HOCHSCHULE
MITTWEIDA
UNIVERSITY OF
APPLIED SCIENCES

BACHELORARBEIT

Herr
Tobias Ussath

Konzeption, Entwurf und prototypische Implementierung einer Software zur Haltung und Organisation von Datensätzen des OpenImmo Standards

Mittweida, 2013

Fakultät **Mathematik/Naturwissenschaften/Informatik**

BACHELORARBEIT

Konzeption, Entwurf und prototypische Implementierung einer Software zur Haltung und Organisation von Datensätzen des OpenImmo Standards

Autor:
Tobias Ussath

Studiengang:
Informatik

Seminargruppe:
IF09w1-B

Erstprüfer:
Prof. Dr.-Ing. Wilfried Schubert

Zweitprüfer:
Prof. Dr.-Ing. Mario Geißler

Einreichung:
Mittweida, 02.08.2013

Verteidigung/Bewertung:
Mittweida, 2013

Bibliografische Angaben

Ussath, Tobias: Konzeption, Entwurf und prototypische Implementierung einer Software zur Haltung und Organisation von Datensätzen des OpenImmo Standards, **??** Seiten, 8 Abbildungen, Hochschule Mittweida, University of Applied Sciences, Fakultät Mathematik/Naturwissenschaften/Informatik

Bachelorarbeit, 2013

Referat

Die Bachelorarbeit verfolgt das Ziel der Erstellung einer zweckmäßigen und updatefähigen Software zum Umgang und zur Verwaltung von Datensätzen des OpenImmo Datenstandards. Der Datenstandard auf Basis der XML-Technologie wird bzgl. des Aufbaus, der Validierung sowie der generellen als auch spezifischen Funktionsweise aufgeführt und erläutert. Des Weiteren erfolgt die Konzeption, der Entwurf und die Implementierung des Prototyps einer möglichen Softwareumsetzung. Die Umsetzung wird analysiert, bewertet und gegenüber anderen Umsetzungsmöglichkeiten verglichen.

I. Inhaltsverzeichnis

II. Abbildungs- und Tabellenverzeichnis

Abbildungen

Tabellen

1 Einleitung

Seit der Einführung des Internets vor knapp 25 Jahren hat es die Art und Weise, wie wir Menschen unseren Alltag gestalten, immer stärker geprägt.

Es ermöglicht die Kommunikation und den Austausch von Daten sowohl zwischen den Menschen als auch den Maschinen über den gesamten Globus. Um diesen Austausch realisieren zu können und zu verbessern, ist es notwendig Protokolle und Übertragungsdaten zu standardisieren. Diese Standards haben mittlerweile einen entscheidenden Einfluss auf viele Bereiche des Lebens und den verschiedenen Branchen der Wirtschaft. Erst eine zweckmäßige und etablierte Standardisierung ermöglicht einen störungsfreien und produktive Austausch.

Ein im Jahr 2001 in der Zeitschrift *Der Immobilienprofi* publizierter Artikel vom Vorstandsvorsitzenden des *Zentrum für interaktive Medien* ZIM e.V., Frank Bitzer, sollte die Entwicklung der Immobilienwirtschaft in Deutschland in den kommenden Jahren nachhaltig verändern. Der Artikel thematisierte die Konzeption eines, auf der XML-Technologie[1] basierenden, Standards zum Austausch von Immobiliendatensätzen. Er hatte zur Folge, dass nur wenige Monate später 30 bedeutende Firmen aus der Software- und Immobilienbranche zu einem Treffen zusammenkamen. Teilnehmer dieser ersten Begegnung waren unter anderem Vertreter der FlowFact AG, Immobilien Scout GmbH und des Verbands der norddeutschen Wohnungswirtschaft. Sie diskutierten die Möglichkeiten, der Umsetzung des von Frank Bitzer erstellten Konzepts.

Dieses Treffen war die Geburtsstunde des *Vereins zur Förderung des Datenaustausches in der Immobilien Wirtschaft* e.V., kurz OpenImmo e.V.. Der Vorsitzende ist seit der Gründung der Vorreiter und Initiator Frank Bitzer.

Die Hauptaufgabe des Vereins liegt seither in der kontinuierlichen Entwicklung des seit dem ersten Treffen angestrebten Datenstandards zur Übertragung aller relevanten Kenndaten eines Immobilienobjekts - dem OpenImmo Standard.

Bereits wenige Jahre nach dem ersten Gedankenaustausch, wurde im Herbst 2003 die Version 1.0 des Standards auf Basis von XML veröffentlicht und bereitgestellt.

Abbildung 1.1: OpenImmo Logo, Quelle: [OI-2013]

[1] XML ist eine Auszeichnungssprache

Die bis heute ständig steigenden Mitgliederzahlen des Vereins, Schnittstellenentwicklungen und Softwareintegrationen des Formats zeugen von dessen Erfolg.

> *"Rund 5500 Downloads der Beschreibung zeigen den Erfolg der Idee (Stand Dezember 2012) Der Verein zählt heute über 36 Fördermitglieder. Mehrere hundert Firmen setzten OpenImmo ein."* [OI-2013]

Der Autor dieser Bachelorarbeit hat die Aufgabe eine Software zur Verwaltung von Immobiliendaten des OpenImmo Formates zu konzipieren und zu entwerfen, sowie eine prototypische Implementierung dieser beizugeben. Die Aufgabenstellung wurde zusammen mit dem Praxispartner, Clicks Online Business in Dresden, erstellt, bei dem der Autor zuvor das Pflichtpraktikum des Studienganges Informatik absolvierte.

Das Unternehmen Clicks Online Business (im Folgenden Praxispartner) wurde 2007 von Herbert Buchhorn gegründet. Die Online Marketing Agentur mit den Schwerpunkten Suchmaschinenoptimierung (SEO) und Suchmaschinenmarketing (SEM) beschäftigt sich seither vornehmlich mit der Gewinnung von potentiellen Kunden für Online-Portale und -shops, durch die Optimierung der Webseiten und gezielte Schaltung von Werbung in Suchmaschinen und sozialen Medien. Zudem liegt die individuelle Erstellung von Webauftritten und Onlineshops im Leistungsspektrum der Firma. [USSATH-2013]

Auf Wunsch des Unternehmens wurde der Prototyp auf Basis des PHP[2] Frameworks[3] Zend Framework 2 (ZF2) entwickelt. Informationen zu diesem Framework können in der Quelle [ZF2-2013], der Entwicklerseite, eingeholt werden.

Während der Arbeit hat sich der Autor die Vorteile der XML-Technologie zu Nutzen gemacht und das Problem der Datenhaltung und Bereitstellung auf eine sehr zweckmäßige Art und Weise ausgearbeitet und umgesetzt. Er wird auf bestehende Vor- und Nachteile der resultierenden Umsetzung eingehen und diese fachlich bewerten.

Zum Zeitpunkt dieser Arbeit befindet sich der OpenImmo Datenstandard in der Version 1.2.5.

1.1 Inhalt und Ziel der Arbeit

In der vorliegenden Bachelorarbeit soll der OpenImmo Datenstandard bzw. das OpenImmo Datenformat vorgestellt und genauer betrachtet werden. Dabei verfolgt jede Überlegung das Ziel der Entwicklung einer möglichste effizienten, zweckmäßigen und

[2] PHP ist eine Programmiersprache, eine serverseitige Skriptsprache

[3] ein Framework ist ein Programmgerüst das den Rahmen/die Rahmenbedingungen (in Form von Klassen, Bibliotheken, u.a.) zur Entwicklung einer Anwendung zur Verfügung stellt

zukunftssicheren Software zur Verwaltung von Datensätzen dieses laut [OI-2013] noch recht jungen Datenstandards.

Es soll Abstand von einer starren Datenhaltungsstruktur genommen werden und eine Möglichkeit zur flexiblen und updatesicheren[4] Verwaltung dieser Datensätze geschaffen werden.

Die Arbeit wird sich strukturiert der Implementierung des Prototyps annähern. Zu Beginn wird der OpenImmo Datensatz vorgestellt und erläutert. Um einen Einblick in den Datenstandard zu erhalten, werden seine generelle Funktionsweise und Datenstruktur kurz dargestellt.

1.2 Idee und Motivation

Die Idee zur Erstellung und Erarbeitung des Prototyps entstand aus Interesse des Praxispartners an der Entwicklung einer entsprechenden Software für diesen Einsatzbereich. Zusammen mit diesem sowie in Absprache und Konsultation mit dem betreuenden Professor der Hochschule Mittweida, Dr.-Ing. Wilfried Schubert, wurde das Thema der Bachelorarbeit formuliert.

Die Umsetzung entspricht nicht der konventionellen Entwicklungsstrategie[5]. Dies spiegelt sich vor allem in der außergewöhnlichen Art der Datenstruktur wider. Die Motivation zu diesem Vorgehen, entstand bei der anfänglichen Recherche des Autors zu dem Thema. Es stellte sich heraus, dass derzeit die meisten Immobilienportale ihr Übersichts- und Detailangebot der Objekte auf eine bestimmte Anzahl von Werten beschränken. Diese Werte stellen die, für die meisten Nutzer interessanten Objektdaten dar. Die Gesamtübersicht aller Informationen erhält der Benutzer meist nur über ein Exposé, welches in Form einer PDF Datei vorliegt. Dies ist sehr zweckmäßig, da somit die Möglichkeit besteht, das Exposé zu speichern oder zu drucken.

Das Ergebnis der Recherche motivierte den Autor als Entwickler, den zu entwickelnden Prototypen hingehend des Zugriffs auf diese beschränkte Anzahl von *wichtigen* Daten maximal zu optimieren. Diese Idee wurde in den Ansätzen anhand der Entwicklung des Prototypen dargelegt.

[4] hier: updatesicher in Bezug auf Neuerscheinungen bzw. Überarbeitungen des OpenImmo Standards. Funktionalität der Software soll in diesem Fall weiterhin bestehen

[5] die hier als *konventionelle Entwicklungsstrategie* bezeichnete Umsetzung wird in Kapitel 3.1 genauer erläutert.

2 Der OpenImmo Datenstandard

2.1 Anwendungsbereich

Der OpenImmo Standard ist ein Datenstandard zur Übertragung von Immobiliendaten. Sein Einsatzbereich ist folglich in den Transport von Daten zwischen verschiedenen Systemen oder Softwareprodukten einzuordnen.

Das Format wird meist als Output der Exportschnittstelle einer Software produziert bzw. von der Importschnittstelle einer Software interpretiert.

Softwareprodukte die den Standard unterstützen sind dabei oft Immobilienportale oder Marklerprogramme.

Der OpenImmo e.V. beschreibt die Vorteile und den Einsatz seines Standards wie folgt.

> *„**Bedienung mehrere Portale***
> *Die Übertragung an mehrere Portale und die anschließende Verwaltung der Objekte wird wesentlich flexibler und einfacher. Die Softwarefirmen können mit einem Datenstandard alle OpenImmo kompatiblen Portale beliefern.*
>
> ***Leichterer Zugang***
> *Da immer mehr Portale den OpenImmo Standard importieren [sic!] kann ein Immobilien Vermittler sich leichter einem Nationalen und einem Lokalen Portale bedienen. Auch die Übergabe der Daten an Zeitungen kann damit erleichtert werden.*
>
> ***Kostenersparnis***
> *OpenImmo erlaubt es Ihnen, auch ohne Service Portale auf einfache Weise mehrere Online Börsen zu bedienen.*
>
> ***Bessere Verwaltung (Löschen, Aktualisieren)***
> *Durch ein modernes Protokoll erlaubt OpenImmo den Immobilien Vermittlern eine wesentlich bessere und fehlerfreie Verwaltung ihrer Objekt[!sic] in den einzelnen Portalen. In der Vergangenheit kam es durch die Kooperationen immer wieder zu 'Datenleichen'.*
>
> ***Beim Einsatz von Standard Office Paketen***
> *Oft werden die Objekte mit gängiger Office Software verwaltet. Sei es Textverarbeitung, Tabellenkalkulation oder Datenbank. Für viele ist dies der Einstieg in die weitergehende Datenverarbeitung.[sic!] OpenImmo erlaubt ihnen auch hier schon den Einsatz des neuen Standards, in dem die Daten in XML exportiert werden, und anschließend konvertiert werden können. Ein*

Umstieg auf eine der OpenImmo kompatiblen Immobilien Vermittler Softwarepakete kann damit erleichtert werden.

Zukunftssicherheit

Die gesamte Software und Internet Branche setzt sukzessive auf OpenImmo. So sollten auch Sie es tun. Bei der Auswahl von Software und Online Vermarktungsportalen bieten Ihnen OpenImmo fähige Partner hohe Investitionssicherheit für die Zukunft.“

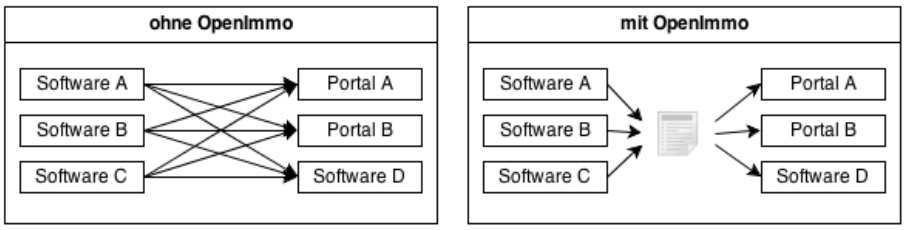
[OI-2013, http://www.openimmo.de/cm_std_info.htm]

Die steigende Anzahl der unterstützenden Softwareprodukte erhöht dabei zeitgleich den Anreiz der Entwickler, bei Softwareneuentwicklungen oder Anpassungen, eine Unterstützung für dieses Format zu realisieren.

Daraus resultierend könnte sich der Standard noch stärker etablieren und zu einem wesentlichen Kriterium bei Softwareprodukten dieser Branche werden, was zur weiteren Festigung der Stellung des Standards in diesem Bereich führen würde. (vgl. [BITZER18-2001])

Wie in [OIINFO-2013] beschrieben, konnte sich so der Datenaustausch in der Immobilienbranche in Zukunft komfortabler und bedeutend effizienter gestalten. Die Abbildung 2.1 nach [OIINFO-2013, S. 2] zeigt die Bedeutung der Standardisierung in einer treffenden Schematisierung.

Abbildung 2.1: Bedeutung des OpenImmo Standards, Quelle: Tobias Ussath nach [OIINFO-2013, S. 2]



2.2 Generelle Funktionsweise

Das OpenImmo Datenformat basiert auf der XML-Technologie. Extensible Markup Language, kurz XML, ist eine Beschreibungssprache zur Darstellung bzw. Haltung von Daten in Textform. Die Darstellung erfolgt dabei in klar strukturierter hierarchischer Form. Ausgehend von dem Wurzel- oder Hauptelement werden alle Datenbereiche und -felder als Tags bzw. Elemente oder Knoten dargestellt. Diesen Elemente können zusätzlich spezielle Eigenschaften mit Hilfe von Attributen zugeordnet werden.

Dabei sind nach [FALLSIDE-2001] Elemente, die Subelemente enthalten oder zusätzliche Attribute tragen, so genannte komplexe Typen (*complexTypes*), während Elemente,

die einen bestimmten Wert eines bestimmten Datentyps, z.B. eine Zeichenkette, eine Zahl oder einen Zeitpunkt enthalten, einfache Typen (*simpleTypes*) sind.

Das Prinzip ermöglicht die Darstellung nahezu jedes möglichen Modells oder Gegenstandes in ein solches Datenmodell. (vgl. [FALLSIDE-2001])

In Listing 2.1 wurde eine solche XML-Struktur zur Veranschaulichung anhand einer einfachen Bestandsführung eines Autohauses aufgestellt. Das Wurzelelement ist in diesem Fall das Element *autobestand*. Der *autobestand* besteht aus einer variablen Anzahl von Autos die jeweils in einem Element *auto*, einem Sub- oder Kindelement des *autobestand* repräsentiert und definiert werden.

Jedes *auto* enthält ein Attribut *kennzeichen* sowie zwei weitere Elemente *hersteller* und *modell*.

Listing 2.1: XML Struktur, Beispiel Autohaus

```
<autobestand>
        <auto kennzeichen="XX-XX-01">
                <hersteller>BMW</hersteller>
                <modell>X3</modell>
        </auto>
        <auto kennzeichen="XX-XX-02">
                <hersteller>Audi</hersteller>
                <modell>A4</modell>
        </auto>
</autobestand>
```

Die Vorteile dieser strukturierten Darstellung und der damit verbundene komfortable Umgang mit dieser Technologie führten dazu, dass XML heute den Kern zahlreicher Datenstandards, wie auch den des OpenImmo Standards ausmacht.

Da die bloße Aufbereitung der Daten in eine Baumstruktur kein Format definiert, bietet die XML-Technologie noch zahlreiche Methoden zur Validierung der erstellten Struktur. Diese Validierung macht den eigentlichen Standard aus. Sie enthält die Vorschriften zum Aufbau der Struktur als auch die Definitionen und Deklarationen der einzelnen Datenfelder (den Elementen und Attributen).

Eine Möglichkeit der Gültigkeitsprüfung eines Datenbestandes ist die Validierung gegen ein XML-Schema. Im Falle der Bestandsführung des Autohauses (Listing 2.1) wäre beispielsweise ein folgendes XML Schema denkbar.

Listing 2.2: XML-Schema, Beispiel Autohaus

```
<xsd:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="autobestand">
    <xsd:complexType>
      <xsd:sequence>
```

```
5         <xsd:element name="auto" maxOccurs="unbounded" minOccurs
              ="0">
6           <xsd:complexType>
7             <xsd:sequence>
8               <xsd:element type="xs:string" name="hersteller"/>
9               <xsd:element type="xs:string" name="modell"/>
10            </xsd:sequence>
11            <xsd:attribute type="xs:string" name="kennzeichen" use
                  ="required"/>
12          </xsd:complexType>
13        </xsd:element>
14      </xsd:sequence>
15    </xsd:complexType>
16  </xsd:element>
17 </xsd:schema>
```

Wird der Autobestand aus Listing 2.1 gegen das hier aufgeführte Schema (Listing 2.2) validiert, wird geprüft, ob die XML-Struktur in der Wurzel *autobestand* beginnt. Dieser *autobestand* darf laut Vorschrift keinen (*minOccurs="0"*) oder beliebig viele (*maxOccurs="unbounded"*) direkte Unterknoten *auto* enthalten. Ein *auto* muss aus den Elementen *hersteller* und *modell*, sowie dem Attribut *kennzeichen* bestehen. Dabei dürfen, wenn Werte vorhanden sind, sich in den Kindknoten *hersteller* und *modell* nur Werte vom Typ *string* befinden. Das Attribut *kennzeichen* ist ein Pflichtfeld (*use="required"*) und muss durch einen Wert vom Typ *string* gesetzt sein.

Dadurch kann über ein solches XML-Schema ein Datenstandard genauestens definiert werden. Es lässt sich schnell überprüfen, ob ein Datenbestand gültig ist und der Erwartung entspricht oder nicht.

Im Falle des OpenImmo Standards kommt ein eben solches XML-Schema zur Definition des Standards zum Einsatz.

2.3 Datenstruktur

Die XML-Struktur der OpenImmo Datensätze ist bedeutend komplexer, als das zuvor aufgeführte Beispiel. Die im Folgenden aufgeführten Erläuterungen zur OpenImmo Datenstruktur, können an dem Inhalt der Beispieldatei der OpenImmo Version 1.2.5 nachvollzogen werden. Einen groben Überblick über eine solche Datendatei soll das Listing 2.3 verschaffen.

Listing 2.3: OpenImmo Version 1.2.5, Datenstruktur *grob*

```
1 <openimmo>
2   <uebertragung art="ONLINE" umfang="VOLL" modus="NEW" version
        ="1.25" sendersoftware="" senderversion="" techn_email=""
        timestamp="2011-11-07T10:00:00" />
```

```
3   <anbieter>
4           <anbieternr />
5           <firma />
6           <openimmo_anid />
7           [...]
8           <immobilie>
9                   <objektkategorie>
10                          <nutzungsart WOHNEN="true" GEWERBE="false"
                                ANLAGE="false" WAZ="false" />
11                          <vermarktungsart KAUF="true" MIETE_PACHT="
                                false" ERBPACHT="false" LEASING="false"
                                />
12                          [...]
13                  </objektkategorie>
14                  [...]
15                  <verwaltung_techn>
16                          <objektnr_intern />
17                          <objektnr_extern />
18                          <openimmo_obid>111</openimmo_obid>
19                          [...]
20                  </verwaltung_techn>
21                  [...]
22          </immobilie>
23          <immobilie>
24                  [...]
25          </immobilie>
26          <impressum />
27          [...]
28  </anbieter>
29  <anbieter>
30        [...]
31  </anbieter>
32 </openimmo>
```

In der aktueller Version lassen sich sämtliche Datenübertragungsvorgänge in der Immobilienbranche abbilden. Es wird hierfür aktuell vom OpenImmo e.V. jeweils eine XML-Schemadatei zur Validierung der Immobilienstammdaten-Übertragung und eine Immobilien-Feedback-Übertragung zur Verfügung gestellt. Im Folgenden wird dieser Sachverhalt ausführlicher betrachtet.

Stammdaten von Immobilien übertragen

Mit Hilfe des OpenImmo Standards lassen sich Daten aller Objekte der Immobilienbranche austauschen. Dies sind neben den klassischen Gebäudeimmobilien u.a. auch Liegenschaften oder Garagen. Alle diese Objekte können anhand ihrer Stammdaten eindeutig in einem OpenImmo Stammdatensatz in einer OpenImmo Datenübertragungsdatei abgebildet werden.

In einer OpenImmo Datenübertragungsdatei lassen sich ein oder mehrere dieser Objekte eines oder verschiedener Anbieter zur Übertragung oder Haltung speichern.

Jede Stammdatendatei beinhaltet neben dem notwendigen Hauptknoten *openimmo* an erster Stelle ein Element zur Kategorisierung der aktuellen Übertragung. Dieses Subelement *uebertragung* beinhaltet keinen Wert, sondern enthält lediglich Eigenschaften in Form von Attributen. Die Attribute beinhalten alle Information zur jeweiligen Datei bzw. Übertragung (siehe Tabelle: 2.1). Diese Dateiinformationen dienen der korrekten Interpretation der Datei beim Import und sollten daher komplett befüllt sein. Auch die Informationen zur Software, *sendersoftware* und *senderversion*, sollten möglichst gesetzt werden. Andere Softwareprodukte haben somit die Möglichkeit Besonderheiten eines spezifischen Programms bei der Verarbeitung der Datei zu berücksichtigen.

Tabelle 2.1: Information über eine OpenImmo Datei (Attribute des Elements *uebertragung*)

Attribut	**Beschreibung**
art	Art der Übertragung. 2 mögliche Werte: ONLINE: Die Übertragung findet online statt, es wird eine Antwort der Empfängersoftware erwartet. OFFLINE: Die Datei wurde exportiert und wird manuell importiert.
umfang	2 mögliche Werte: VOLL: Kompletter Datensatz TEIL: Nur zu überschreibende Daten werden übertragen.
modus	3 mögliche Werte: NEW: neue Datensätze CHANGE: Datensätze sind zu ändern DELETE: löschen
version	Die OpenImmo Version, nach der die Datei erstellt wurde und durch welche sie validiert werden soll.
sendersoftware	Name der Software durch welche die Datei generiert wurde
senderversion	Version der Software
techn_mail	Kontaktmailadresse des Supports bzw. Entwicklers der erstellenden Software
timestamp	genauer Zeitpunkt der Dateierstellung

Die Aufführung der zu übertragenden Stammdaten erfolgt unmittelbar nach dem Element *uebertragung*.

Die Objekte bzw. Immobilien (in der OpenImmo XML-Struktur die Elemente mit dem Namen *immobilie*) werden immer als Kindelement einem Element *anbieter* zugeordnet. Dies hat den Grund, dass in der Regel jedem Objekt mindestens ein Anbieter, das heißt Markler, Verkäufer oder Inhaber, zugeordnet ist.

Ein Element *immobilie* enthält die Stammdaten des Immobilienobjektes. Diese Daten sind alle möglichen Daten, Definitionen und Kategorisierungen einer Immobilie. Neben

der geographischen Lage, bestehen hier beispielsweise ebenfalls die Möglichkeiten zur Angabe des Interieurs oder der Position des Objekts in einem größeren Gebäude oder auf einem Grundstück. Diese Struktur verdeutlicht das praxisnahe Konzept.

Die Informationen zur technischen Verarbeitung der Immobilie als Datenobjekt des OpenImmo Standards befinden sich im Kindknoten *techn_verwaltung* des jeweiligen Elements *immobilie*. Hier sind vor allem die Werte für OpenImmo Objekt-ID (Element *openimmo_obid*), die Objektnummer extern (*Element objektnr_extern*) und Objektnummer intern (Element *objektnr_intern*) von großer Bedeutung.

Die OpenImmo Objekt-ID identifiziert die Immobilie eindeutig innerhalb aller Immobilien-Softwareprodukte bzw. Immobilienportalen. Die interne und externe Objektnummer dienen zur internen Verwaltung.

Das Element *anbieter* besitzt neben den einzelnen Immobilien noch Elemente, um die Eigenschaften des Anbieters zu setzen. Diese Eigenschaften bzw. Daten zu dem Anbieter sind im Allgemeinen die Daten zur Spezifizierung des Anbieters als Firma oder Privatperson. Diese Daten sind Anbieternummer, Firma, OpenImmo Anbieter-ID sowie die Daten des Impressums, inklusive Adresse und Steuernummern. Die OpenImmo Anbieter-ID erfüllt die Aufgabe der eindeutigen Identifizierung des Anbieters im gesamten OpenImmo Pool, das heißt jeder Anbieter ist innerhalb des OpenImmo Datenverkehrs eindeutig in jeder Software über diese ID identifizierbar. Die Anbieternummer hingegen dient der Einordnung des Anbieters in das interne System.

Wird ein Objekt, aus welchem Grund auch immer, ohne Anbieter übertragen, so muss sich das Element des Objekts innerhalb eines ”leeren” Anbieters befinden. ”Leerer” Anbieter bedeutet in diesem Zusammenhang, dass der Anbieterknoten neben dem ihm zugeordneten Objekten (Element *immobilie*) alle herkömmlichen Anbieter Unterknoten enthält, sich in diesen jedoch keine Werte befinden.

Feedback übertragen

> *„Feedback Datensatz der vom Portal an den Anbieter (Makler / Software) übertragen wird. Eine Datei mit einem / mehreren Objekt(en) und jeweils einem / mehreren Kundenanfrage(n)“*
> [OISTANDARD-2013, Dokument openimmo-feedback-125.xsd]

Die *Feedback* Dateien dienen zur Darstellung bzw. Übermittlung von Kommunikationsdaten bzgl. einer Immobilie. In diesen Datensätzen werden beispielsweise Interessenanfragen zum Kauf einer Immobilie übermittelt. (Siehe Listing 2.4)

Listing 2.4: OpenImmo Version 1.2.5, Feedback Datensatzstruktur

```
<openimmo_feedback>
        <version>1.2.5</version>
        <sender>
                <name>immoabc</name>
                <datum>2011-11-11</datum>
                <makler_id>446544</makler_id>
        </sender>
        <objekt>
                <portal_unique_id>4eu73u734da8s38d</
                   portal_unique_id  >
                <portal_obj_id>P12345678</portal_obj_id  >
                <anbieter_id>MA-1024</anbieter_id  >
                <oobj_id>MA-1024-8874</oobj_id >
                <vermarktungsart>KAUF</vermarktungsart >
                <bezeichnung>Haus zum kauf</bezeichnung>
                <interessent>
                        [...]
                </interessent>
        </objekt>
</openimmo_feedback>
```

Diese Datensätze enthalten neben der OpenImmo Version eine Kurzübersicht über die Absendersoftware, sowie dem gewünschten Objekt oder Objekten und dessen Interessent.

Die *Feedback* Datensatzverwaltung wurde bei der Implementierung des Prototyps nicht umgesetzt und wird daher an dieser Stelle nicht tiefgehender betrachtet.

2.4 Validierung und Gültigkeit

Validiert werden die genannten Strukturen über die von OpenImmo gelieferten beiden XML-Schema Dateien. Diese Dateien spezifizieren, ob, wie und in welcher Form die verschiedenen Felder und Attribute ausgefüllt werden sollen bzw. dürfen. (zum Vergleich, siehe Kapitel 2.2)

In diesem Kapitel wird der Autor lediglich auf die Validierung der OpenImmo Datensätze eingehen. Diese Abgrenzung hat den bereits erwähnten Grund, dass die Feedback-Übertragung nicht Bestandteil dieser Arbeit ist.

In der OpenImmo Schemadatei werden den Elementen, welche in einem OpenImmo Datensatz enthalten sind, entsprechende komplexe und einfache Typen zugeordnet. Die komplexen Typen, wie z.B. das Wurzelelement oder das zuvor erwähnte Element *uebertragung*, enthalten dabei weitere Elemente und/oder Attribute, während die einfachen Typen, wie das Attribut *art* des Elements *uebertragung*, lediglich Werte definierter

Datentypen enthalten. (siehe Kapitel 2.2)

Dabei verlaufen die Definitionen letztendlich so, dass nahezu jeder Pfad in dem hierarchischen *Baum* der Datenübertragungsdatei in einem Element des Typs *simpleType* endet.

In Listing 2.5 wird zur Verdeutlichung der erklärten Zusammenhänge die Definition des Wurzelknotens aus der Schemadatei abgebildet.

Listing 2.5: OpenImmo Version 1.2.5, Definition des Hauptelements in der Schemadatei

```
<xsd:element name="openimmo">
        <xsd:annotation>
                <xsd:appinfo>0</xsd:appinfo>
                <xsd:documentation>Dokument Element</xsd:
                    documentation>
                <xsd:documentation xml:lang="en"></xsd:
                    documentation>
        </xsd:annotation>
        <xsd:complexType>
                <xsd:sequence>
                        <xsd:element ref="uebertragung"/>
                        <xsd:element ref="anbieter" maxOccurs="
                            unbounded"/>
                        <xsd:element ref="user_defined_simplefield"
                             minOccurs="0" maxOccurs="unbounded"/>
                        <xsd:element ref="user_defined_anyfield"
                            minOccurs="0" maxOccurs="unbounded"/>
                </xsd:sequence>
        </xsd:complexType>
</xsd:element>
```

Somit enthält die OpenImmo Schemadatei alle Bestimmungen der zugelassenen Datentypen und Werte der einzelnen optionalen und notwendigen Datenangaben.

Dabei werden bestimmte Attribute oder Elementwerte auch auf eine Auswahl möglicher Werte beschränkt. Wie in Tabelle 2.1 bereits aufgeführt, ist beispielsweise das Attribut *umfang* nur gültig, wenn es einen der Werte *TEIL* oder *VOLL* enthält. Zu diese Zweck sind in der Schemadatei spezielle einfache Typen definiert.

Durch dieses Vorgehen kann eine Vorschrift zum Datenstandard erschaffen werden. Dieses Validierung stößt jedoch zum Teil an seine Grenzen.

Ein Beispiel für solche Grenzen wären die Daten des Elements *techn_verwaltung*, welche zur technischen Verwaltung der Datensätze bereitstehen.(vgl. 2.3).

Die hier enthaltene OpenImmo Objekt-ID soll eine Immobilie innerhalb aller Portale und Programme eindeutig identifizieren. Diese Nummer lässt sich dementsprechend nicht

über eine starres Schema prüfen. Das Schema könnte in diesem Zusammenhang lediglich die Zusammensetzung der Zeichenkette prüfen. Diese Zusammensetzung befindet sich in stetiger Entwicklung und wird Teil der nächsten Versionen des Standards werden.

Weitere Hinweise hierzu können den Dokumentationsabschnitten der OpenImmo Dateien der Version 1.2.5 entnommen werden. In diesen werden folgende Informationen zu dem Thema bereitgestellt.

> *„Eindeutiger Identifikator der Immobilie innerhalb aller Immobilienbörsen-Portale, der Aufbau ist definiert (s. Protokoll v. 28.11.01), wird aber nicht durch die XSD überprüft, da hier in absehbarer Zeit durch Verwendung von Verschlüsselung eine Änderungen [sic!] stattfindet“*
> [OISTANDARD-2013, Dokument openimmo_125.xsd]

Durch das Fehlen einer zentralen Überwachungseinheit bzw. Datenbank zum Abgleich dieser Nummer, stellt dieser Teil eine große Herausforderung an die Entwickler bzw. den Verein dar.

Demzufolge sind im erwähnten Zusammenhang in Zukunft noch größere Veränderungen zu erwarten.

Das OpenImmo XML-Schema der OpenImmo Version 1.2.5 befindet sich auszugsweise im Anhang A.1. Der dort abgebildete Auszug umfasst das Schema von Zeile 1 bis Zeile 682 der insgesamt über 3000 Zeilen. Aufgrund der Komplexität musste von der kompletten Darstellung des Schemas an dieser Stelle abgesehen werden.

3 Konzeption

Das OpenImmo Datenformat basiert, wie zuvor dargestellt, auf der XML-Technologie. Entsprechende Softwareentwicklungen müssen daher über die Werkzeuge zum Umgang mit dieser Technologie verfügen.

Da es sich bei OpenImmo Datensätzen um Übertragungsdaten handelt, dienen sie zur Import- und Exportgenerierung der jeweiligen softwareinternen Datenstrukturen.

3.1 Klassische Datenorganisation einer Software mit Import/Export Funktion

Immobilienprogramme, die den OpenImmo Standard unterstützen, verfügen über die Möglichkeit Datensätze des OpenImmo Formats zu im- bzw. exportieren. Der OpenImmo Datenstandard beeinflusst dabei in der Regel in keiner Weise die internen Abläufe oder Strukturen.

Es existiert eine konsequente Trennung von der Datenstruktur der Software und der Datenstruktur des zu importierenden Formats. Dies hat zum einen den Grund, dass die Software meist unabhängig und nicht mit dem Ziel, ein bestimmtes Format zu unterstützen, entsteht. Zum anderen kann die Entwicklung somit meist stärker an die Bedürfnisse des angestrebten Einsatzgebietes angepasst werden.

Eine solche Software lässt sich grob wie in Abbildung 3.1 dargestellt schematisieren.

Abbildung 3.1: Klassisches Konzept, Quelle: Tobias Ussath

Die zu importierten Daten lassen sich dementsprechend nach den eigenen Vorstellungen in die Software integrieren und es entsteht kein Raum für Kompromisse bzgl. der eigenen Softwareentwicklung.

Die Entwicklung der Software verläuft unabhängig von den zu importierenden Datensätzen bzw. -formaten. Die Schnittstelle regelt den Im-/Export, validiert die ein- und ausgehenden Dateien und reguliert das korrekte Mapping der internen Struktur auf die des externen Formats.

Wie viele verschiedene Arten von Datenformaten die Software später unterstützten kann, muss zu keinem Stand der Entwicklung feststehen. Die Schnittstelle kann jederzeit erweitert oder abgeändert werden.

Die Datenstruktur der Software ist dabei, jedenfalls im Bereich der Immobilienverwaltung, sehr komplex. Die Softwarearchitektur muss in diesem Fall von Beginn an gut durchdacht und auf den Anwendungsfall ausgelegt werden. Es muss schon frühzeitig bei der Entwicklung des Datenbankschemas darauf geachtet werden, dass Daten die später schnell oder oft zur Verfügung stehen sollen, keine hohen Zugriffszeiten benötigen. Diese hohen Zugriffszeiten könnten durch eine unzweckmäßige Verknüpfung von Datenbanktabellen oder eine zu große Verstreuung der relevanten Daten in der Datenbank entstehen. Da zu Beginn einer Entwicklung nur sehr schwer abzuschätzen ist, in welche Richtung sich die Bedürfnisse des Anwenderkreises in Zukunft entwickeln könnten, sind die Ansprüche an den Softwarearchitekten besonders hoch.

3.2 Für den Prototyp optimierte Datenorganisation

Im Fall dieser Bachelorarbeit liegt das Hauptinteresse des Praxispartners in der Entwicklung eines Onlinetools zur Verwaltung von Immobiliendaten, speziell Daten des OpenImmo Formats. Es steht somit bereits zu Beginn fest, dass es sich um ein Tool handelt, welches lediglich mit Datensätzen dieses Typs in Berührung kommen soll und wird. Die importierten Immobilien sollen dabei auch in jeglicher Art und Weise bearbeitet werden können.

In erster Linie zielt die Entwicklung darauf ab, eine Webanwendung zu schaffen, welche Immobilien an mögliche Kunden anbieten kann. Das bedeutet, es wird nach einer Lösung gesucht, die prägnanten Daten einer Immobilie bereitzustellen. Der in dieser Arbeit entwickelte Prototyp soll dabei lediglich den Verwaltungsbereich der Objektdaten umfassen. Der Autor hat weiterhin den Anspruch, eine updatefähige und im Idealfall eine von der Version des Datenformats unabhängige Software zu schaffen.

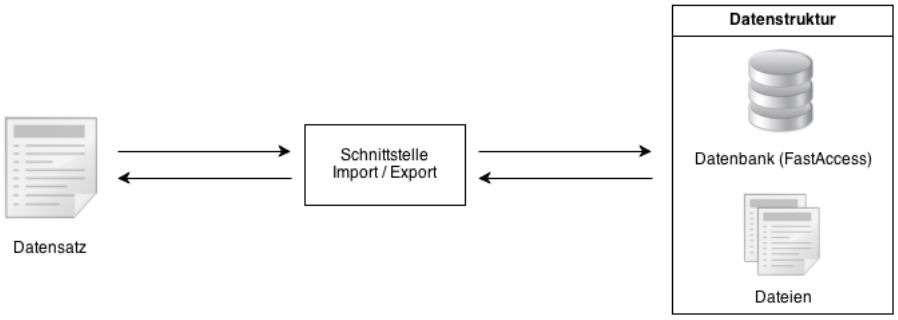
Diese Rahmenbedingungen und die Überlegungen aus Kapitel 3.1 führen zu dem Resultat, die Haltung der Daten für den Einsatzbereich zu optimieren. Diese Optimierung soll auf eine ungewöhnliche Art und Weise geschehen.

Nach Möglichkeit soll die Datenstruktur nicht von Anfang an von der Software streng vorgegeben werden. Der Autor hat sich dazu entschieden, für die Umsetzung der Ziele die Datenhaltung nach einem vorgegebenen Regelsatz aus der OpenImmo Version dynamisch zu generieren.

Dabei dient die OpenImmo Version nicht nur der Validierung der Import- und Exportdateien, sondern auch der Unterstützung der internen Datenstruktur. Es erfolgt daher

gegenüber dem Prinzip aus Kapitel 3.1 eine Aufhebung der Trennung von interner und externer Datenstruktur.

Abbildung 3.2: Schematische Überlegungen zur Konzeption, Quelle: Tobias Ussath



Das Prinzip beruht auf den folgenden Ansätzen. Nach dem Import wird jedes Datenobjekt mit eindeutigen Kenndaten und einem Verweis auf eine Datendatei (die Datendatei dieses Objekts) in der Datenbank vermerkt (siehe Abbildung 3.2). Die Software separiert nun Daten für den Schnellzugriff (*FastAccess*) und Daten für die dauerhafte Speicherung, die auf keine schnellen Zugriffszeiten angewiesen sind.

Alle Werte des Schnellzugriffs werden in der Datenbank nach einem entsprechendem Schema abgelegt. Daten geringerer Priorität bleiben in den Dateien gespeichert. Des Weiteren werden in einer *FastAccess*-Konfigurationstabelle die Feld- bzw. Knotenadressen der Werte vermerkt, welche in der Software für den Schnellzugriff zur Verfügung stehen sollen. Über diese Tabelle werden die objektspezifischen Schnellzugriffswerte, welche sich in einer separaten *FastAccess*-Wertetabelle befinden, mit den jeweiligen Objekten verknüpft.

Über die Administration der Software soll man Zugriff auf alle Daten der Immobilien haben und diese ebenfalls bearbeiten sowie verwalten können. Auch die Festlegung der *FastAccess*-Daten, die gleichzeitig den Schnellansichts- bzw. Übersichtsdaten des späteren Kundenbereiches entsprechen, finden an dieser Stelle statt.

Im Kundenbereich besteht zukünftig die Möglichkeit, die Immobilien nach diesen *FastAccess*-Daten zu durchsuchen, zu filtern und anzuzeigen bzw. aufzulisten. Eine Komplettübersicht wäre über ein, bei Aufruf als PDF generiertes, Exposé denkbar.

Nach diesen Überlegungen würde die Software in der Lage sein, die gestellten Anforderungen zu erfüllen.

Der im letzten Teil dieses Kapital erwähnte Kundenbereich bzw. Gästebereich sollte nur einen Ausblick auf die weitergehende Entwicklung darstellen. Er ist aufgrund des geringen Zeitfensters dieser Arbeit kein Bestandteil der folgenden Ausarbeitungen und des Prototyps.

4 Entwurf und prototypische Implementierung der Software

Aufgrund der durch den Praxispartner gegebenen Rahmenbedingungen, erfolgt die Entwicklung des Prototyps auf Basis des Zend Framework 2. Der Prototyp ist die erste Arbeit, die der Autor mit Hilfe diesem Frameworks umsetzt. Nicht zuletzt dieser Umstand erhöhte den Entwicklungsaufwand und -zeitraum, da eine Einarbeitungszeit und das Erlernen bestimmter Verwendungen des Frameworks erforderlich waren.

4.1 Zend Framework 2

Das Zend Framework 2 (ZF2) ist eine Entwicklung der Firma Zend. Das flexible Open-Source PHP-Framework eignet sich sowohl für kleinere Projekte als auch für professionelle Webanwendungen.

Nach Installation des Zend Frameworks kann mit der Erstellung und Installation einzelner Module begonnen werden. Die große Stärke des Frameworks besteht darin, dass ab Version 2.0 diese Module unabhängig und übertragbar (auf andere ZF2 Anwendungen) arbeiten. Sie können als vollkommen eigenständige Unterprogramme angesehen werden. Aus diesem Grund entsteht der Prototyp zu dieser Arbeit als eigenständig funktionierendes ZF2 Modul.

Es wird davon abgesehen, an dieser Stelle auf die nähere Funktionsweise von ZF2 einzugehen. Das Modul wird im Folgenden überwiegend alleinstehend betrachtet, ohne größeren Bezug auf die zugrunde liegende Skeleton Application des ZF2 zu nehmen. Nähere Informationen sowie eine anfängliche Dokumentation zu dem Framework, findet man auf [ZF2-2013], der Entwicklerseite.

4.2 Entwurf

4.2.1 Architektur

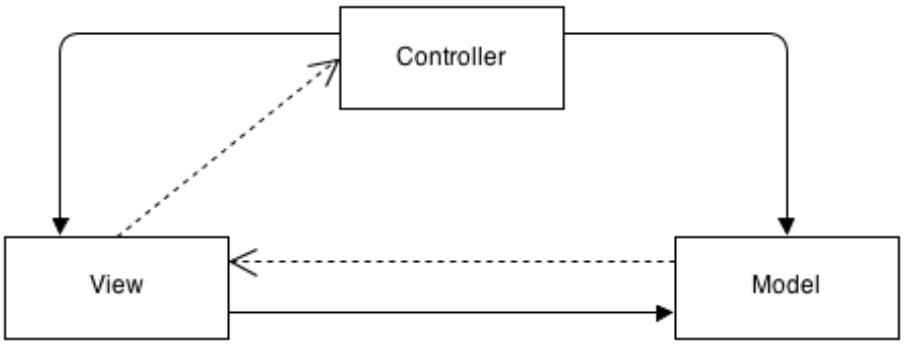
Der elementare Bestandteil des Prototypen ist in einem einzigen eigenständigen Modul, dem Modul *Backend*, verwirklicht. Es erledigt die Kernaufgaben der Datenverwaltung und stellt die Konfigurationsoberflächen zur Verfügung. Alle Dateien des Moduls befinden sich im ZF2 Modulordner in einem gleichnamigen Unterordner (`/module/Backend`).

Das Modul wird in der Datei `/config/application.config.php` im System angemel-

det und kann in der moduleigenen Konfigurationsdatei
`/module/Backend/config/module.config.php` konfiguriert werden.

Das Modul wurde nach dem Prinzip der Model-View-Controller Architektur (Abbildung 4.1) entworfen. Durch die Trennung von Präsentation, Logik und Daten entsteht somit ein strukturiertes, einfach erweiterbares und zu wartendes Softwareprojekt. Programmteile können an den verschiedenen Stellen der Software komfortable wiederverwendet und modifiziert werden. Bei Fehlern oder Updates reicht es somit meist aus, nur einzelne dieser Bestandteile zu überarbeiten oder auszutauschen.

Abbildung 4.1: Schematisierung der Model-View-Controller Architektur, Quelle: Tobias Ussath



Die Controller werden in der Konfigurationsdatei des Moduls registriert. Von nun an sind diese im System angemeldet. Alle mit *Action* endenden Methodennamen können in der Konfiguration als *action* eines Controllers angesprochen werden. So kann über den *Router* eine entsprechende *Route* definiert werden. Über diese Route wird die entsprechende *Action* ausgeführt. Gibt eine solche *Action* eine *ViewModel* Instanz zurück, so erwartet ZF2 eine gleichnamige (gleicher Name wie die *Action*) Template-Datei im Ordner `/view/Backend/CONTROLLER-NAME/` des Moduls zur Ausgabe der Daten (z.B. die Template-Datei der `Action`-Methode `importAction()` des `ImportController`: `/view/Backend/import/import`). Anderweitig wirft das Framework eine *Exception*.

Dem `ViewModel` können als Parameter im Controller weitere Instanzen oder andere konfigurierte Elemente der Präsentationsschicht übergeben werden. Benötigte Datenobjekte oder Mapper werden dabei über die Models eingeholt.

Nach diesem Prinzip erfolgt die Entwicklung des Moduls, der Prototypsoftware. Auf die einzelnen Controller und deren Funktionsweise wird im späteren Abschnitt dieses Kapitels in Punkt 4.3 genauer eingegangen. Zunächst sollen die Datenstruktur und das Datenbankschema näher erläutert werden.

4.2.2 Datenstruktur und Datenbankschema

Die entwickelte Datenbankstruktur arbeitet eng mit dem Dateisystem zusammen. Es wird daher im Folgenden in diesem Zusammenhang von einen Datenhaltungsschema gesprochen.

Das Datenhaltungsschema stellt den Hauptpunkt der Optimierung dar. Durch seine ungewöhnliche Struktur und Arbeitsweise wird versucht, die in den vorangegangenen Punkten genannten Ziele und Optimierungen zu erreichen. Das Schema ist in Abbildung 4.2 dargestellt.

In der Datenbanktabelle ***oi_schema_version*** werden die OpenImmo Schemaversionen gespeichert. Diese Tabelle ist aktuell mit keiner anderen Tabelle oder Datei verknüpft, da sie lediglich die Schemaversionen hält und bereitstellt. Es wurde sich an dieser Stelle für die Speicherung in einer Datenbank entschieden, um die Verwaltung und Zugriffe auf die einzelnen Versionen zu vereinfachen. In der Tabelle wird ebenfalls die aktuell aktive Schemaversion durch die Spalte *active* gekennzeichnet. So kann im Falle des Imports schnell auf die richtige Version zur Validierung automatisch zugreifen werden. Das Schema an sich wird in Klartext in der Spalte *schema* der Tabelle abgelegt.

Das abgelegte aktive Schema wird zum Import/Export und zu jeglichen programminternen Datensatzänderungen zur Validierung benötigt. Es ist ansonsten in der aktuellen Version des Prototyps in keiner Weise mit anderen Datensätzen verknüpft.

Ein importierter OpenImmo Datensatz wird in der Tabelle ***oi_object*** angelegt. Er wird anhand seiner *openimmo_obid* eindeutig eingetragen. Von nun an ist er durch eine *ID* eindeutig im System identifizierbar. In der *oi_object*-Tabelle werden seine *openimmo_obid*, *objektnr_intern* und *objektnr_extern* gesichert. Es erfolgt ebenfalls die Eintragung der *oi_offerer_id*, welche ein Verweis auf die Tabelle ***oi_offerer*** darstellt, in der sich die entsprechenden Anbieter der Objekte bzw. Immobilien befinden.

Alle Daten der Immobilien, das bedeutet der gesamte Knoten *immobilie* der OpenImmo Datendatei (siehe Kapitel 2.3), werden in einer Datei mit der eindeutigen *ID* des Tabelleneintrags benannt, im Ordner `/data/oi_object` gespeichert. Auf diese Daten kann nun durch die Verknüpfung *"Dateiname = ID des Tabelleintrages"* jederzeit zugegriffen werden. Das Gleiche geschieht mit den Daten des Anbieters. Der gesamte Knoten *anbieter*, exklusive den Immobilienelementen (alle Elemente *immobilie*), wird in einer Datei gespeichert. Der Name der Datei ist der Wert der Spalte *ID* des Tabelleneintrags des Anbieters. Die Dateien der Anbieter werden hierbei jedoch im Ordner `/data/oi_offerer` gesichert.

Im Folgenden zu den Schnellzugriffsdaten, den *FastAccess*-Daten. Diese Daten werden in den Tabellen mit Präfix *oi_fa* hinterlegt.

Die Tabelle ***oi_fa_config*** enthält die eigentlichen *FastAccess*-Felder, das heißt sie verfügt über die Pfade der Daten eines Datensatzes bzw. einer Immobilie, welche per Schnellzugriff erreichbar sein sollen. Dabei gilt die Zuordnung global. Festlegungen für spezielle Datensätze sind im aktuellen Prototyp nicht umgesetzt, da sie für das aktuelle Ziel unzweckmäßig erscheinen. Die hinterlegten Pfade sind die XPath-Abfragen (vgl. [CLARK-2013]) für die in ein DOM-Objekt überführten OpenImmo Datensätze.

Das DOM-Objekt bildet dabei die Schnittstelle zum Zugriff auf die einzelnen Elemente der XML-Struktur. Dadurch können Elemente direkt ausgelesen, geändert oder gelöscht werden. Mit Hilfe von XPath können Elemente in solch einem DOMDocument (dem DOM-Objekt) per Abfrage bzgl. spezieller Merkmalen (wie z.B. einem bestimmten Attribut) adressiert und ausgewertet werden.

In der Tabelle ***oi_fa_config*** ist außerdem zu jedem Pfad ein Titel zur Darstellung gespeichert und ob es sich bei dem aktuellen Pfad um einen Anbieterpfad, das heißt um einen Pfad in den Anbieterdateien, oder einen Objektpfad, das heißt um einen Pfad in den Immobiliendateien handelt.

Diese Pfade sind nötig, um die richtigen Daten aus den XML-Datensätzen auslesen und in die Datenbank schreiben zu können. Diese Thematik wird in späteren Kapiteln genauer behandelt.

In der Tabelle ***oi_fa_data*** sind nun die *FastAccess*-Daten enthalten. Diese Tabelle beinhaltet die Zuordnung der Werte zu *FastAccess*-Felder und den Objekten oder Anbieter (in der Tabelle *entity_id*). Dabei kann die Spalte *entity_id* sowohl mit einem Datensatz der Tabelle *oi_offerer* als auch mit einem Datensatz *oi_object* verknüpft sein. Dies ist möglich, da in der Tabelle *oi_fa_config* eindeutig definiert ist, ob es sich bei dem jeweiligen *FastAccess*-Feld um einen Eintrag des Typen *object* oder *offerer* handelt.

Um diese Zusammenhänge klarer darstellen zu können, wird in Abbildung 4.2 das Datenhaltungsschema vereinfacht schematisiert.

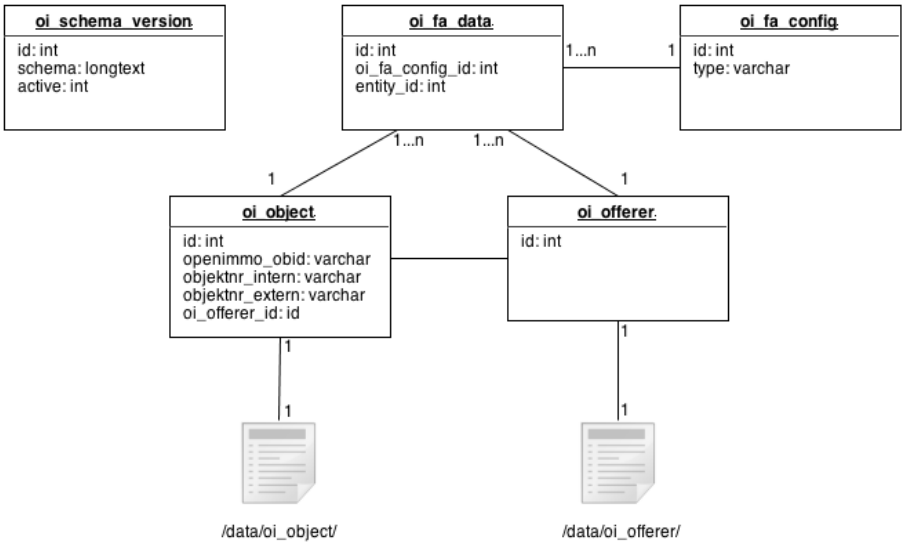
4.3 Implementierung

Die Software unterscheidet in den Programmabläufen zwischen Controller- (Steuerung), Model- (Daten) und View- (Präsentation) Komponenten . (Kapitel 4.2.1)

Den Bestandteil der **Steuerung** übernehmen die verschiedenen Controller. Dabei wurden diese Controller anhand der thematischen Aufgabenbereiche erstellt.

In Tabelle 4.1 erhält man eine Kurzübersicht über die Controller des aktuellen Prototyps.

Abbildung 4.2: abstrakte schematische Darstellung des Datenhaltungsschemas, Quelle: Tobias Ussath



Die Aufgaben der **Präsentation** übernimmt in einem ZF2 Modul in vieler Hinsicht die Skeleton Application. Für vielerlei Arten der Anzeige kann auf in ZF2 vorhandene Klassen in Form von Vererbung zurückgegriffen und deren Methoden zu Nutze gemacht werden. Beispiele dafür wären die Eingabeformulare der Software, die durch Ableitung der ZF2 Klasse `Form` sehr komfortabel generiert werden konnten.

Die Repräsentation der **Daten** erfolgt in den Models. Hierbei wurde ein `Entity` Model und `Table` Model mit jeweils den Grundfunktionalitäten des jeweiligen Bereichs geschaffen.

Das `Entity` Model enthält dabei die Methoden, die jedes einzelne der intern verwendeten Datenmodelle (Objekte) benötigt. Dies ist zum aktuellen Stand die Methode `exchangeArray(array $array)`, welche ein übergebendes `Array` in das entsprechende `Object` umwandelt. Alle für die Programmlogik benötigten Datenmodelle erweitern dieses Grundmodell.(siehe Listing 4.1)

Listing 4.1: Klasse `Offerer`, Erweitert die Klasse `Entity`

```
1 use Backend\Model\Entity;
2 class Offerer extends Entity{
3     public $id;
4     public $openimmo_anid;
5 }
```

Das allgemeine `Table` Model enthält die von jedem spezifischen Tabellenmapper genutzten Basisfunktionen wie `deleteRow(int $id)`, zum Löschen eines Objektes, oder `getRow(int $id)`, zum Auslesen eines Objektes aus der Datenbank. Die spezifischen Mapper leiten von dieser Klasse ab und können diese Methoden somit sofort ohne erneute Implementierung nutzen.

Tabelle 4.1: Controller des Protoyps

Controller	**Beschreibung**
EstateController	Steuerung der Arbeit mit den Datensätzen der Immobilienobjekte (u.a. editieren, löschen, duplizieren)
FastAccessController	FastAccess-Daten- und Konfigurationssteuerung; Arbeit mit den Konfigurations- und den Wertedatensätzen der Schellzugrifffelder
ImportController	Import und Export von Dateien der aktiven OpenImmo Version
IndexController	Startabläufe der Software bzw. des Moduls; im aktuellen Stand des Prototyps nur eine Begrüßung
OffererController	Steuerung der Arbeit mit den Datensätzen der Anbieterdaten der Immobilien
SchemaController	Import, Export und Verwaltung der OpenImmo Versionen
UserController	Erweiterung der Benutzersteuerung um einen Adminbereich; aufgrund des beschränkten Zeitrahmens der Arbeit vernachlässigt und nicht fertig implementiert

4.3.1 Import/Export

Hinter der Klasse `ImportController` verbirgt sich, anders als der Name vermuten lässt, die Klasse für die Import- und Exportsteuerung der Software. Die Klasse erweitert die ZF2 Klasse `AbstractActionController`.

Der Import ist dabei über die relative URL `admin/data` bzw. dem Menüpunkt *Import/-Export* im Menü des Prototypen erreichbar.

Bei der Importroutine erfolgt der Aufruf der Methode `importAction()` der Klasse (siehe Kapitel 4.2.1). Die Methode erfüllt im Programmablauf drei Stationen des Imports. Bei Erstaufruf instantiiert diese ein `ImportForm`, das Formular zur Auswahl der Datei, welche importiert werden soll und gibt anschließend eine `ViewModel` Instanz zurück. Dieses `ViewModel` erhält zuvor das `ImportForm` als Parameter. Die Rückgabe des `ViewModel` bewirkt, dass die ZF2 Skeleton Application, die durch das `ImportForm` generierten Daten auf das entsprechende Template rendert. Das Template wird von ZF2 unter `module/Backend/view/backend/import/import.phtml` erwartet. Bei diesem Vorgang wird lediglich ein leeres Formular ausgegeben. Es werden somit jegliche anderen Vorgänge der Methode außer Acht gelassen. (siehe Listing 4.2)

Listing 4.2: ImportController, Erstaufruf der Methode `importAction()`

```
1 $fieldsets = array();
2 $messages = array();
3 $form = new ImportForm('importform');
4 [...]
5 return new ViewModel(array(
6                 'messages' => $messages,
```

```
7                   'fieldsets' => $fieldsets,
8                   'form' => $form,
9           ));
```

In der Klasse `ImportForm` werden die notwendigen Felder und Validatoren konfiguriert. Da das Formular bei diesem Erstaufruf keine Daten enthält, findet an dieser Stelle keine Prüfung auf Gültigkeit statt.

Wird eine Datei ausgewählt und das Formular bestätigt, erfolgt per `request` ein weiterer Aufruf der Methode `importAction`. Dies ist die zweite Station.

Wie beim Erstaufruf wird eine Instanz des `ImportForm` erzeugt. Anschließend wird geprüft ob ein `request` stattfand und die Daten werden per `$this->setData($post)` in das Formular übertragen. Die Validierung erfolgt über die Formularinstanz.

Zur korrekten Validierung, der an die Software übergebenen OpenImmo Datei, wurde ein XML-Schemavalidator implementiert. Dieser wird genau wie die ZF2-eigenen Validatoren konfiguriert. Der Validator prüft dabei Daten vom Typ `File`. Nach Prüfung auf Dateiendung wird der Inhalt der Datei ausgelesen und in ein DOM-Objekt überführt. Dieses DOM-Objekt wird anschließend gegen das aktuell aktive OpenImmo XML Schema validiert.

War die Validierung erfolgreich, wird die Formularinstanz verworfen, da zur Darstellung der in der Datei zur Verfügung stehenden Immobilienobjekte ein anderes Formular benötigt wird. Das neue Formular wird mit den aufgearbeiteten Daten bestückt und per `ViewModel` zur Ausgabe gebracht.

Das Absenden dieses Formulars führt zum Import der Datensätze in die Datenbank. Dabei erfolgt neben dem Eintrag in die Tabellen `oi_object` und `oi_offerer` auch eine Prüfung der aktuellen *FastAccess*-Konfiguration. Je nach Resultat dieser Prüfung erfolgt das Schreiben der *FastAccess*-Daten in die Tabelle `oi_fa_data`.

4.3.2 *FastAccess*-Konfiguration und -Daten

Die Konfiguration der *FastAccess*-Daten kann im aktuellen Prototyp vollzogen werden, nachdem mindestens ein Immobiliendatensatz im System vorliegt. Aus diesem Datensatz werden die Pfade der XML-Struktur ermittelt.

Die notwendigen Methoden zu dieser Konfiguration liegen im Controller `FastAccessController`.

Der Controller erstellt sowohl die *FastAccess*-Konfiguration für die Immobilienobjekte

`object` als auch für die Anbieterobjekte `offerer`.

Die Methode `indexAction()` ist für den Immobilienteil des *FastAccess*-Bereichs zuständig. Wird die Methode aufgerufen, erfolgt die Ermittlung aller möglichen Wert- und Attributpfade. Dies geschieht, indem ein Immobilienobjekt geladen wird, in diesem Fall die Datendatei der Immobilie, und dessen Struktur in ein SimpleXML-Objekt konvertiert wird.

SimpleXML-Objekte bieten viele Möglichkeiten zur Handhabung der Werte und der Struktur von XML-Daten. So lassen sich mit Hilfe dieser Objekte die XML-Daten mit geringen Aufwand in ein `Array` umwandeln. (vgl. [SXML-2001])

Das SimpleXML-Objekte der Immobilien wird durch Verwendung dieser Technologie in ein `Array` umgewandelt. Dieses `Array` wird im Anschluss mit Hilfe einer rekursiven Methode durchlaufen, welche ein `Array` zurück gibt, das alle in der XML-Struktur enthaltenen Pfade enthält. Die Pfade werden in einer Form zurückgegeben, welche sich von XPath interpretieren lässt.

Durch ein tabellenartiges dynamisches Formular lassen sich nun alle gewünschten Schnellzugriffspfade inklusive Titel in der Konfiguration hinterlegen. Nach Absenden des Formulars, werden die Pfade in der Tabelle `oi_fa_config` gesichert und im Zuge dessen erfolgt der Aufruf der controllereigenen Methode `saveFastAccessData()`. Die Methode `saveFastAccessData()` durchläuft nun alle vorhandenen Immobilienobjekte und schreibt die Werte aus deren Dateien per entsprechender Zuordnung in die Tabelle `oi_fa_data`.

4.3.3 Benutzeroberfläche der Webanwendung

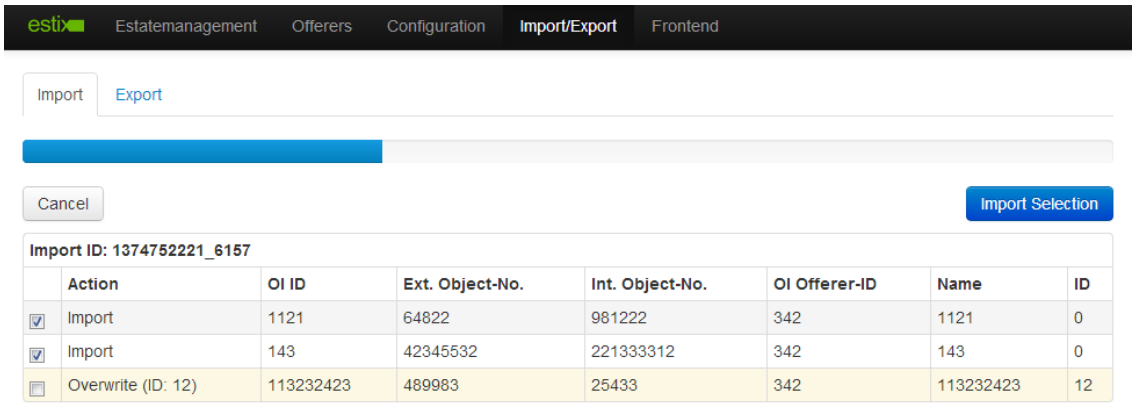
Das Webinterface bzw. die GUI des Prototypen wurde mit Hilfe des CSS Frameworks Bootstrap realisiert. Informationen zum Framework und dessen Anwendung können auf [BS-2013] eingeholt werden. Abbildung 4.3 zeigt das Design der Software. Dargestellt ist der Import von OpenImmo Datensätzen.

Bei der Entwicklung wurde Wert auf Übersichtlichkeit, Einheitlichkeit und benutzerfreundliche Bedienung gelegt. Sämtliche Systemmeldungen und Texte der Oberfläche wurden in englischer Sprache formuliert.

Der Prototyp der Software erhielt den Namen *Estix*.

Bei der Oberfläche ist zu bedenken, dass es sich bei der Umsetzung lediglich um das Backenddesign, das heißt dem Design der Administrationsansicht der Datensätze und Anzeigen des Kundenbereichs, handelt. Der Kunden- bzw. Besucherbereich (Frontend),

Abbildung 4.3: Weboberfläche der Software. Auswahl der zu importierenden Immobilien, Quelle: Tobias Ussath



kann sich später vom Erscheinungsbild der Administrationsoberfläche vollkommen differenzieren.

5 Analyse der Umsetzung

Das Resultat der Implementierung ist ein funktionierender Prototyp der geplanten Software. Der Prototyp erfüllt die gesetzten Ansprüche aus technischer Sicht. Der Prototyp unterstützt in der aktuellen Version folgende Funktionalitäten.

Konfiguration der Software

- Es ist möglich, OpenImmo Versionsdateien zur Validierung der OpenImmo Datenübertragungen in die Software zu importieren und zu nutzen.
- Zur Konfiguration der Software kann festgelegt werden, welche Daten für einen Schnellzugriff zur Verfügung stehen sollen. Diese *FastAccess*-Konfiguration erfolgt durch dynamische Auslesen.
- Es können *FastAccess*-Daten sowohl für Immobilien als auch für Anbieter konfiguriert und gespeichert werden.

Benutzung der Software

- Immobilien- und Anbieterdaten können komfortabel im- und exportiert werden.
- Datensätze der Immobilien und Anbieter können editiert und gelöscht werden. Es besteht die Möglichkeit Datensätze beider Formate zu erstellen.

5.1 Bewertung der Umsetzung

Der Prototyp erfüllt, wie aufgelistet, die gesetzten Anforderungen. Dabei ist er in allen Punkten der Umsetzung jedoch wirklich nur als Prototyp einer Software zu sehen. Dies spiegelt sich in vielen Punkten der Implementierung wider. Die Software ist im aktuellen Entwicklungsstand nicht für den Produktiveinsatz geeignet, bietet jedoch in allen Punkten die notwendigen Ansätze zur konstruktiven Weiterentwicklung. Im Folgenden sollen kurz die aktuellen Schwachstellen aufgedeckt und mögliche zukünftige Meilensteine erörtert werden.

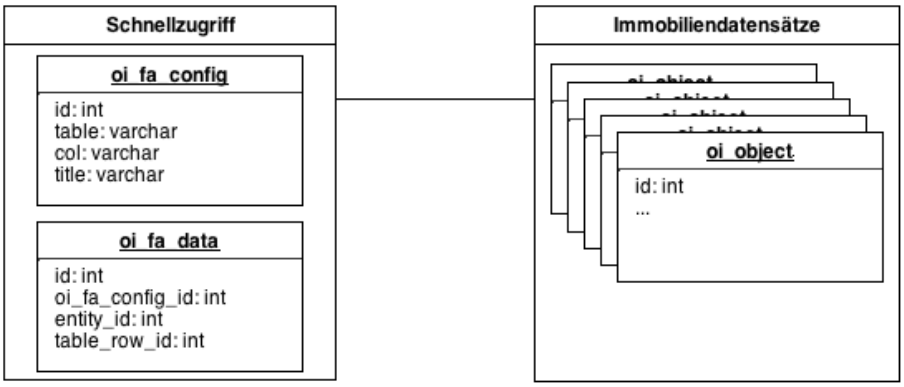
Datenhaltungsstruktur

Die Datenhaltungsstruktur erfüllt im aktuellen Stadium die Anforderungen und grundsätzlichen Prinzipien der Konzeption, erfüllt jedoch nicht den Standard einer aktuellen Softwareentwicklung. Hierzu müsste zunächst an folgenden Punkten angesetzt werden.

Die **Redundanz** bei der Datenhaltung ist in der aktuellen Version durch das gezielte Speichern bestimmter Werte in den Dateien als auch in der Datenbank sehr stark ausgeprägt. Dies gilt es zu vermeiden und wurde an dieser Stelle nur in dieser Art und Weise realisiert, um eine Umsetzung der Aufgabenstellung in kurzer Zeit verwirklichen zu können. So soll der Prototyp auf die Vorzüge des Schnellzugriffs aufmerksam machen, welche durch die Haltung der Stammdaten in Dateien noch stärker zum Ausdruck gebracht wird.

Die nächsten Entwicklungsphasen dieses Softwareteils, sollte Änderungen in die Richtung folgender Ausarbeitungen enthalten. Es macht Sinn, die Daten nicht doppelt zu speichern. Zu diesem Zweck sollten die Datensätze der Immobilien und Anbieter, wie im klassischem Konzept (Kapitel 3.1) in der Datenbank gespeichert werden. Somit könnte man von der Haltung der Datensätze in Dateien vollkommen absehen. Um die Verknüpfung zu der *FastAccess*-Konfiguration zu realisieren, wäre es denkbar die Tabelle `oi_fa_daten` in der Form abzuändern, dass die Spalten `oi_fa_config_id`, `entity_id` und `table_row_id` enthält. Des Weiteren wäre so eine Verknüpfung der Datenwerte mit direkten Pfad durch die Tabelle `oi_fa_config` möglich und zweckmäßig. Der Pfad bzw. die Verknüpfung würde an dieser Stelle durch die Hinterlegung des Tabellennamens geschehen. Ein Beispiel zur Umsetzung dieses Prinzip wurde grob in Abbildung 5.1 visualisiert. Nach diesem Schema könnte ein Direktzugriff (Schnellzugriff) auf die wichtigen Daten der Datensätze stattfinden.

Abbildung 5.1: Überarbeitetes Datenhaltungsschema, Aufhebung von Redundanz und Optimierung der Datenhaltung, Quelle: Tobias Ussath



Die aktuell zusätzliche Haltung der Datensätze in Dateien führt dabei jedoch nur zu einem Leistungsverlust im Administrationsbereich. Die Schnellzugriffsdaten werden in der aktuellen Version, sowie in der zuvor erarbeiteten Verbesserung, in etwa gleicher Geschwindigkeit ausgelesen. Die neue Version behebt lediglich den unökonomischen Ansatz der Speicherung der Daten in dieser Datenhaltungsstruktur.

Diese fehlende Redundanz würde sich auch bei der Verwaltung der Datensätze bemerkbar machen. Es müssten zum Beispiel bei Änderung der Datensätze nicht die Daten in Datendatei und Tabelle geändert werden.

Import/Export und Anlegen von Datensätzen und ihre Validierung

Die Verwaltung der Datensätze erfordert im Prototypen noch an vielen Punkten der Software die Existenz mindestens eines Datensatzes des jeweiligen Typs zum Auslesen der notwendigen Pfade. Dies ist aktuell darauf zurückzuführen, dass die Software über noch kein fertiges Werkzeug zur kompletten Interpretation der OpenImmo XML-Schemadateien verfügt. Eine Implementierung einer solchen Softwarekomponente hätte bedeutende Vorteile bei den Programmabläufen.

Notwendige Felder könnten anhand dieses XML-Schema-Parsers aus dem Schema heraus generiert werden. Ebenfalls die Validierung der Eingabefelder könnte optimiert werden. Die Eingabefelder könnten anhand der in der Schemadatei definierten Datentypen und Informationen dargestellt werden. Beispielsweise könnte für ein Eingabefeld eines Wertes des Typ Boolean automatisch ein geeignetes Dropdown Feld oder eine Checkbox ausgegeben werden.

Benutzeroberfläche

Die Benutzeroberfläche bedarf an vielen Stellen noch Verbesserungen hinsichtlich der Benutzerfreundlichkeit. Um den aktuellen Prototypen nutzen zu können, erfordert es aktuell noch ein gewisses technisches Verständnis. Das Anlegen der *FastAccess*-Konfiguration beruht stark auf der XML-Struktur, was mit der dynamischen Generierung dieser in Verbindung steht. Hier werden in Zukunft geeignete Templates hinterlegt bzw. bei der Entwicklung des zuvor erwähnten XML-Schema-Parsers genauer darauf eingegangen.

Updatesicherheit, Flexibilität und Dynamik der Software

Trotz der notwendigen Verbesserungen bzw. der notwendigen Weiterentwicklung der Software, sind durch die Entwicklung des Prototyps zahlreiche Erfolge erzielt worden.

Der Prototyp ist komplett funktionell und arbeitet dabei an allen Aufgabenbereichen des Programms dynamisch. Jegliche Einstellungen und Abläufe reagieren auf die aktuelle Konfigurationen und Datensätze. Dadurch gewinnt man ein hohes Maß an Flexibilität. Das Programm kann mit allen aktuellen OpenImmo Versionen betrieben werden. Durch die Speicherung der Datensätze in Dateien und das Setzen der *FastAccess*-Felder und Werte, sind die Programmabläufe von keiner vom Entwickler festgelegten Struktur abhängig.

Zu importierende OpenImmo Schema Versionen werden auf Gültigkeit geprüft und können ab Aktivierung genutzt werden. Jeder nach der Aktivierung eines Schemas angelegter oder importierter Datensatz richtet sich von da an, sei es beim Importprozess oder bei der Validierung des Änderungsformular, nach dem aktiven Schema.

Für Inbetriebnahme ist somit lediglich der Import des gewünschten OpenImmo Versionsschema und eines gültigen Datensatzes notwendig.

5.2 Vor- und Nachteile gegenüber konventionellen Umsetzungsstrategien

Im Folgenden soll eine kurze Gegenüberstellung der aktuellen Umsetzung des Prototyps gegenüber der konventionellen Umsetzung stattfinden. Dieser Teil wird, angesichts dass es sich bei dieser Entwicklung um einen Prototyp und keine einsatzfähige Produktivsoftware handelt, nur kurz beleuchtet.

Komplexität der Datenhaltungsstruktur

Ein **Vorteil** bei der Entwicklung des Prototypen ist die aktuell noch sehr überschaubare Datenhaltungsstruktur. Die Umsetzung der kompletten Immobiliendaten als Datenbankschema, wäre an dieser Stelle bedeutend komplexer ausgefallen, je nach Einsatzbereich der Software womöglich jedoch zweckmäßiger.

Redundanz der Daten

Die Redundanz der Datenhaltung im Prototyp ist ein klarer **Nachteil** gegenüber anderen Umsetzungen. Daten werden durch das aktuelle System der Schnellzugriffsfelder an mehreren Orten in gleicher Ausführung gespeichert. Dies sollte in aktuellen Softwareentwicklungen vermieden werden und schafft zusätzlichen Fehleranfälligkeit bei den Datenoperationen.

Zugriffszeiten auf relevante Daten

Wurde der Prototyp korrekt konfiguriert, das heißt alle notwendigen Schnellzugriffsfelder wurden entsprechend angelegt, so erfolgt eine Abfrage der Daten im Normalfall bedeutend schneller als bei vielen anderen Datenstrukturen, in die das OpenImmo Format, bzw. Immobiliendaten dieser Komplexität, sinnvoll überführt werden könnten. Dies ist an dieser Stelle somit als klarer **Vorteil** der erarbeiteten Software gegenüber der klassischen Umsetzung (3.1) zu sehen. Die Zugriffszeit müsste sich jedoch bei dem aktuellen Datenhaltungsschema ab einer gewissen Anzahl an Datensätzen immer mehr der Zugriffszeit eines konventionellen komplexen Datenbankschema anpassen.

Konfigurationsaufwand

Der Konfigurationsaufwand ist bei der aktuellen Umsetzung ein **Nachteil**. Andere Softwareentwicklungen erfordern bei gleichem Anwendungsbereich keine derartige Einstellungen und Wissen über den verwendeten Datenstandard bzw. der verwendeten Technologie. In diesem Fall wäre das der OpenImmo Standard und die XML-Technologie.

Flexibilität

Die Software kann für jeden Anwendungsbereich eigenständig ohne Entwicklungsaufwand konfiguriert werden. Dieser **Vorteil** sollte auch bei der Weiterentwicklung stets einbezogen werden. Dies ist ein Merkmal, das nahezu bei keinem aktuellen Softwareprodukt vorhanden ist.

6 Zusammenfassung und Ausblick

Das Thema lautete, ”Konzeption, Entwurf und prototypische Implementierung einer Software zur Haltung und Organisation von Datensätzen des OpenImmo Standards”. Die Erarbeitung erfolgte mit Unterstützung und in den Räumlichkeiten des Praxispartner Clicks Online Business in Dresden.

Ziel der Arbeit war es, anhand einer Software eine Möglichkeit zur Verwaltung von Datensätzen des Immobilienübertragungsformats OpenImmo zu entwickeln und umzusetzen. Es sollte dabei die Option bestehen, die Datensätze importieren und exportieren zu können, sie zu bearbeiten und zu verwalten. Die Entwicklung beschränkte sich aufgrund der Bearbeitungszeit der Bachelorarbeit von nur wenigen Wochen auf ein Produkt mit prototypischen Charakter.

Zusätzlich zu diesen thematischen Rahmenbedingungen sollte der Wunsch des Praxispartners, den Prototypen mit Hilfe des etablierten PHP-Frameworks ZF2 zu erstellen, berücksichtigt werden.

Diese für die Entwicklung wesentliche Bedingung beeinflusste die Bearbeitung der Aufgabe in einem zu Beginn unerwarteten Ausmaß. Somit begann die Einarbeitung in das Thema, neben der Recherche zu dem Datenformat OpenImmo, mit der Aneignung der Grundlagen des für den Autor bis dato unbekannten ZF2. Diese zusätzliche Einarbeitungszeit in das Framework ergab sich als sehr aufwendig und beeinflusste den Zeitplan der Bachelorarbeit nicht unerheblich. Viele Verfahrensweisen mit der für den Autor vertrauten Programmiersprache PHP mussten durch die Verwendung der frameworkeigenen Bibliotheken oftmals neu erlernt werden.

Trotz dieser Herausforderungen wurde der Prototyp fertiggestellt. Er erfüllt in der aktuellen Version die genannten Anforderungen und wurde mit Hinblick auf eine Weiterentwicklung strukturiert nach der MVC-Architektur entworfen.

Das besondere Merkmal der Umsetzung ist die ungewöhnliche Haltung der Daten in Form von Datei- und Datenbankeinträgen. Die daraus resultierende Redundanz wird dabei nicht als Schwachpunkt der Entwicklung gesehen, sondern lediglich als Etappenlösung auf dem Weg zur einsatzfähigen Produktivsoftware. Mögliche Schritte zur Weiterentwicklung dieser auf die Prototypphase beschränkten Lösung wurden in Kapitel 5 ausgiebig dargelegt.

Die Art der Umsetzung verdeutlicht die Vorteile der Nutzung des *FastAccess*-Tabellenprinzips. Dadurch sollte dargelegt werden, dass die Performance beim Abruf komplexer Datengebilde, wie die Datensätze von Immobilienobjekten, durch die Beschränkung auf die wesentlichen Werte bzgl. der Zugriffszeit stark verbessert werden kann.

Um das Vorgehen allgemein zu betrachten, wäre es ebenfalls denkbar, diese Art der Umsetzung zur Aufwertung veralteter bzw. historisch gewachsener Datenbankschemen anzuwenden. In diesen meist sehr komplexen und unzweckmäßigen Datenbankgebilden ist die Performance meist nur durch eine Neuentwicklung der Software zu erhöhen. Die zeitaufwändige und kostspielige Entwicklung könnte durch den Einsatz einer Programmkomponente zur Verwaltung der Schnellzugriffsdaten herausgezögert werden.

Die Aufgabenstellung wurde erfüllt und die aus dieser Arbeit sowie dem Prototyp gewonnenen Erkenntnisse öffnen Spielraum für eine Weiterentwicklung. Die zudem erworbenen Kenntnisse über das Framework ZF2 und den OpenImmo Datenstandard stellen sich für den Autor und künftige Entwicklungen als sehr nützlich dar.

Anhang A: OpenImmo Standard

A.1 XML-Schema

Listing A.1: oistandard]openimmo_125.xsd - OpenImmo Version 1.2.5 Schema, Quelle: [OISTANDARD-2013, Dokument openimmo_125.xsd, Zeile 1 - 682 von 3129 (ohne Leerzeilen)]

```
1  <?xml version='1.0' encoding='iso-8859-1'?>
2  <xsd:schema xmlns:xsd='http://www.w3.org/2001/XMLSchema' version='1.0'>
3  <!--
4  $$DOKU:
5  XML Schema Beschreibung des OpenImmo Datensatzes.
6  $COPYRIGHT: OpenImmo e.v.
7  OpenImmo und das OpenImmo Logo sind eingetragen Marken.
8  Nutzung gemaess Lizenzbedingung  unter www.openimmo.de
9  We are sorry, but there ist no english version available !
10 $VERSION: 1.2.5
11 ========================================
12 $VERSIONDATE: 2011-11-12
13 $HISTORY:
14 $CREATED: 2011-10-15, FB, Frank Bitzer, Openimmo e.V. ,    Marcus Hertel, FIO SYSTEMS AG
15 $CHANGE: Markiert mit $V120
16 $CHANGE: Markiert mit $V121
17 $CHANGE: Markiert mit $V122   7-9.2009
18 $CHANGE: Markiert mit $V123   August 2010
19 $CHANGE : 1.2.3.a September
20 $CHANGE : Markiert mit $V124  September/Oktober 2011
21 $CHANGE : Markiert mit $V125  November 2012
22 $$END
23 -->
24 <!-- Technische Hinweise aus der W3 Doku:
25  An instance of a datatype boolean can have the following legal literals {true, false, 1, 0}. -->
26 <xsd:include schemaLocation='opim-iso.xsd'><!-- import von land und waehrung -->
27   <xsd:element name='openimmo'>
28     <xsd:annotation>
29       <xsd:appinfo>0</xsd:appinfo>
30       <xsd:documentation>Dokument Element</xsd:documentation>
31 <xsd:documentation xml:lang='en'></xsd:documentation>
32     </xsd:annotation>
33     <xsd:complexType>
34       <xsd:sequence>
35         <xsd:element ref='uebertragung'/>
36         <xsd:element ref='anbieter' maxOccurs='unbounded'/>
37         <xsd:element ref='user_defined_simplefield' minOccurs='0' maxOccurs='unbounded'/>
38         <xsd:element ref='user_defined_anyfield' minOccurs='0' maxOccurs='unbounded'/>
39       </xsd:sequence>
40     </xsd:complexType>
41   </xsd:element>
42   <!-- ========== ROOT ============= -->
43   <xsd:element name='uebertragung'>
44     <xsd:annotation>
45       <xsd:documentation>Uebertragungsangaben</xsd:documentation>
46 <xsd:documentation xml:lang='en'></xsd:documentation>
47     </xsd:annotation>
48     <xsd:complexType>
49       <xsd:attribute name='art' use='required'>
50         <xsd:simpleType>
51           <xsd:restriction base='xsd:string'>
52             <xsd:enumeration value='ONLINE'/>
53             <xsd:enumeration value='OFFLINE'/>
54           </xsd:restriction>
55         </xsd:simpleType>
56         <!--Je nach Art der Uebertragung wird eine Antwort erwartet/zurueckgegeben-->
57       </xsd:attribute>
58       <xsd:attribute name='umfang' use='required'>
59         <xsd:simpleType>
60           <xsd:restriction base='xsd:string'>
61             <xsd:enumeration value='TEIL'/>
62             <xsd:enumeration value='VOLL'/>
63           </xsd:restriction>
64         </xsd:simpleType>
65         <!--Umfang gibt an ob es sich um einen Vollabgleich, welchee den kompletten Datenbestand wiederspiegelt
66           oder um einen Teilabgleich, wo nur einzufuegende, zu aendernde oder zu loeschen Objekte uebertragen werden (
                 inkrementell),
67           handelt-->
68       </xsd:attribute>
69       <xsd:attribute name='modus'><!--$V120-->
70         <xsd:simpleType>
```

```
71            <xsd:restriction base='xsd:string'>
72              <xsd:enumeration value='NEW'/>
73              <xsd:enumeration value='CHANGE'/>
74              <xsd:enumeration value='DELETE'/>
75            </xsd:restriction>
76          </xsd:simpleType>
77          <!--Informierender Charakter fuer Loeschen ohne <immo> ELement-->
78        </xsd:attribute>
79        <xsd:attribute name='version' type='xsd:string' use='required'/><!--$V120-->
80        <!--Version = OPENIMMO! Version im Format: Major.Minor.Revision-->
81        <xsd:attribute name='sendersoftware' type='xsd:string' use='required'/>
82        <xsd:attribute name='senderversion' type='xsd:string' use='required'/><!--$V120-->
83        <!--Version der Exportsoftware: Major.Minor-->
84        <xsd:attribute name='techn_email' type='xsd:string' use='optional'/>
85        <!--Die Email des techn. Ansprechpartners, Adresse fuer die Antwort (log-file) des Imports-->
86        <xsd:attribute name='regi_id' type='xsd:string' use='optional'/><!--$V120-->
87        <!--Registrieungskey aus der Lizenzvereinbarung, falls erfolgt-->
88        <xsd:attribute name='timestamp' type='xsd:dateTime' use='optional'/><!--$V122-->
89      </xsd:complexType>
90    </xsd:element>
91    <xsd:element name='anbieter'>
92      <xsd:annotation>
93        <xsd:documentation>Anbieterangaben</xsd:documentation>
94  <xsd:documentation xml:lang='en'></xsd:documentation>
95      </xsd:annotation>
96      <xsd:complexType>
97        <xsd:sequence>
98          <xsd:element ref='anbieternr' minOccurs='0'/>
99          <xsd:element ref='firma'/>
100         <xsd:element ref='openimmo_anid'/>
101         <xsd:element ref='lizenzkennung' minOccurs='0'/><!--$V120-->
102         <xsd:element ref='anhang' minOccurs='0'/>
103         <xsd:element ref='immobilie' minOccurs='0' maxOccurs='unbounded'/><!--$V120 min=0-->
104         <xsd:element ref='impressum' minOccurs='0'/>
105         <xsd:element ref='impressum_strukt' minOccurs='0'/><!--$V120-->
106         <xsd:element ref='user_defined_simplefield' minOccurs='0' maxOccurs='unbounded'/>
107         <xsd:element ref='user_defined_anyfield' minOccurs='0' maxOccurs='unbounded'/>
108         <xsd:element ref='user_defined_extend' minOccurs='0' maxOccurs='unbounded'/><!--$V120-->
109       </xsd:sequence>
110     </xsd:complexType>
111   </xsd:element>
112   <xsd:element name='immobilie'>
113     <xsd:annotation>
114       <xsd:documentation>Angaben einer einzelnen Immobile</xsd:documentation>
115 <xsd:documentation xml:lang='en'></xsd:documentation>
116     </xsd:annotation>
117     <xsd:complexType>
118       <xsd:sequence>
119         <xsd:element ref='objektkategorie'/>
120         <xsd:element ref='geo'/>
121         <xsd:element ref='kontaktperson'/>
122         <xsd:element ref='weitere_adresse' minOccurs='0' maxOccurs='unbounded'/>
123         <xsd:element ref='preise' minOccurs='0'/>
124         <xsd:element ref='bieterverfahren' minOccurs='0'/><!--$V120-->
125         <xsd:element ref='versteigerung' minOccurs='0'/><!-- $V122 -->
126         <xsd:element ref='flaechen' minOccurs='0'/>
127         <xsd:element ref='ausstattung' minOccurs='0'/>
128         <xsd:element ref='zustand_angaben' minOccurs='0'/>
129         <xsd:element ref='bewertung' minOccurs='0'/><!--$V120-->
130         <xsd:element ref='infrastruktur' minOccurs='0'/>
131         <xsd:element ref='freitexte' minOccurs='0'/>
132         <xsd:element ref='anhaenge' minOccurs='0'/>
133         <xsd:element ref='verwaltung_objekt' minOccurs='0'/>
134         <xsd:element ref='verwaltung_techn'/>
135         <xsd:element ref='user_defined_simplefield' minOccurs='0' maxOccurs='unbounded'/>
136         <xsd:element ref='user_defined_anyfield' minOccurs='0' maxOccurs='unbounded'/>
137         <xsd:element ref='user_defined_extend' minOccurs='0' maxOccurs='unbounded'/><!--$V120-->
138       </xsd:sequence>
139     </xsd:complexType>
140   </xsd:element>
141   <xsd:element name='objektkategorie'>
142     <xsd:complexType>
143       <xsd:sequence>
144         <xsd:element ref='nutzungsart'/>
145         <xsd:element ref='vermarktungsart'/>
146         <xsd:element ref='objektart'/>
147         <xsd:element ref='user_defined_simplefield' minOccurs='0' maxOccurs='unbounded'/>
148         <xsd:element ref='user_defined_anyfield' minOccurs='0' maxOccurs='unbounded'/>
149         <xsd:element ref='user_defined_extend' minOccurs='0' maxOccurs='unbounded'/><!--$V120-->
150       </xsd:sequence>
151     </xsd:complexType>
152   </xsd:element>
153   <xsd:element name='geo'>
154     <xsd:complexType>
155       <xsd:sequence>
156         <xsd:choice>
157           <xsd:sequence>
158             <xsd:element ref='plz'/>
```

```
159             <xsd:element ref='ort' minOccurs='0'/>
160             <xsd:element ref='geokoordinaten' minOccurs='0'/>
161           </xsd:sequence>
162           <xsd:sequence>
163             <xsd:element ref='ort'/>
164             <xsd:element ref='geokoordinaten' minOccurs='0'/>
165           </xsd:sequence>
166           <xsd:element ref='geokoordinaten'/>
167         </xsd:choice>
168         <xsd:element ref='strasse' minOccurs='0'/>
169         <xsd:element ref='hausnummer' minOccurs='0'/>
170         <xsd:element ref='bundesland' minOccurs='0'/>
171         <xsd:element ref='land' minOccurs='0'/>
172         <xsd:element ref='gemeindecode' minOccurs='0'/>
173         <xsd:element ref='flur' minOccurs='0'/>
174         <xsd:element ref='flurstueck' minOccurs='0'/>
175         <xsd:element ref='gemarkung' minOccurs='0'/>
176         <xsd:element ref='etage' minOccurs='0'/>
177         <xsd:element ref='anzahl_etagen' minOccurs='0'/><!--$V120-->
178         <xsd:element ref='lage_im_bau' minOccurs='0'/>
179         <xsd:element ref='wohnungsnr' minOccurs='0'/>
180         <xsd:element ref='lage_gebiet' minOccurs='0'/>
181         <xsd:element ref='regionaler_zusatz' minOccurs='0'/>
182         <xsd:element ref='karten_makro' minOccurs='0'/><!--$V120-->
183         <xsd:element ref='karten_mikro' minOccurs='0'/><!--$V120-->
184         <xsd:element ref='virtuelletour' minOccurs='0'/><!--$V120-->
185         <xsd:element ref='luftbildern' minOccurs='0'/><!--$V120-->
186         <xsd:element ref='user_defined_simplefield' minOccurs='0' maxOccurs='unbounded'/>
187         <xsd:element ref='user_defined_anyfield' minOccurs='0' maxOccurs='unbounded'/>
188         <xsd:element ref='user_defined_extend' minOccurs='0' maxOccurs='unbounded'/><!--$V120-->
189       </xsd:sequence>
190     </xsd:complexType>
191   </xsd:element>
192   <xsd:element name='kontaktperson'>
193     <xsd:complexType>
194       <xsd:sequence>
195         <xsd:choice>
196           <xsd:sequence>
197             <xsd:element ref='email_zentrale'/>
198             <xsd:element ref='email_direkt' minOccurs='0'/>
199             <xsd:element ref='tel_zentrale' minOccurs='0'/>
200             <xsd:element ref='tel_durchw' minOccurs='0'/>
201             <xsd:element ref='tel_fax' minOccurs='0'/>
202             <xsd:element ref='tel_handy' minOccurs='0'/>
203           </xsd:sequence>
204           <xsd:sequence>
205             <xsd:element ref='email_direkt'/>
206             <xsd:element ref='tel_zentrale' minOccurs='0'/>
207             <xsd:element ref='tel_durchw' minOccurs='0'/>
208             <xsd:element ref='tel_fax' minOccurs='0'/>
209             <xsd:element ref='tel_handy' minOccurs='0'/>
210           </xsd:sequence>
211           <xsd:sequence>
212             <xsd:element ref='tel_zentrale'/>
213             <xsd:element ref='tel_durchw' minOccurs='0'/>
214             <xsd:element ref='tel_fax' minOccurs='0'/>
215             <xsd:element ref='tel_handy' minOccurs='0'/>
216           </xsd:sequence>
217           <xsd:sequence>
218             <xsd:element ref='tel_durchw'/>
219             <xsd:element ref='tel_fax' minOccurs='0'/>
220             <xsd:element ref='tel_handy' minOccurs='0'/>
221           </xsd:sequence>
222           <xsd:sequence>
223             <xsd:element ref='tel_fax'/>
224             <xsd:element ref='tel_handy' minOccurs='0'/>
225           </xsd:sequence>
226           <xsd:element ref='tel_handy'/>
227         </xsd:choice>
228         <xsd:element ref='name'/>
229         <xsd:element ref='vorname' minOccurs='0'/>
230         <xsd:element ref='titel' minOccurs='0'/>
231         <xsd:element ref='anrede' minOccurs='0'/>
232         <xsd:element ref='position' minOccurs='0'/>
233         <xsd:element ref='anrede_brief' minOccurs='0'/>
234         <xsd:element ref='firma' minOccurs='0'/>
235         <xsd:element ref='zusatzfeld' minOccurs='0'/>
236         <xsd:element ref='strasse' minOccurs='0'/>
237         <xsd:element ref='hausnummer' minOccurs='0'/>
238         <xsd:element ref='plz' minOccurs='0'/>
239         <xsd:element ref='ort' minOccurs='0'/>
240         <xsd:element ref='postfach' minOccurs='0'/>
241         <xsd:element ref='postf_plz' minOccurs='0'/>
242         <xsd:element ref='postf_ort' minOccurs='0'/>
243         <xsd:element ref='land' minOccurs='0'/>
244         <xsd:element ref='email_privat' minOccurs='0'/>
245         <xsd:element ref='email_sonstige' minOccurs='0' maxOccurs='unbounded'/>
246         <xsd:element ref='email_feedback' minOccurs='0' /><!--$V123: fuer das Feedback aus dem Portal-->
```

```
247          <xsd:element ref='tel_privat' minOccurs='0'/>
248          <xsd:element ref='tel_sonstige' minOccurs='0' maxOccurs='unbounded'/>
249          <xsd:element ref='url' minOccurs='0'/>
250          <xsd:element ref='adressfreigabe' minOccurs='0'/>
251          <xsd:element ref='personennummer' minOccurs='0'/>
252          <xsd:element ref='immobilientreuhaenderid' minOccurs='0'/><!--$V125 -->
253          <xsd:element ref='foto' minOccurs='0'/><!--$V125 -->
254          <xsd:element ref='freitextfeld' minOccurs='0'/>
255          <xsd:element ref='user_defined_simplefield' minOccurs='0' maxOccurs='unbounded'/>
256          <xsd:element ref='user_defined_anyfield' minOccurs='0' maxOccurs='unbounded'/>
257          <xsd:element ref='user_defined_extend' minOccurs='0' maxOccurs='unbounded'/><!--$V120-->
258        </xsd:sequence>
259      </xsd:complexType>
260    </xsd:element>
261    <xsd:element name='weitere_adresse'>
262      <xsd:complexType>
263        <xsd:sequence>
264          <xsd:element ref='vorname' minOccurs='0'/>
265          <xsd:element ref='name' minOccurs='0'/>
266          <xsd:element ref='titel' minOccurs='0'/>
267          <xsd:element ref='anrede' minOccurs='0'/>
268          <xsd:element ref='anrede_brief' minOccurs='0'/>
269          <xsd:element ref='firma' minOccurs='0'/>
270          <xsd:element ref='zusatzfeld' minOccurs='0'/>
271          <xsd:element ref='strasse' minOccurs='0'/>
272          <xsd:element ref='hausnummer' minOccurs='0'/>
273          <xsd:element ref='plz' minOccurs='0'/>
274          <xsd:element ref='ort' minOccurs='0'/>
275          <xsd:element ref='postfach' minOccurs='0'/>
276          <xsd:element ref='postf_plz' minOccurs='0'/>
277          <xsd:element ref='postf_ort' minOccurs='0'/>
278          <xsd:element ref='land' minOccurs='0'/>
279          <xsd:element ref='email_zentrale' minOccurs='0'/>
280          <xsd:element ref='email_direkt' minOccurs='0'/>
281          <xsd:element ref='email_privat' minOccurs='0'/>
282          <xsd:element ref='email_sonstige' minOccurs='0' maxOccurs='unbounded'/>
283          <xsd:element ref='tel_durchw' minOccurs='0'/>
284          <xsd:element ref='tel_zentrale' minOccurs='0'/>
285          <xsd:element ref='tel_handy' minOccurs='0'/>
286          <xsd:element ref='tel_fax' minOccurs='0'/>
287          <xsd:element ref='tel_privat' minOccurs='0'/>
288          <xsd:element ref='tel_sonstige' minOccurs='0' maxOccurs='unbounded'/>
289          <xsd:element ref='url' minOccurs='0'/>
290          <xsd:element ref='adressfreigabe' minOccurs='0'/>
291          <xsd:element ref='personennummer' minOccurs='0'/>
292          <xsd:element ref='freitextfeld' minOccurs='0'/>
293          <xsd:element ref='user_defined_simplefield' minOccurs='0' maxOccurs='unbounded'/>
294          <xsd:element ref='user_defined_anyfield' minOccurs='0' maxOccurs='unbounded'/>
295          <xsd:element ref='user_defined_extend' minOccurs='0' maxOccurs='unbounded'/><!--$V120-->
296        </xsd:sequence>
297        <xsd:attribute name='adressart' type='xsd:string' use='required'>
298        <!--Frei Angabe ueber die Art der Adresse z.B. Hausmeister, Kontakt -->
299        </xsd:attribute>
300      </xsd:complexType>
301    </xsd:element>
302    <xsd:element name='preise'>
303      <xsd:complexType>
304        <xsd:sequence>
305          <xsd:element ref='kaufpreis' minOccurs='0'/>
306          <xsd:element ref='nettokaltmiete' minOccurs='0'/>
307          <xsd:element ref='kaltmiete' minOccurs='0'/>
308          <xsd:element ref='warmmiete' minOccurs='0'/>
309          <xsd:element ref='nebenkosten' minOccurs='0'/>
310          <xsd:element ref='heizkosten_enthalten' minOccurs='0'/><!--$V120-->
311          <xsd:element ref='heizkosten' minOccurs='0'/>
312          <xsd:element ref='zzg_mehrwertsteuer' minOccurs='0'/>
313          <xsd:element ref='mietzuschlaege' minOccurs='0'/>
314          <xsd:element ref='pacht' minOccurs='0'/>
315          <xsd:element ref='erbpacht' minOccurs='0'/>
316          <xsd:element ref='hausgeld' minOccurs='0'/>
317          <xsd:element ref='abstand' minOccurs='0'/>
318          <xsd:element ref='preis_zeitraum_von' minOccurs='0'/>
319          <xsd:element ref='preis_zeitraum_bis' minOccurs='0'/>
320          <xsd:element ref='preis_zeiteinheit' minOccurs='0'/>
321          <xsd:element ref='mietpreis_pro_qm' minOccurs='0'/>
322          <xsd:element ref='kaufpreis_pro_qm' minOccurs='0'/>
323          <xsd:element ref='provisionspflichtig' minOccurs='0'/><!--$V122-->
324          <xsd:element ref='provision_teilen' minOccurs='0'/><!--$V124-->
325          <xsd:element ref='innen_courtage' minOccurs='0'/>
326          <xsd:element ref='aussen_courtage' minOccurs='0'/>
327          <xsd:element ref='courtage_hinweis' minOccurs='0'/><!--$V123-->
328          <xsd:element ref='waehrung' minOccurs='0'/>
329          <xsd:element ref='mwst_satz' minOccurs='0'/>
330          <xsd:element ref='mwst_gesamt' minOccurs='0'/><!--$V125-->
331          <xsd:element ref='freitext_preis' minOccurs='0'/>
332          <xsd:element ref='x_fache' minOccurs='0'/>
333          <xsd:element ref='nettorendite' minOccurs='0'/><!--$V123: bleibt wegen Kompatibilitaet bestehen, wird in spaeteren
                 Versionen entfernt-->
```

```
334         <xsd:element ref='nettorendite_soll' minOccurs='0'/><!--$V123-->
335         <xsd:element ref='nettorendite_ist' minOccurs='0'/><!--$V123-->
336         <xsd:element ref='mieteinnahmen_ist' minOccurs='0'/>
337         <xsd:element ref='mieteinnahmen_soll' minOccurs='0'/>
338         <xsd:element ref='erschliessungskosten' minOccurs='0'/>
339         <xsd:element ref='kaution' minOccurs='0'/>
340         <xsd:element ref='kaution_text' minOccurs='0'/><!-- $V124 -->
341         <xsd:element ref='geschaeftsguthaben' minOccurs='0'/>
342         <xsd:element ref='stp_carport' minOccurs='0'/>
343         <xsd:element ref='stp_duplex' minOccurs='0'/>
344         <xsd:element ref='stp_freiplatz' minOccurs='0'/>
345         <xsd:element ref='stp_garage' minOccurs='0'/>
346         <xsd:element ref='stp_parkhaus' minOccurs='0'/>
347         <xsd:element ref='stp_tiefgarage' minOccurs='0'/>
348         <xsd:element ref='stp_sonstige' minOccurs='0' maxOccurs='unbounded'/>
349         <xsd:element ref='richtpreis' minOccurs='0'/><!--$V124-->
350         <xsd:element ref='user_defined_simplefield' minOccurs='0' maxOccurs='unbounded'/>
351         <xsd:element ref='user_defined_anyfield' minOccurs='0' maxOccurs='unbounded'/>
352         <xsd:element ref='user_defined_extend' minOccurs='0' maxOccurs='unbounded'/><!--$V120-->
353       </xsd:sequence>
354     </xsd:complexType>
355   </xsd:element>
356   <xsd:element name='bieterverfahren'><!--$V120-->
357     <xsd:annotation>
358       <xsd:documentation>Angaben zum Bieterverfahren</xsd:documentation>
359 <xsd:documentation xml:lang='en'></xsd:documentation>
360     </xsd:annotation>
361     <xsd:complexType>
362       <xsd:sequence>
363         <xsd:element name='beginn_angebotsphase' type='xsd:date' minOccurs='0'/>
364         <xsd:element name='besichtigungstermin' type='xsd:date' minOccurs='0'/>
365         <xsd:element name='besichtigungstermin_2' type='xsd:date' minOccurs='0'/>
366         <xsd:element name='beginn_bietzeit' type='xsd:dateTime' minOccurs='0'/>
367         <xsd:element name='ende_bietzeit' type='xsd:dateTime' minOccurs='0'/>
368         <xsd:element name='hoechstgebot_zeigen' type='xsd:boolean' minOccurs='0'/>
369         <xsd:element name='mindestpreis' type='xsd:decimal' minOccurs='0'/>
370         <xsd:element ref='user_defined_simplefield' minOccurs='0' maxOccurs='unbounded'/>
371         <xsd:element ref='user_defined_anyfield' minOccurs='0' maxOccurs='unbounded'/>
372         <xsd:element ref='user_defined_extend' minOccurs='0' maxOccurs='unbounded'/>
373       </xsd:sequence>
374     </xsd:complexType>
375     <!--$V120  Hinweis: dateTime: 1999-05-31T13:20:00.000-05:00-->
376   </xsd:element>
377   <xsd:element name='versteigerung'><!--$V122-->
378     <xsd:annotation>
379       <xsd:documentation>Angaben zu einer Versteigerung</xsd:documentation>
380 <xsd:documentation xml:lang='en'></xsd:documentation>
381     </xsd:annotation>
382     <xsd:complexType>
383       <xsd:sequence>
384         <xsd:element name='aktenzeichen' type='xsd:string' minOccurs='0'/>
385         <xsd:element name='zvtermin' type='xsd:dateTime' minOccurs='0'/>
386         <xsd:element name='zusatztermin' type='xsd:dateTime' minOccurs='0'/>
387         <xsd:element name='amtsgericht' type='xsd:string' minOccurs='0'/>
388         <xsd:element name='verkehrswert' minOccurs='0'/><!--$V125  -->
389       </xsd:sequence>
390     </xsd:complexType>
391   </xsd:element>
392   <xsd:element name='flaechen'>
393     <xsd:complexType>
394       <xsd:sequence>
395         <xsd:element ref='wohnflaeche' minOccurs='0'/>
396         <xsd:element ref='nutzflaeche' minOccurs='0'/>
397         <xsd:element ref='gesamtflaeche' minOccurs='0'/>
398         <xsd:element ref='ladenflaeche' minOccurs='0'/>
399         <xsd:element ref='lagerflaeche' minOccurs='0'/>
400         <xsd:element ref='verkaufsflaeche' minOccurs='0'/>
401         <xsd:element ref='freiflaeche' minOccurs='0'/>
402         <xsd:element ref='bueroflaeche' minOccurs='0'/>
403         <xsd:element ref='bueroteilflaeche' minOccurs='0'/>
404         <xsd:element ref='fensterfront' minOccurs='0'/>
405         <xsd:element ref='verwaltungsflaeche' minOccurs='0'/>
406         <xsd:element ref='gastroflaeche' minOccurs='0'/>
407         <xsd:element ref='grz' minOccurs='0'/>
408         <xsd:element ref='gfz' minOccurs='0'/>
409         <xsd:element ref='bmz' minOccurs='0'/>
410         <xsd:element ref='bgf' minOccurs='0'/>
411         <xsd:element ref='grundstuecksflaeche' minOccurs='0'/>
412         <xsd:element ref='sonstflaeche' minOccurs='0'/>
413         <xsd:element ref='anzahl_zimmer' minOccurs='0'/>
414         <xsd:element ref='anzahl_schlafzimmer' minOccurs='0'/>
415         <xsd:element ref='anzahl_badezimmer' minOccurs='0'/>
416         <xsd:element ref='anzahl_sep_wc' minOccurs='0'/>
417         <!--  'anzahl_balkon_terrassen' $V123: nicht mehr verwenden , $V124  Entfernt -->
418         <xsd:element ref='anzahl_balkone' minOccurs='0'/><!--$V123-->
419         <xsd:element ref='anzahl_terrassen' minOccurs='0'/><!--$V123-->
420         <xsd:element ref='anzahl_logia' minOccurs='0'/><!--$V125-->
421         <xsd:element ref='balkon_terrasse_flaeche' minOccurs='0'/>
```

```
422         <xsd:element ref='anzahl_wohn_schlafzimmer' minOccurs='0'/>
423         <xsd:element ref='gartenflaeche' minOccurs='0'/>
424         <xsd:element ref='kellerflaeche' minOccurs='0'/>
425         <xsd:element ref='fensterfront_qm' minOccurs='0'/>
426         <xsd:element ref='grundstuecksfront' minOccurs='0'/>
427         <xsd:element ref='dachbodenflaeche' minOccurs='0'/>
428         <xsd:element ref='teilbar_ab' minOccurs='0'/>
429         <xsd:element ref='beheizbare_flaeche' minOccurs='0'/>
430         <xsd:element ref='anzahl_stellplaetze' minOccurs='0'/>
431         <xsd:element ref='plaetze_gastraum' minOccurs='0'/>
432         <xsd:element ref='anzahl_betten' minOccurs='0'/>
433         <xsd:element ref='anzahl_tagungsraeume' minOccurs='0'/>
434         <xsd:element ref='vermietbare_flaeche' minOccurs='0'/>
435         <xsd:element ref='anzahl_wohneinheiten' minOccurs='0'/>
436         <xsd:element ref='anzahl_gewerbeeinheiten' minOccurs='0'/>
437         <xsd:element ref='einliegerwohnung' minOccurs='0'/>
438         <xsd:element ref='user_defined_simplefield' minOccurs='0' maxOccurs='unbounded'/>
439         <xsd:element ref='user_defined_anyfield' minOccurs='0' maxOccurs='unbounded'/>
440         <xsd:element ref='user_defined_extend' minOccurs='0' maxOccurs='unbounded'/><!--$V120-->
441       </xsd:sequence>
442     </xsd:complexType>
443   </xsd:element>
444   <xsd:element name='ausstattung'>
445     <xsd:complexType>
446       <xsd:sequence>
447         <xsd:element ref='ausstatt_kategorie' minOccurs='0'/><!-- $V122-->
448         <xsd:element ref='wg_geeignet' minOccurs='0'/>
449         <xsd:element ref='raeume_veraenderbar' minOccurs='0'/>
450         <xsd:element ref='bad' minOccurs='0'/>
451         <xsd:element ref='kueche' minOccurs='0'/>
452         <xsd:element ref='boden' minOccurs='0'/>
453         <xsd:element ref='kamin' minOccurs='0'/>
454         <xsd:element ref='heizungsart' minOccurs='0'/>
455         <xsd:element ref='befeuerung' minOccurs='0'/>
456         <xsd:element ref='klimatisiert' minOccurs='0'/>
457         <xsd:element ref='fahrstuhl' minOccurs='0'/>
458         <xsd:element ref='stellplatzart' minOccurs='0' maxOccurs='unbounded'/>
459         <xsd:element ref='gartennutzung' minOccurs='0'/>
460         <xsd:element ref='ausricht_balkon_terrasse' minOccurs='0'/>
461         <xsd:element ref='moebliert' minOccurs='0'/>
462         <xsd:element ref='rollstuhlgerecht' minOccurs='0'/>
463         <xsd:element ref='kabel_sat_tv' minOccurs='0'/>
464         <xsd:element ref='dvbt' minOccurs='0'/><!--$V120-->
465         <xsd:element ref='barrierefrei' minOccurs='0'/>
466         <xsd:element ref='sauna' minOccurs='0'/>
467         <xsd:element ref='swimmingpool' minOccurs='0'/>
468         <xsd:element ref='wasch_trockenraum' minOccurs='0'/>
469         <xsd:element ref='wintergarten' minOccurs='0'/><!--$V124-->
470         <xsd:element ref='dv_verkabelung' minOccurs='0'/>
471         <xsd:element ref='rampe' minOccurs='0'/>
472         <xsd:element ref='hebebuehne' minOccurs='0'/>
473         <xsd:element ref='kran' minOccurs='0'/>
474         <xsd:element ref='gastterrasse' minOccurs='0'/>
475         <xsd:element ref='stromanschlusswert' minOccurs='0'/>
476         <xsd:element ref='kantine_cafeteria' minOccurs='0'/>
477         <xsd:element ref='teekueche' minOccurs='0'/>
478         <xsd:element ref='hallenhoehe' minOccurs='0'/>
479         <xsd:element ref='angeschl_gastronomie' minOccurs='0'/>
480         <xsd:element ref='brauereibindung' minOccurs='0'/>
481         <xsd:element ref='sporteinrichtungen' minOccurs='0'/>
482         <xsd:element ref='wellnessbereich' minOccurs='0'/>
483         <xsd:element ref='serviceleistungen' minOccurs='0' maxOccurs='unbounded'/>
484         <xsd:element ref='telefon_ferienimmobilie' minOccurs='0'/>
485         <xsd:element ref='breitband_zugang' minOccurs='0'/><!--$V120-->
486         <xsd:element ref='umts_empfang' minOccurs='0'/><!--$V120-->
487         <xsd:element ref='sicherheitstechnik' minOccurs='0'/>
488         <xsd:element ref='unterkellert' minOccurs='0'/>
489         <xsd:element name='abstellraum' type='xsd:boolean' minOccurs='0'/><!--$V120-->
490         <xsd:element name='fahrradraum' type='xsd:boolean' minOccurs='0'/><!--$V120-->
491         <xsd:element name='rolladen' type='xsd:boolean' minOccurs='0'/><!--$V120-->
492         <xsd:element ref='dachform' minOccurs='0'/><!--$V122-->
493         <xsd:element ref='bauweise' minOccurs='0'/><!--$V122-->
494         <xsd:element ref='ausbaustufe' minOccurs='0'/><!--$V122-->
495         <xsd:element ref='energietyp' minOccurs='0'/><!--$V123-->
496         <xsd:element name='bibliothek' type='xsd:boolean' minOccurs='0'/><!--$V123-->
497         <xsd:element name='dachboden' type='xsd:boolean' minOccurs='0'/><!--$V123-->
498         <xsd:element name='gaestewc' type='xsd:boolean' minOccurs='0'/><!--$V123-->
499         <xsd:element name='kabelkanaele' type='xsd:boolean' minOccurs='0'/><!--$V123-->
500         <xsd:element name='seniorengerecht' type='xsd:boolean' minOccurs='0'/><!--$V123-->
501         <xsd:element ref='user_defined_simplefield' minOccurs='0' maxOccurs='unbounded'/>
502         <xsd:element ref='user_defined_anyfield' minOccurs='0' maxOccurs='unbounded'/>
503         <xsd:element ref='user_defined_extend' minOccurs='0' maxOccurs='unbounded'/><!-- $V120-->
504       </xsd:sequence>
505     </xsd:complexType>
506   </xsd:element>
507   <xsd:element name='zustand_angaben'>
508     <xsd:complexType>
509       <xsd:sequence>
```

```
510         <xsd:element ref='baujahr' minOccurs='0'/>
511         <xsd:element ref='letztemodernisierung' minOccurs='0'/><!-- $V123-->
512         <xsd:element ref='zustand' minOccurs='0'/>
513         <xsd:element ref='alter' minOccurs='0'/>
514         <xsd:element ref='bebaubar_nach' minOccurs='0'/>
515         <xsd:element ref='erschliessung' minOccurs='0'/>
516         <xsd:element ref='erschliessung_umfang' minOccurs='0'/><!-- $V125-->
517         <xsd:element ref='altlasten' minOccurs='0'/>
518         <xsd:element ref='energiepass' minOccurs='0'/><!-- $V120-->
519         <xsd:element ref='verkaufstatus' minOccurs='0'/><!-- $V123-->
520         <xsd:element ref='user_defined_simplefield' minOccurs='0' maxOccurs='unbounded'/>
521         <xsd:element ref='user_defined_anyfield' minOccurs='0' maxOccurs='unbounded'/>
522         <xsd:element ref='user_defined_extend' minOccurs='0' maxOccurs='unbounded'/><!--$V120-->
523       </xsd:sequence>
524     </xsd:complexType>
525   </xsd:element>
526   <xsd:element name='bewertung'><!--$V120-->
527       <xsd:annotation>
528         <xsd:documentation>Container fuer detailierte Bewertungs Parmater</xsd:documentation>
529   <xsd:documentation xml:lang='en'></xsd:documentation>
530       </xsd:annotation>
531       <xsd:complexType>
532         <xsd:sequence>
533           <xsd:element name='feld' minOccurs='0' maxOccurs='unbounded'>
534             <xsd:complexType>
535               <xsd:sequence>
536                 <xsd:element name='name' type='xsd:string'/>
537                 <xsd:element name='wert' type='xsd:string'/>
538                 <xsd:element name='typ' type='xsd:string' minOccurs='0' maxOccurs='unbounded'/>
539                 <xsd:element name='modus' type='xsd:string' minOccurs='0' maxOccurs='unbounded'/>
540               </xsd:sequence>
541             </xsd:complexType>
542           </xsd:element>
543         </xsd:sequence>
544       </xsd:complexType>
545     </xsd:element>
546     <xsd:element name='infrastruktur'>
547       <xsd:complexType>
548         <xsd:sequence>
549           <xsd:element ref='zulieferung' minOccurs='0'/>
550           <xsd:element ref='ausblick' minOccurs='0'/>
551           <xsd:element ref='distanzen' minOccurs='0' maxOccurs='unbounded'/>
552           <xsd:element ref='distanzen_sport' minOccurs='0' maxOccurs='unbounded'/>
553           <xsd:element ref='user_defined_simplefield' minOccurs='0' maxOccurs='unbounded'/>
554           <xsd:element ref='user_defined_anyfield' minOccurs='0' maxOccurs='unbounded'/>
555           <xsd:element ref='user_defined_extend' minOccurs='0' maxOccurs='unbounded'/><!--$V120-->
556         </xsd:sequence>
557       </xsd:complexType>
558     </xsd:element>
559     <xsd:element name='freitexte'>
560       <xsd:complexType>
561         <xsd:sequence>
562           <xsd:element ref='objekttitel' minOccurs='0'/>
563           <xsd:element ref='dreizeiler' minOccurs='0'/>
564           <xsd:element ref='lage' minOccurs='0'/>
565           <xsd:element ref='ausstatt_beschr' minOccurs='0'/>
566           <xsd:element ref='objektbeschreibung' minOccurs='0'/>
567           <xsd:element ref='sonstige_angaben' minOccurs='0'/>
568           <xsd:element ref='user_defined_simplefield' minOccurs='0' maxOccurs='unbounded'/>
569           <xsd:element ref='user_defined_anyfield' minOccurs='0' maxOccurs='unbounded'/>
570           <xsd:element ref='user_defined_extend' minOccurs='0' maxOccurs='unbounded'/><!--$V123-->
571         </xsd:sequence>
572       </xsd:complexType>
573     </xsd:element>
574     <xsd:element name='anhaenge'>
575       <xsd:complexType>
576         <xsd:sequence>
577           <xsd:element ref='anhang' minOccurs='0' maxOccurs='unbounded'/>
578           <xsd:element ref='user_defined_simplefield' minOccurs='0' maxOccurs='unbounded'/>
579           <xsd:element ref='user_defined_anyfield' minOccurs='0' maxOccurs='unbounded'/>
580           <xsd:element ref='user_defined_extend' minOccurs='0' maxOccurs='unbounded'/><!--$V123-->
581         </xsd:sequence>
582       </xsd:complexType>
583     </xsd:element>
584     <xsd:element name='verwaltung_objekt'>
585       <xsd:complexType>
586         <xsd:sequence>
587           <xsd:element ref='objektadresse_freigeben' minOccurs='0'/>
588           <xsd:element ref='verfuegbar_ab' minOccurs='0'/>
589           <xsd:element ref='abdatum' minOccurs='0'/>
590           <xsd:element ref='bisdatum' minOccurs='0'/>
591           <xsd:element ref='min_mietdauer' minOccurs='0'/>
592           <xsd:element ref='max_mietdauer' minOccurs='0'/>
593           <xsd:element ref='versteigerungstermin' minOccurs='0'/>
594           <xsd:element ref='wbs_sozialwohnung' minOccurs='0'/>
595           <xsd:element ref='vermietet' minOccurs='0'/>
596           <xsd:element ref='gruppennummer' minOccurs='0'/>
597           <xsd:element ref='zugang' minOccurs='0'/>
```

```
598           <xsd:element ref='laufzeit' minOccurs='0'/>
599           <xsd:element ref='max_personen' minOccurs='0'/>
600           <xsd:element ref='nichtraucher' minOccurs='0'/>
601           <xsd:element ref='haustiere' minOccurs='0'/>
602           <xsd:element ref='geschlecht' minOccurs='0'/>
603           <xsd:element ref='denkmalgeschuetzt' minOccurs='0'/>
604           <xsd:element ref='als_ferien' minOccurs='0'/>
605           <xsd:element ref='gewerbliche_nutzung' minOccurs='0'/>
606           <xsd:element ref='branchen' minOccurs='0'/>
607           <xsd:element ref='hochhaus' minOccurs='0'/>
608           <xsd:element ref='user_defined_simplefield' minOccurs='0' maxOccurs='unbounded'/>
609           <xsd:element ref='user_defined_anyfield' minOccurs='0' maxOccurs='unbounded'/>
610           <xsd:element ref='user_defined_extend' minOccurs='0' maxOccurs='unbounded'/><!--$V120-->
611         </xsd:sequence>
612       </xsd:complexType>
613     </xsd:element>
614     <xsd:element name='verwaltung_techn'>
615         <xsd:complexType>
616           <xsd:sequence>
617             <xsd:element ref='objektnr_intern' minOccurs='0'/>
618             <xsd:element ref='objektnr_extern'/>
619             <xsd:element ref='aktion'/>
620             <xsd:element ref='aktiv_von' minOccurs='0'/>
621             <xsd:element ref='aktiv_bis' minOccurs='0'/>
622             <xsd:element ref='openimmo_obid'/>
623             <xsd:element ref='kennung_ursprung' minOccurs='0'/>
624             <xsd:element ref='stand_vom'/>
625             <xsd:element ref='weitergabe_generell' minOccurs='0'/>
626             <xsd:element ref='weitergabe_positiv' minOccurs='0'/>
627             <xsd:element ref='weitergabe_negativ' minOccurs='0'/>
628             <xsd:element ref='gruppen_kennung' minOccurs='0'/><!--$V125-->
629             <xsd:element ref='master' minOccurs='0'/><!--$V125-->
630             <xsd:element ref='sprache' minOccurs='0'/><!--$V125-->
631             <xsd:element ref='user_defined_simplefield' minOccurs='0' maxOccurs='unbounded'/>
632             <xsd:element ref='user_defined_anyfield' minOccurs='0' maxOccurs='unbounded'/>
633             <xsd:element ref='user_defined_extend' minOccurs='0' maxOccurs='unbounded'/><!--$V120-->
634           </xsd:sequence>
635         </xsd:complexType>
636       </xsd:element>
637       <!--ab hier die global definierten einzelnen Elemente -->
638       <xsd:element name='openimmo_anid' type='xsd:string'>
639         <xsd:annotation>
640           <xsd:documentation>openimmo-Anbieter ID. Aufbau:
641             Kennbuchstabe: ([O]bjekt|[A]nbieter) {K};
642             OI Mitglied Firmen Kennung, LIN= Lizenznehmer,  3 chars, {P};
643             Timestamp (date-time) 17 Stellen, {YMThmst};
644             Zufall 10 Stellen; {R};
645             Form: KPPPYYYYMMTThhmmsstttRRRRRRRRRR;
646             Bsp:  OABC20011128124930123asd43fer34;
647           </xsd:documentation>
648     <xsd:documentation xml:lang='en'></xsd:documentation>
649         </xsd:annotation>
650         <!--eindeutiger Identifikator des Anbieters innerhalb aller Immobilienboersen-Portale.-->
651         <!--wird aber nicht durch die XSD ueberprueft, da hier in absehbarer Zeit durch Verwendung von Verschluesselung eine
                Aenderungen stattfindet-->
652       </xsd:element>
653       <xsd:element name='lizenzkennung' type='xsd:string'>
654         <xsd:annotation>
655           <xsd:documentation>Lizenzkennung aus der Zuteilung ueber Nutzungslizenz.  Mind 3, Max 8 Chars</xsd:documentation>
656     <xsd:documentation xml:lang='en'></xsd:documentation>
657         </xsd:annotation>
658       </xsd:element>
659       <xsd:element name='strasse' type='xsd:string'>
660         <xsd:annotation>
661           <xsd:documentation>Strasse</xsd:documentation>
662     <xsd:documentation xml:lang='en'></xsd:documentation>
663         </xsd:annotation>
664       </xsd:element>
665       <xsd:element name='hausnummer' type='xsd:string'>
666         <xsd:annotation>
667           <xsd:documentation>Hausnummer, alphanummerisch</xsd:documentation>
668     <xsd:documentation xml:lang='en'></xsd:documentation>
669         </xsd:annotation>
670       </xsd:element>
671       <xsd:element name='plz' type='xsd:string'>
672         <xsd:annotation>
673           <xsd:documentation>PLZ, Pflichtfeld, alternativ mit Ort, Geokoordinaten</xsd:documentation>
674     <xsd:documentation xml:lang='en'></xsd:documentation>
675         </xsd:annotation>
676       </xsd:element>
677       <xsd:element name='ort' type='xsd:string'>
678         <xsd:annotation>
679           <xsd:documentation>Ort, Pflichtfeld, alternativ mit PLZ, Geokoordinaten</xsd:documentation>
680     <xsd:documentation xml:lang='en'></xsd:documentation>
681         </xsd:annotation>
682       </xsd:element>
683       [...]
```

Literaturverzeichnis

[BITZER17-2001] BITZER, FRANK: *Immer Ärger mit dem Datenaustausch: Die Übergabe von Daten zwischen unterschiedlichen Anwendern und Computersystem gehört trotz aller Fortschritte der EDV immer noch zu den großen Hindernissen*. In: Immobilienprofi (2001), Heft 17, S. 21-22

[BITZER18-2001] BITZER, FRANK: *Datenaustausch in der Immobilienwelt (2): Wie kann der Austausch von Daten gestaltet werden, um möglichst viele Partner reibungslos einzubinden.* In: Immobilienprofi (2001), Heft 18, S. 29

[BS-2013] BOOTSTRAP CSS FRAMEWORK: Website, URL: <http://twitter.github.io/bootstrap/>, *abgerufen am: 02.07.2013*

[CLARK-2013] CLARK, JAMES: *XML Path Language (XPath)* (1999), URL: <http://www.w3.org/TR/1999/REC-xpath-19991116/>, *abgerufen am: 15.07.2013*

[EGGERT-2013] EGGERT, RALF: *Zend Framework 2* - 1. Aufl. München: Addison-Wesley, 2013

[FALLSIDE-2001] FALLSIDE, DAVID C.: XML Schema Teil 0: Einführung (2001), URL: <http://www.edition-w3.de/TR/2001/REC-xmlschema-0-20010502/>, *abgerufen am: 10.07.2013*

[IPROFI18-2001] BERGHAUS, WERNER (HRSG.): *XML Kommt: Der Internet-Datenstandard wird früher oder später auch für die Immobilien-Branche gelten. Die Frage stellt sich nur: wann?*. In: Immobilienprofi (2001), Heft 18, S. 30

[IPROFI37-2005] BERGHAUS, WERNER (HRSG.): *Freiheit für Ihre Daten*. In: Immobilienprofi (2005), Heft 37, S. 40

[OI-2013] OPENIMMO E.V. (HRSG.): Website URL: <http://www.openimmo.de>, *abgerufen am: 25.06.2013*

[OIINFO-2013] OPENIMMO E.V. (HRSG.): PDF zum Informationsvortrag, URL: <http://www.openimmo.de/oi_itessen.pdf>, *abgerufen am: 20.07.2013*

[OISTANDARD-2013] OPENIMMO E.V. (HRSG.): Dateien und Dokumentation zum OpenImmo Standard Versionen 1.2.1 bis 1.2.5, URL: <http://www.openimmo.de/cm_std_downs.htm>, Zugriff nach Registrierung möglich *abgerufen am: 15.05.2013*

[OIWP-2013] OPENIMMO E.V. (HRSG.): Whitepaper: URL: <http://www.openimmo.de/oi_info_0405.pdf>, *abgerufen am: 20.07.2013*

[ROMER-2013] ROMER, MICHAEL: *Webentwicklung mit Zend Framework 2* - eBook, Stand: Juni 2013

[SXML-2001] PHP.NET: SimpleXML, URL: <http://php.net/manual/de/book.simplexml.php>, *abgerufen am: 30.07.2013*

[USSATH-2013] USSATH, TOBIAS: *Praktikumsbericht, Clicks Online Business: Internet Marketing Agentur* - Dresden, 2013

[ZF2-2013] ZEND FRAMEWORK 2: Website, URL: <http://framework.zend.com>, *abgerufen am: 25.07.2013*