacht `\"o` erzeugt. Da dies, insbesondere wenn man längere deutsche Texte schreibt und keinen geeigneten Texteditor zur Hand hat, recht umständlich ist, ist es über weitere Pakete möglich, Umlaute weiter vereinfacht (`"o`) oder direkt einzugeben. Hierbei ist jedoch dann die Kodierung der Quelldatei zu beachten. Aus diesem Grund verwendet OpenCodex die vereinfachte Eingabe nicht, sondern verwendet für Sonderzeichen generell deren Umschreibung.

10 beginnt mit einem %-Zeichen. Dieses leitet einen Kommentar im Quelltext ein, der Rest der Zeile wird ignoriert. Wollte man im Text ein %-Zeichen verwenden, so müsste man `\%` schreiben.

11 definiert erneut einen Unterabschnitt.

12 verwendet den von dem Paket `textalpha` bereitgestellten Befehl `\TextGreek`, um einen Text mit lateinischen Buchstaben in deren Äquivalent im griechischen Alphabet darzustellen. Der Akzent über dem Buchstaben ω wird durch die vorangestellte Zeichenfolge `\'` erzeugt.

13 definiert einen neuen Abschnitt und zeigt, dass der LaTeX-Schriftzug auch in Überschriften gesetzt werden kann.

14 ist wieder normaler Fließtext.

15 leitet durch `\[` den Mathematik-Modus zum Setzen von Formeln ein. Der Befehl `\sum` erzeugt das Summenzeichen Σ, der Unterstrich _ leitet den Text darunter ein bzw. das Zirkumflex-Zeichen ^ den Text darüber. Ähnliches gilt für das Integral `\int`. Der Befehl `\mathrm{<Text>}` erlaubt das Setzen von aufrechtem Text innerhalb von Formeln.

16 zeigt, dass es über den Befehl `\frac{<Zähler>}{<Nenner>}` möglich ist, Brüche (engl. *frac*tions), auch geschachtelt, zu setzen. Der Befehl `\]` beendet den Mathematik-Modus und somit die Formel.
Zu beachten ist, dass der Zeilenumbruch von Zeile 15 auf 16 im Quelltext weder die Formel umbricht noch einen neuen Absatz generiert. Um in LaTeX einen manuellen Zeilenumbruch ohne neuen Absatz zu bekommen, verwendet man die Zeichenfolge `\\`. Will man einen neuen Absatz erhalten, so genügt ein doppelter Zeilenumbruch, also eine leere Zeile im Quelltext.

17 zeigt an, dass das Dokument hier zu Ende ist; jegliche Zeichenketten, die im Quelltext danach kommen, werden ignoriert.

Die folgende Seite ist das unveränderte Beispieldokument, ausgegeben als PDF-Datei; aufgrund des Seitenstils *empty* enthält die Seite keine Seitenzahl.

1 *Hello, World!* in different languages

This little example should illustrate some of the capabilities of LaTeX.

1.1 German

Hallo, schöne Welt!

1.2 Greek

Γεια σου κόσμε!

2 LaTeX can do math very well

This (probably senseless) formula shows some of LaTeX's math-features:

$$\sum_{i=6}^{12} x_i = \int_0^{\pi} \mathrm{e}^{-\sqrt[3]{y}}\,\mathrm{d}y + \frac{1+x}{\frac{x^2}{2y}}$$

2.3.3 Pakete[21]

Die Verwendung von LaTeX mit seinen Standardeinstellungen erzeugt meist gute bis sehr gute Ergebnisse. Oftmals kann die Qualität eines Dokumentes durch geringfügige manuelle Eingriffe weiter verbessert werden. Da jedoch bei OpenCodex das Setzen eines Dokumentes für den Benutzer verborgen im Hintergrund abläuft, muss der automatisch generierte Code so generisch wie möglich gestaltet sein. Hierzu sind eine Reihe von Paketen notwendig bzw. hilfreich. OpenCodex verwendet eine Vielzahl von Paketen, manche ergeben lediglich kleine Änderungen oder bieten selten verwendete Befehle. Die folgende Aufstellung geht deshalb nur auf die wichtigsten ein.

2.3.3.1 KOMA-Script – A bundle of versatile classes and packages

Die Dokumente, welche aus den „Standard“-Dokumentklassen erzeugt werden, sind auf die typographischen Gewohnheiten im anglo-amerikanischen Raum optimiert. Deshalb entwickelte Frank Neukam Anfang der 1990er Jahre auf die Regeln europäischer Typografie optimierte Stile für LaTeX, genannt Script. Als Mitte 1994 durch den Versionswechsel von LaTeX 2.09 auf LaTeX 2$_\varepsilon$ viele Änderungen nötig waren, beschloss Markus Kohm, die Stile zu überarbeiten und die Script-Sammlung zu adaptieren. KOMA-Script war geboren. In den folgenden Jahren hat sich KOMA-Script, v. a. wegen seiner Wandelbarkeit, weit über den deutschsprachigen Raum hinaus zu einer häufig genutzten Alternative zu den Standardklassen entwickelt. [KM11, Kapitel 1.3]

Eine Gesetzessammlung in OpenCodex wird im Dokumentstil `scrbook` gesetzt. Wie der Name bereits vermuten lässt ist dieser Stil (als KOMA-Script-Ersatz der Klasse `book`) zum Setzen von Büchern gedacht. Der Satz ist hierbei standardmäßig auf zweiseitigen Druck (d. h. es wird angenommen, dass ein Blatt vorne und hinten bedruckt wird) angepasst, die oberste Gliederungseinheit ist ein Kapitel (`\chapter{<Kapitelname>}`); dies entspricht in OpenCodex einem Gesetz einer Sammlung. Ein neues Kapitel beginnt immer auf der rechten Seite, es wird also eventuell eine leere (linke) Seite eingefügt. Ferner wird eine Kopfzeile aktiviert, welche die Seitenzahl enthält und auf jeder linken Seite zusätzlich den Namen des Kapitels.

Da ein Fassungsvergleich nur ein Gesetz und somit ein Kapitel umfasst, wird hierfür der Dokumentstil `scrartcl` – die verbesserte Version von `article` – verwendet. Hierbei ist der Satz einseitig (d. h. es wird angenommen, dass ein Blatt nur vorne), die Kopfzeile

[21] Im Englischen wird in diesem Zusammenhang zwischen einem komplexen Paket, einem *bundle*, und einem einfachen Paket, einem *package*, unterschieden, wobei Ersteres meist eine Vielzahl von Letzteren enthält. Zur Vereinfachung wird dieser Unterscheidung im Weiteren nicht Rechnung getragen.

mit Seitenzahl muss manuell aktiviert werden und die höchste Gliederungseinheit ist ein Abschnitt (`\section{<Abschnittsname>}`); dieser entspricht in OpenCodex einer Änderung in einem Paragrafen bzw. Artikel.

2.3.3.2 babel – Multilingual support for Plain TEX or LATEX

LATEX ist, wie bereits erwähnt, für sich allein genommen auf das Erstellen (US-)englischer Texte ausgelegt. Dies betrifft neben typografischer Eigenheiten auch diverse Bezeichnungen wie „abstract“, „bibliography“, „list of . . . “ und die Regeln der automatischen Silbentrennung.

Das Paket `babel` erlaubt es, die Sprache in der LATEX „denkt“, festzulegen und innerhalb eines Dokumentes zu ändern. Man bindet das Paket über den Befehl `\usepackage [<Sprachen>]{babel}` ein. In den optionalen Parametern werden die gewünschten Sprachen, durch Beistrich getrennt, angegeben, wobei die letzte Angabe die aktuell zu ladende Sprache ist. Über den Befehl `\selectlanguage{<Sprachname>}` kann innerhalb des Dokumentes die aktuell zu verwendende Sprache umgeschaltet werden. In OpenCodex ist als einzige Sprache `naustrian`, also österreichisches Deutsch mit neuer Rechtschreibung, geladen, da die Daten aus dem RIS keine Sprachinformationen enthalten und mit äußerst wenig anderssprachigem Text in einem Gesetz zu rechnen ist. [Bra08]

2.3.3.3 multicol – Intermix single and multiple columns

OpenCodex erlaubt es, Gesetzestexte ein-, zwei- oder dreispaltig zu setzen. Da jedoch nicht jedes Element einer Gesetzessammlung mehrspaltig gesetzt werden kann[22], ist es nicht möglich, die KOMA-Script-eigene Funktion für zweispaltigen Satz zu verwenden, sondern es muss der mehrspaltige Satz innerhalb eines Dokumentes an- und wieder abgeschaltet werden können.

Dies ist über das Paket `multicol` möglich. Es bietet hierzu die Umgebung `multicols`. Beim Öffnen der Umgebung wird die Anzahl der Spalten als zweiter Parameter angegeben (z. B. `\begin{multicols}{2}`). Eine Umgebung in LATEX wird immer mit `\begin {<Umgebungsname>}` begonnen und endet mit `\end{<Umgebungsname>}`, wobei beim Öffnen, je nach Umgebung, die Angabe weiterer Parameter nötig oder möglich ist. Es kann mit `multicol` also durch die Verwendung verschiedener Umgebungen innerhalb einer Seite beliebig oft zwischen ein- und zwei- (oder mehr-)spaltigem Satz gewechselt werden. [Mit11]

[22] Insbesondere Tabellen und breite Grafiken verursachen Probleme.

2.3.3.4 tabu – Flexible LaTeX tabulars

Dieses Paket wird nur bei Gesetzessammlungen verwendet.

Eine Tabelle wird in LaTeX üblicherweise in einer `tabular`-Umgebung gesetzt. Deren Funktionsumfang ist oft ausreichend, hat jedoch einige Nachteile:

- Es ist lediglich möglich, einzelne Spaltenbreiten fix anzugeben; die Angabe eines Verhältnises zueinander ist nicht möglich.
- Die Tabellenbreite wird automatisch errechnet und kann unter Umständen die gewünschte Seitenbreite überschreiten, d. h. die Tabelle ragt über den rechten Rand hinaus.
- Eine Tabelle kann maximal eine Seite lang sein. Ist der Inhalt länger, so steht er über den unteren Rand hinaus, die Tabelle wird nicht auf die nächste Seite umgebrochen.

Es existieren verschiedene Pakete um diese Probleme zu behandeln, das umfangreichste und für OpenCodex am besten geeignete ist wohl das Paket `tabu`, in eigener Schreibweise $\tau_{\aleph}b\subset$. Es bietet mit `longtabu` eine Tabellenumgebung, die über mehrere Seiten hinweg gehen kann. Definiert man z. B. eine Tabelle mit dem Befehl
`\begin{longtabu} to .8\textwidth{|X[1]|X[2]|X[1]|}`
so hat diese folgende Eigenschaften:

- Die Tabelle geht, so der Platz benötigt wird, über mehrere Seiten. Hat man mittels `\endhead` bzw. `\endfoot` einen Tabellenkopf/-fuss definiert, wird dieser auf jeder neuen Seite erneut angezeigt.
- Die Tabelle nimmt 80% des für Text zur Verfügung stehenden Platzes ein.
- Die Spalten werden mit einer Linie getrennt.
- Die Spaltenbreiten haben das Verhältnis 1:2:1, die mittlere Spalte ist also doppelt so breit wie die beiden äußeren.

[Che11]

2.3.3.5 adjustbox – Graphics package-alike macros for “general” boxes

Normalerweise werden Bilder in LaTeX mit Hilfe des Befehls `\includegraphics [<Optionen>]{<Pfad/zur/Bilddatei>}` aus dem Paket `graphicx` eingebunden. In den Optionen kann man zwar die gewünschte Breite und/oder Höhe des Bildes angeben, das Bild wird aber, wenn es kleiner ist, auf die angegebene Größe gestreckt. Ist dieses Verhalten nicht gewünscht, so bietet das Paket `adjustbox` Abhilfe. Wenn man es über

`\usepackage[export]{adjustbox}` lädt, gibt es die Möglichkeit, eine *maximale* Breite oder Höhe anzugeben, an den Befehl zum Einbetten von Grafiken weiter. Man kann also dann z. B. über den Befehl `\includegraphics[max width=.45\textwidth]{<Pfad>}` angeben, dass das Bild maximal 45% der Textbreite einnehmen darf. [Sch12]

2.3.3.6 imakeidx – A package for producing multiple indexes

Dieses Paket wird nur bei Gesetzessammlungen verwendet.

Das Paket `imakeidx` erlaubt es, mehrere Stichwortverzeichnisse in einem LATEX-Dokument anzulegen. Nach dem Laden über `\usepackage{imakeidx}` kann mit dem Befehl `\makeindex[name=<Bezeichnung>,title=<Überschrift>]` ein neuer benannter Index erzeugt werden, wobei dessen Titel, also z. B. „Schlagwortverzeichnis“ mit angegeben wird. Über den Befehl `\index[<Bezeichnung>]{<Stichwort>}` fügt man im Fließtext einen Anker für das Stichwort ein, auf den dann im Verzeichnis, zusammen mit der korrekten Seitenzahl, referenziert wird. Mit `\printindex[<Bezeichnung>]` wird das übergebene Stichwortverzeichnis im Text gesetzt. In OpenCodex werden die Schlagworte der Einzeldokumente verwendet, um pro Gesetz einen Index zu erzeugen; Details zur technischen Umsetzung finden sich in 3.2.4.2. [BG13]

2.3.3.7 paracol – Multiple columns with texts “in parallel”

Dieses Paket wird nur bei Fassungsvergleichen verwendet.

In 2.3.3.3 wurde bereits die Möglichkeit dargestellt, mehrspaltigen Text zu setzen. Man hat jedoch hierbei keinen direkten Zugriff auf die einzelnen Spalten, sondern der Text läuft von links nach rechts in die jeweils nächste Spalte. Beim Fassungsvergleich von OpenCodex ist es jedoch nötig, zwei Spalten separat mit Text zu befüllen.

Das Paket `paracol` bietet diese Funktion. Mit `\begin{paracol}{2}` wird eine Umgebung mit zwei separat ansprechbaren Spalten definiert. Diese können dann innerhalb dieser Umgebung in den beiden Umgebungen `leftcolumn` und `rightcolumn` mit Text gefüllt werden. [Nak12]

2.4 JavaServer Faces

Die folgenden Ausführungen über JavaServer Faces (JSF) basieren auf dem Tutorial der Java Platform, Enterprise Edition 6 (Java EE 6) [Jen+13].

JSF ist eine Technologie der Java EE, die es ermöglicht, auf einfache Weise Benutzeroberflächen für Java Web-Anwendungen zu erzeugen. Die mit JSF gebauten Benutzeroberflächen sind aus Komponenten aufgebaut, welche an serverseitige Variablen in speziellen Java-Klassen gebunden werden können. Seit der Version 2.0 wird in JSF Asynchronous JavaScript and XML (AJAX) nativ unterstützt. Mit AJAX ist es möglich, Teile einer Webseite beim Client auszutauschen, ohne dass die Seite komplett neu geladen werden muss.

Eine JSF-Seite ist eine XML-Datei, die den Aufbau der gewünschten Benutzeroberfläche beschreibt. Daraus generiert der Anwendungsserver eine XHTML-Datei, die an den Client übermittelt wird. Die JSF-Seite wird mit der JSF-Expression Language (EL) an Variablen und Methoden im *Managed Bean* gebunden. Ein Managed Bean ist eine normale Java-Klasse, welche mit der Annotation `@ManagedBean(name="beanName")` versehen wird. Sie muss kein besonderes Interface implementieren. Die Bindung einer Variable im Bean an ein Feld der JSF-Seite erfolgt mit Ausdrücken der Form `#{beanName.variableName}`. Hierbei wird vorausgesetzt, dass im Bean die Methoden `public String getVariableName()` und – so die Variable auch gesetzt werden soll – `public void setVariableName(String variable)` existieren.

Der normale Geltungsbereich (engl. *scope*) eines Managed Beans ist ein HTTP-Request. Hierbei ist zu beachten, dass das Lesen von Werten eines Formulars beim GET-Request zum Darstellen der Seite für den Benutzer und das anschließende Zurückschreiben geänderter Werte bei einem POST-Request zwei unabhängige Requests darstellen. Es wird also beim Setzen der Werte eine neue Instanz des Beans erzeugt. Um dieses Verhalten zu ändern kann man, erneut über Annotationen, die „Lebensdauer“ einer Klasse steuern. In OpenCodex werden die beiden Einstellungen `@SessionScoped` und `@ApplicationScoped` verwendet. Erstere bedeutet, dass die Java-Klasse erzeugt wird, wenn sie in einer Session das erste Mal verwendet wird (*lazy instantiation*) und aktiv bleibt, solange die Session existiert. Das Ende einer Session ist entweder nach Ablauf einer konfigurierbaren maximalen Inaktivitätsdauer oder wenn die Sitzung programmatisch über den statischen Aufruf `FacesContext.getCurrentInstance().getExternalContext().invalidateSession();` beendet wird. Bei Zweiterer wird eine Instanz des Beans ebenfalls bei erster Benutzung erzeugt. Diese existiert dann, bis das Programm beendet wird und

wird von allen Sessions gemeinsam verwendet. Über die parametrierte Annotation `@ManagedBean(eager=true)` kann man konfigurieren, dass das Bean sofort beim Programm- bzw. Sessionstart erzeugt wird und nicht erst beim ersten Zugriff.

Das Beispiel in den Listings 4 und 5 zeigt eine einfache JSF-Seite, welche die Variable `name` aus einem Bean nachlädt.

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:p="http://primefaces.org/ui">
   <h:head>
      <title>Sample Page</title>
   </h:head>
   <h:body>
      Hello <h:outputText value="#{myBean.name}" />!
   </h:body>
</html>
```

Listing 4: Quelltext einer einfachen JSF-Seite.

```
@ManagedBean(name="myBean")
@SessionScoped
public class MyBean {
  private String name;
  public MyBean() {
    // initialize class
    name = "World";
  }
  public String getName() {
    return name;
  }
  public void setName(String n) {
    name=n;
  }
}
```

Listing 5: Quelltext eines einfachen Managed Beans.

2.4.1 PrimeFaces

JSF setzt auf die Verwendung von Komponentenbibliotheken. Diese bieten eine Vielzahl von Elementen für eine Benutzeroberfläche. Die Elemente sind dabei oft stark an Komponenten angelehnt, die von „normalen“ Desktop-Benutzeroberflächen bekannt sind.

Für die Oberfläche von OpenCodex wird die Sammlung PrimeFaces[23] verwendet, da sie im Vergleich mit den Konkurrenten RichFaces[24] und IceFaces[25] am umfangreichsten die benötigten Funktionalitäten bietet. Unter 3.3 werden die wichtigsten dieser Funktionen und Komponenten diskutiert. Abbildung 2 zeigt exemplarisch die PrimeFaces-Komponenten zur Datumsauswahl.

Abbildung 2: Kalender- bzw. Datumsauswahlfeld-Komponente aus PrimeFaces.

Die Integration von PrimeFaces erfolgt wie das Einbinden einer normalen Bibliothek in Java. Nach dem Einbinden und der Definition eines eigenen Namensraums für PrimeFaces-Komponenten (siehe Attribut `xmlns:p` in Listing 4) stehen in den JSF-Seiten die Komponenten zur Verwendung bereit. Das Aussehen einer mit PrimeFaces designten Seite kann in der Projektdatei `web.xml` über den Kontext-Parameter `primefaces.THEME` verändert werden. Es stehen standardmässig die Themes des jQuery ThemeRoller[26] zur Verfügung, es können jedoch auch eigene Themes gestaltet werden. In OpenCodex kommt der Theme *humanity* zum Einsatz.

[23] Siehe `http://primefaces.org`, (22.07.2013)
[24] Siehe `http://www.jboss.org/richfaces`, (22.07.2013)
[25] Siehe `http://www.icesoft.org/java/projects/ICEfaces`, (22.07.2013)
[26] Siehe `http://jqueryui.com/themeroller`, (22.07.2013)

3 Implementierung

Die Implementierung von OpenCodex erfolgte in Java 7. Als Entwicklungsumgebung kam die Netbeans IDE[27] in Version 7.3 zum Einsatz. Netbeans enthält in den Download-Versionen „Java EE“ und „All“ bereits den Java-Anwendungsserver GlassFish Server Open Source Edition 3.1.2.2, auf welchem OpenCodex läuft.

3.1 Struktur

OpenCodex ist in zwei Package-Äste unterteilt, `opencodex.backend` und `opencodex.frontend`. Die Struktur der Anwendung innerhalb des Servers, die interne Kommunikation mit der Caching-Datenbank bzw. der LaTeX-Installation und die externe Kommunikation mit dem Client bzw. dem RIS-Server sind in Abbildung 3 ersichtlich.

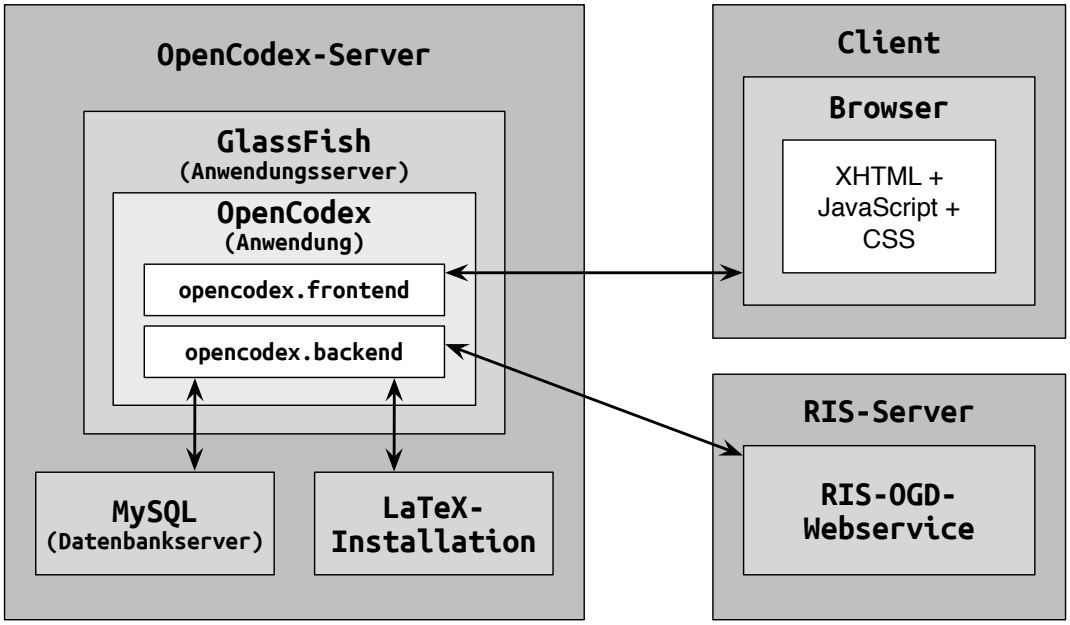


Abbildung 3: Struktur und Kommunikation von OpenCodex.

[27] Siehe `http://netbeans.org`, (22.07.2013)

3.2 Back-End

Das Back-End ist in folgende Unter-Packages gegliedert:

indexloader Klassen zum Laden, Parsen und Abbilden des Index des Bundesrechts, siehe 3.2.5.

jaxb Automatisch erzeugte Klassen, welche für die XSD-Definitionen der Antwortdokumente der Webservice-Anfragen erzeugt wurden. Die Klassen werden aus den Dokument-Definitionen[28] vom JAXB-Parser generiert. Die *Java Architecture for XML Binding* (JAXB) erlaubt es, ausgehend von XML-Schemata – in Form von XSD-Dateien – Java-Klassen zu erzeugen. Nach dem jeweiligen Schema valide XML-Dokumente können dann in Objekte umgewandelt werden (genannt *Unmarshalling*) bzw. kann umgekehrt aus den Objekten eine XML-Datei serialisiert werden (*Marshalling*).

lawloader Klassen, welche das Laden von Gesetzestexten ermöglichen. Weiter untergliedert in drei Packages:

- **data** Enthält Datenklassen, welche die abgerufenen bzw. zwischengespeicherten Dokumente repräsentieren und die Schnittstelle zur Caching-Datenbank.
- **documents** Klassen zum parallelisierten Abrufen von Dokumenten.
- **ids** Klassen zum parallelisierten Senden von Suchanfragen.

pdfgen Klassen, welche aus den abgerufenen XML-Dokumenten LaTeX-Code erzeugen (siehe 3.2.4).

utils Hilfs- und Konfigurationsklassen.

versioncompare Klassen zum Erstellen von Versionsvergleichen (siehe 3.2.4.3).

ws Automatisch aus der WSDL-Datei des RIS-Webservices generierte Klassen, mit Hilfe derer OpenCodex mit dem Webservice kommuniziert.

3.2.1 Webservice-Schnittstelle

In Netbeans kann man über die Benutzeroberfläche einen *Web Service Client* zu einem Projekt hinzufügen. Abbildung 4 zeigt die Eingabemaske hierzu.

[28] Die XSD-Dateien unter `http://www.data.gv.at/datensatz/?id=31430a9f-c8ba-4654-ab68-c9c3dff0361b`, (22.07.2013).

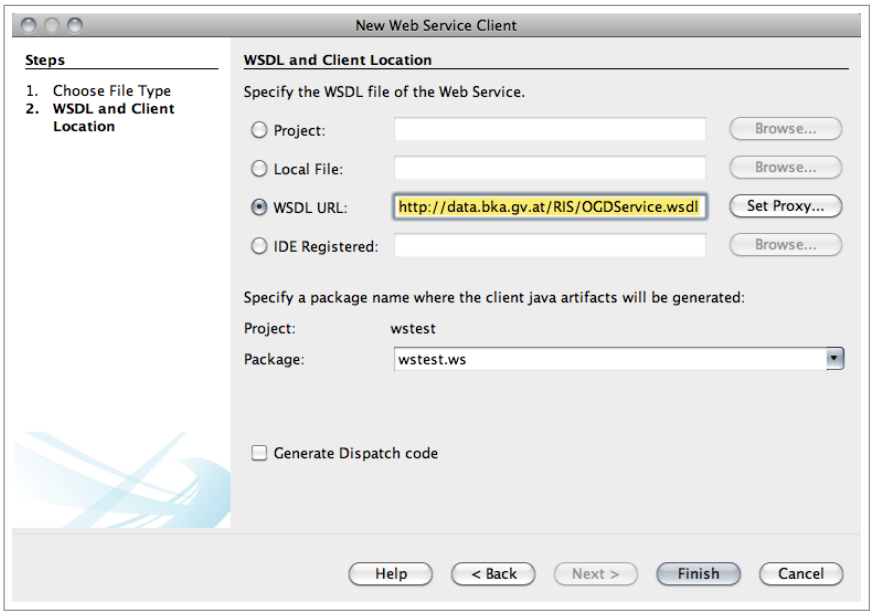


Abbildung 4: Eingabemaske zum Anlegen eines Web Service Clients in Netbeans.

Hat man keine graphische Entwicklungsumgebung zur Verfügung, kann das seit Java Version 6 mitgelieferte Konsolenprogramm `wsimport` verwendet werden, um aus einer WSDL-Datei oder der URL dazu den entsprechenden Code zu generieren. Mit Hilfe von Parametern kann das für den Code gewünschte Package und der Ausgabepfad angegeben werden.

Der Webservice bietet drei Methoden: Mit der Methode `String getDocument (String application, String docId)` kann man anhand einer Dokumentnummer das zugehörige Dokument abrufen. Als Parameter `application` wird bei der verwendeten Version des Webservices (1.3.1) nur „Br“ für das *Bundesrecht konsolidiert* akzeptiert. Das Format des gelieferten Dokumentes ist in einer Reihe von kommentierten XSD-Dateien unter `http://www.data.gv.at/datensatz/?id=31430a9f-c8ba-4654-ab68-c9c3dff0361b` definiert; die Definition der eigentlichen Dokument-Nutzdaten ist in der Datei `http://data.bka.gv.at/RIS/XSD/RISJudikaturNutzdaten.xsd` zu finden. Die Methode `String request(String application, String query)` dient der Suche nach relevanten Dokumentnummern; hier ist die Anwendung ebenfalls „Br“. Mit der Methode `String version()` kann die Versionsnummer des Webservice überprüft werden.

Die `query` ist ein XML-Dokument mit vorgegebenem Format. Da in OpenCodex nicht alle Suchmöglichkeiten des Webservices benötigt werden, ist es nicht nötig, das Suchdokument dynamisch zu generieren. Es wurde deshalb eine statische Vorlage erzeugt,

die dann mit Hilfe der *Freemarker Template Engine Library*[29] mit den benötigten Suchdaten gefüllt wird. Hierzu werden in einer Datencontainer-Klasse die gewünschten Suchdaten (Suchwort oder Index, Gültigkeitsdatum, Von-/Bis-Paragraf/Artikel, Seitennummer) zusammengefasst und mittels des Methodenaufrufs `template.process (Object searchData, Writer out)` in der Variable `out` zusammengefügt. Diese wird dann dem Webservice als Suchanfrage übergeben.

Wenn der Webservice nicht verfügbar ist, bleibt eine Anfrage standardmäßig hängen. Man kann jedoch einen Timeout-Wert hinterlegen. Hierzu ist jeweils ein Wert für den Verbindungs- und Request-Timeout zu setzen. Da die einschlägigen Informationsquellen (Postings auf `stackoverflow.com` bzw. Foreneinträge)[30] hierzu nicht eindeutig sind, wurde die Einstellung an verschiedenen Stellen getroffen, wie in Listing 6 ersichtlich.

```
1  service = new OGDService().getOGDServiceSoap12();
2  System.setProperty("sun.net.http.retryPost", "false");
3  Map<String, Object> rc = ((BindingProvider) service).getRequestContext();
4  rc.put("com.sun.xml.ws.connect.timeout", TIMEOUT);
5  rc.put("com.sun.xml.ws.request.timeout", TIMEOUT);
6  rc.put("com.sun.xml.internal.ws.connect.timeout", TIMEOUT);
7  rc.put("com.sun.xml.internal.ws.request.timeout", TIMEOUT);
8  rc.put("sun.net.client.defaultConnectTimeout", TIMEOUT);
9  rc.put("sun.net.client.defaultReadTimeout", TIMEOUT);
10 System.setProperty("sun.net.client.defaultConnectTimeout", "" + TIMEOUT);
11 System.setProperty("sun.net.client.defaultReadTimeout", "" + TIMEOUT);
```

Listing 6: Setzen der Timeout-Werte für den Webservice.

Zusätzlich wurde festgestellt, dass der Webservice nicht zu 100 % zuverlässig funktioniert und manche Anfragen nicht beantwortet, obwohl er verfügbar ist. Aus diesem Grund wird jede Anfrage bis zu drei Mal wiederholt, bevor ein Fehler gemeldet wird. Das Timeout- und Retry-Handling wurde in eine Webservice-Wrapperklasse gepackt, über die der weitere Zugriff auf den Webservice erfolgt.

[29] Siehe `http://freemarker.sourceforge.net`, (22.07.2013)

[30] Siehe `http://stackoverflow.com/questions/2148915/how-do-i-set-the-timeout-for-a-jax-ws-webservice-client`, (22.07.2013); `http://stackoverflow.com/questions/13967069/how-to-set-timeout-for-jax-ws-webservice-call`, (22.07.2013); `http://www.java-forum.org/soa/107354-jax-ws-client-timeout-problem.html`, (22.07.2013)

3.2.2 Parallelisierung und Caching

Beim Testen der Webservice-Funktionalität wurde festgestellt, dass eine einzelne Dokumentenabfrage im Durchschnitt knapp eine Sekunde dauert[31]. Der Großteil der Abrufdauer ist dabei die Wartezeit, bis der Webservice eine Antwort schickt. Geht man von umfangreichen Gesetzen wie dem Allgemeinen bürgerlichen Gesetzbuch (ABGB) mit weit über 1000 Einzeldokumenten aus, würde der Abruf eines solchen Gesetzes 15 Minuten und mehr dauern. Dies würde die Bedienung von OpenCodex extrem verlangsamen und für den Benutzer uninteressant gestalten. Aus diesem Grund wurden zwei Maßnahmen getroffen: Parallelisierung des Webservice-Zugriffs und lokales Caching.

3.2.2.1 Parallelisierter Webservice-Zugriff

Der Dokumentenabruf und die Suche nach Dokumentnummern erfolgt parallelisiert. Es werden bis zu 10 Webservice-Anfragen gleichzeitig gestellt. Der Wert 10 wurde experimentell ermittelt: Bei der schrittweisen Erhöhung des Parallelitätsgrades wurde ein signifikanter Geschwindigkeitszuwachs festgestellt, der jedoch beim gleichzeitigen Abruf von 10 Dokumenten nahezu stagnierte (siehe Tabelle 2 in Abschnitt 4.1). Auch litt über diesem Wert die Stabilität der Webservice-Anfragen (keine Antwort innerhalb des Timeouts). Die Suchergebnisse werden als „Seiten“ zu jeweils maximal 100 Dokumentnummern geliefert, ein Verhalten, das für einen Webservice eigentlich nicht gewünscht bzw. zeitgemäß ist und stark darauf hindeutet, dass der Webservice lediglich ein Wrapper für die RIS-Bundesrecht Webseite ist. Das Suchergebnis der ersten „Seite“ gibt aber an, wie viele Dokumentnummern insgesamt gefunden wurden. Die weiteren Seiten werden alle parallel vom Webservice abgefragt und die Ergebnisse zusammengefasst. Eine abzurufende Liste von Dokumenten, bestehend aus den Dokumentnummern, wird in 10 gleich große Teillisten aufgespalten und 10 Instanzen eines `DocumentCrawler`s laden die Dokumente „ihrer“ Teilliste in je einem eigenen Thread.

3.2.2.2 Caching-Datenbank

Ein Dokument ist, wie eingangs bereits erwähnt, wenn es einmal angelegt wurde, unveränderlich; eine Gesetzesänderung führt neue Dokumente ein. Deshalb ist es möglich, dass Dokumente, die einmal über den Webservice geladen wurden, in einer lokalen[32]

[31] Die genauere Dauer hängt von der Auslastung und der Komplexität der Dokumente ab; beim testweisen Abrufen der Straßenverkehrsordnung ergab sich ein Durchschnittswert von 760 ms.

[32] Es wäre theoretisch auch möglich, die Datenbank vom Server auszulagern, eine lokale Datenbank ist jedoch in diesem Fall ausreichend.

Datenbank abgelegt werden. Wird das Dokument erneut benötigt, kann es direkt aus der Datenbank bezogen werden und muss nicht erneut über den Webservice geladen werden.

Bei einer größeren Anzahl von Dokumenten wurde durch die beiden Maßnahmen eine Beschleunigung um einen Faktor von über 3000 ermittelt (siehe auch 4.1).

Abbildung 5 auf der nächsten Seite zeigt den Weg von der Suche bis zu den Dokumenten als XML-Inhalte. Die einzelnen Schritte sind folgende:

1 Eine Suchanfrage wird von dem `BookCreator` an den `DocumentIDsLoader` gesendet.
2 Dieser ruft parallel die `request()` Methode des Webservice auf.
3 Der Webservice antwortet jeweils mit einer „Seite“ von bis zu 100 Dokumentnummern.
4 Der `DocumentIDsLoader` fasst alle Nummern zu einer Liste zusammen und gibt diese dem `BookCreator` als Antwort auf dessen Suchanfrage zurück.
5 Der `BookCreator` gibt die Liste an den `DocumentManager` weiter, um von ihm die Dokumente zu erhalten.
6 ⎫ Der `DocumentManager` lässt vom `DocumentCache` die Nummern der bekannten
7 ⎭ Dokumente herausfiltern.
8 Der `DocumentManager` weist den `DocumentLoader` an, die unbekannten Dokumente nachzuladen.
9 Dieser ruft, erneut parallel, die `getDocument()` Methode des Webservice auf.
10 Jede Dokumentanfrage wird vom Webservice mit dem Dokument als XML-Inhalt beantwortet.
11 Der `DocumentLoader` speichert die neuen Dokumente im `DocumentCache` ab.
12 Der `DocumentManager` ruft aus dem (nunmehr vollständigen) `DocumentCache` die Dokumente anhand deren Nummern ab.
13 Der `DocumentCache` ruft die gewünschten Dokumente aus der Datenbank ab und gibt sie in Form von `Doc`-Objekten an den `DocumentManager` zurück.
14 Dieser reicht die Liste von `Doc`s an den `BookCreator` weiter.

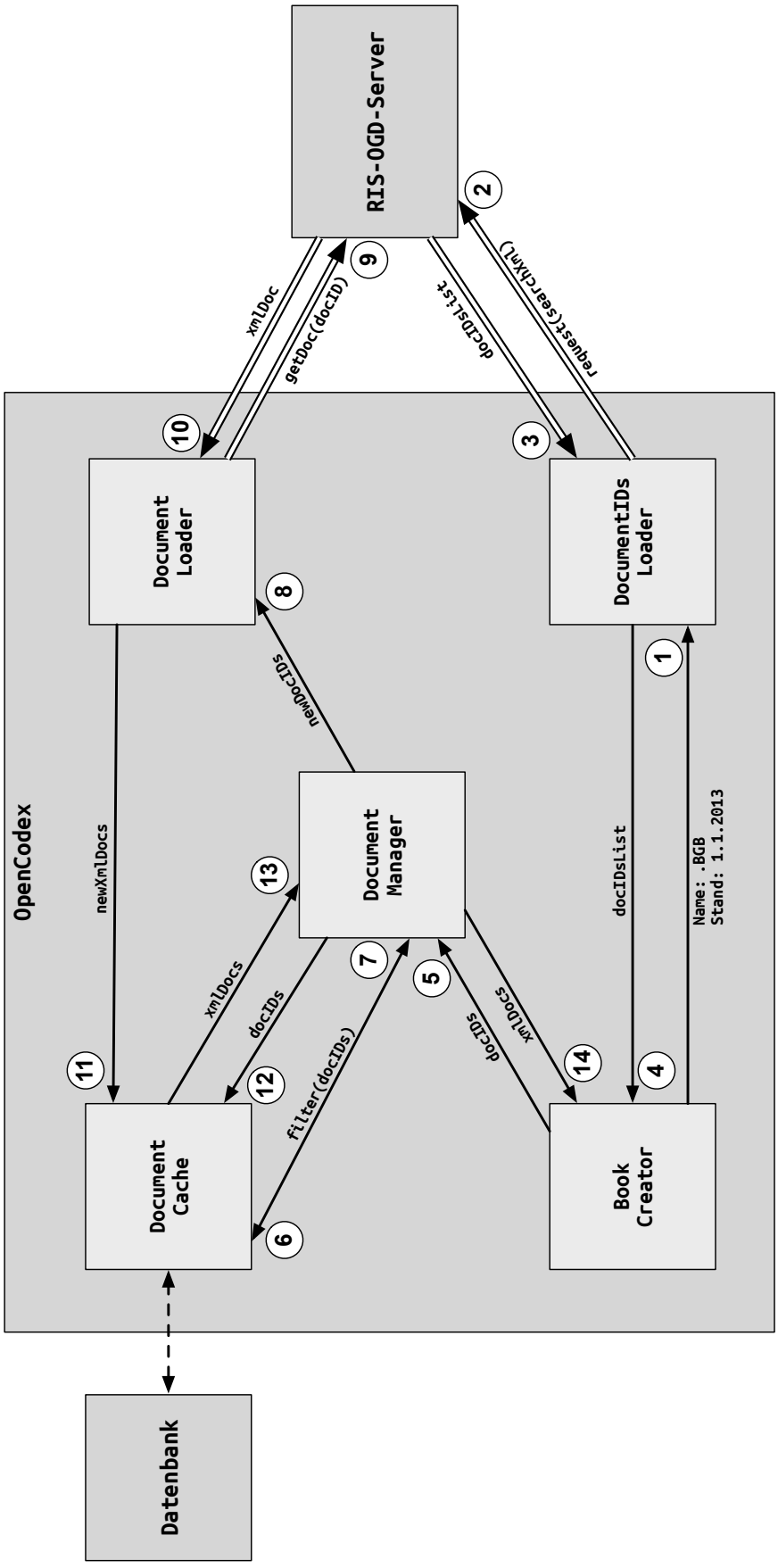


Abbildung 5: Kommunikation mit dem Webservice und Interaktion mit dem Cache im Detail.

3.2.3 Paragraf/Artikel/Anlage

Eine Liste von `Doc`s ist standardmäßig unsortiert. Die Dokumentnummer als Sortierkriterium ist ungeeignet, da, wenn ein Paragraf/Artikel durch eine neuere Version ersetzt wird, auch eine neue Dokumentnummer vergeben wird, die dann nicht mehr in das zuvor fortlaufende Schema passt. Zum Sortieren der Liste wird also das Element `ArtikelParagraphAnlage` des XML-Dokumentes gewählt. Das Feld ist eine im Schema leider nur vage bestimmte („Gliederungsart der Rechtsvorschrift z.B § 5, Art. 10, Anl 8“) Zeichenkette. Deshalb wurde versucht, dessen mögliche Struktur programmatisch zu erfassen. Es wurden durch manuelle Recherche im RIS sieben mögliche Formate[33] identifiziert, die auftreten können:

- `Art. #(?) Anl. #(?)/*(*)`
- `Art. #(?) Anl. #(?)`
- `Art. #(?) § #(?)`
- `Anl. #(?)/*(*)`
- `Anl. #(?)`
- `Art. #(?)`
- `§ #(?)`

Die Klasse `ParArt` erkennt aus einer Zeichenkette, welches dieser Formate vorliegt und speichert die Einzelbestandteile. Die Klasse überschreibt die Methoden `equals(Object o)` und `compareTo(ParArt o)` um zwei Instanzen dieser Klasse zu vergleichen. Damit ist es möglich, zwei `Doc`-Objekte, welche je ein `ParArt`-Objekt halten, mittels dieser zu sortieren. Man kann also eine Liste von Dokumenten über die gewohnte Methode `Collections.sort()` für die Weiterverarbeitung sortieren.

3.2.4 Dokumentenerstellung

Die Erstellung eines PDF-Dokumentes besteht aus dem Aufbereiten der XML-Daten in kompilierbaren LaTeX-Code und dem Umwandeln desselben in ein PDF-Dokument. Letzteres geschieht mit dem Programm *pdflatex* als eigener Prozess ausserhalb des Java-Programms.

Die aufbereiteten Daten für ein Gesetzbuch werden in einer Instanz der Klasse `Book` gehalten. Der Aufbau und der geschachtelt abgelegte Inhalt dieser werden in der Abbildung 6 ersichtlich.

[33] Es gelte dabei folgende Notation: `#` ist eine beliebige ganze Zahl $\geqslant 0$, `?` ist ein beliebiger Buchstabe, `*` ist eine Ziffer oder ein Buchstabe. Ein Ausdruck in `()` ist optional.

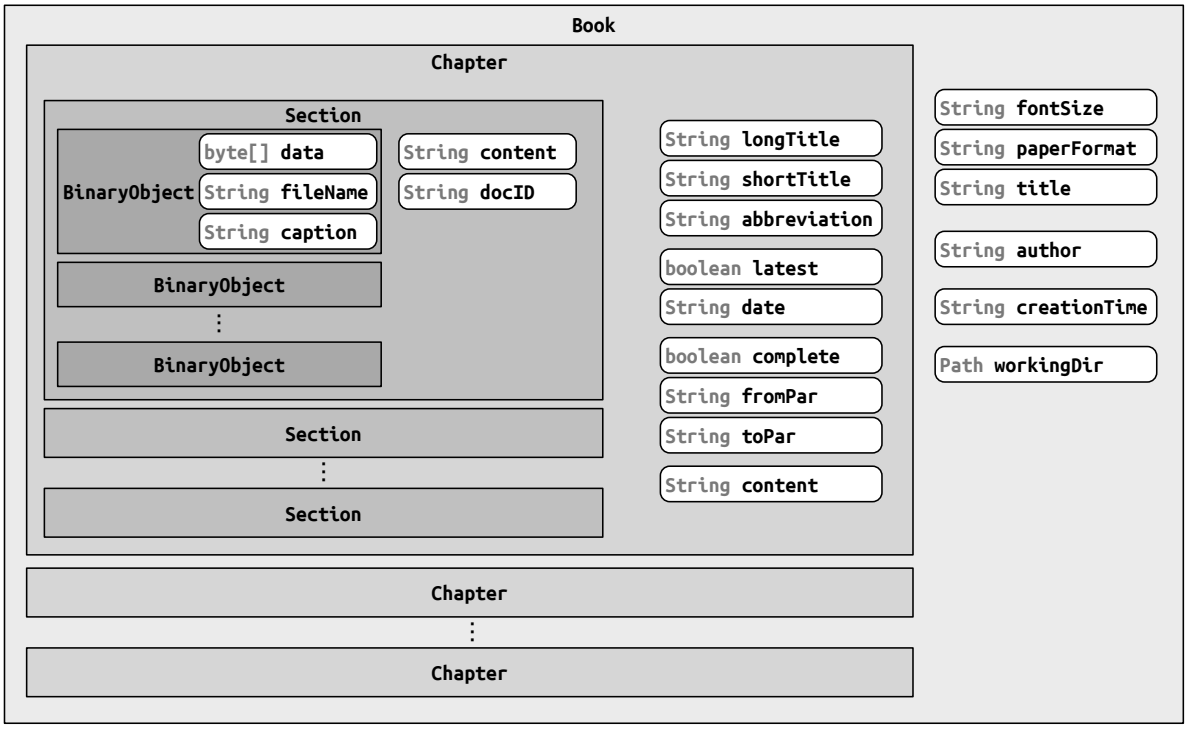


Abbildung 6: Aufbau und Inhalt der Container-Klasse `Book`.

Ein `Chapter` repräsentiert ein Gesetz in dem Gesetzbuch, eine `Section` ist der Inhalt eines Einzeldokumentes, also ein Paragraf, ein Artikel oder eine Anlage und wird aus einem `Doc`-Objekt generiert. Hierzu wird in der Klasse `DocumentContentParser` die Methode `public Section process(Doc doc)` benutzt. Diese startet den Parsingvorgang, der aus dem XML-Dokument LaTeX-Code erzeugt.

Bevor der XML-Code geparst wird, müssen (X)HTML-Formatierungselemente (z. B. `<br/>`, `<b>...</b>`, etc.), welche in das XML eingebettet sind, durch die entsprechenden LaTeX-Formatierungsbefehle ersetzt werden. Auch benannte Zeichenentitäten für Umlaute und Sonderzeichen (z. B. `&szlig;` oder `&AElig;`) müssen ersetzt werden, Details siehe 3.2.4.1.

Die Parser-Methoden nutzen den XML-Parser der HTML-Parser-Bibliothek *jsoup*[34], der aus den XML-Elementen `Element`-Objekte erzeugt. Je nachdem, um welche XML-Tags es sich bei den Elementen handelt, wird die jeweilige Methode zum Parsen des zu erwartenden Element-Inhalts aufgerufen. Die Möglichkeit von Java 7, ein `switch()`-Statement auf `String`-Objekte anzuwenden erleichtert die Lesbarkeit des Codes.

[34] Siehe `http://jsoup.org`, (22.07.2013)

In vom RIS gelieferte XML-Dokumente können Binärdateien als Base64-kodierte[35] Zeichenketten eingebunden sein. Der den eigentlichen Base64-Inhalt umgebende Tag spezifiziert dabei das jeweilige Dateiformat. Die Dokumentation im Nutzdaten-Stylesheet ist bei den möglichen Formaten nicht eindeutig („Gibt den Binärtyp an: jpeg | tiff | pdf | . . . “), deshalb wurde bei der zuständigen Stelle nach einer taxativen Liste der möglichen Formate angefragt. Tabelle 1 zeigt die nach Anhang C.1.2 möglichen Formate und deren Behandlung in OpenCodex.

Format	**Behandlung in OpenCodex**
XML	Wird direkt als formatiertes Code-Listing mittels des LaTeX-Paketes *listings* eingebunden.
RTF	*Nicht unterstützt.* (siehe 4.2.3)
DOC	*Nicht unterstützt.* (siehe 4.2.3)
DOCX	*Nicht unterstützt.* (siehe 4.2.3)
ODT	*Nicht unterstützt.* (siehe 4.2.3)
PDF	Wird direkt eingebunden mittels `\includegraphics{}`.
GIF	Wird mit Hilfe von Java ImageIO in das PNG-Format umgewandelt und dann mit `\includegraphics{}` eingebunden.
JPG	Wird direkt eingebunden mittels `\includegraphics{}`.
TIF(F)	Wird mit Hilfe von Java ImageIO in das JPG-Format umgewandelt und dann mit `\includegraphics{}` eingebunden.

Tabelle 1: Mögliche Binärdateien und deren Behandlung in OpenCodex.

3.2.4.1 HTML-Formatierungen und XHTML-Entitäten

Vor dem eigentlichen Verarbeiten müssen sowohl formatierende HTML-Tags als auch Zeichenentitäten ersetzt werden. Dies geschieht durch einfaches Suchen und Ersetzen in der Zeichenkette, die das (ungeparste) XML-Dokument darstellt. Für das Ersetzen von öffnenden, schließenden und leeren Tags wurde je ein `Map<String,String>`-Objekt erzeugt und manuell mit den entsprechenden Ersetzungen gefüllt. So wird etwa ein „`<i>`“-Tag, also ein öffnender Tag für kursiv geschriebenen Text, mit dem LaTeX-Äquivalent „`\emph{`“ ersetzt; der schließende Tag „`</i>`“ wird durch „`}`“ ersetzt.

XHTML verwendet laut [Pem02, Anhang A.2] die Zeichenentitäten aus HTML 4, definiert in [JRH99, Kapitel 24], und laut [Pem02, Anhang C.16] zusätzlich die Entität `&apos;`. Die definierten Entitäten können sowohl benannt vorkommen, als auch über ihre

[35] *Base64*, aktuell definiert in [RFC4648, Kapitel 4], ist ein Verfahren, mit dem ein beliebiger Binärinhalt in 64 gewöhnliche ASCII-Zeichen (`a-zA-Z0-9/+`) umgewandelt werden kann. Hierzu werden nach einer Tabelle je drei Oktette in vier Zeichen gewandelt. Der umgekehrte Weg ist genauso möglich.

Unicode-Nummer. Es wurde deshalb eine Erweiterung zu einer gewöhnlichen `HashMap <String,String>` geschrieben, welche den benannten Entitätsnamen, die numerische Unicode-Darstellung und das Unicode-Zeichen selbst auf die jeweilige LaTeX-Darstellung des Zeichens abbildet. Die Entsprechungsliste XHTML → LaTeX wurde händisch erstellt. Diese Datenstruktur wird statisch intitialisiert. Um den Such-/Ersetzvorgang zu beschleunigen wird dafür die Methode `public static String replaceEach(String text, String[] searchList, String[] replacementList)` aus der Bibliothek *Apache Commons Lang 3*[36] in der Klasse `org.apache.commons.lang3.StringUtils` verwendet. Dafür werden aus der Map XHTML → LaTeX die Keys und Entries jeweils in ein eigenes Array gespeichert.

Zu beachten ist beim Erstellen von mehreren Replacement-Maps, dass sowohl Java als auch LaTeX und XML (bzw. der XML-Parser) für verschiedene Sonderzeichen eine besondere Behandlung benötigen, bzw. manche Zeichen geschützt (engl. *escaped*) werden müssen. So muss etwa ein „\“ in Java als „\\“ geschrieben werden. Wendet man verschiedene Maps nacheinander an, muss auch beachtet werden, dass vorhergegangene Ersetzungen nicht wieder zerstört werden, indem ein (Steuer-)Zeichen erneut ersetzt, oder durch den XML-Parser von *jsoup* falsch interpretiert wird. Bei der Ersetzung der formatierenden HTML-Tags wird etwa das Zeichen „\“ nicht direkt ausgegeben, da es bei der darauffolgenden Zerlegung des XML-Dokuments durch *jsoup* selbst erneut ersetzt werden würde. Es muss also zunächst durch das HTML-Äquivalent `&#92;` ersetzt werden, um später von *jsoup* in das korrekte Zeichen „\“ umgewandelt zu werden.

3.2.4.2 Schlagwortverzeichnis

Die vom Webservice gelieferten XML-Dokumente enthalten ein Element *Schlagworte*, in dem eine Beistrich-getrennte Liste von Schlagworten steht, welche auf den jeweiligen Paragrafen/Artikel passen. Aus diesen Schlagworten wird pro Gesetz ein Schlagwortverzeichnis generiert. LaTeX beherrscht von sich aus diese Funktion, allerdings nur ein Verzeichnis, genannt Index, pro Dokument. Mit Hilfe des Paketes *imakeidx* (siehe 2.3.3.6) kann man pro Gesetz, also pro `Chapter`, in der Präambel einen benannten Index definieren. Der Indexname ist im `Chapter`-Objekt hinterlegt und besteht aus einem Zeitstempel. In der Verarbeitung jedes Einzeldokuments wird nach der Überschrift je Schlagwort ein Indexeintrag definiert. Am Ende eines Kapitels wird per `\printindex [<IndexName>]` das Schlagwortverzeichnis im Dokument ausgegeben. Da LaTeX intern ja nur *einen* Index beherrscht, muss dieser beim Kompilieren mit dem Tool *splitindex* in die benannten Indizes aufgeteilt werden; Details siehe 3.2.4.4.

[36] Siehe `http://commons.apache.org/proper/commons-lang`, (22.07.2013)

3.2.4.3 Vergleichserstellung

Zum Vergleich zweier Fassungen eines Gesetzes wird zunächst für die alte und neue Fassung die Liste mit Dokumenten (D_{old} und D_{new}) geladen. Anschließend wird ermittelt, was sich in dem gegebenen Zeitraum in dem Gesetz geändert hat. Pro Gliederungseinheit gibt es vier Möglichkeiten: *gelöscht*, *hinzugefügt*, *verändert* oder *unverändert*. Mit Hilfe einfacher Mengenoperationen kann der jeweilige Fall unter Berücksichtigung der Dokumentnummer num_d und der Paragraf/Artikel/Anlage-Angabe paa_d eines jeden Einzeldokuments ermittelt werden. D bezeichnet hierbei eine Menge von Dokumenten, d ein einzelnes Dokument. Die Gleichheit zweier Dokumente bei den Mengenoperationen beruht auf der Gleichheit der Paragraf/Artikel/Anlage-Angabe. Es gilt folgendes:

$$
\begin{aligned}
D_{deleted} &= D_{old} \backslash D_{new} \\
D_{added} &= D_{new} \backslash D_{old} \\
D_{changed_{pot}} &= D_{old} \cap D_{new} \\
D_{changed_{real}} &= D_{changed_{pot}}(d \mid (paa_{d_{old}} = paa_{d_{new}}) \wedge (num_{d_{old}} \neq num_{d_{new}})) \\
D_{unchanged} &= D_{changed_{pot}} \backslash D_{changed_{real}}
\end{aligned}
$$

Entfernte, neu hinzugefügte und unveränderte Paragrafen werden im Ausgabedokument je nach Einstellung gar nicht oder direkt dargestellt. Bei einem geänderten Paragrafen gibt es zwei Möglichkeiten der Darstellung: Entweder im zweispaltigen Satz, links die alte Version, rechts daneben die neue (siehe Abbildung 7) oder als *ein* Fließtext, in dem die Änderungen direkt markiert sind (siehe Abbildung 8).

Abhängig davon, wie umfangreich die Änderungen in einem Paragrafen sind, gibt die eine oder die andere Darstellungsvariante mehr Sinn. Kleine Änderungen sind schneller zu erkennen, wenn sie direkt im Text markiert sind. Bei großen Änderungen kann dies jedoch rasch unübersichtlich wirken, sodass die zweispaltige Variante vorzuziehen ist. OpenCodex bietet die Möglichkeit, automatisch, je nach Umfang der Änderungen, die jeweils bessere Darstellungsform zu verwenden.

Stand 07.06.2010

Beachte: Abs. 1: Verfassungsbestimmung

Organisation und Geschäftsführung der Datenschutzkommission

§ 38.[I] (1) **(Verfassungsbestimmung)** Die Datenschutzkommission hat sich eine Geschäftsordnung zu geben, in der eines ihrer Mitglieder mit der Führung der laufenden Geschäfte zu betrauen ist (geschäftsführendes Mitglied). Diese Betrauung umfaßt auch die Erlassung von verfahrensrechtlichen Bescheiden und von Mandatsbescheiden im Registrierungsverfahren gemäß § 20 Abs. 2 oder § 22 Abs. 3. Inwieweit einzelne fachlich geeignete Bedienstete der Geschäftsstelle der Datenschutzkommission zum Handeln für die Datenschutzkommission oder das geschäftsführende Mitglied ermächtigt werden, bestimmt die Geschäftsordnung.

(2) Für die Unterstützung in der Geschäftsführung der Datenschutzkommission hat der Bundeskanzler eine Geschäftsstelle einzurichten und die notwendige Sach- und Personalausstattung bereitzustellen. Er hat das Recht, sich jederzeit über alle Gegenstände der Geschäftsführung der Datenschutzkommission beim Vorsitzenden und dem geschäftsführenden Mitglied zu unterrichten.

(3) Die Datenschutzkommission ist vor Erlassung von Verordnungen anzuhören, die auf der Grundlage dieses Bundesgesetzes ergehen oder sonst wesentliche Fragen des Datenschutzes unmittelbar betreffen.

(4) Die Datenschutzkommission hat spätestens alle zwei Jahre einen Bericht über ihre Tätigkeit zu erstellen und in geeigneter Weise zu veröffentlichen. Der Bericht ist dem Bundeskanzler zur Kenntnis zu übermitteln.

[I] Abs. 1: Verfassungsbestimmung

Stand 07.06.2013

Beachte: Abs. 1: Verfassungsbestimmung

Organisation und Geschäftsführung der Datenschutzkommission

§ 38.[I] (1) **(Verfassungsbestimmung)** Die Datenschutzkommission hat sich eine Geschäftsordnung zu geben, in der eines ihrer Mitglieder mit der Führung der laufenden Geschäfte zu betrauen ist (geschäftsführendes Mitglied). Diese Betrauung umfaßt auch die Erlassung von verfahrensrechtlichen Bescheiden und von Mandatsbescheiden im Registrierungsverfahren gemäß § 20 Abs. 2 oder § 22 Abs. 3. Inwieweit einzelne fachlich geeignete Bedienstete der Geschäftsstelle der Datenschutzkommission zum Handeln für die Datenschutzkommission oder das geschäftsführende Mitglied ermächtigt werden, bestimmt die Geschäftsordnung.

(2) Der Bundeskanzler kann sich beim Vorsitzenden der Datenschutzkommission über die Gegenstände der Geschäftsführung unterrichten. Dem ist vom Vorsitzenden der Datenschutzkommission nur insoweit zu entsprechen, als dies nicht der völligen Unabhängigkeit der Kontrollstelle im Sinne von Art. 28 Abs. 1 UAbs. 2 der Richtlinie 95/46/EG zum Schutz natürlicher Personen bei der Verarbeitung personenbezogener Daten und zum freien Datenverkehr, ABl. Nr. L 281 vom 23.11.1995 S. 31, widerspricht.

(3) Die Datenschutzkommission ist vor Erlassung von Verordnungen anzuhören, die auf der Grundlage dieses Bundesgesetzes ergehen oder sonst wesentliche Fragen des Datenschutzes unmittelbar betreffen.

(4) Die Datenschutzkommission hat spätestens alle zwei Jahre einen Bericht über ihre Tätigkeit zu erstellen und in geeigneter Weise zu veröffentlichen. Der Bericht ist dem Bundeskanzler zur Kenntnis zu übermitteln.

[I] Abs. 1: Verfassungsbestimmung

Abbildung 7: Beispiel Vergleichsdokument im zweispaltigen Satz (Vergleich von § 38 DSG vom 07.06.2010 und vom 07.06.2013).

Organisation und Geschäftsführung der Datenschutzkommission

§ 38.[I] (1) **(Verfassungsbestimmung)** Die Datenschutzkommission hat sich eine Geschäftsordnung zu geben, in der eines ihrer Mitglieder mit der Führung der laufenden Geschäfte zu betrauen ist (geschäftsführendes Mitglied). Diese Betrauung umfaßt auch die Erlassung von verfahrensrechtlichen Bescheiden und von Mandatsbescheiden im Registrierungsverfahren gemäß § 20 Abs. 2 oder § 22 Abs. 3. Inwieweit einzelne fachlich geeignete Bedienstete der Geschäftsstelle der Datenschutzkommission zum Handeln für die Datenschutzkommission oder das geschäftsführende Mitglied ermächtigt werden, bestimmt die Geschäftsordnung.

(2) ~~Für die Unterstützung in~~ Der Bundeskanzler kann sich beim Vorsitzenden der Datenschutzkommission über die Gegenstände der Geschäftsführung ~~der Datenschutzkommission hat der Bundeskanzler eine Geschäftsstelle einzurichten und die notwendige Sach- und Personalausstattung bereitzustellen.Er hat das Recht, sich jederzeit über alle Gegenstände der Geschäftsführung der Datenschutzkommission beim Vorsitzenden und dem geschäftsführenden Mitglied zu unterrichten~~unterrichten. Dem ist vom Vorsitzenden der Datenschutzkommission nur insoweit zu entsprechen, als dies nicht der völligen Unabhängigkeit der Kontrollstelle im Sinne von Art. 28 Abs. 1 UAbs. 2 der Richtlinie 95/46/EG zum Schutz natürlicher Personen bei der Verarbeitung personenbezogener Daten und zum freien Datenverkehr, ABl. Nr. L 281 vom 23.11.1995 S. 31, widerspricht.

(3) Die Datenschutzkommission ist vor Erlassung von Verordnungen anzuhören, die auf der Grundlage dieses Bundesgesetzes ergehen oder sonst wesentliche Fragen des Datenschutzes unmittelbar betreffen.

(4) Die Datenschutzkommission hat spätestens alle zwei Jahre einen Bericht über ihre Tätigkeit zu erstellen und in geeigneter Weise zu veröffentlichen. Der Bericht ist dem Bundeskanzler zur Kenntnis zu übermitteln.

Abbildung 8: Beispiel Vergleichsdokument im Fließtext (Vergleich von § 38 DSG vom 07.06.2010 und vom 07.06.2013).

Es wird ein relatives Maß der Differenz $Diff_{rel}$ zwischen dem alten Text t_{old} mit der Länge l_{old} und dem neuen Text t_{new} mit der Länge l_{new} definiert. Dieses errechnet sich folgendermaßen:

$$\begin{aligned} l_{short} &= min(l_{old}, l_{new}) \\ l_{long} &= max(l_{old}, l_{new}) \\ Diff_{rel} &= \frac{LevenshteinDistance(t_{old}, t_{new}) - (l_{long} - l_{short})}{l_{short}} \end{aligned}$$

Die Levenshtein-Distanz ist eine von Wladimir Iossifowitsch Lewenstein[37] in [Lev66] definierte Maßzahl der Differenz zweier Zeichenketten. Die Distanz der beiden Zeichenketten erhöht sich für jedes *Einfügen*, *Löschen* und *Ersetzen* eines Buchstaben jeweils um eins. In OpenCodex kommt die Implementierung `StringUtils.getLevenshteinDistance(oldText, newText)` aus der Bibliothek *Apache Commons Lang 3* zum Einsatz.

Die Levenshtein-Distanz zweier unterschiedlich langer Texte ist mindestens so groß wie der Längenunterschied der beiden Texte. Diesen Längenunterschied kann man zum Zwecke der Normierung von der ermittelten Distanz abziehen. Der so gewonnene Wert wird nun noch durch die geringere Länge geteilt, um einen relativen (und somit vergleichbaren) Wert zwischen 0 und 1 zu erhalten. Bei der Division ist zu beachten, dass die kürzere Zeichenkette nicht leer sein darf (z. B. Paragraf aufgehoben), da sonst eine Division durch Null auftreten würde.

Da in OpenCodex nicht nur der einfache Klartext verglichen wird, sondern dieser bereits mit LaTeX-Markup versehen ist, entspricht die errechnete Distanz nicht der eigentlichen Textdistanz. Es wäre mit großem Aufwand verbunden, aus dem zu vergleichenden Code das LaTeX-Markup und insbesondere die bereits umschriebenen Sonderzeichen (siehe 3.2.4.1) für den Textvergleich erneut herauszufiltern, um nur den reinen Text vergleichen zu können. Es wurde durch Ausprobieren der Schwellwert der Textdifferenz von 0,15 ermittelt, ab dem die Ausgabe zweispaltig erfolgt.

Der zweispaltige Satz erfolgt, wie in Abschnitt 2.3.3.7 beschrieben, mittels der `paracol`-Umgebung. Das Erzeugen von Änderungsmarkierungen direkt im Text geschieht mit Hilfe des Programms *latexdiff*[38]. Es generiert aus zwei .tex Dateien eine neue, in der

[37] Der russische Name „Владимир Иоссифович Левенштейн“ wird je nachdem ob man Transkription (aussprachebasiert) oder Transliteration (schriftbasiert) verwendet, im Deutschen unterschiedlich geschrieben. Für die Distanz bzw. den Algorithmus ist aber die im Englischen übliche Schreibweise „Levenshtein“ gebräuchlich.

[38] Siehe `http://www.ctan.org/tex-archive/support/latexdiff`, (22.07.2013)

die Änderungen hervorgehoben sind; die Art der Hervorhebung kann über die LaTeX-Präambel geändert werden. Das Programm wird wie der externe LaTeX-Compiler aufgerufen, wie im nächsten Abschnitt beschrieben. Der von *latexdiff* generierte Code wird in das Ausgabedokument des Fassungsvergleichs integriert.

3.2.4.4 Externer Kompiliervorgang

Sowohl der Buch- als auch der Vergleichserstellungsprozess erzeugt eine .tex Datei. Diese kann nicht von OpenCodex selbst verarbeitet werden, sondern wird mit Hilfe des Programms *pdflatex* in ein PDF-Dokument umgewandelt. Um Tabellenformatierungen und eventuelle interne Referenzen im Ausgabedokument richtig zu setzen, sind drei Programmläufe von *pdflatex* nötig. Nach dem ersten Programmlauf werden durch den Aufruf des Programms *splitindex* die Schlagwortverzeichnisse kapitelweise aufgeteilt und aus Rohdaten in setzbare Indexdaten umgewandelt. Der Dokumenten-Kompiliervorgang in OpenCodex umfasst also folgende Programmaufrufe:

pdflatex → *splitindex* → *pdflatex* → *pdflatex*

Der Aufruf eines externen Programms erfolgt mit Hilfe der Bibliothek *JProc*[39]. Listing 7 zeigt deren Verwendung: Im Konstruktor wird der Pfad zur auszuführenden Binärdatei benötigt; anschließend können Parameter übergeben, das Arbeitsverzeichnis definiert und eine maximale Laufzeit des externen Programms angegeben werden. Mit der Methode `run()` wird das externe Programm gestartet.

```
1 ProcBuilder proc = new ProcBuilder(pathToBinary);
2 proc.withArg(pathToTexFile);
3 proc.withWorkingDirectory(workingDir);
4 proc.withTimeoutMillis(timeOut);
5 ProcResult result = proc.run();
6 if (result.getExitValue() == 0) { /* success */ }
```

Listing 7: Aufruf des externen Programms *pdflatex*.

3.2.5 Index des Bundesrechts

Der Index des Bundesrechts ist unter der URL `http://www.ris.bka.gv.at/UI/Bund/Bundesnormen/IndexBundesrecht.aspx` zu finden. Er ist in Sachgebiete, Hauptgruppen und Untergruppen in folgender Form gegliedert:

[39] Siehe `http://code.google.com/p/jproc`, (22.07.2013)

1 Sachgebiet
10 Hauptgruppe
10/10 Untergruppe

Der Index liegt nicht als gut maschinenlesbare XML-Datei, sondern nur als HTML-Webseite vor. Diese wird im `IndexLoader` mit Hilfe von *jsoup* eingelesen. *jsoup* erlaubt es, HTML-Tags mit der Cascading Style Sheet (CSS) Selektor-Syntax[40] auszuwählen. Der gelesene Index wird in den Datencontainern `Field`, `MainGroup` und `SubGroup` gespeichert und hierarchisch geschachtelt/verknüpft. Der Index wird programmatisch über den statischen Aufruf `IndexLoader.loadIndex()` geladen. Die Methode liefert ein sortiertes Set von `Field`-Objekten.

3.2.6 Utilities und Konfiguration

Die Klasse `Util` enthält statische Methoden für verschiedene Zwecke wie dem Laden oder Speichern von Dateiinhalten, dem Konvertieren von Binärdaten oder dem Schnellzugriff auf Datums- und Zeitstempel-Strings. Die Klasse `Conf` gewährt über statische Methoden Zugriff auf sämtliche konfigurierbare und/oder programmweit benötigte Zeichenketten wie spezielle Ordnernamen, die Pfade zu den externen Programmen, Informationen für die Datenbankverbindung oder die aktuelle Versionsnummer des Programms. Die Klasse ermöglicht es, gewisse Variablen davon aus einer externen Konfigurationsquelle zu beziehen. Dazu muss eine Klasse das Interface `ExternalConf` implementieren und sich selbst über den Aufruf `Conf.setExternalConf(this);` bei der Konfigurationsklasse einhängen. Dies birgt zwar eine potentielle Sicherheitslücke, ist jedoch nötig, um das Back-End unabhängig vom Front-End betreiben zu können[41].

3.3 Front-End

Das Front-End gliedert sich in einen Teil mit JSF-Seiten, welche dem Benutzer vom Server zur Verfügung gestellt werden und dann im Browser laufen, und einen Teil mit Java-Beans, welche die JSF-Seiten auf der Server-Seite stützen.

Der JSF-Teil des Front-Ends ist nicht hierarchisch gegliedert, der Java-Teil des Front-Ends untergliedert sich in folgende Pakete:

[40] Details hierzu sind in der CSS-Spezifikation [Bos+11] in Kapitel 5: *Selectors* zu finden.
[41] Es könnte theoretisch mit dem unveränderten Back-End eine andere, z. B. lokale Benutzeroberfläche entwickelt werden, siehe auch 4.2.4.

beans Java EE Beans, welche mit der Benutzeroberfläche direkt interagieren.

data Datencontainer, in denen die von der Benutzeroberfläche gesammelten bzw. vom Back-End gelieferten Daten gehalten werden.

exceptions Klassen zur Behandlung eines auftretenden Session-Timeouts.

3.3.1 JSF-Seiten

Die Benutzeroberfläche selbst besteht aus einer Hauptseite `index.xhtml`; darin eingebettet werden die Hilfe-Seite `help.xhtml` und das Impressum `imprint.xhtml`. Die Hauptseite wird mit der PrimeFaces-Komponente `<p:layout fullPage="true">` seitenfüllend gestaltet. Die Komponente erzeugt ein BorderLayout mit fünf Panels:

- Das *Nord*-Panel besteht aus Logo, Anwendungstitel und Programmversion.
- Das *West*-Panel dient bei der Sammlungserstellung als Hauptmenü. Es kann die aktuelle Sammlung verworfen werden, Einstellungen betreffend der Formatierung getroffen werden, die Sammlung lokal auf dem Computer des Anwenders gespeichert bzw. davon geladen werden und die Dokumentenerstellung gestartet werden.
- Das *Ost*-Panel bleibt ungenutzt.
- Das *Süd*-Panel enthält eine Tabelle mit den Einträgen (=Gesetzen) der aktuellen Sammlung. Die Einträge können darin umsortiert, bearbeitet und gelöscht werden.
- Das *Zentrum*-Panel enthält fünf Registerkarten:
  - *Suche Bundesrecht* Die Default-Registerkarte. Hier kann der Benutzer zur aktuellen Sammlung – über die Suche nach Namen oder nach dem Index des Bundesrechts – Gesetze hinzufügen. Abbildung 9 zeigt die Benutzeroberfläche mit dem Suchfenster und drei Gesetzen in der aktuellen Sammlung.
  - *Sammlungsarchiv* Hier erhält der Benutzer Zugriff auf das Sammlungsarchiv, kann die darin befindlichen Sammlungen einsehen und bewerten, und eine Sammlung laden.
  - *Fassungsvergleich* In dieser Registerkarte kann der Benutzer das zu vergleichende Gesetz auswählen, Einstellungen zum Vergleich treffen und das gewünschte Vergleichsdokument generieren lassen. Abbildung 10 zeigt den Fassungsvergleich mit eingestelltem Gesetz.
  - *Hilfe* Zeigt den Hilfe-Text an, der auch in Anhang A.1 zu finden ist.
  - *Impressum* Zeigt das Impressum an.

  Das Panel ist mit Hilfe von Registerkarten ausgeführt und nicht auf verschiedene Unterseiten aufgeteilt, da somit beim Aufruf der Seite bereits deren gesamter Inhalt geladen wird und bei einer Benutzerinteraktion nichts nachgeladen werden

muss. Die Bedienung wirkt für den Benutzer dadurch flüssiger und einem „nativen“ Programm ähnlicher.

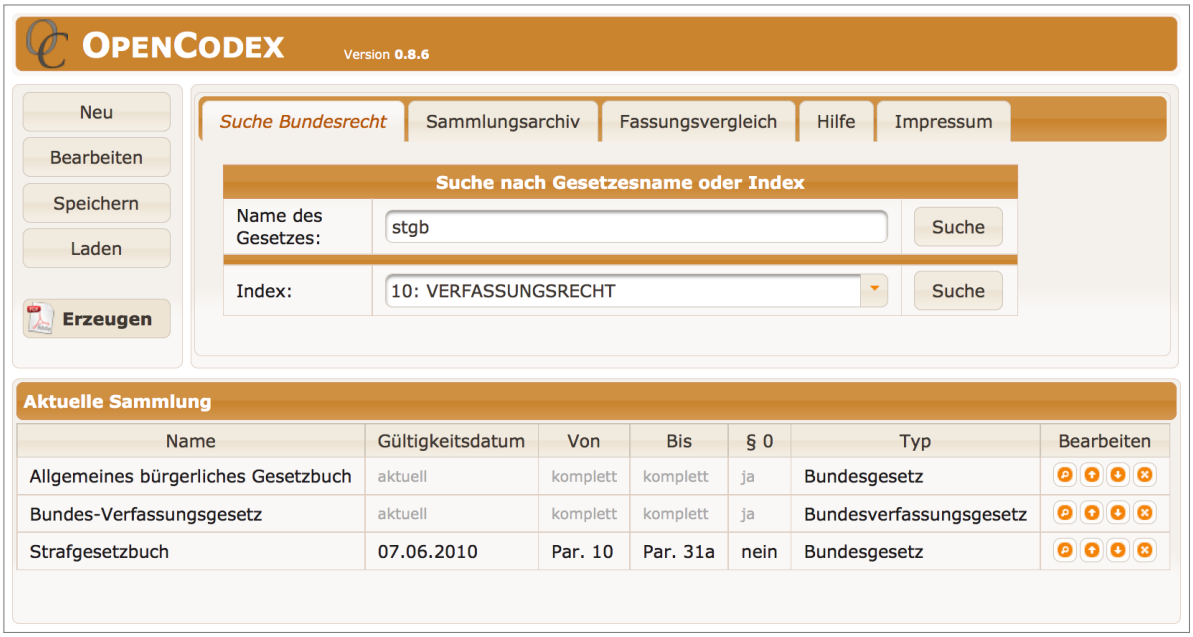


Abbildung 9: Screenshot der Web-Oberfläche zur Sammlungserstellung.

Abbildung 10: Screenshot der Web-Oberfläche zum Fassungsvergleich.

Ist eine der drei Registerkarten *Fassungsvergleich*, *Hilfe* oder *Impressum* aktiv, werden die West- und Süd-Panels ausgeblendet, da sie nicht benötigt werden. Diese Funktio-

nalität wird von PrimeFaces selbst nicht zur Verfügung gestellt, deshalb wurde sie in JavaScript implementiert. Listing 8 zeigt den entsprechenden Codeausschnitt.

```
1 function handleTabShow(index) {
2   if (index[0].id === 'centerForm:centerTab:centerTabCompare' || index[0].id
       === 'centerForm:centerTab:centerTabHelp' || index[0].id === 'centerForm
     :centerTab:centerTabImprint') {
3     wholePage.layout.hide('south');
4     wholePage.layout.hide('west');
5   } else {
6     wholePage.layout.show('west');
7     wholePage.layout.show('south');
8   }
9 }
```

Listing 8: JavaScript-Code zum Anzeigen/Verstecken der West- und Süd-Panels.

Wie in 2.4.1 bereits erwähnt, wird eine Benutzeroberfläche in JSF aus Komponenten zusammengebaut. Die folgende Aufstellung beschreibt die in OpenCodex meistverwendeten Komponenten[42]:

p:commandButton ist eine gewöhnliche Schaltfläche, die ein Ereignis auslöst. Die Komponente unterstützt AJAX, d. h. es kann ausgewählt werden, welcher Teil der Seite an den Server geschickt (Attribut `process`) und welcher Teil der Seite vom Server aktualisiert (Attribut `update`) werden soll. Die Auswahl geschieht durch Selektoren, wobei PrimeFaces sowohl direkt über die Element-ID auf Seitenelemente referenzieren kann, als auch die *jQuery Selector API*[43] verwendet, welche Seitenelemente aufgrund von Eigenschaften auswählt. Die Komponente wird an eine Methode im Managed Bean oder an einen JavaScript-Aufruf (z. B. Anzeigen eines Dialoges) gebunden.

p:dataTable stellt Daten tabellarisch dar. Diese Komponente wird an eine Liste (oder ein Array) im Bean gebunden. Wird das Attribut `var` gesetzt, kann man in den Tabellenspalten auf das iterierende Objekt der Liste oder des Arrays zugreifen.

p:column repräsentiert eine Spalte in einer Tabelle. Über die in der Tabelle benannte Variable kann man auf Felder der einzelnen Instanzen des zu iterierenden Objektes zugreifen.

[42] „`p:`“ zeigt, dass es sich um eine Komponente aus dem PrimeFaces-Namensraum handelt, „`f:`“ sind JSF-eigene Komponenten aus dem Namensraum JSF-Core.
[43] Siehe `http://api.jquery.com/category/selectors`, (22.07.2013)

p:tabView definiert eine Registerkartenleiste. Mit dem Attribut `onTabShow` kann eine JavaScript-Funktion definiert werden, die beim Registerwechsel aufgerufen wird. Die Methode in Listing 8 wird darüber eingebunden.

p:tab ist eine Registerkarte. Alle Komponenten, die sich innerhalb dieses Elements befinden, werden auf der Registerkarte angezeigt.

p:selectOneMenu definiert ein DropDown-Auswahlmenü. Die Menüpunkte können einzeln über `f:selectItem`-Komponenten hinzugefügt werden. Alternativ kann über *eine* `f:selectItems`-Komponente auf eine Liste von Objekten referenziert werden; dabei muss dann die Bezeichnung und der Wert eines jeden Eintrags, ähnlich wie bei einer Tabelle über eine benannte Variable, auf eine Variable im Managed Bean referenzieren. Der ausgewählte Eintrag wird im Bean auf ein String-Objekt referenziert.

p:calendar erlaubt die Auswahl eines Datums. Die Komponente erscheint zunächst wie ein normales Texteingabefeld. Klickt man jedoch in die Komponente, erscheint ein Kalender, aus dem das gewünschte Datum ausgewählt werden kann. Die Namen für Wochentage und Monate werden normalerweise bei dem Kalender in englisch angezeigt. Mit dem Setzen des Attributs `lang="de_AT"` kann die Sprache geändert und mit dem Attribut `locale="de_AT"` kann das Verhalten (erster Tag der Woche, Anzeige von AM/PM bei Uhrzeiten) auf Österreich angepasst werden. Dazu muss jedoch in den Ressourcen von PrimeFaces eine Datei mit Namen `primefaces_locale_de_AT.js` enthalten sein, welche die dafür nötigen Einstellungen enthält. Eine Kalender-Komponente referenziert auf ein Date-Objekt im Managed Bean.

p:dialog erstellt ein Dialogfenster. Dialoge werden häufig im modalen Modus (Attribut `modal="true"`) betrieben, d. h. das Dialogfenster ist ganz im Vordergrund und der Rest der Webseite bleibt gesperrt bis das Fenster geschlossen wird. Über das Attribut `widgetVar` kann dem Dialog ein seitenweit bekannter Name gegeben werden. Man kann somit an einer beliebigen Stelle im Code der Seite (meist als Aktion einer Schaltfläche) mit dem JavaScript-Aufruf `dialogName.show();` das Dialogfeld öffnen.

p:ajax fügt zu einer nicht-AJAX-Komponente AJAX-Funktionalitäten hinzu. Beispiele hierfür sind der Kalender oder das Auswahlmenü. Mit dem Element `<p:ajax event="dateSelect" process="@this"/>` innerhalb eines `p:calendar`-Elements wird etwa eingestellt, dass bei der Datumsauswahl dieses sofort (per AJAX) an den Server übertragen werden soll.

Zusätzlich zur Hauptseite existieren vier weitere JSF-Seiten, die allerdings ob ihres administrativen Charakters nicht von der Hauptseite aus verlinkt sind. Die Seite `archiveadmin.xhtml` erlaubt das Einsehen, Freischalten und Löschen der Sammlungen im Sammlungsarchiv. Die Seite `settings.xhtml` bietet Zugriff auf die konfigurierbaren Einstellungen von OpenCodex sowie Wartungsmaßnahmen um alte erstellte Dokumente zu löschen. Diese beiden Seiten sind durch ein Passwort, das über die Einstellungen verändert werden kann, vor unberechtigtem Zugriff geschützt. Die Seite `stats.xhtml` zeigt Nutzungs- und Datenbankstatistiken zu OpenCodex an. Kommt es zu einem wie in 2.4 erklärten Sitzungs-Timeout (die Dauer der Inaktivität ist auf 30 Minuten konfiguriert), wird der Benutzer bei der nächsten Interaktion mit OpenCodex auf die Seite `expired.xhtml` umgeleitet, welche ihn informiert, dass die Sitzung abgelaufen ist.

3.3.2 Managed Beans

Auf der Server-Seite werden die JSF-Seiten von sieben Beans gestützt:

AppBean hält für die sitzungsbasierten Beans eine gemeinsame Instanz des Dokumentenmanagers, welcher eine Datenbankverbindung hält, und des Datenbankmanagers für das Sammlungsarchiv. Diese Objekte werden nur einmal für die Anwendung benötigt und sollten nicht bei jeder neuen Sitzung neu erzeugt werden, da das Aufbauen einer Datenbankverbindung zeitintensiv sein kann und nur eine beschränkte Anzahl an Datenbankverbindungen gleichzeitig genutzt werden kann.

MainBean ist die Hauptklasse für die Gesetzbucherstellung. Die Klasse hält die aktuelle Sammlung und die auf der Seite getroffenen Einstellungen; sie bearbeitet die Suchanfragen und bereitet die Suchergebnisse für die Darstellung im Browser des Benutzers auf.

CompareBean hält die Daten für eine Vergleichserstellung und bearbeitet die Gesetzessuche und die Dokumentenerstellung eines Vergleichs.

ArchiveBean lädt die Archivdaten aus der Datenbank, bereitet sie für die Darstellung beim Benutzer auf und verarbeitet Sammlungsbewertungen von Benutzern. Damit ein Benutzer eine Sammlung nur einmal bewerten kann, wird von dieser Klasse bei der Bewertung in seinem Browser ein Cookie mit allen bereits bewerteten Sammlungs-IDs abgelegt. Die Lebensdauer des Cookies ist auf den Maximalwert eingestellt, da eine Bewertung ja dauerhaft nur einmal möglich sein soll. Das Speichern der schon bewerteten Sammlungen ist nötig, da aufgrund der fehlenden Anmeldung auf dem Server keine Benutzerinformationen abgelegt werden können.

ArchiveAdminBean bietet die administrativen Funktionen *Freischalten* und *Löschen* für Einträge des Sammlungsarchivs.

SettingsBean implementiert das Interface `ExternalConf` des Back-Ends und erlaubt somit, Einstellungen desselben zu beeinflussen. Die dazugehörige Seite `settings.xhtml` stellt eine einfache Benutzeroberfläche hierzu zur Verfügung.

StatBean bereitet die Daten für die Statistikseite `stats.xhtml` auf. Bei jeder Sammlungs- und Vergleichserstellung wird ein Eintrag in eine spezielle Log-Datei geschrieben, aus der Tages-, Wochen-, Monats- und Jahresstatistiken zur Nutzung von OpenCodex erzeugt werden.

Abbildung 11 zeigt zum einen die Verknüpfungen der Managed Beans untereinander, also welches Bean auf welches andere zugreift (gestrichelte Pfeile). Zum anderen werden die Interaktionen der Beans und der einzelnen JSF-Seiten dargestellt (durchgehende Pfeile).

3.3.3 Serialisierung und Sammlungsarchiv

Mit den Funktionen *Speichern* und *Laden* einer Sammlung kann der Benutzer die Zusammenstellung selbst, also nicht das fertig generierte Gesetzbuch, sondern dessen Struktur, auf seinen lokalen Rechner laden und später von dort wieder in OpenCodex importieren. Beide Funktionen arbeiten im Hintergrund mit dem Serialisieren von Daten. Die allgemeinen Einstellungen zu einer Sammlung werden in einer Instanz der Klasse `BookProperties` gehalten und die Gesetze der Sammlung in einer `List<LawEntry>`. Die Klasse `SavedBook` bildet zusammen mit einem Zeitstempel des Speicherzeitpunktes einen Container für die beiden genannten Konstrukte. Die Klassen sind mit Annotationen aus dem Package `javax.xml.bind` versehen. Somit kann eine Instanz über den statischen Aufruf `static void JAXB.marshal(SavedBook, OutputStream)` in eine XML-

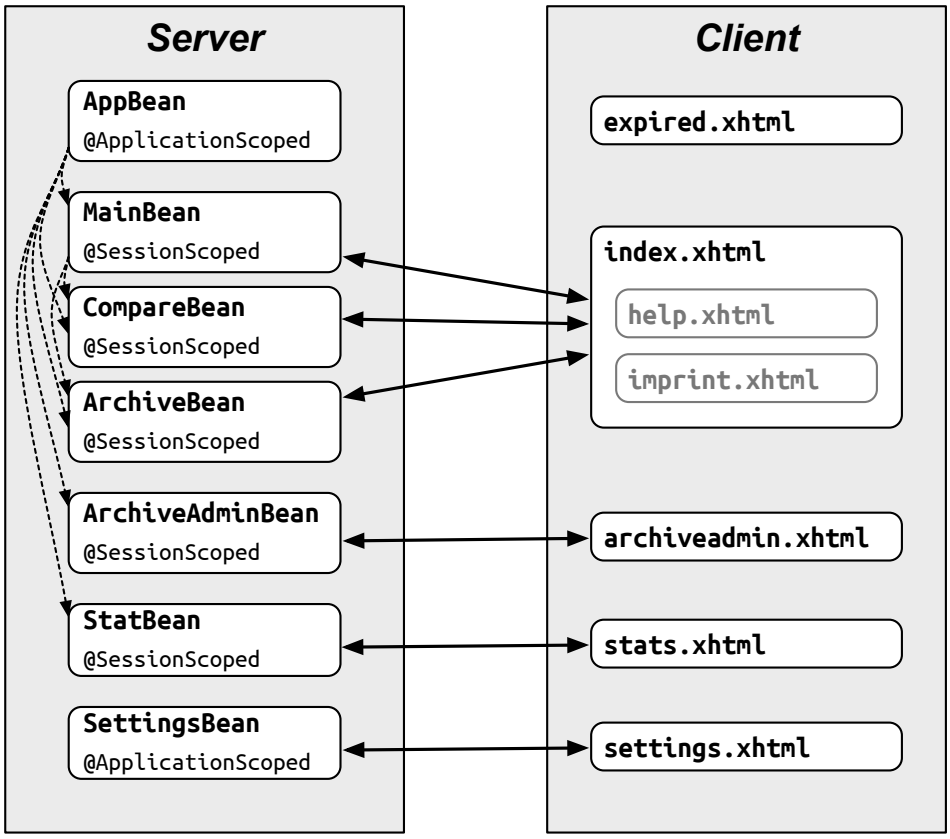


Abbildung 11: Verknüpfungen zwischen den Managed Beans untereinander (Server) bzw. den Beans und den JSF-Seiten (Client).

Datei serialisiert werden. Über den Aufruf `static T JAXB.unmarshal(InputStream, Class<T>)` wird die XML-Datei erneut in Java-Objekte umgewandelt.

Das Sammlungsarchiv arbeitet ebenfalls mit den XML-serialisierten Daten. Wird eine Sammlung in das Archiv veröffentlicht, werden deren Daten serialisiert. Anstatt sie in eine Datei zu schreiben und dem Benutzer zum Download anzubieten, werden sie aber, zusammen mit den weiters angegebenen Informationen (Sammlungsname, Beschreibung, Autor), in eine Tabelle in einer Datenbank gespeichert. Es wird hierfür die Caching-Datenbank genutzt, eine andere Datenbank wäre jedoch auch denkbar, da es sich bei dem Archiv nicht um eine zeitkritische Anwendung handelt.

Eine Sammlung im Archiv ist erst dann sichtbar, wenn sie von einem Administrator freigeschaltet wurde. Deshalb wird dieser immer, wenn eine neue Sammlung in das Archiv gestellt wird, per Email darüber benachrichtigt. Bestätigt er die Sammlung, wird im Tabelleneintrag des Sammlungsarchivs eine Schaltervariable gesetzt. Erst wenn diese aktiviert ist, wird der Eintrag im Archiv angezeigt.

Jede Sammlung kann in ihrer Qualität von 1 bis 5 bewertet werden. Hierzu werden pro Tabelleneintrag zwei weitere Felder mitgespeichert: Ein Zähler i, wie viele Bewertungen bereits abgegeben wurden und der aktuelle Durchschnittswert a. Kommt eine neue Bewertung r hinzu, ergeben sich die neuen Werte:

$$\begin{aligned} i_{new} &= i_{old} + 1 \\ a_{new} &= \frac{i_{old} \times a_{old} + r}{i_{new}} \end{aligned}$$

3.3.4 Behandlung des Sitzungs-Timeout

Nach 30 Minuten Inaktivität des Benutzers kommt es, wie bereits erwähnt, zu einem Sitzungs-Timeout. Es tritt auf der Server-Seite in Form einer `ViewExpiredException` auf. Diese bleibt für gewöhnlich unbehandelt und die Benutzeroberfläche beim Client reagiert ungewollt bzw. gar nicht mehr. Deshalb können in der JSF-Konfigurationsdatei `faces-config.xml` im XML-Element `<factory> ... </factory>` Elemente mit der Bezeichnung `<exception-handler-factory> ... </exception-handler-factory>` abgelegt werden. Darin kann auf voll qualifizierte Klassen, die das Interface `javax.faces.context.ExceptionHandlerFactory` implementieren, verwiesen werden. Tritt ein Fehler auf, wird er an eine Instanz des von der Factory erzeugten `ExceptionHandler`s gegeben und kann davon behandelt werden, ehe er geworfen wird. Im Fall von OpenCodex durchsucht der `ViewExpiredExceptionHandler` die Fehlerliste nach `ViewExpiredException`s, navigiert den Benutzer auf die Seite `expired.xhtml` und löscht die entsprechenden Fehler anschließend, ohne dass sie geworfen werden.[44] Andere auftretende `Exception`s werden von dem Handler nicht bearbeitet und an die darüber liegende Instanz, in dem Fall der GlassFish Anwendungsserver, weitergegeben. Dieser schreibt die auftretenden Fehler in eine programmweite Logdatei; bei schwerwiegenden Fehlern kann es zum Zurücksetzen der Sitzung kommen, dies sollte jedoch nicht auftreten, da die potentiellen Fehler in der Anwendung berücksichtigt und behandelt werden.

[44] Inspiriert durch: `http://weblogs.java.net/blog/edburns/archive/2009/09/03/dealing-gracefully-viewexpiredexception-jsf2`, (22.07.2013)

3.4 Testen

Im Entwicklungsprozess stellte sich rasch heraus, dass ein automatisiertes Testen, z. B. mit dem Framework *JUnit*[45], nicht sinnvoll ist, da das Erstellen von Testcases mit sehr großem Aufwand verbunden wäre. Es müsste pro Testcase händisch ein Referenzdokument aus den XML-Gesetzesdaten erstellt werden, gegen das dann verglichen werden könnte. Ferner ist die Lauffähigkeit des generierten LaTeX-Codes ohnehin nur sichergestellt, wenn ein Versuch, den Code mit einem entsprechenden Compiler in ein PDF umzuwandeln, glückt. Daher wurden über entsprechende Programmaufrufe „manuell" Sammlungen generiert und diese in PDFs umgewandelt. Es wurden zum Testen insbesondere das Datenschutzgesetz 2000 als „normales" Gesetz und das ABGB als umfangreiches Gesetz verwendet. Mit Hilfe der Straßenverkehrsordnung wurden die Integration und interne Umwandlung von Binärdaten (Bilder) getestet, da einige der Paragrafen Abbildungen von Verkehrszeichen beinhalten. Ein geglückter Kompiliervorgang stellte die Lauffähigkeit des LaTeX-Codes sicher, eine Sichtprüfung die Qualität des Endergebnisses.

Durch den Einsatz der Logging API *Apache Log4j 2*[46] ist performantes und komfortables Logging möglich. Durch die Verwendung verschiedener Log-Levels kann die Menge der Ausgaben (sogar im laufenden Betrieb) verändert werden. Debug-Ausgaben halfen bei der Entwicklung, Fehlerstellen im Code, Probleme beim Zusammensetzen von Befehlen (z. B. SQL-Statements zur Datenbankabfrage) und laufzeitintensive Programmteile zu identifizieren. Zur Laufzeitoptimierung, insbesondere beim Parsen der XML-Dokumente in LaTeX-Code, wurde darüber hinaus der in NetBeans integrierte Profiler verwendet.

Das Testen der Benutzeroberfläche ist ebenfalls nicht automatisiert möglich, da mittels einer automatischen Test-Suite keine Bedienung im Browser simuliert werden kann. Die Benutzeroberfläche wurde also in verschiedenen Browsern händisch bezüglich der richtigen Darstellung und der Bedienbarkeit getestet (siehe nächster Abschnitt).

3.5 Browser-Kompatibilität

Die Browser-Kompatibilität von OpenCodex hängt stark von der Kompatibilität der Komponentenbibliothek *PrimeFaces* ab. Diese ist hauptsächlich für die Generierung der XHTML-Seiten, dem darin eingebetteten JavaScript-Code sowie für die Definitionen in

[45] Siehe `http://junit.org`, (22.07.2013)
[46] Siehe `http://logging.apache.org/log4j/2.x`, (22.07.2013)

den verknüpften CSS-Dateien verantwortlich. Die Stylingdefinitionen von *PrimeFaces* gehen dabei nicht auf browserspezifische Attribute der Form `-moz-*` für Firefox oder `-webkit-*` für Safari bzw. Chrome ein.

Die Anwendung wurde mit folgenden Browser-Versionen als lauffähig und kompatibel getestet:

- Firefox 19 – 22
- Safari 6
- Chrome 25 – 27
- Internet Explorer 9 – 10

Auf älteren Versionen dieser Browser *kann* die Anwendung ebenfalls lauffähig sein, es kann jedoch, insbesondere beim Internet Explorer $\leqslant 7$, zu Darstellungs- und Bedienungsproblemen kommen.

Tests mit den Browsern Safari Mobile 6.0 und Google Chrome Mobile 27 zeigten, dass Teile der Benutzeroberfläche auch auf mobilen Browsern funktionieren, andere Teile jedoch nicht. Da OpenCodex nicht auf mobile Endgeräte abzielt, wurden dahingehend keine weiteren Bemühungen unternommen; es wäre jedoch eine eigene mobile Benutzeroberfläche vorstellbar (siehe 4.2.4).

Bezüglich der Auflösung des Browserfensters, auf die hin die Seite optimiert werden soll, wurde aufgrund einiger Statistiken[47, 48] die Entscheidung getroffen, eine Auflösung von mindestens 1024×768 px vorauszusetzen. Diese Display-Auflösung ergibt nach [SO09] eine effektive Weite des Browser-Fensters von 960 px. Insbesondere die Tabelle mit den aktuellen Einträgen einer Sammlung benötigt diese Breite, um die Fülle der Informationen übersichtlich darstellen zu können. Benutzer mit kleineren Auflösungen müssen horizontales Scrollen in Kauf nehmen.

3.6 Sicherheitsaspekte der Anwendung

Da alle von OpenCodex verwendeten Daten offen und frei zugänglich sind und keine benutzerspezifischen Daten gespeichert werden, sind für die Anwendung keine erweiterten Sicherheits- bzw. Verschlüsselungsmaßnahmen nötig. Aus diesem Grund ist die Anwendung unverschlüsselt über HTTP erreichbar, die gesicherte Verbindung wurde im Server deaktiviert. Auch die Kommunikation mit dem Webservice erfolgt unverschlüsselt, da

[47] Siehe `http://www.w3schools.com/browsers/browsers_display.asp`, (22.07.2013)
[48] Siehe `http://www.w3counter.com/globalstats.php`, (22.07.2013)

dieser keine HTTPS-Verbindungen vorsieht. Dies stellt eine potentielle Sicherheitslücke dar, da es über einen Man-in-the-Middle Angriff möglich wäre, die erhaltenen Gesetzestexte zu manipulieren. Das Einschleusen von schadhaftem JavaScript-Code in das PDF ist hingegen nicht möglich, da jeglicher Code, der in den Ausgangsdaten enthalten ist, entweder, wenn er im nicht zu druckenden Bereich der Daten ist, von OpenCodex ignoriert wird, oder, wenn er im druckbaren Bereich (Nutzdaten der Dokumente) ist, im Ausgabedokument lediglich dargestellt, nie jedoch ausgeführt, werden würde. Die Lücke des möglichen Man-in-the-Middle Angriffs kann nur von dem Betreiber des Webservices geschlossen werden.

Von den Benutzern eingegebene Daten wie der Sammlungstitel können keine Manipulationen an der Seite vornehmen (z. B. durch Einschleusen von schadhaftem HTML- oder CSS-Code), da die Daten von den darstellenden PrimeFaces-Komponenten standardmässig nicht direkt dargestellt werden, sondern darin enthaltenes Markup umschrieben (*escaped*) wird.

Um eine Manipulation der Caching-Datenbank und der darin abgelegten Gesetzestexte zu verhindern, wird zum einen der Zugriff auf die Datenbank von Außen verhindert (der Daemon `mysqld` akzeptiert nur Verbindungen vom *localhost*). Zum anderen werden die programmatischen Datenbankzugriffe mittels `PreparedStatement`s effektiv gegen SQL-Injections geschützt.

Die Administrationsoberfläche des GlassFish-Anwendungsservers stellt eine weitere potentielle Sicherheitslücke dar, weil darüber die Anwendung selbst ausgetauscht werden kann. OpenCodex könnte auf dem Server durch eine für den Benutzer (oder auch den Anwendungsserver) schadhafte Anwendung ersetzt werden. Deshalb ist die Administrationsoberfläche nur über verschlüsselte HTTPS-Verbindungen erreichbar und mit einem starken Passwort abgesichert.

Die einzige Stelle, an der in OpenCodex Informationen von Benutzern für alle weiteren Benutzer sichtbar gespeichert werden, ist das Sammlungsarchiv. Um Missbrauch vorzubeugen wird jede Sammlung und deren von einem Benutzer eingegebene Daten, bevor sie öffentlich sichtbar sind, administrativ freigeschaltet.

4 Zusammenfassung

Mit OpenCodex wurde eine Web-Anwendung entwickelt, mit der man Gesetzbücher individuell zusammenstellen und als PDF ausgeben kann. Die Gesetze bzw. ein Ausschnitt daraus sowie deren Gültigkeitsdatum sind frei aus den Daten der RIS-Anwendung *Bundesrecht konsolidiert* wählbar. Das resultierende PDF-Dokument kann bezüglich Schriftgröße und Ausgabe-Seitenformat eingestellt werden. Als zweite Funktion bietet OpenCodex – in der gleichen Benutzeroberfläche integriert – die Möglichkeit zwei Fassungen eines Gesetzes, also dasselbe Gesetz zu zwei Gültigkeitsdaten, zu vergleichen. Die Anzeige der Änderungen erfolgt entweder zweispaltig oder in einem gemeinsamen Text, in dem die konkreten Änderungen hervorgehoben sind.

4.1 Probleme und Lösungen

Bei der Entwicklung von OpenCodex traten einige Probleme auf, die es zu lösen galt:

- Die ersten Experimente mit der Webservice-Schnittstelle zeigten schnell, dass diese sehr langsam und nicht auf das Abrufen einer großen Anzahl von Dokumenten ausgelegt ist. Die Stabilität der Schnittstelle ist ebenfalls nicht zu 100 % gegeben. Bei der darauf folgenden schrittweisen Erhöhung des Parallelitätsgrads (Anzahl der Dokumente, die gleichzeitig parallel abgerufen werden) zeigte sich, dass bis zu einem Wert von etwa 10 die gewünschte und erwartete Beschleunigung eintritt. Der fehlenden Stabilität der Schnittstelle wird mit bis zu drei Wiederholungsversuchen begegnet, eine Maßnahme, die sich als praktikabel herausstellte.
- Der parallele Dokumentenabruf verbesserte das Geschwindigkeitsproblem, bei umfangreichen Gesetzen war die erzielte Geschwindigkeit jedoch immer noch inakzeptabel. Aus diesem Grund wurde ein datenbankgestützter Caching-Mechanismus implementiert. Bereits einmal abgerufene Dokumente können also aus der lokalen Datenbank bezogen werden und müssen nicht erneut über den Webservice abgefragt werden.

Eine Messung ergab für den Dokumentenabruf des gesamten ABGB zum Stand 01.07.2013 (1363 Einzeldokumente mit einer Gesamtgröße von 5,34 MB) folgende Zeiten bzw. Beschleunigungsfaktoren:

Abruf	**Dauer in Sekunden**	**Speedup**
1 Abrufinstanz	983	–
2 Abrufinstanzen	556	1,76
5 Abrufinstanzen	270	3,64
10 Abrufinstanzen	189	5,20
20 Abrufinstanzen	182	5,40
Abruf aus dem Cache	0,26	3780

Tabelle 2: Speedup-Faktoren abhängig vom Parallelisierungsgrad und dem Caching.

Bei den gemessenen Zeiten ist anzumerken, dass diese, abhängig von der Auslastung des Webservices, schwanken. Die dargestellten Werte wurden aus drei Einzelmessungen gemittelt und stellen lediglich einen Richtwert dar. Aus den Werten lassen sich folgende Aussagen ableiten: Der Speedup ist nicht konstant steigend, sondern stagniert bei zehn Instanzen, mit denen parallel abgerufen wird. Das lokale Caching bringt einen signifikanten Geschwindigkeitszuwachs.

- Die gelieferten Daten liegen als schema-definierte XML-Daten vor, die auch HTML-Formatierungen enthalten. Für diesen Anwendungsfall konnte kein passender Parser gefunden werden, der aus den XML-Daten sinnvollen LaTeX-Code generierte. Es musste also selbst ein geeigneter Parser implementiert werden. Insbesondere die Verarbeitung von Sonderzeichen bereitete hierbei Probleme, da diese in den beiden Sprachen sehr unterschiedlich behandelt werden und Java als dritte Sprache zusätzliches Escaping erforderte.
- Kurz vor Abschluss der Entwicklung wurde festgestellt, dass die Lizenzen der gewählten Caching-Datenbank *H2 Database Engine*[49], nämlich die *Mozilla Public License 1.1* und die *Eclipse Public License 1.0*, nicht mit der GNU General Public License (GPL) kompatibel sind und die Datenbank deshalb nicht in OpenCodex integriert werden kann. Da sich der Umbau der Anwendung auf eine externe MySQL-Datenbank darauf beschränkte, den JDBC-Treiber und den Datenbank-Verbindungs-String auszutauschen, wurde dieses Problem rasch gelöst.

[49] Siehe `http://www.h2database.com`, (22.07.2013)

4.2 Mögliche Erweiterungen

OpenCodex könnte an verschiedenen Stellen erweitert und verändert werden. Die folgenden Punkte sollen dazu Inspiration geben und „Angriffspunkte“ im Code aufzeigen.

4.2.1 Landesrecht, Gemeinderecht und andere RIS-Anwendungen

Der Webservice sieht durch den Parameter `application` prinzipiell vor, dass andere RIS-Anwendungen als das *Bundesrecht konsolidiert* unterstützt werden. Die Dokumentation zeigt noch keine derartigen Möglichkeiten auf [50], nach C.2.2 sind aber für das Jahr 2013 derartige Erweiterungen geplant.

Die Anwendung müsste dazu im Back-End, falls die Suchschnittstelle ansonsten unverändert bleibt, lediglich leicht in den Klassen `DocumentIDsLoaderWS` bzw. `SearchData` verändert werden. Ändert sich die Suchschnittstelle, wären ebenfalls Änderungen im (generierten) Webservice-Code bzw. im Wrapper nötig. Sollte sich die Struktur der vom Webservice gelieferten Dokumente ändern, wären je nach Umfang Änderungen, bzw. eine separate Neuimplementierung der Klasse `DocumentContentParser` erforderlich.

Das Front-End müsste um weitere Suchmöglichkeiten ergänzt werden. Dies wäre einfach als neuer Reiter neben der *Suche Bundesrecht* möglich. Da die RIS-Anwendungen der Landesrechte je Bundesland eine andere Suchmaske haben, ist auch damit zu rechnen, dass der Web-Service je Land eine Schnittstelle bietet. Es müsste also entweder pro Land ein eigener Reiter mit Suchseite implementiert oder über ein Auswahlfeld das Land gewählt werden. Man müsste dann davon abhängig die Suchanfrage an die jeweilige Landessuche im Back-End weiterleiten.

4.2.2 Anpassungen der Ausgabe

Das Ausgabe-PDF könnte anders gestaltet werden. Es wäre denkbar, dem Benutzer nicht nur die Einstellung des Seitenformates, der Schriftgröße und der Anzahl der Spalten zu bieten, sondern verschiedene Templates anzubieten, aus denen er auswählen kann. Damit könnte etwa die Lesbarkeit weiter erhöht und an Menschen mit Beeinträchtigungen angepasst oder Gesetzestexte als Lernunterlagen für Studierende aufbereitet werden (z. B. größerer Zeilenabstand oder mehr Seitenrand für Notizen und Anmerkungen). Hierzu wären im Back-End und im Front-End Änderungen nötig: Im Back-End müssten

[50] Aus [Öst12, S. 2]: „Derzeit immer ‚Br‘ (Bundesrecht konsolidiert)“.

die Templates hinterlegt und deren Auswahl ermöglicht werden. In das Front-End müsste die Template-Auswahl, am besten mit Vorschaubildern, integriert und gegebenenfalls um templateabhängige Parameter erweitert werden.

4.2.3 Erweiterte Binärdateiunterstützung

Wie in Tabelle 1 schon beschrieben, werden von OpenCodex nicht alle potentiell auftretenden Binärdatenformate unterstützt. Bei den nicht unterstützten Formaten handelt es sich um Microsoft Word und OpenDocument-Text Dateien. Es gibt keine direkte Möglichkeit, diese in ein LaTeX-Dokument einzubinden. Eine Möglichkeit, den Inhalt der Dokumente dennoch in das Ausgabe-PDF zu integrieren wäre, die Dokumente in das PDF-Format umzuwandeln und das resultierende Dokument über den Befehl `\includegraphics{}` (einbinden als Bild) oder `\includepdf{}` (einbinden als eigene PDF-Seiten) in die Ausgabe zu integrieren.

Die Umwandlung diverser Office-Formate in PDF ist mit LibreOffice[51] im sogenannten „headless“-Modus, also aus einer Kommandozeile heraus, ohne die Anzeige der grafischen Benutzeroberfläche, möglich. Hierzu muss lediglich ein Befehl der Form:
`libreoffice --headless --invisible --convert-to pdf /pfad/zum/ursprungsdokument.doc`
aufgerufen werden. Dies wäre aus OpenCodex heraus über JProc genauso wie der Aufruf des externen LaTeX-Compilers möglich. Es wäre dazu allerdings eine LibreOffice-Installation auf dem Server nötig. Die Auswahl der unterstützten Formate würde dann weit über die wenigen vom Webservice gelieferten hinausgehen; es könnten alle Formate integriert werden, die LibreOffice umwandlen kann.

4.2.4 Alternative Benutzeroberfläche

Das Back-End von OpenCodex ist nicht von der aktuellen, konkreten Implementierung des Front-Ends abhängig. Deshalb wäre es denkbar, alternative Benutzeroberflächen zu verwenden. Man könnte etwa OpenCodex als eigenständige Anwendung auf dem Computer des Benutzers installieren. Vorteile wären die für den Benutzer vertrautere und somit schnellere Bedienung eines „nativen“ Programms genauso wie die Resourcenschonung am Anwendungsserver. Nachteile ergeben sich beim Caching: Der Benutzer kann, aufgrund des unabhängigen Caches, nicht von den bereits für andere Nutzer geladenen Dokumenten profitieren. Ein weiterer Nachteil ergibt sich aus der nicht trivialen Installation der benötigten LaTeX-Umgebung bzw. der Konfiguration von OpenCodex, um

[51] Siehe `http://www.libreoffice.org`, (22.07.2013)

diese Umgebung zu verwenden. Ein Ausweg daraus wäre es, OpenCodex mit einer integrierten, portablen LaTeX-Umgebung auszuliefern, was jedoch auf der anderen Seite das Auslieferungspaket auf mindestens mehrere hundert Megabytes vergrößern würde.

Denkbar wäre auch, nur das Front-End als native Anwendung auf dem Benutzer-PC zu haben, das dann mit einem fest konfigurierten Back-End auf dem Compile-Server kommuniziert. Hierzu müsste neben der Front-End-Anwendung (=Client) lediglich ein Wrapper für das Back-End geschrieben werden, welcher die Kommunikation – z. B. direkt XML über eine TCP-Verbindung oder als SOAP-Webservice – mit den Clients für das eigentliche Back-End übersetzt. Die Vorteile des gemeinsamen Dokumentencaches und der zentralen LaTeX-Installation wären hierbei genauso gegeben wie die vertraute und schnelle Benutzeroberfläche.

Dieses Konzept könnte auch auf mobilen Geräten als Apps umgesetzt werden. Hierfür müsste lediglich die Front-End Anwendung für die iOS- bzw. Android-Plattform implementiert werden. Der Nutzen bleibt hierbei jedoch fraglich, da die Ausgabe von OpenCodex nicht auf das Lesen auf kleinen Bildschirmen optimiert ist und für beide Systeme ohnehin RIS-Apps existieren.

Eine alternative Web-Benutzeroberfläche wäre ebenfalls möglich. Das Front-End könnte etwa in PHP implementiert und mit Hilfe von *Apache Thrift*[52] an das Java-Back-End gebunden werden.

Es könnte ferner mit Hilfe der Komponentenbibliothek *PrimeFaces Mobile*[53] eine eigens auf mobile Geräte optimierte Version der schon vorhandenen Benutzeroberfläche geschaffen werden. Man müsste dann lediglich eine Seite zur Browser-Erkennung der eigentlichen Benutzeroberfläche vorschalten.

4.2.5 User-Verwaltung

Anstatt eine erstellte Sammlung lokal zu sichern bzw. im Archiv zu veröffentlichen könnte man OpenCodex mit einer Benutzeranmeldung versehen. Es würden dann pro Benutzer die getroffenen Einstellungen gespeichert bzw. dessen Sammlungen, allerdings nur für ihn, abrufbar. Für Benutzer ergibt sich dabei die Hürde einer Anmeldung mit den damit verbundenen Unannehmlichkeiten: Auswahl und Merken von Benutzernamen und Passwort, Verifizieren der E-Mail-Adresse, Passwort-Reset bei vergessenen Passwörtern, man kann die Plattform nicht „sofort“ testen, sondern muss sich erst anmelden, etc.

[52] Siehe `http://thrift.apache.org`, (22.07.2013)
[53] Siehe `http://www.primefaces.org/showcase-labs/mobile/index.jsf`, (22.07.2013)

Technologisch müsste OpenCodex gegen unberechtigte Zugriffe abgesichert werden, da die Benutzerdaten sensibler zu behandeln sind als die ohnehin frei verfügbaren Gesetzestexte. Die Anwendung dürfte, zumindest im Anmeldebereich, nur noch über HTTPS erreichbar sein. Vor die eigentliche Oberfläche müsste ein Anmeldebildschirm geschaltet werden, und die Benutzerinformationen müssten in einer Datenbank abgelegt werden.

A Handbuch

A.1 Benutzer

Als Handbuch für die Benutzer von OpenCodex *diene der Hilfe-Text der Anwendung:*

Einführung

OpenCodex bietet zwei Möglichkeiten eine Gesetzessammlung zu erstellen:

1. Zusammenstellen einer eigenen Sammlung
2. Laden einer Sammlung aus dem Archiv

Das Zusammenstellen einer eigenen Sammlung ermöglicht es Ihnen, komplett individualisierte Gesetzbücher zu erstellen. Sie können hierbei auf den gesamten Datenbestand der RIS-Anwendung „Bundesrecht konsolidiert“ zugreifen. Die Suche und das Hinzufügen von Gesetzen, Verordnungen, Kundmachungen, etc. erfolgt über den Reiter *Suche Bundesrecht*.

Alternativ können Sie eine vorgefertigte Sammlung aus dem Archiv (Reiter *Sammlungsarchiv*) laden. Die verfügbaren Sammlungen werden sowohl von dem Betreiber als auch von den Benutzern von OpenCodex zur Verfügung gestellt und redaktionell betreut.

Eigene Sammlung

Wenn Sie OpenCodex laden, können Sie sofort beginnen, Rechtsvorschriften in eine Sammlung zu integrieren. Dies geschieht im Reiter *Suche Bundesrecht* entweder über den Namen des Gesetzes oder über die Suche nach den Indizes des Bundesrechts[54].

Bei der Suche nach dem Namen können Sie die gängigen, offiziellen Abkürzungen (z. B.: „B-VG“, „ABGB“, „StVO“) verwenden oder auch den (Lang-)Titel des Gesetzes. Groß-/Kleinschreibung wird hierbei nicht beachtet. Sie können bei der Suche auch den Platzhalter * verwenden; dies kann allerdings je nach Suchbegriff zu sehr vielen Suchergebnissen führen.

[54] Siehe `http://www.ris.bka.gv.at/UI/Bund/Bundesnormen/IndexBundesrecht.aspx`, (22.07.2013)

Nach abgeschlossener Suche erscheint ein Fenster mit den Ergebnissen. Sie können die gewünschten Gesetze mit einem Häkchen in der linken Spalte auswählen. Wollen Sie alle gefundenen Gesetze übernehmen, so können Sie das Häkchen in der Titelzeile der Tabelle setzen. Das in der Auswahlbox *Gültigkeitsdatum der Gesetze* ausgewählte Datum gibt den Stichtag an, zu dem die Gesetze, welche in die Sammlung aufgenommen werden, Gültigkeit haben sollen. Dies können Sie später für jeden Eintrag individuell bearbeiten.

Die Schaltfläche *Bearbeiten* am linken Rand ermöglicht es, Titel und Untertitel der Sammlung einzugeben, sowie das gewünschte Papierformat und die Schriftgröße zu wählen. Hier können Sie auch eine erstellte Sammlung im Archiv veröffentlichen.

Die Schaltfläche *Speichern* erlaubt es, eine Zusammenstellung auf Ihren Computer herunterzuladen und somit bei Ihnen lokal zu speichern. Wenn Sie später diese Zusammenstellung in OpenCodex weiterverwenden wollen, können Sie über den Button *Laden* diese wieder in OpenCodex zurückholen.

Sind Sie mit dem Hinzufügen von Gesetzen fertig, können Sie über die Schaltfläche *Erstellen* die Dokumentenerzeugung starten. Ein Fortschrittsbalken gibt Auskunft über den Status des Erstellvorgangs. Die Erstelldauer ist davon abhängig, wie umfangreich Ihre Sammlung ist und wieviele der von Ihnen verwendeten RIS-Einzeldokumente in OpenCodex bereits verwendet und deshalb lokal auf dem Server zwischengespeichert wurden. Ist der Erstellvorgang abgeschlossen, so wird eine Vorschau auf die erstellte Sammlung gezeigt. In dem Vorschaufenster können Sie diese als PDF-Dokument herunterladen.

Sammlung aus dem Archiv

Über den Reiter *Sammlungsarchiv* gelangen Sie zu der Liste der gespeicherten Sammlungen. Neben Name und Beschreibung wird auch angezeigt, wie andere Benutzer eine Sammlung bewertet haben. Über die Schaltfläche erscheint ein Fenster mit weiteren Informationen zu der Sammlung. Dort können Sie diese mittels der Rating-Sterne bewerten. Die Bewertung wird über an den Server übermittelt. Die Sammlung wird mit dem Button geladen.

Mit einer geladenen Archiv-Sammlung können Sie weiter verfahren (bearbeiten, speichern, erstellen) wie mit einer selbst erstellten Sammlung. Ihre Änderungen wirken sich nicht auf die Sammlung im Archiv aus.

Bearbeiten eines Eintrages

Die Tabelle am unteren Rand der Seite zeigt die Einträge der aktuellen Gesetzessammlung an. In der rechten Spalte befinden sich vier Buttons zum Bearbeiten des jeweiligen Eintrags:

- Mit den Schaltflächen und können Sie die Reihenfolge der Rechtsvorschriften in einer Sammlung bearbeiten.
- Mit der Schaltfläche kann ein Eintrag aus der Sammlung entfernt werden.
- Über die Schaltfläche können Sie verschiedene Einstellungen für die gewählte Rechtsvorschrift treffen:
  - Mit der Schaltfläche *§0 inkludieren* können Sie einstellen, ob der Paragraph Null eines Gesetzes inkludiert werden soll oder nicht. Dieser enthält neben verschiedenen Metainformationen wie In- und Außerkrafttretensdatum oftmals ein Inhaltsverzeichnis des Gesetzes.
  - Die Schaltfläche *Gesamtes Gesetz* gibt an, ob die gesamte Rechtsvorschrift in die Sammlung aufgenommen werden soll. Wird sie abgewählt, so ist es über die beiden Auswahlmenüs *Von* und *Bis* möglich, lediglich einen Bereich des Gesetzes auszuwählen. Aufgrund von Einschränkungen der verfügbaren Daten ist es hierbei nicht möglich, einen Bereich über das eigentliche Gesetz und dessen Anhänge zu wählen (ein Bereich von § 23 bis § 42 ist möglich, ein Bereich von § 42 bis Anl. 5 ist nicht möglich).
  - Wird die Einstellung *aktuell* ausgewählt, so wird immer die aktuelle Version der Rechtsvorschrift verwendet, auch wenn die Sammlung gespeichert und zu einem späteren Zeitpunkt wieder geladen wird oder die Sammlung aus dem Archiv geladen wird.
    Wird hingegen als Datum der heutige Tag gewählt, so bleibt das gesetzte Datum erhalten.

Eigene Sammlung veröffentlichen

Haben Sie Sammlungen erstellt, welche oftmals benötigt werden, weil sie etwa (ähnlich wie kommerziell erhältliche Gesetzessammlungen) alle Gesetze eines bestimmten Themengebietes enthalten, können Sie diese allen Benutzern von OpenCodex zugänglich machen. Hierzu benutzen Sie die Schaltfläche *Veröffentlichen ...* in dem Dialogfenster „Bearbeiten“. Diese ist nur sichtbar, wenn die Sammlung selbst erstellt und nicht aus dem Archiv geladen wurde. Nach Betätigen der Schaltfläche öffnet sich ein weiteres Dialogfeld, in welchem Sie den Titel der Sammlung bearbeiten, sowie eine Beschreibung der Sammlung für die anderen Benutzer eingeben können. Auch ist es möglich, den Autor anzugeben.

Nach einem Klick auf die Schaltfläche *Veröffentlichen* wird diese im Archiv aufgenommen, jedoch noch nicht für alle Benutzer veröffentlicht. Die Sammlung wird zunächst von dem Betreiber von OpenCodex geprüft und ist nach dessen Bestätigung für alle Benutzer verfügbar.

Vergleich zweier Gesetzesfassungen

OpenCodex ermöglicht es Ihnen, zwei Fassungen, also den jeweiligen Stand eines Gesetzes zu zwei gegebenen Zeitpunkten, zu vergleichen. Hierzu benützen Sie den Reiter *Fassungsvergleich*:

Über die Suchfunktion in der oberen Zeile wählen Sie das Gesetz, dessen Fassungen verglichen werden sollen. Mit Hilfe der beiden Datumsauswahlfelder *Gültigkeit alt* und *Gültigkeit neu* können Sie die zu vergleichenden Daten wählen und angeben, ob das gesamte Gesetz verglichen werden soll, oder nur ein Ausschnitt daraus.

Unterhalb dieser Angaben können Sie einige Einstellungen zur Darstellung der Gesetzesänderungen treffen:

- Bei der *Darstellung Gesetzesänderung* haben Sie drei Möglichkeiten zur Auswahl:
  - *Zweispaltig* bedeutet, dass die alte und die neue Version eines Paragrafen/Artikels in zwei Spalten nebeneinander dargestellt wird.
  - *Nur Änderungen* heisst, dass die Änderungen, soweit möglich, direkt im Text hervorgehoben werden. Dabei werden entfernte Passagen rot und durchgestrichen gekennzeichnet, neu hinzugefügte Passagen werden in blau und durch Unterstreichung hervorgehoben.
  - *Dynamisch* gibt an, dass die Art der Darstellung, je nach Umfang der Änderungen, dynamisch zwischen *zweispaltig* und *Nur Änderungen* vom System automatisch ausgewählt wird.
- Die Angabe *Unveränderte Paragrafen/Artikel anzeigen* gibt an, ob im generierten Dokument Paragrafen bzw. Artikel, welche zwischen den beiden Daten unverändert blieben, angezeigt werden sollen oder nicht.
- Das Auswahlfeld *Text gelöschter Paragrafen/Artikel anzeigen* gibt an, ob der Text von veralteten Paragrafen bzw. Artikeln, welche in der alten Fassung enthalten waren, in der neuen jedoch nicht mehr, angezeigt werden soll. Soll der Text nicht angezeigt werden, so enthält das Ausgabedokument lediglich den Hinweis, dass der entsprechende Paragraf/Artikel gelöscht wurde.
- Unter *Papierformat* bzw. *Schriftgröße* können die Ausgabeoptionen des Dokumentes eingestellt werden.

Browser-Kompatibilität

OpenCodex wurde mit folgenden Browsern getestet:

- Firefox 19
- Safari 6
- Chrome 25
- Internet Explorer 9

Bitte verwenden Sie einen aktuellen Browser, wenn Sie unsere Webapplikation nutzen wollen. Bei anderen Browsern bzw. Browserversionen können während der Verwendung Probleme auftreten. Die Anwendung ist auf eine Bildschirmauflösung von mindestens 1024x768 Pixel optimiert.

A.2 Administrator

OpenCodex läuft auf einem *Apache GlassFish™ Server Open Source Edition* Version `3.1.2.2 (build 5)`. Dieser läuft in einem auf *Microsoft Hyper-V* virtualisierten Debian Linux, Version `6.0.6`, Kernel Version `2.6.32-5-686`. Die Caching-Datenbank läuft auf derselben virtuellen Maschine in Form des MySQL-Servers *mysqld* in der Version `5.1.66-0+squeeze1`.

Als LaTeX-Distribution wurde TeXLive 2012 installiert, welches das benötigte Programm *pdflatex* in der Version `3.1415926-2.4-1.40.13` installiert und das Programm *splitindex* in Version `0.1` mitbringt. Das Script *latexdiff* ist in Version `1.0.2` installiert.

Der GlassFish Server ist über die URL
`https://opencodex.fim.uni-linz.ac.at:4848`
administrierbar. Ein Neustart des Daemons kann über eine SSH-Verbindung auf dem Server als Root-Benutzer mittels des Befehls `/etc/init.d/glassfish restart` angestoßen werden. Der Neustart kann bis zu ca. einer Minute dauern.

Das Logging von OpenCodex geschieht in die Datei
`/opt/glassfish3/glassfish/domains/domain1/logs/server.log`,
die Log-Datei wird ab einer Größe von 2 MB automatisch versioniert. Die Dokumente werden in dem Ordner
`/opt/glassfish3/glassfish/domains/domain1/config/output/latex`
generiert und kompiliert. Anschliessend werden die fertigen PDF-Dokumente von dem Programm in den Ordner
`/opt/glassfish3/glassfish/domains/domain1/applications/OpenCodex/created`
kopiert. Über die passwortgeschützte Seite
`http://opencodex.fim.uni-linz.ac.at/settings.xhtml`
können diese beiden Ordner bereinigt werden.

B Beispieldokumente

Dieser Abschnitt enthält Ausschnitte aus einer Gesetzessammlung und aus einem Fassungsvergleich.

B.1 Sammlung

Die beiden folgenden Seiten sind der Ausschnitt § 54 bis § 56a aus der Straßenverkehrsordnung, da hier sowohl normale Paragrafen als auch die nahtlose Integration von Binärdateien ersichtlich sind. Zur Erstellung wurden die Standardeinstellungen gewählt, also Seitenformat *DIN A4*, Schriftgröße *9*, Spaltenanzahl *Zwei*. Das generierte Dokument enthielte neben Titelei, Inhaltsverzeichnis und eigentlichem Dokumenteninhalt auch ein Schlagwortverzeichnis. Es werden hier jedoch nur die Inhaltsseiten wiedergegeben.

1. Bundesgesetz vom 6. Juli 1960, mit dem Vorschriften über die Straßenpolizei erlassen werden

Aktuelle Fassung vom 10. Juni 2013.

Achtung, diese Sammlung umfasst nicht die ganze Rechtsvorschrift, sondern lediglich § 54 bis § 56a!

Achtung, §0 ist nicht enthalten!

§ 54. Zusatztafeln.

(1) Unter den in den §§ 50, 52 und 53 genannten Straßenverkehrszeichen sowie unter den in § 38 genannten Lichtzeichen können auf Zusatztafeln weitere, das Straßenverkehrszeichen oder Lichtzeichen erläuternde oder wichtige, sich auf das Straßenverkehrszeichen oder Lichtzeichen beziehende, dieses erweiternde oder einschränkende oder der Sicherheit oder Leichtigkeit des Verkehrs dienliche Angaben gemacht werden.

(2) Die Angaben und Zeichen auf Zusatztafeln müssen leicht verständlich sein. Insbesondere kann auch durch Pfeile in die Richtung der Gefahr oder des verkehrswichtigen Umstandes gewiesen werden.

(3) Die Zusatztafeln sind Straßenverkehrszeichen. Sie sind, sofern sich aus den Bestimmungen des § 53 Z 6 nichts anderes ergibt, rechteckige, weiße Tafeln; sie dürfen das darüber befindliche Straßenverkehrszeichen seitlich nicht überragen.

(4) Zusatztafeln dürfen nicht verwendet werden, wenn ihre Bedeutung durch ein anderes Straßenverkehrszeichen (§§ 50, 52 und 53) zum Ausdruck gebracht werden kann.

(5) Die nachstehenden Zusatztafeln bedeuten:

a)

Eine solche Zusatztafel gibt die Entfernung bis zu der Straßenstelle an, auf die sich das betreffende Straßenverkehrszeichen bezieht.

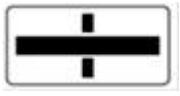
b)

Eine solche Zusatztafel gibt die Länge eines Straßenabschnittes an, für den das betreffende Straßenverkehrszeichen gilt, wie etwa eine längere Gefahrenstelle, die Länge einer Verbots- oder Beschränkungsstrecke u. dgl.

c)

Eine solche Zusatztafel unter dem Zeichen „Vorrang geben“ kündigt das Zeichen „Halt“ an (§ 48 Abs. 6).

d)



Eine solche Zusatztafel unter den Zeichen „Vorrang geben“ oder „Halt“ zeigt an, dass die Querstraße eine Vorrangstraße ist.

e)

Eine solche Zusatztafel unter den Zeichen „Vorrang geben“, „Halt“ oder „Vorrangstraße“ zeigt an, dass eine Straße mit Vorrang einen besonderen Verlauf nimmt (§ 19 Abs. 4).

f)

Diese Zusatztafel weist darauf hin, dass das Straßenverkehrszeichen bei Schneelage oder Eisbildung auf der Fahrbahn zu beachten ist.

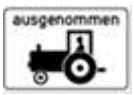
g)

Diese Zusatztafel weist darauf hin, dass das Straßenverkehrszeichen bei nasser Fahrbahn zu beachten ist. Die Symbole der Zusatztafeln nach lit. f und g dürfen auch auf einer Zusatztafel nebeneinander angebracht werden.

h)

Eine solche Zusatztafel unter dem Zeichen „Halten und Parken verboten“ zeigt an, dass das Halte- und Parkverbot nicht für Fahrzeuge gilt, die nach der Bestimmung des § 29b Abs. 4 gekennzeichnet sind.

i)



Eine solche Zusatztafel unter dem Zeichen „Überholen verboten“ zeigt an, dass Zugmaschinen, Motorkarren, selbstfahrende Arbeitsmaschinen und vierrädrige Leichtkraftfahrzeuge überholt werden dürfen.

j)

Eine solche Zusatztafel unter dem Zeichen „Halten und Parken verboten“ zeigt eine Abschleppzone (§ 89a Abs. 2 lit. b) an.

k)

Diese Zusatztafel darf nur verwendet werden, wenn auf einer Fahrbahn mit mehreren Fahrstreifen für dieselbe Fahrtrichtung Straßenverkehrszeichen oberhalb eines Fahrstreifens angebracht sind; sie zeigt an, dass das Straßenverkehrszeichen nur für diesen Fahrstreifen gilt.

l)

Diese Zusatztafel darf nur in Verbindung mit einem Straßenverkehrszeichen verwendet werden, das auf einer Verkehrsinsel, einem Fahrbahnteiler oder einer ähnlichen baulichen Einrichtung, die die Fahrbahn in mehrere Fahrstreifen für dieselbe Fahrtrichtung aufteilt, angebracht ist. Sie zeigt an, dass das Straßenverkehrszeichen nur für den Fahrstreifen gilt, der links an der Trennungseinrichtung vorbeiführt.

E. Verkehrsleiteinrichtungen.

§ 55. Bodenmarkierungen auf der Straße.

(1) Zur Sicherung, Leitung und Ordnung des fließenden und des ruhenden Verkehrs können auf der Straße Bodenmarkierungen angebracht werden; sie können als Längsmarkierungen, Quermarkierungen, Richtungspfeile, Schraffen, Schriftzeichen, Symbole u. dgl. ausgeführt werden.

(2) Längs- oder Quermarkierungen, die ein Verbot oder Gebot bedeuten, wie Sperrlinien (§ 9 Abs. 1), Haltelinien vor Kreuzungen (§ 9 Abs. 3 und 4) und Längsmarkierungen, die dazu dienen, den Fahrbahnrand auszuzeigen (Randlinien), sind als nicht unterbrochene Linien anzuführen.

(3) Längs- oder Quermarkierungen, die dazu dienen, den Verkehr zu leiten oder zu ordnen (Leit- oder Ordnungslinien) und Längsmarkierungen, die dazu dienen, die Fahrbahn von anderen Verkehrsflächen, wie Einmündungen, Ausfahrten u. dgl., abzugrenzen (Begrenzungslinien), sind als unterbrochene Linien auszuführen.

(4) Sperrflächen sind als schräge, parallele Linien (Schraffen), die durch nichtunterbrochene Linien begrenzt sind, auszuführen. Parkverbote können mit einer Zickzacklinie kundgemacht werden.

(5) Wenn die Anlage einer Straße entsprechende Fahrmanöver zuläßt, kann unmittelbar neben einer Sperrlinie eine Leitlinie angebracht werden (§ 9 Abs. 1). Wenn es die Verkehrsverhältnisse erfordern, daß in jeder Fahrtrichtung zumindest zwei Fahrstreifen durch Markierung gekennzeichnet werden, dann sind zum Trennen der Fahrtrichtungen zwei Sperrlinien nebeneinander anzubringen.

(6) Bodenmarkierungen, ausgenommen die Darstellung von Verkehrszeichen, sind in weißer Farbe auszuführen; Zickzacklinien sind jedoch in gelber, Kurzparkzonen in blauer Farbe auszuführen. Wenn es erforderlich ist, eine durch Bodenmarkierungen zum Ausdruck gebrachte Verkehrsregelung vorübergehend durch eine andere Regelung zu ersetzen, sind die dafür notwendigen Bodenmarkierungen in einer anderen Farbe auszuführen.

(7) Bodenmarkierungen können dem jeweiligen Stand der Wissenschaft und Technik entsprechend durch Beschichten der Fahrbahn, durch Aufbringen von Belägen, durch den Einbau von Kunst- oder Natursteinen oder von Formstücken, durch Aufbringen von Fahrstreifenbegrenzern u. dgl. dargestellt werden.

(8) Abweichend von Abs. 6 sind die in § 24 Abs. 1 lit. p und 3 lit. a genannten Linien in gelber Farbe auszuführen; die in § 24 Abs. 3 lit. a angeführten Linien sind überdies abweichend von Abs. 2 als unterbrochene Linien auszuführen. Die genannten Linien sind außerhalb einer allenfalls vorhandenen Randlinie anzubringen und können bei Vorhandensein eines Gehsteigs auch auf diesem in einer Entfernung von nicht mehr als 0,30 m zum Fahrbahnrand angebracht werden.

(9) (Anm.: aufgehoben durch BGBl. Nr. 518/1994)

§ 56. Schutzwegmarkierungen.

(1) In Ortsgebieten sind auf Straßenstellen, wo ständig betriebene Lichtzeichen zur Regelung des Verkehrs oder zur Abgabe blinkenden gelben Lichtes vorhanden sind, auch Schutzwege (§ 2 Abs. 1 Z 12) in entsprechender Anzahl anzulegen, sofern für den Fußgängerverkehr nicht in anderer Weise, etwa durch Über- oder Unterführungen, Vorsorge getroffen ist.

(2) Auf anderen als den in Abs. 1 bezeichneten Straßenstellen sind Schutzwege dann anzulegen, wenn es Sicherheit und Umfang des Fußgängerverkehrs erfordern. Die Benützung solcher Schutzwege ist durch Lichtzeichen zu regeln.

(3) Solange es die Verkehrsverhältnisse nicht erfordern, kann von einer Regelung des Verkehrs durch Lichtzeichen bei den in Abs. 2 genannten Schutzwegen Abstand genommen werden. In diesem Falle ist der Schutzweg mit blinkendem gelbem Licht (§ 38 Abs. 3) oder mit dem Hinweiszeichen nach § 53 Z 2a („Kennzeichnung eines Schutzweges“) zu kennzeichnen.

§ 56a. Radfahrerüberfahrtmarkierungen

(1) Im Ortsgebiet sind auf Straßenstellen, wo ständig betriebene Lichtzeichen zur Regelung des Verkehrs oder zur Abgabe blinkenden gelben Lichtes vorhanden sind, auch Radfahrerüberfahrten (§ 2 Abs. 1 Z 12a) anzulegen, sofern dies in Fortsetzung von Radfahrstreifen, Radwegen oder Geh- und Radwegen erforderlich ist und für den Fahrradverkehr nicht in anderer Weise, etwa durch Über- oder Unterführungen, Vorsorge getroffen ist.

(2) Auf anderen als den in Abs. 1 bezeichneten Straßenstellen sind Radfahrerüberfahrten dann anzulegen, wenn es die Sicherheit und der Umfang des Fahrradverkehrs erfordern. Die Benützung solcher Radfahrerüberfahrten ist durch Lichtzeichen zu regeln.

(3) Solange es die Verkehrsverhältnisse nicht erfordern, kann von einer Regelung des Verkehrs durch Lichtzeichen bei den in Abs. 2 genannten Radfahrerüberfahrten Abstand genommen werden. In diesem Falle ist die Radfahrerüberfahrt mit blinkendem gelben Licht oder mit dem Hinweiszeichen nach § 53 Abs. 1 Z 2b („Kennzeichnung einer Radfahrerüberfahrt“) zu kennzeichnen.

B.2 Vergleich

Die folgenden vier Seiten enthalten den kompletten Vergleich des Datenschutzgesetzes (DSG 2000) vom Stand 10.06.2010 und 10.06.2013 mit den Standardeinstellungen, also *Gesamtes Gesetz*, Darstellungsart Gesetzesänderung *Dynamisch*, Papierformat *DIN A4*, Schriftgröße *9*.

Datenschutzgesetz 2000

Vergleich Fassung vom *10.06.2010* und vom *10.06.2013*

Generiert mit OpenCodex 0.8.6
am 10. Juni 2013

Inhaltsverzeichnis

Art. 2 § 36 wurde geändert:

Zusammensetzung der Datenschutzkommission

§ 36. (1) Die Datenschutzkommission besteht aus sechs Mitgliedern, die auf Vorschlag der Bundesregierung vom Bundespräsidenten für die Dauer von fünf Jahren bestellt werden. Wiederbestellungen sind zulässig. Die Mitglieder müssen rechtskundig sein. Ein Mitglied muß dem Richterstand angehören.

(2) Die Vorbereitung des Vorschlages der Bundesregierung für die Bestellung der Mitglieder der Datenschutzkommission obliegt dem Bundeskanzler. Er hat dabei Bedacht zu nehmen auf:

1. einen Dreiervorschlag des Präsidenten des Obersten Gerichtshofs für das richterliche Mitglied,
2. einen Vorschlag der Länder für zwei Mitglieder,
3. einen Dreiervorschlag der Bundeskammer für Arbeiter und Angestellte für ein Mitglied,
4. einen Dreiervorschlag der Wirtschaftskammer Österreich für ein Mitglied.

Alle vorgeschlagenen Personen sollen Erfahrung auf dem Gebiet des Datenschutzes besitzen.

(3) Ein Mitglied ist aus dem Kreise der rechtskundigen Bundesbediensteten vorzuschlagen.

(3a) Die Mitglieder der Datenschutzkommission üben diese Funktion neben ihnen sonst obliegenden beruflichen Tätigkeiten aus.

(4) Für jedes Mitglied ist ein Ersatzmitglied zu bestellen. Das Ersatzmitglied tritt bei Verhinderung des Mitglieds an dessen Stelle. Die Funktionsperiode des Ersatzmitglieds endet mit der Funktionsperiode des Mitglieds; für den Fall der vorzeitigen Beendigung der Funktionsperiode des Mitglieds gilt Abs. 8.

(5) Der Datenschutzkommission können nicht angehören:

1. Mitglieder der Bundesregierung oder einer Landesregierung sowie Staatssekretäre;
2. Personen, die zum Nationalrat nicht wählbar sind.

(6) Hat ein Mitglied der Datenschutzkommission Einladungen zu drei aufeinanderfolgenden Sitzungen ohne genügende Entschuldigung keine Folge geleistet oder tritt bei einem Mitglied ein Ausschließungsgrund des Abs. 5 nachträglich ein, so hat dies nach seiner Anhörung die Datenschutzkommission festzustellen. Diese Feststellung hat den Verlust der Mitgliedschaft zur Folge. Im übrigen kann ein Mitglied der Datenschutzkommission nur aus einem schwerwiegenden Grund durch Beschluß der Datenschutzkommission, dem mindestens drei ihrer Mitglieder zustimmen müssen, seines Amtes für verlustig erklärt werden. Die Mitgliedschaft endet auch, wenn das Mitglied seine Funktion durch schriftliche Erklärung an den Bundeskanzler zurücklegt. Die Mitgliedschaft des richterlichen Mitglieds sowie des Mitglieds aus dem Kreis der rechtskundigen Bundesbediensteten endet auch, wenn diese aus ihren Dienstverhältnissen zum Bund ausscheiden, in den Ruhestand übertreten oder in den Ruhestand versetzt werden. Bei Richtern steht dem Ausscheiden eine Dienstzuteilung nach § 78 des Richter- und Staatsanwaltschaftsdienstgesetzes, BGBl. Nr. 305/1961, gleich. Die Mitgliedschaft der übrigen Mitglieder endet am 31. Dezember des Jahres, in dem sie das 65. Lebensjahr vollenden.

(7) Auf die Ersatzmitglieder sind die Abs. 2, 3, 5 und 6 wie auf Mitglieder anzuwenden.

(8) Scheidet ein Mitglied wegen Todes, freiwillig oder gemäß Abs. 6 vorzeitig aus, so wird das betreffende Ersatzmitglied (Abs. 4) Mitglied der Datenschutzkommission bis zum Ablauf der Funktionsperiode des ausgeschiedenen Mitglieds. Unter Anwendung der Abs. 2 und 3 ist für diese Zeit ein neues Ersatzmitglied zu bestellen. Scheidet ein Ersatzmitglied vorzeitig aus, ist unverzüglich ein neues Ersatzmitglied zu bestellen.

(9) Die Mitglieder und Ersatzmitglieder der Datenschutzkommission haben für die Anreise zu den Sitzungen der Datenschutzkommission sowie für in Ausübung ihrer Funktion erforderliche sonstige Dienstreisen Anspruch auf Ersatz der Reisekosten (Gebührenstufe 3) durch den Bundeskanzler nach Maßgabe der für Bundesbedienstete geltenden Rechtsvorschriften. Sie haben ferner Anspruch auf eine der Zeit und dem Arbeitsaufwand entsprechende Vergütung, die auf Antrag des Bundeskanzlers von der Bundesregierung durch Verordnung festzusetzen ist.

Art. 2 § 37 wurde geändert:

Weisungsfreiheit der Datenschutzkommission

§ 37. (1) Die Mitglieder der Datenschutzkommission sind in Ausübung ihres Amtes unabhängig und an keine Weisungen gebunden.

(2) Die ~~in der~~ Datenschutzkommission ist eine Dienstbehörde und Personalstelle. Zur Unterstützung der Datenschutzkommission ist eine Geschäftsstelle ~~der Datenschutzkommission tätigen Bediensteten unterstehen fachlich~~ eingerichtet. Im Bundesfinanzgesetz ist die notwendige Sach- und Personalausstattung sicherzustellen. Die Bediensteten der Geschäftsstelle unterstehen nur den Weisungen des Vorsitzenden ~~oder des geschäftsführenden Mitglieds der Datenschutzkommission~~der Datenschutzkommission. Der Vorsitzende der Datenschutzkommission übt die Diensthoheit über die Bediensteten in der Geschäftsstelle aus.

Art. 2 § 38 wurde geändert:

Stand 10.06.2010

Beachte: Abs. 1: Verfassungsbestimmung

Organisation und Geschäftsführung der Datenschutzkommission

§ 38.[I] (1) **(Verfassungsbestimmung)** Die Datenschutzkommission hat sich eine Geschäftsordnung zu geben, in der eines ihrer Mitglieder mit der Führung der laufenden Geschäfte zu betrauen ist (geschäftsführendes Mitglied). Diese Betrauung umfaßt auch die Erlassung von verfahrensrechtlichen Bescheiden und von Mandatsbescheiden im Registrierungsverfahren gemäß § 20 Abs. 2 oder § 22 Abs. 3. Inwieweit einzelne fachlich geeignete Bedienstete der Geschäftsstelle der Datenschutzkommission zum Handeln für die Datenschutzkommission oder das geschäftsführende Mitglied ermächtigt werden, bestimmt die Geschäftsordnung.

(2) Für die Unterstützung in der Geschäftsführung der Datenschutzkommission hat der Bundeskanzler eine Geschäftsstelle einzurichten und die notwendige Sach- und Personalausstattung bereitzustellen. Er hat das Recht, sich jederzeit über alle Gegenstände der Geschäftsführung der Datenschutzkommission beim Vorsitzenden und dem geschäftsführenden Mitglied zu unterrichten.

(3) Die Datenschutzkommission ist vor Erlassung von Verordnungen anzuhören, die auf der Grundlage dieses Bundesgesetzes ergehen oder sonst wesentliche Fragen des Datenschutzes unmittelbar betreffen.

(4) Die Datenschutzkommission hat spätestens alle zwei Jahre einen Bericht über ihre Tätigkeit zu erstellen und in geeigneter Weise zu veröffentlichen. Der Bericht ist dem Bundeskanzler zur Kenntnis zu übermitteln.

[I] Abs. 1: Verfassungsbestimmung

Stand 10.06.2013

Beachte: Abs. 1: Verfassungsbestimmung

Organisation und Geschäftsführung der Datenschutzkommission

§ 38.[I] (1) **(Verfassungsbestimmung)** Die Datenschutzkommission hat sich eine Geschäftsordnung zu geben, in der eines ihrer Mitglieder mit der Führung der laufenden Geschäfte zu betrauen ist (geschäftsführendes Mitglied). Diese Betrauung umfaßt auch die Erlassung von verfahrensrechtlichen Bescheiden und von Mandatsbescheiden im Registrierungsverfahren gemäß § 20 Abs. 2 oder § 22 Abs. 3. Inwieweit einzelne fachlich geeignete Bedienstete der Geschäftsstelle der Datenschutzkommission zum Handeln für die Datenschutzkommission oder das geschäftsführende Mitglied ermächtigt werden, bestimmt die Geschäftsordnung.

(2) Der Bundeskanzler kann sich beim Vorsitzenden der Datenschutzkommission über die Gegenstände der Geschäftsführung unterrichten. Dem ist vom Vorsitzenden der Datenschutzkommission nur insoweit zu entsprechen, als dies nicht der völligen Unabhängigkeit der Kontrollstelle im Sinne von Art. 28 Abs. 1 UAbs. 2 der Richtlinie 95/46/EG zum Schutz natürlicher Personen bei der Verarbeitung personenbezogener Daten und zum freien Datenverkehr, ABl. Nr. L 281 vom 23.11.1995 S. 31, widerspricht.

(3) Die Datenschutzkommission ist vor Erlassung von Verordnungen anzuhören, die auf der Grundlage dieses Bundesgesetzes ergehen oder sonst wesentliche Fragen des Datenschutzes unmittelbar betreffen.

(4) Die Datenschutzkommission hat spätestens alle zwei Jahre einen Bericht über ihre Tätigkeit zu erstellen und in geeigneter Weise zu veröffentlichen. Der Bericht ist dem Bundeskanzler zur Kenntnis zu übermitteln.

[I] Abs. 1: Verfassungsbestimmung

Art. 2 § 60 wurde geändert:

Beachte: Abs. 8: Verfassungsbestimmung

Inkrafttreten

§ 60.[II] (1) *(Anm.: Durch Art. 2 § 2 Abs. 1 Z 24 und Abs. 2 Z 71, BGBl. I Nr. 2/2008, als nicht mehr geltend festgestellt.)*

(2) Die übrigen Bestimmungen dieses Bundesgesetzes treten ebenfalls mit 1. Jänner 2000 in Kraft.

(3) §§ 26 Abs. 6 und 52 Abs. 1 und 2 in der Fassung des Bundesgesetzes BGBl. I Nr. 136/2001 treten mit 1. Jänner 2002 in Kraft.

(4) § 48a Abs. 5 in der Fassung des Bundesgesetzes BGBl. I Nr. 135/2009 tritt mit 1. Jänner 2010 in Kraft.

(5) Das Inhaltsverzeichnis, § 4 Abs. 1 Z 4, 5, 7 bis 9, 11 und 12, § 8 Abs. 1, 2 und 4, § 12 Abs. 1, die Umnummerierung der Absätze in § 13, § 16 Abs. 1 und 3, § 17 Abs. 1, 1a und 4, § 19 Abs. 1 Z 3a und Abs. 2, die Umnummerierung der Absätze in § 19, die §§ 20 bis 22a samt Überschriften, § 24 Abs. 2a, § 24 Abs. 4, § 26 Abs. 1 bis 8 und 10, § 28 Abs. 3, § 30 Abs. 2a, 5 bis 6a, die §§ 31 und 31a samt Überschriften, § 32 Abs. 1, 4, 6 und 7, § 34 Abs. 1, 3 und 4, § 36 Abs. 3, 3a und 9, § 39 Abs. 5, § 40 Abs. 1 und 2, § 41 Abs. 2 Z 4a, § 42 Abs. 1 Z 1, § 42 Abs. 5, § 46 Abs. 1 Z 2 und 3, Abs. 2 bis 3a, § 47 Abs. 4, § 49 Abs. 3, § 50 Abs. 1 bis 2a, der 9a. Abschnitt, § 51, § 52 Abs. 2 und 4, § 55, § 61 Abs. 6 bis 9 sowie § 64 in der Fassung des Bundesgesetzes BGBl. I Nr. 133/2009 treten mit 1. Jänner 2010 in Kraft. Gleichzeitig treten § 4 Abs. 1 Z 10, § 13 Abs. 3 sowie § 51 Abs. 2 außer Kraft.

(6) § 36 Abs. 6 in der Fassung des Bundesgesetzes BGBl. I Nr. 133/2009 tritt am 1. Juli 2010 in Kraft.

(6a) §37 Abs. 2, §38 Abs. 2 und §61 Abs. 9 in der Fassung des Bundesgesetzes BGBl. I Nr. 57/2013 treten mit 1. Mai 2013 in Kraft.

(7) Das Inhaltverzeichnis, § 5 Abs. 4, § 10 Abs. 2, § 12 Abs. 4, § 13 Abs. 1, 2 Z 2, Abs. 3, 4 und 6, § 16 Abs. 1, § 17 Abs. 1, § 18 Abs. 2, § 19 Abs. 1 Z 6 und Abs. 2, § 20 Abs. 2 und 5 Z 2, § 21 Abs. 1 Z 3, § 22 Abs. 2 bis 4, § 22a Abs. 1, 3 bis 5, § 23 Abs. 2, § 26 Abs. 2, 5 und 7, § 27 Abs. 5 und 7, die Überschrift zu § 30, § 30 Abs. 1, 2, 2a, 4 bis 6a, die Überschrift zu § 31, § 31 Abs. 1, 2, 5, 6 und 8, § 31a, § 32 Abs. 5 bis 7, § 34 Abs. 3 und 4, die Überschrift zu § 35, § 35 Abs. 1, §§ 36 bis 40 samt Überschriften, § 41 Abs. 2 Z 1, § 44 Abs. 6 und 8, § 46 Abs. 2 Z 3 und Abs. 3, § 47 Abs. 3 und 4, § 48a Abs. 2, § 50 Abs. 1 und 2, § 50b Abs. 2, § 50c Abs. 1, § 52 Abs. 2 Z 2 und 3 sowie Abs. 5, § 54 Abs. 2 und § 61 Abs. 8 bis 10 in der Fassung des Bundesgesetzes BGBl. I Nr. 83/2013 treten mit 1. Jänner 2014 in Kraft. Gleichzeitig treten § 41 Abs. 2 Z 4a und die DSK-Vergütungsverordnung, BGBl. II Nr. 145/2006, außer Kraft. Die für die Bestellung des Leiters der Datenschutzbehörde

[II] Abs. 8: Verfassungsbestimmung

und seines Stellvertreters notwendigen organisatorischen und personellen Maßnahmen können bereits vor Inkrafttreten des Bundesgesetzes BGBl. I Nr. 83/2013 getroffen werden.

(8) **(Verfassungsbestimmung)** § 2 Abs. 2 und § 35 Abs. 2 in der Fassung des Bundesgesetzes BGBl. I Nr. 83/2013 treten mit 1. Jänner 2014 in Kraft.

Art. 2 § 61 wurde geändert:

Beachte: Abs. 4: Verfassungsbestimmung

Übergangsbestimmungen

§ 61.[III] (1) Meldungen, die vor Inkrafttreten dieses Bundesgesetzes an das Datenverarbeitungsregister erstattet wurden, gelten als Meldungen im Sinne des § 17, soweit sie nicht im Hinblick auf das Entfallen von Meldepflichten gemäß § 17 Abs. 2 oder 3 gegenstandslos geworden sind. Desgleichen gelten vor Inkrafttreten dieses Bundesgesetzes durchgeführte Registrierungen als Registrierungen im Sinne des § 21.

(2) Soweit nach der neuen Rechtslage eine Genehmigung für die Übermittlung von Daten ins Ausland erforderlich ist, muß für Übermittlungen, für die eine Genehmigung vor Inkrafttreten dieses Bundesgesetzes erteilt wurde, eine Genehmigung vor dem 1. Jänner 2003 neu beantragt werden. Wird der Antrag rechtzeitig gestellt, dürfen solche Übermittlungen bis zur rechtskräftigen Entscheidung über den Genehmigungsantrag fortgeführt werden.

(3) Datenschutzverletzungen, die vor dem Inkrafttreten dieses Bundesgesetzes stattgefunden haben, sind, soweit es sich um die Feststellung der Rechtmäßigkeit oder Rechtswidrigkeit eines Sachverhalts handelt, nach der Rechtslage zum Zeitpunkt der Verwirklichung des Sachverhalts zu beurteilen; soweit es sich um die Verpflichtung zu einer Leistung oder Unterlassung handelt, ist die Rechtslage im Zeitpunkt der Entscheidung in erster Instanz zugrundezulegen. Ein strafbarer Tatbestand ist nach jener Rechtslage zu beurteilen, die für den Täter in ihrer Gesamtauswirkung günstiger ist; dies gilt auch für das Rechtsmittelverfahren.

(4) **(Verfassungsbestimmung)** Datenanwendungen, die für die in § 17 Abs. 3 genannten Zwecke notwendig sind, dürfen auch bei Fehlen einer im Sinne des § 1 Abs. 2 ausreichenden gesetzlichen Grundlage bis 31. Dezember 2007 vorgenommen werden, in den Fällen des § 17 Abs. 3 Z 1 bis 3 jedoch bis zur Erlassung von bundesgesetzlichen Regelungen über die Aufgaben und Befugnisse in diesen Bereichen.

(5) Manuelle Datenanwendungen, die gemäß § 58 der Meldepflicht unterliegen, sind, soweit sie schon im Zeitpunkt des Inkrafttretens dieses Bundesgesetzes bestanden haben, dem Datenverarbeitungsregister bis spätestens 1. Jänner 2003 zu melden. Dasselbe gilt für automationsunterstützte Datenanwendungen gemäß § 17 Abs. 3, für die durch die nunmehr geltende Rechtslage die Meldepflicht neu eingeführt wurde.

(6) Videoüberwachungen, die vor dem Inkrafttreten der §§ 50a bis 50e registriert wurden, bleiben in ihrer registrierten Form rechtmäßig, wenn sie den am 31. Dezember 2009 geltenden datenschutzrechtlichen Bestimmungen genügen und die Datenschutzkommission keine Befristung verfügt hat. Hat die Datenschutzkommission hingegen eine Befristung einer solchen Videoüberwachung verfügt, bleibt diese bis zum Ablauf der Befristung, längstens aber bis zum 31. Dezember 2012 rechtmäßig.

(7) Soweit in einzelnen Vorschriften Verweise auf das Datenschutzgesetz, BGBl. Nr. 565/1978, enthalten sind, gelten diese bis zu ihrer Anpassung an dieses Bundesgesetz sinngemäß weiter.

(8) Die Verordnung nach § 16 Abs. 3 ist vom Bundeskanzler nach Maßgabe der technischen Möglichkeiten des Datenverarbeitungsregisters bis spätestens 1. ~~Jänner~~ September 2012 neu zu erlassen. Bis zum Inkrafttreten dieser Verordnung sind die §§ 16 bis 22, § 30 Abs. 3 und 6 sowie § 40 Abs. 1 (letzterer mit Ausnahme des Verweises auf § 31a Abs. 3) in der Fassung vor dem Bundesgesetz BGBl. I Nr. 133/2009 anzuwenden; § 22a, § 30 Abs. 2a und 6a, § 31a Abs. 1 und 2 sowie § 32 Abs. 7 sind bis dahin nicht anzuwenden. § 31 Abs. 3 in der Fassung vor dem Bundesgesetz BGBl. I Nr. 133/2009 ist bis dahin zusätzlich weiter anzuwenden. Die Erklärung, ob eine Datenanwendung einen oder mehrere der in § 18 Abs. 2 Z 1 bis 4 genannten Tatbestände erfüllt (§ 19 Abs. 1 Z 3a), ist der Datenschutzkommission bei im Zeitpunkt des Inkrafttretens der neuen Verordnung nach § 16 Abs. 3 registrierten Datenanwendungen anlässlich der ersten über eine Streichung hinausgehenden Änderungsmeldung zu melden, die nach diesem Zeitpunkt erstattet wird. Eine Meldung allein im Hinblick auf § 19 Abs. 1 Z 3a ist nicht erforderlich.

(9) Bedienstete des Bundeskanzleramtes, die ausschließlich oder überwiegend Aufgaben besorgen, die in den Wirkungsbereich der Geschäftsstelle der Datenschutzkommission fallen, werden mit Inkrafttreten des BGBl. I Nr. 57/2013 als Bedienstete der Datenschutzkommission übernommen. Der Bundeskanzler hat mit Bescheid festzustellen, welche Beamten des abgebenden Bundeskanzleramtes ausschließlich oder überwiegend Aufgaben besorgen, die in den entsprechenden Wirkungsbereich der übernehmenden Datenschutzkommission fallen. Für vertraglich Bedienstete gilt dies mit der Maßgabe, dass an die Stelle des Bescheides eine Dienstgebererklärung tritt.

[III] Abs. 4: Verfassungsbestimmung

C E-Mail-Verkehr

C.1 Anfragen bzgl. des RIS-Webservice

C.1.1 Anfrage:

```
Sehr geehrte Damen und Herren,

ich entwickle gerade im Zuge meiner Masterarbeit an der Johannes Kepler Universität Linz eine Applikation, welche die Daten des RIS-Webservices verwendet.

Hierzu haben sich nun ein paar Fragen ergeben:

* Bei <binary>-Elementen innerhalb der Nutzdaten ist in der xsd lediglich folgendes angegeben: "Gibt den Binärtyp an: jpeg | tiff | pdf | ..." Gibt es eine taxative Liste der möglichen vorkommenden Typen?

* Gibt es eine dem "Bundesrecht konsolidiert" ähnliche Möglichkeit, automatisiert, also per Webservice, auf die Landesrechte zuzugreifen? Die Methoden "request" und "getDocument" würden dies ja prinzipiell mit der Variable "application" vorsehen. Wird in der iOS-RIS-App der Inhalt der Webseiten geparst, oder wird hier eine andere Möglichkeit genutzt?

* Gibt es eine Möglichkeit, zu Testzwecken alle Daten zu erhalten, die bei dem Webservice hinterlegt sind?

Vielen Dank im Voraus für Ihre Hilfe!

Beste Grüße
Josef Schaitl
```

Listing 9: E-Mail Anfrage vom 13.12.2012 an `ris.it@bka.gv.at`.

C.1.2 Antwort von Herrn Mag. Helmut Weichsel:

```
Sehr geehrter Herr Schaitl!

ad 1)
Anlagen zu Dokumenten können in folgenden Formaten verfügbar sein:
Ø  XML
Ø  RTF
Ø  DOC
Ø  DOCX
Ø  ODT
Ø  PDF
Ø  GIF
Ø  JPG
Ø  TIFF
Ø  TIF

Bei einem Hauptdokument sind folgende Dateiformate möglich: DOC, DOCX, PDF, XML

ad 2)
Nein, derzeit werden via Open Government Data nur die Daten der RIS-Anwendung
    „Bundesrecht konsolidiert“ angeboten. Für die iOS Variante der „RIS-App“ wird
    eine andere Schnittstelle verwendet.

ad 3)
Grundsätzlich besteht die Möglichkeit, auch anderer RIS-Anwendungen im Format XML -
     gegen Entgelt - auf zB DVD zu erhalten.

Mit freundlichen Grüßen

Helmut Weichsel

_____________________________________________
Mag. Helmut Weichsel

Bundeskanzleramt Österreich
e-Government - Programm- und Projektmanagement (Abt. I/13)
Ballhausplatz 2
1014 Wien
Tel.:       (+43 1) 53115/204211
Fax:       (+43 1) 53109/204211
E-Mail: helmut.weichsel@bka.gv.at
```

Listing 10: E-Mail Antwort vom 14.12.2012 von `helmut.weichsel@bka.gv.at`.

C.2 Nachfrage bzgl. weiterer RIS-Applikationen

C.2.1 Anfrage, Bezug nehmend auf C.1.2:

```
Sehr geehrter Herr Weichsel,

ad 2)
Ist diese Schnittstelle frei verfügbar? Wenn ja, wo erhalte ich hierzu
    Informationen? Wenn nein, warum nicht (Stichwort: Open Government)?

Besten Dank im Voraus!

MfG
Josef Schaitl
```

Listing 11: E-Mail Anfrage vom 14.12.2012 an `helmut.weichsel@bka.gv.at`.

C.2.2 Antwort von Herrn Mag. Helmut Weichsel:

```
Sehr geehrter Herr Schaitl!

Nein, diese Schnittstelle ist nicht frei verfügbar. Das Bundeskanzleramt
    beabsichtigt, im kommenden Jahr weitere RIS-Anwendungen im Rahmen von OGD zur
    Verfügung zu stellen.

Mit freundlichen Grüßen

Helmut Weichsel
```

Listing 12: E-Mail Antwort vom 14.12.2012 von `helmut.weichsel@bka.gv.at`.

Literaturverzeichnis

[BG13] Claudio Beccari und Enrico Gregorio. *The package imakeidx*. Handbuch zum Paket, Version 1.3. März 2013.

[BM04] Paul V. Biron und Ashok Malhotra. *XML Schema Part 2: Datatypes Second Edition*. W3C Recommendation. `http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/`. W3C, Okt. 2004.

[Boo+04] David Booth u. a. *Web Services Architecture*. W3C Note. `http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/`. W3C, Feb. 2004.

[Bos+11] Bert Bos u. a. *Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification*. W3C Recommendation. `http://www.w3.org/TR/2011/REC-CSS2-20110607`. W3C, Juni 2011.

[Bra08] Johannes Braams. *Babel, a multilingual package for use with LaTeX's standard document classes*. Handbuch zum Paket, Version 3.9e. Apr. 2008.

[BSM96] Tim Bray und C. M. Sperberg-McQueen. *Extensible Markup Language (XML)*. World Wide Web Consortium, Working Draft WD-xml-961114. Nov. 1996.

[Bra+08] Tim Bray u. a. *Extensible Markup Language (XML) 1.0 (Fifth Edition)*. W3C Recommendation. `http://www.w3.org/TR/2008/REC-xml-20081126`. W3C, Nov. 2008.

[Bra+09] Tim Bray u. a. *Namespaces in XML 1.0 (Third Edition)*. W3C Recommendation. `http://www.w3.org/TR/2009/REC-xml-names-20091208/`. W3C, Dez. 2009.

[BM13] Bundeskanzleramt und Mitglieder der Cooperation OGD Österreich. *Hintergrund-Infos | data.gv.at*. Online unter `http://data.gv.at/hintergrund-infos/` sowie direkte Unterseiten davon. Apr. 2013.

[Che11] Florent Chervet. *tabu and longtabu – Flexible LaTeX tabulars*. Handbuch zum Paket, Version 2.8. Feb. 2011.

[Chr+01] Erik Christensen u. a. *Web Services Description Language (WSDL) 1.1.* W3C Note. `http://www.w3.org/TR/wsdl`. W3C, März 2001.

[Cic50] Marcus Tullius Cicero. *De re publica – Über das Gemeinwesen.* Eigenverlag, ca. 50 v. Chr.

[Eib+12] Gregor Eibl u. a. *Rahmenbedingungen für Open Government Data Plattformen.* Dokumentversion 1.1.0. Juli 2012.

[FW04] David C. Fallside und Priscilla Walmsley. *XML Schema Part 0: Primer Second Edition.* W3C Recommendation. `http://www.w3.org/TR/2004/REC-xmlschema-0-20041028/`. W3C, Okt. 2004.

[Ges13] Gesetzbuch24.de. *Topaktuell – individuell – Gesetzbuch 24 – Textsammlungen nach Maß.* Online unter `http://www.gesetzbuch24.de/topaktuelle_individuelle_gesetze/`. 2013.

[Gud+07] Martin Gudgin u. a. *SOAP Version 1.2 Part 1: Messaging Framework (Second Edition).* W3C Recommendation. `http://www.w3.org/TR/2007/REC-soap12-part1-20070427/`. W3C, Apr. 2007.

[Har04] Elliotte Rusty Harold. *Effective XML.* Kapitelweise auch online verfügbar unter: `http://www.cafeconleche.org/books/effectivexml/`. Addison-Wesley, 2004.

[Hel04] Otto Hellwig. „EDV-Koordination im Bundeskanzleramt: Impulse von Dr. Winter, die noch heute im E-Government wirken“. In: *Von der Verwaltungsinformatik zum E-Government, Festschrift für Arthur Winter zum 60. Geburtstag.* Hrsg. von Robert Traunmüller u. a. Wien: ADV Handelsgesellschaft m.b.H., 2004, S. 123–132.

[JRH99] Ian Jacobs, David Raggett und Arnaud Le Hors. *HTML 4.01 Specification.* W3C Recommendation. `http://www.w3.org/TR/1999/REC-html401-19991224`. W3C, Dez. 1999.

[Jen+13] Eric Jendrock u. a. *The Java EE 6 Tutorial.* Online verfügbar unter: `http://docs.oracle.com/javaee/6/tutorial/doc/index.html` und den verlinkten Unterseiten. Jän. 2013.

[Knu98] Donald E. Knuth. *Digital Typography. CSLI Lecture Notes Number 78.* Stanford, CA, USA: CSLI Publications, 1998.

[Knu68] Donald E. Knuth. *The Art of Computer Programming, Volume 1: Fundamental Algorithms.* Reading, MA, USA: Addison-Wesley Publishing Company, 1968.

[Knu69] Donald E. Knuth. *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*. Reading, MA, USA: Addison-Wesley Publishing Company, 1969.

[Knu73] Donald E. Knuth. *The Art of Computer Programming, Volume 3: Sorting and Searching*. Reading, MA, USA: Addison-Wesley Publishing Company, 1973.

[Knu11] Donald E. Knuth. *The Art of Computer Programming, Volume 4: Combinatorial Algorithms*. Reading, MA, USA: Addison-Wesley Publishing Company, 2011.

[Knu84] Donald E. Knuth. *The TEXbook*. Reading, MA, USA: Addison-Wesley Publishing Company, 1984.

[KM11] Markus Kohm und Jens-Uwe Morawski. *KOMA-Script – Die Anleitung*. Handbuch zum Paket, Version 3. Apr. 2011.

[LS04] Friedrich Lachmayer und Helga Stöger. „Österreichs Weg zur authentischen elektronischen Publikation – Zur Projektgeschichte des RIS“. In: *Von der Verwaltungsinformatik zum E-Government, Festschrift für Arthur Winter zum 60. Geburtstag*. Hrsg. von Robert Traunmüller u. a. Wien: ADV Handelsgesellschaft m.b.H., 2004, S. 133–142.

[Lam13] Leslie Lamport. *The Writings of Leslie Lamport: LATEX: A Document Preparation System*. Online unter `http://research.microsoft.com/en-us/um/people/lamport/pubs/pubs.html#latex`. März 2013.

[LZ00] Leslie Lamport und Günther M. Ziegler. „How (LA)TEX changed the face of Mathematics“. In: *Mitteilungen der Deutschen Mathematiker-Vereinigung* 1 (2000), S. 49–51.

[Lev66] Vladimir I. Levenshtein. „Binary codes capable of correcting deletions, insertions, and reversals“. In: *Soviet Physics Doklady* 10.8 (1966), S. 707–710.

[MM99] Ashok Malhotra und Murray Maloney. *XML Schema Requirements*. W3C Note. `http://www.w3.org/TR/1999/NOTE-xml-schema-req-19990215`. W3C, Feb. 1999.

[ML07] Nilo Mitra und Yves Lafon. *SOAP Version 1.2 Part 0: Primer (Second Edition)*. W3C Recommendation. `http://www.w3.org/TR/2007/REC-soap12-part0-20070427/`. W3C, Apr. 2007.

[Mit11] Frank Mittelbach. *multicol – An environment for multicolumn output*. Handbuch zum Paket, Version 1.7a. Juni 2011.

[Mor+07] Jean-Jacques Moreau u. a. *SOAP Version 1.2 Part 2: Adjuncts (Second Edition)*. W3C Recommendation. `http://www.w3.org/TR/2007/REC-soap12-part2-20070427/`. W3C, Apr. 2007.

[Nak12] Hiroshi Nakashima. *Package paracol: Yet Another Multi-Column Package to Typeset Columns in Parallel*. Handbuch zum Paket, Version 1.1. Mai 2012.

[Öst12] Bundekanzleramt Österreich. *RIS-OGDServices*. Handbuch. `http://data.bka.gv.at/RIS/Documents/RIS_OGD_Dokumentation.pdf`. Republik Österreich – Bundeskanzleramt, Okt. 2012.

[Öst13] Bundeskanzleramt Österreich. *Bundeskanzleramt RIS Informationsangebote – Bundesrecht*. Online unter `http://www.ris.bka.gv.at/Bund/`. Apr. 2013.

[Par03] Parlamentsdirektion. *Stenographisches Protokoll der 35. Sitzung des Nationalrates der Republik Österreich, XXII. Gesetzgebungsperiode*. Okt. 2003.

[Pem02] Steven Pemberton. *XHTML™ 1.0 The Extensible HyperText Markup Language (Second Edition)*. W3C Recommendation. `http://www.w3.org/TR/2002/REC-xhtml1-20020801`. W3C, Aug. 2002.

[RFC4648] S. Josefsson. *The Base16, Base32, and Base64 Data Encodings*. RFC 4648 (Proposed Standard). Internet Engineering Task Force, Okt. 2006. URL: `http://www.ietf.org/rfc/rfc4648.txt`.

[Sch12] Martin Scharrer. *The adjustbox Package*. Handbuch zum Paket, Version 1.0. Mai 2012.

[SO09] Stack Overflow User ‚aximili'. *Recommended website resolution (width and height)?* Online unter `http://stackoverflow.com/questions/456250/recommended-website-resolution-width-and-height`. Jän. 2009.

[SMW94] Werner Robert Svoboda, Andreas Manak und Werner Weinguny. *Elektronische Rechtsinformation in Österreich – Alle Juristischen Datenbanken im Überblick*. Bd. 71. Schriftenreihe Österreichische Computer Gesellschaft. Wien München: R. Oldenbourg, 1994.

[Tho+04] Henry S. Thompson u. a. *XML Schema Part 1: Structures Second Edition*. W3C Recommendation. `http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/`. W3C, Okt. 2004.

Josef K. Schaitl

Lebenslauf

Rechte Donaustraße 1/2
4020 Linz
✉ *josef.schaitl@gmail.com*

Ausbildung

seit 04/11	**Master Informatik**, *Johannes Kepler Universität (JKU)*, Linz. Hauptfach: Netzwerke und Sicherheit Nebenfach: Pervasive Computing
Masterarbeit	OpenCodex – Erzeugung individualisierter Gesetzbücher aus den Daten des RIS
10/07–03/11	**Bachelor Informatik**, *JKU*, Linz. Abschluss: Bachelor of Science
10/03–09/07	**Diplomstudium Mechatronik**, *JKU*, Linz.
10/94–06/03	**Abitur**, *Gymnasium Pfarrkirchen*, Bayern. Mathematisch-naturwissenschaftlicher Zweig Schwerpunkte: Englisch & Physik, Regelschuldauer: 9 Jahre

Berufliche Erfahrungen

04/09–11/12 **Teammitglied Karrieretag**, *Kepler Society, JKU*, Linz.

- Planung der Karrieremesse auf der JKU
- Aktualisierung und Adaption der Homepage
- Hauptverantwortlicher für Aufbau und Technik

02/10–09/10 **Systemadministrator**, *Abteilung für Wirtschaftspädagogik, JKU*, Linz.

- Betreuung der IT des Schulungsraumes
- Wartung des Abteilungsservers und der Druckerinfrastruktur
- Auswahl, Anschaffung und Inbetriebnahme neuer Hardware
- EDV-Support für die Mitarbeiter der Abteilung

10/05–03/11 **Systemadministrator**, *KHG-Heim*, Linz.

- Leitung des selbstverantwortlichen Administratorenteams
- Installation und Administration von Linux Mail-, Web-, File- und Print-Servern
- Aufbau und Pflege einer LDAP-Benutzerdatenbank
- Wartung der Netzwerkinfrastruktur
- Support für Bewohner und Heimleitung

11/00–09/06 **Verkäufer Einzelhandel**, *Expert Theiner*, Pfarrkirchen, Bayern.

- Kundenberatung und Verkauf
- Servicemassnahmen

Sprachen

Deutsch	**Muttersprache**
Englisch	**Fließend**
Französisch	**Grundkenntnisse**

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Masterarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe.

Die vorliegende Masterarbeit ist mit dem elektronisch übermittelten Textdokument identisch.

Linz, 22. Juli 2013 _____

Josef K. Schaitl