

Projektdokumentation zum Thema Smartphone-Recycling

Datenmodellierung mit XML

This version:

-

Latest version:

-

Previous versions:

-

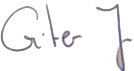
Editors:



Jennifer Steinbinder, 2262360



Tobias Linke, 2432648



Josef Eiter, 2318404

Nürnberg, 16. Januar 2015

Abstract

Diese Dokumentation handelt sich um die Projektarbeit für das Fach Datenmodellierung mit XML. Das Projekt realisiert eine technische Umsetzung einer Smartphone-Recycling Anwendung mit XML und XSLT.

Status des Dokuments

Dieses Dokument ist momentan noch in Bearbeitung.

Inhaltsverzeichnis

- 1 [Einleitung](#)
 - 1.1 [Problemstellung und Ziel](#)
- 2 [Geschäftsprozessmodell](#)
- 3 [Datenmodell](#)
 - 3.1 [Initiales Datenmodell](#)
 - 3.2 [Neues Datenmodell](#)
- 4 [Richtlinien zur Gestaltung der XML Dateien](#)
 - 4.1 [Elemente und Attribute](#)
 - 4.2 [Kommentare](#)
- 5 [Document Type Definition \(DTD\)](#)
 - 5.1 [Gerät](#)
 - 5.2 [Hersteller](#)
 - 5.3 [Kunde](#)
 - 5.4 [Recyclinghof](#)
 - 5.5 [Weitere Definitionen](#)
- 6 [Entwicklung XML-Schemata](#)
 - 6.1 [Aufbau und Struktur der XML Dateien](#)
 - 6.2 [Validierung](#)
 - 6.3 [Testfälle](#)
- 7 [XML Path Language](#)
 - 7.1 [Einarbeitung](#)
 - 7.2 [Document Order](#)
- 8 [Entwicklung der Views mit XSLT](#)
 - 8.1 [Vorbereitung](#)
 - 8.2 [Geräteliste](#)
 - 8.3 [Herstellerübersicht](#)
 - 8.4 [Kundenliste](#)
 - 8.5 [Übersicht Kunden-Recyclinghof fehlende Postleitzahl](#)
 - 8.6 [Anzeige welcher Recyclinghof passt genau zur Postleitzahl des Kunden](#)
 - 8.7 [Anzeige welcher Recyclinghof ist in der Nähe des Kunden](#)
 - 8.8 [Aufbau der Anwendung](#)
 - 8.9 [Probleme](#)
- 9 [Mögliche Erweiterungen und Einsatzbereiche](#)

- 10 [Verwendete Tools](#)
- 11 [Fazit](#)
- 12 [Aufgabenverteilung](#)
- A [Anhang](#)
- A.1 [Präsentation](#)
- A.2 [Quellcode](#)

1 Einleitung

1.1 Problemstellung und Ziel

Die Zuweisung der Projekte führte dazu, dass Josef Eiter, Tobias Linke und Jennifer Steinbinder das Thema "Recycling" zu bearbeiten hatten. Da aktuelle Diskussionen rund um das unnötige Aufbewahren alter Handys geführt werden, hat sich das Projektteam genau auf diese Situation spezialisiert. Das Handyrecycling wird mit Hilfe der Datenmodellierung mit XML zu einem ausführlichen Informationssystem abgebildet. Ziel ist es, einen Überblick über die Datensicht des Informationssystems "Handyrecycling" zu verschaffen.

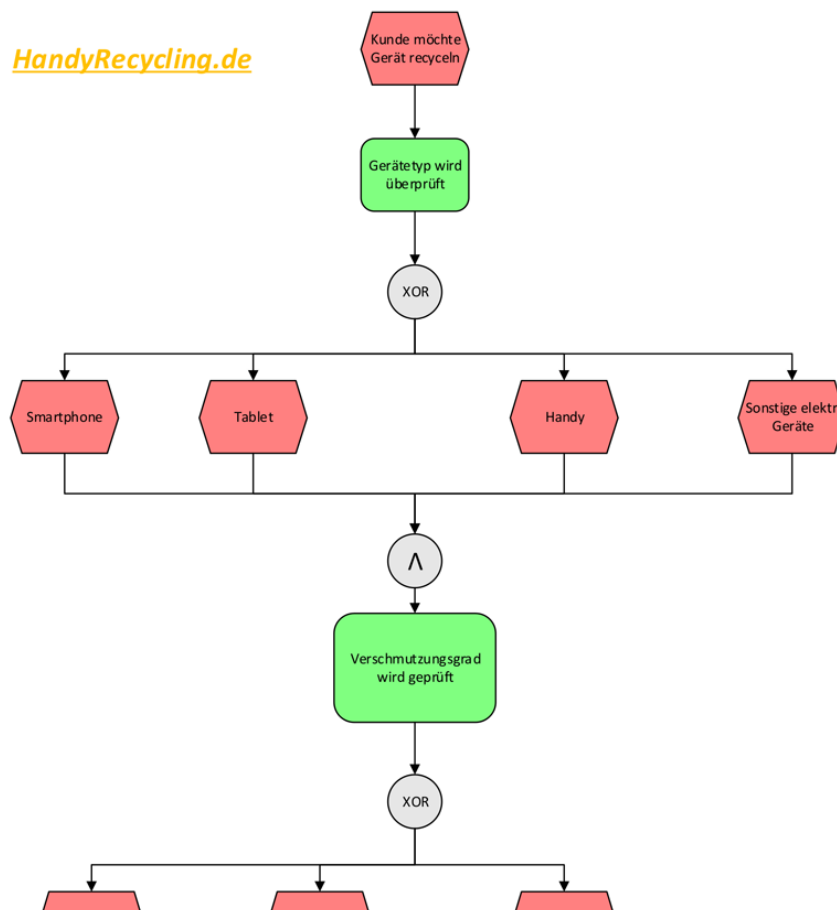
Das Projektteam ist wie folgt strukturiert vorgegangen. Die Phasen der Datenmodellierung wurden gemeinsam erarbeitet. Mit Hilfe eines Geschäftsprozessmodells wurde ein konzeptuelles Schema entworfen. Das konzeptuelle Schema wurde in die logische bzw. relationale Strukturen des Modellierungswerkzeugs überführt. Analog zu den relationalen Strukturen wurden die dazugehörigen XML-Dokumente erstellt. Parallel wurde die Document Type Definition zu jeder XML-Datei deklariert. Anschließend wurde der Umgang mit XPath-Ausdrücke sich angeeignet. Durch das nun verschaffte Know-How der XML Path Language und der Sprache XSLT Views erzeugt.

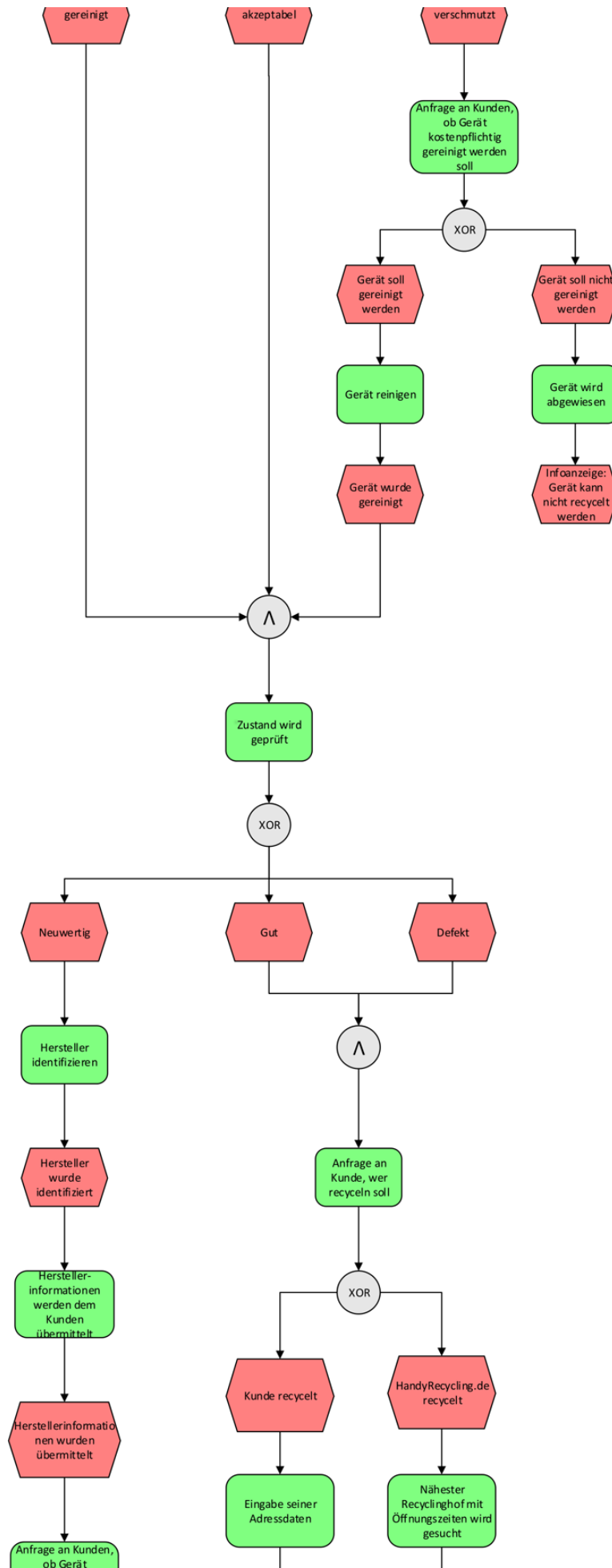
2 Geschäftsprozessmodell

Zur Darstellung des Vorgehens bei unserer Onlineanwendung "Handyrecycling.de" wurde ein EPK-Modell ([siehe Abbildung 1](#)) modelliert. Es beschreibt den Prozess des Handyrecycling-Projekts.

Gedanke:

Der Kunde möchte sein Gerät recycling. Zuerst wird ihm eine Auswahl des Gerätetyps angezeigt: Smartphone, Tablet, Handy oder sonstige elektrische Geräte. Folglich muss er den Verschmutzungsgrad angeben. Dazu kann der Kunde unter folgenden Zuständen auswählen: gereinigt, akzeptabel und verschmutzt. Falls das Gerät verschmutzt ist, wird ihm eine kostenpflichtige Säuberung angeboten. Wird diese verneint, wird ihm in einer Anzeige mitgeteilt, dass sein Gerät leider nicht recycelt werden kann. Bei einem gesäuberten Gerät kann nun unser Recyclingprozess weitergeführt werden. Nun wird der Zustand des Geräts geprüft. Bei einem neuwertigen Gerät wird daraufhin der Hersteller des Geräts identifiziert. Angrenzend entscheidet der Kunde, ob er sein Gerät selbst an den Hersteller verschickt oder Handyrecycling.de kostenpflichtig das Gerät an den Hersteller versendet. Ist jedoch das Gerät stark gebraucht, entscheidet der Kunde, ob er unseren Service, dass das Gerät von uns aus zum Recyclinghof versendet wird, in Anspruch nimmt oder er selber das Gerät recyceln möchte. Entscheidet er sich für das Letztere wird ihm mit Hilfe seiner Adressdaten der nächste Recyclinghof angezeigt.





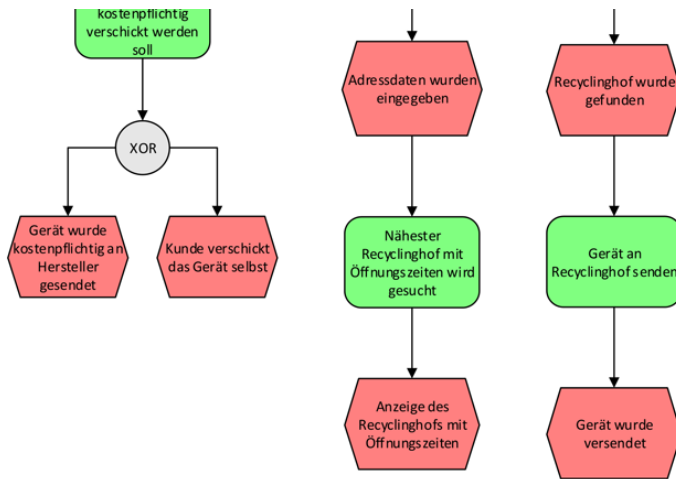


Abbildung 1: EPK Modell Geschäftsprozess

3 Datenmodell

Während der Entwicklung unseres Projekts wurde das Startdatenmodell aufgrund von Unstimmigkeiten und Unklarheiten erweitert und verändert.

3.1 Initiales Datenmodell

Der erste Entwurf (siehe Abbildung 2) hatte ein konkretes Produkt fokussiert, das von einem Kunden zurückgegeben wird. Nach den ersten Tests und Abbildung unserer Anwendungsfälle wurde jedoch klar, dass dieses Datenmodell nicht genau das leistet, was unsere Applikation benötigt. Zudem hatte dieses Modell die Schwachstelle, dass die Geräte immer eindeutig pro Kunde vorhanden sind d. h. es gibt keine Bibliothek von Geräten, die zum Beispiel als Auswahl für die Neuerfassung von Kunden genutzt werden kann.

Das Datenmodell wurde komplett überarbeitet und es wurde mit einigen Elementen erweitert (siehe Abschnitt [3.2 Neues Datenmodell](#)).

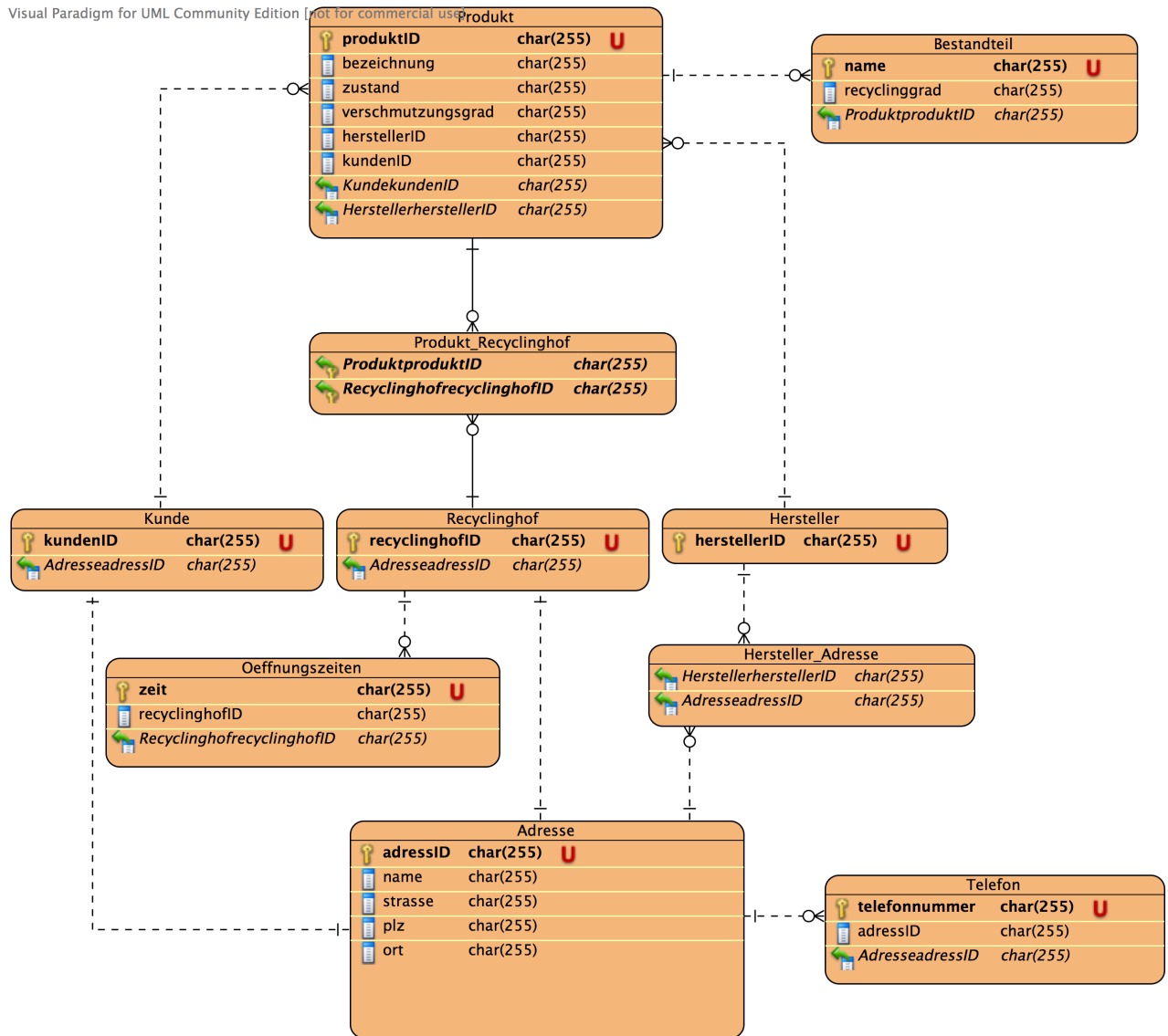


Abbildung 2: Initiales Datenmodell

3.2 Neues Datenmodell

Da unser initiales Datenmodell einige Schwachstellen hatte und nicht genau unseren Anwendungsfällen entsprach musste dieses nochmal überarbeitet werden. Es wurden einige Element hinzugefügt. Irrelevante Informationen wurden weggelassen und das Datenmodell so schlank wie möglich zu halten.

Das Datenmodell in Abbildung 3 enthält nun Elemente die unsere Geräte, Hersteller, Kunden und Recyclinghöfe als Liste speichern. Durch diese kleine aber wichtige Änderung können in unserer Anwendung alle zum Beispiel alle Geräte oder Kunden ausgegeben werden.

Weitere Anpassungen wie zum Beispiel das hinzufügen des Elements Spezialgebiet, das die Spezialisierung eines Recyclinghofs angibt, ist es nun möglich passend auf das Gerät des Kunden zu filtern und einen passenden Recyclinghof zu finden.

Das Erweitern des Elements Bestandteile um die Attribute Anteil und Recyclinggrad ermöglicht es zu einem bestimmten Gerät die Zusammensetzung von Bestandteile anzugeben. Der Recyclinggrad gibt an zu wie viel Prozent das jeweilige Bestandteil recyclebar ist.

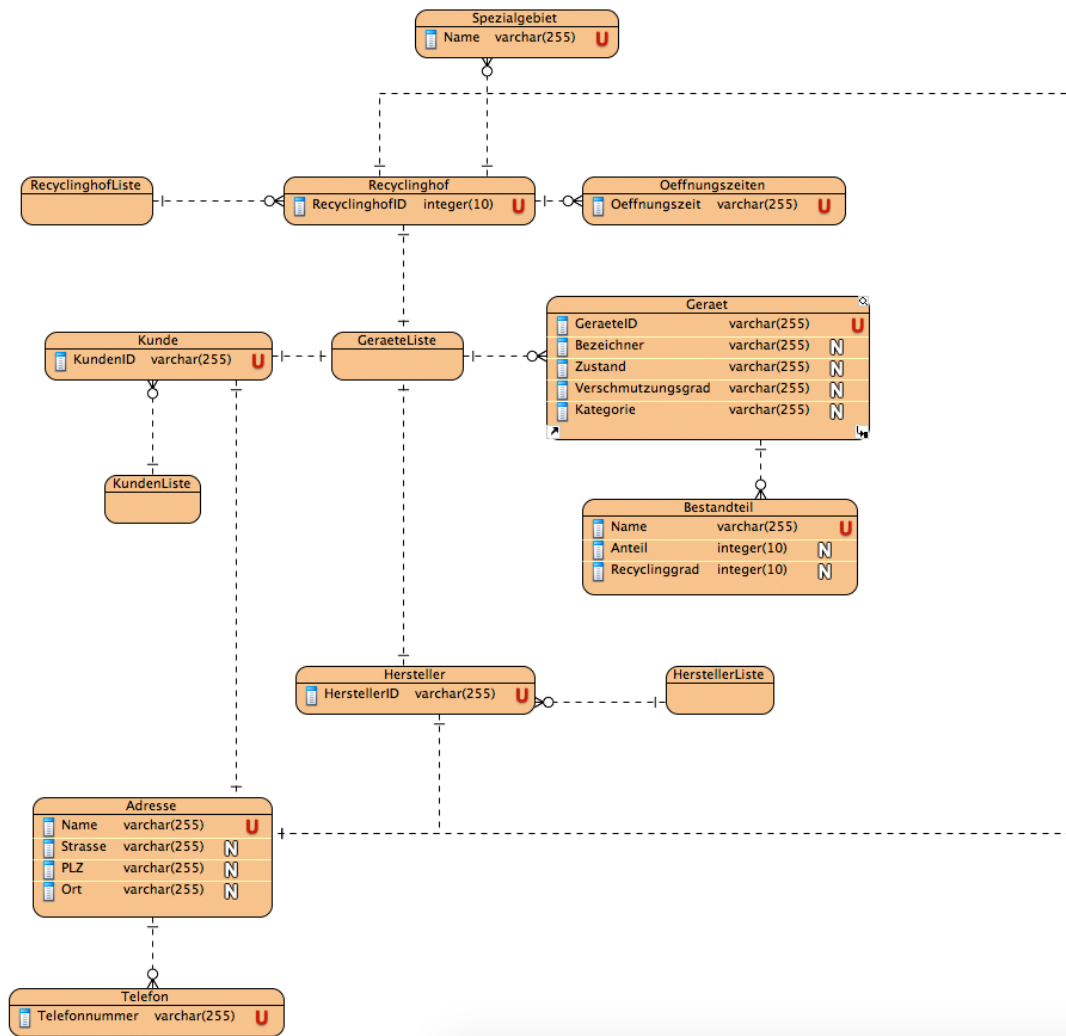


Abbildung 3: Neues Datenmodell

4 Richtlinien zur Gestaltung der XML Dateien

Da gemeinsam an einem Projekt gearbeitet wird, macht es durchaus Sinn gewisse Konventionen für die Gestaltung des XML Codes und allgemeine Regeln festzulegen. Diese Regeln sollten auch bei Änderungen oder Erweiterungen eingehalten werden um die Konsistenz der Datenstruktur zu gewährleisten.

4.1 Elemente und Attribute

Die allgemeine Zeichenkodierung wurde über alle XML-Files hinweg wie folgt festgelegt:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

Da unsere Anwendung nur in Deutschland verwendet werden soll und die Inhalte auf Deutsch sind, sollen die Elementnamen und Attribute auf Deutsch sein. Elementnamen und Attribute sind in der Regel klein zu schreiben. Bei allen XML-Dokumenten wurde ein genaues Wurzelement ausgewählt. Das Informationssystem hat immer die Wurzel "datenbank", in der alle vier Listen eingepflegt werden: geraeteListe, herstellerListe, kundenListe und recyclinghofListe. An diesem Beispiel werden zum Einen die Vorgabe, dass das Element mit einem kleinen Buchstaben beginnt und zum Anderen auf Deutsch gehalten wird, eingehalten. Jedes darauffolgende Wort in demselben Element beginnt mit einem großen Buchstaben: geraeteListe.

Es wurden hier also folgende Regeln beachtet:

- Namen dürfen Buchstaben, Ziffern und andere Zeichen enthalten
- Namen dürfen Groß- und Kleinschreibung beinhalten
- Namen sollten ausdrucksstark und kurz sein

Um den Inhalt mancher Elemente zu beschreiben wurden einerseits PCDATA und andererseits CDATA verwendet. Als Beispiel wird die herstellerListe betrachtet. Parsed Character Data (PCDATA) wurde bei Elementen angewendet, die einen unstrukturierten Inhalt haben. PCDATA sind Dokumentenabschnitte, die der XML-Parser gemäß den XML-Syntax-Regeln verarbeitet. Ein Beispiel für einen Inhalt des Elements "oeffnungszeiten" ist:

```
<oeffnungszeiten>Montag 9.30 Uhr - 18 Uhr</oeffnungszeiten>
<oeffnungszeiten>Dienstag 9.30 Uhr - 18 Uhr</oeffnungszeiten>
```

```
<oeffnungszeiten>Mittwoch 9.30 Uhr - 18 Uhr</oeffnungszeiten>
<oeffnungszeiten>Donnerstag 9.30 Uhr - 18 Uhr</oeffnungszeiten>
<oeffnungszeiten>Freitag 9.30 Uhr - 18 Uhr</oeffnungszeiten>
<oeffnungszeiten>Samstag 9.30 Uhr - 15 Uhr</oeffnungszeiten>
<oeffnungszeiten>Sonntag und Feiertag geschlossen</oeffnungszeiten>
```

Unparsed Character Data (CDATA) wurde in Attributen verwendet. Der Term CDATA markiert Texte, die ein XML-Parser nicht verarbeiten soll. Ein Beispiel eines Attributs, das mit CDATA deklariert worden ist, ist das Attribut recyclinghof:

```
<recyclinghof recyclinghofID="HofMitte">
</recyclinghof>
```

Hier ist der Inhalt der recyclinghofID von der Verarbeitung des XML-Pasers ausgeschlossen. Attribute beschreiben ein Element. Das Element "recyclinghof" wird hier also mit dem Attributnamen "recyclinghofID" und dem Attributwert "HofMitte" beschrieben. Ein Element kann auch beliebig viele Attribute tragen, somit ist die Entscheidung ziemlich schnell getroffen worden, dass beispielsweise das Element "geraet" mehrere Attribute hat, da ein Gerät mehrere Eigenschaften hat:

```
<geraet geraetID="AiP5" geraetBezeichnung="iPhone 5" zustand="neuwertig" verschmutzungsgrad="akzeptabel" geraetekategorie="S"
</geraet>
```

Das Gerätebeispiel hat also eine genaue ID, eine Bezeichnung, einen Zustand, einen Verschmutzungsgrad sowie eine dazugehörige Gerätekategorie. Die Richtlinie, dass Attribute unterschiedliche Namen tragen müssen, wurde strengst beachtet. Die Reihenfolge zwischen den einzelnen Eigenschaften sind in einem Attribut völlig gleichgültig.

Durch das zusätzliche Einfügen von Elementen und Attributen können XML Dokumente leicht erweitert werden. Die Erweiterbarkeit von XML kann Missverständnisse in der Kommunikation unter den Projektmitgliedern auslösen, deswegen einigten sich die Teammitglieder darauf, dass falls ein Name eines Attributs oder Elements schon einmal vergeben worden ist, dieser nun reserviert ist und keinesfalls für eine andere Bedeutung verwendet werden durfte.

Es wurden also folgende Punkte des Well-formed XML eingehalten:

- Jedes Anfangs-Tag muss ein zugehöriges Ende-Tag haben
- Elemente dürfen sich nicht überlappen
- XML-Dokumente haben genau ein Wurzel-Element
- Element-Namen müssen bestimmten Namenskonventionen entsprechen
- XML beachtet grundsätzlich Groß- und Kleinschreibung
- XML belässt Whitespace im Text
- Ein Element darf niemals zwei Attribute mit dem selben Namen beinhalten

4.2 Kommentare

Da Kommentare nicht innerhalb von XML-Tags verwendet werden dürfen, werden diese nach Möglichkeit über den betreffenden XML-Tag verfasst. Die Klammerung der Kommentare erfolgte durch standardgemäßer Form: <!--Kommentar-->. Außerdem wurde vorallem die Angabe befolgt, dass "-" Zeichen in keinem Kommentar verwendet wurden. Im Allgemeinen sollten die Kommentare beschreibend und aussagekräftig sein. Fehlgeschlagene Versuche wurden in den XML Dokumenten beibehalten aber mit Hilfe der Kommentarfunktion auskommentiert, damit sie keinen Einfluss auf die Validierung des XML Dokuments hat. Um die Document Type Definition zu beschreiben wurde auch innerhalb der DTD einige Kommentare erfasst.

5 Document Type Definition (DTD)

Document Type Definitions (DTDs) werden in einem separaten Dokument ausgelagert um die Dateigröße der einzelnen XML-Dokumenten möglichst gering zu halten. Die DTD-Datei(en) werden in das XML Dokument ganz eingebunden. Eine Menge formal überprüfbarer Regeln zur Beschreibung der gemeinsamen strukturellen Merkmale einer Menge von Dokumenten ist ein Dokumententyp. Dementsprechend ist eine DTD eine Grammatik. Das Validieren der Document Typ Definition und der zugehörigen XML Dokumente erfolgte durch das Online-Tool von W3C <http://validator.w3.org>. Da mit verschiedenen Editoren wie Notepad++ und TextEdit gearbeitet worden ist, wurde entschlossen, dass nach jeder Änderung des Files mit dem W3C Validator validiert wird. Die DTD wurde extern lokal im System deklariert:

```
<!DOCTYPE datenbank SYSTEM "datenbank.dtd">
```

Elemente werden in der DTD wie folgt definiert:

XML	DTD
<geraet>	<!ELEMENT geraet>

Die Reihenfolge von Elementen in einem Dokument ist in der DTD fest deklariert:

```
<!ELEMENT adresse (name+ , strasse? , plz? , ort? , telefon+)>
```

Diese Ausdrücke über das Element "adresse" zeigt folgendes: Jede Adresse hat mindestens einen Namen. Die Straße, Postleitzahl sowie der Ort sind optional anzugeben. Jedoch muss mindestens eine Telefonnummer des Kunden angegeben werden. Der Kunde sollte bei Rückfragen oder Notfällen unter einer Telefonnummer erreichbar sein.

Ein Attribut wurde in der DTD wie folgt deklariert:

```
<!ELEMENT geraet (bestandteile+)>
<!--ATTLIST geraet geraetID NMTOKEN #REQUIRED-->
<!--ATTLIST geraet geraetBezeichnung CDATA #REQUIRED-->
<!--ATTLIST geraet zustand (neuwertig | gut | defekt) #REQUIRED-->
<!--ATTLIST geraet verschmutzungsgrad (gereinigt | akzeptabel | verschmutzt) #REQUIRED-->
<!--ATTLIST geraet geraetekategorie CDATA #REQUIRED-->
<!--ATTLIST geraet hersteller CDATA #REQUIRED-->
```

Die Grammatik für das Element Gerät beschreibt hier, welche Attribute und Attributwerte in einem XML-Dokument für das Element "geraet" vorkommen dürfen. Die Attribute geraetID, geraetBezeichnung, zustand, verschmutzungsgrad, geraetekategorie und hersteller wurden außerhalb des betroffenen Elements durch "ATTLIST" deklariert.

Hier haben wir mehrere Attribute, die durch NMTOKEN beschrieben worden sind. Das heißt es sind bei einem NMTOKEN Namen führende Ziffern und Satzzeichen am Anfang des Namens erlaubt. Der Default-Wert #REQUIRED besagt, dass das XML-Dokument an dieser Stelle einen Wert spezifizieren muss.

In der XML-Datei selbst werden die Kindelemente des übergeordneten Elements um einen Tab eingerückt, um die Lesbarkeit der XML-Dokumente zu erhöhen.

Bei der Vergabe von IDs wird der Datentyp NMTOKEN verwendet.

Im Rahmen dieses Projekts wurden keine Entities (Abkürzungen für Zeichenfolgen) verwendet.

5.1 Gerät

Für unseren Anwendungsfall müssen zu den Geräten verschiedene Informationen gespeichert werden.

Die DTD für Geräte sieht wie folgt aus:

```
<!--ELEMENT geraeteListe (geraet*)-->
<!--ATTLIST geraeteListe geraeteListeID NMTOKEN #REQUIRED-->
<!--ELEMENT geraet (bestandteile+)-->
<!--ATTLIST geraet geraetID NMTOKEN #REQUIRED-->
<!--ATTLIST geraet geraetBezeichnung CDATA #REQUIRED-->
<!--ATTLIST geraet zustand (neuwertig | gut | defekt) #REQUIRED-->
<!--ATTLIST geraet verschmutzungsgrad (gereinigt | akzeptabel | verschmutzt) #REQUIRED-->
<!--ATTLIST geraet geraetekategorie CDATA #REQUIRED-->
<!--ATTLIST geraet hersteller CDATA #REQUIRED-->
```

Elemente:

Element	Beschreibung
geraeteListe	Ist ein Container für alle Geräte die in unserer Datenbank vorhanden sind. Mit diesem Container ist es möglich eine Liste der Geräte auszugeben.
geraet	Beschreibt ein bestimmtes Gerät. Es beinhaltet alle relevanten Information zu diesem Gerät.

Attribute geraet:

Attribut	Beschreibung
geraeteListeID	Diese Nummer identifiziert genau eine Geräteleiste eines bestimmten Kunden. Jeder Kunde hat seine eigene Liste, in der jedes abgegebene Gerät vermerkt wird und diese Liste ist mit einer ID beschrieben.
bestandteile+	Ein Gerät besteht aus mehreren Bestandteilen (mindestens einem Bestandteil). Bestandteile können zum Beispiel Glas, Kunststoff oder Kupfer sein. Diese Information ist für den Recyclingvorgang relevant.
geraetID	Ist ein eindeutiger Bezeichner für ein Gerät.
geraetBezeichnung	Bezeichnung ein bestimmtes Geräts. Beispiel: Samsung Galaxy S3
zustand	Beschreibt den Zustand eines Geräts. Der Zustand kann neuwertig, gut oder defekt sein.
verschmutzungsgrad	Beschreibt den Verschmutzungsgrad eines vom Kunden gebrachten Geräts. Ist das Gerät sehr stark verschmutzt wird es ggf. nicht angenommen. Es können gereinigt, akzeptabel oder verschmutzt angegeben werden.
geraetekategorie	Beschreibt ein bestimmtes Gerät. Es beinhaltet alle relevanten Information zu diesem Gerät.
hersteller	Beschreibt den Hersteller des abgegebenen Geräts.

5.2 Hersteller

Alle Geräte besitzen einen Hersteller. Die Hersteller werden in der Datenbank mit einer Adresse gespeichert, da es sein kann, dass ein Gerät beim Hersteller selbst recycelt werden kann.

Die DTD für Hersteller sieht wie folgt aus:


```
<!ELEMENT herstellerListe (hersteller*)>
<!ELEMENT hersteller (adresse+)>
<!ATTLIST hersteller herstellerID CDATA #REQUIRED>
```

Elemente:

Element	Beschreibung
herstellerListe	Ist ein Container für alle Hersteller, die in unserer Datenbank vorhanden sind. Mit diesem Container ist es möglich, eine Liste der Hersteller auszugeben.
hersteller	Beschreibt ein bestimmtes Gerät. Es beinhaltet alle relevanten Information zu diesem Gerät.

Attribute hersteller:

Attribut	Beschreibung
adresse+	Adresse des Herstellers. Er kann mehrere Adressen haben. Es muss aber mindestens eine Adresse angegeben werden.
herstellerID	Ist ein eindeutiger Bezeichner für einen Hersteller.

5.3 Kunde

Für unseren Anwendungsfall müssen zu den Kunden verschiedene Informationen gespeichert werden.

Die DTD für Kunde sieht wie folgt aus:

```
<!ELEMENT kundenListe (kunde*)>
<!ELEMENT kunde (adresse?)>
<!ATTLIST kunde kundenID CDATA #REQUIRED>
<!ATTLIST kunde geraeteListe NMTOKEN #REQUIRED>
```

Elemente:

Element	Beschreibung
kundenListe	Ist ein Container für alle Kunden die in unserer Datenbank vorhanden sind. Mit diesem Container ist es möglich eine Liste der empfangenen Kunden auszugeben.
kunde	Beschreibt einen bestimmten Kunden. Es beinhaltet alle relevanten Information zu diesem Kunden.

Attribute kunde:

Attribut	Beschreibung
kundenID	Jeder Kunde hat eine eindeutige ID, mit der er identifiziert werden kann.
geraeteListe	Ist die Liste der abgegebenen Geräte zu dem zugehörigen Kunden.

5.4 Recyclinghof

Damit die passenden Recyclinghöfe gespeichert und für unsere Anwendung passend angezeigt werden können, werden diese ebenfalls in unsere XML Datenbank mitaufgenommen.

Die DTD für Recyclinghöfe sieht wie folgt aus:

```
<!-- Recyclinghof -->
<!ELEMENT recyclinghofListe (recyclinghof*)>
<!ELEMENT recyclinghof (adresse , oeffnungszeiten+, spezialgebiet+)>
<!ELEMENT oeffnungszeiten (#PCDATA)>
<!ELEMENT spezialgebiet (#PCDATA)>
<!ATTLIST recyclinghof recyclinghofID CDATA #REQUIRED>
```

Elemente:

Element	Beschreibung
recyclinghofListe	Ist ein Container für alle Recyclinghöfe, die in unserer Datenbank vorhanden sind. Mit diesem Container ist es möglich eine Liste von Recyclinghöfen auszugeben.
recyclinghof	Ist der konkrete Recyclinghof. Es beinhaltet alle relevanten Information zu eineme Recyclinghof wie zum Beispiel Adresse und Öffnungszeiten.
oeffnungszeiten	Hier werden die Öffnungszeiten eines Recyclinghofs angegeben.
spezialgebiet	Manche Recyclinghöfe haben sich auf eine bestimmte Geräteart spezialisiert, die sie zurücknehmen. Das Element "spezialgebiet" beinhaltet diese Information.

Attribute recyclinghof:

Attribut	Beschreibung
recyclinghofID	Ist ein eindeutiger Bezeichner für einen Recyclinghof.

5.5 Weitere Definitionen

Neben den Elementen Gerät, Recyclinghof, Hersteller und Kunde gibt es noch Elemente die relevante Informationen wie zum Beispiel Adresse etc. enthalten.

Diese Elemente sind:

- **adresse**

Das Element adresse wird von den Elementen Recyclinghof, Hersteller und Kunde verwendet. Das Element enthält für eine Adresse typische Unterelemente wie zum Beispiel Straße, PLZ, Ort usw.

Die DTD für das Element ist wie folgt aufgebaut:

```
<!ELEMENT adresse (name+ , strasse? , plz? , ort? , telefon+)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT strasse (#PCDATA)>
<!ELEMENT plz (#PCDATA)>
<!ELEMENT ort (#PCDATA)>
<!ELEMENT telefon (#PCDATA)>
```

- **bestandteile**

Die Bestandteile eines bestimmten Geräts werden in dem Element "bestandteile" gespeichert. Dieses Element ist sozusagen ein Container für Elemente vom Typ "bestandteil".

Die DTD für das Element ist wie folgt aufgebaut:

```
<!ELEMENT bestandteile (bestandteil+)>
```

- **bestandteil**

Die Informationen zu den Bestandteilen werden im Element "bestandteil" abgelegt. Das Element beinhaltet wichtige Attribute zu einem Bestandteil wie name, anteil und recyclinggrad.

Die DTD für das Element ist wie folgt aufgebaut:

```
<!ELEMENT bestandteil (#PCDATA)>
<!ATTLIST bestandteil name CDATA #REQUIRED>
<!ATTLIST bestandteil anteil CDATA #REQUIRED>
<!ATTLIST bestandteil recyclinggrad CDATA #REQUIRED>
```

6 Entwicklung XML-Schemata

Bei der Entwicklung des XML-Schemas wurden folgende Anforderungen bearbeitet und erfüllt:

- Flexibilität: Das Schema wird für eine Klasse von Dokumenten entworfen
- Zukunftssicherheit: Das Schema sollte durch seine Struktur zukünftige Ergänzungen und Erweiterungen leicht machen
- Komplexität: Gekoppelt mit Pflegeaufwand und Akzeptanz durch Benutzer

Folgende Phasen der Document-Type-Definition-Entwicklung wurden abgeschlossen:

- Die Vorbereitung: Das Schema wurde vorerst auf Nutzbarkeit, Validierbarkeit und Komplexität geprüft.
- Das Konzeptionelle Schema: Es wurde analysiert, welche Daten benötigt werden. Um das perfekte Ergebnis zu erreichen, hat sich das Team erst einmal den zu bearbeiteten Geschäftsprozess genauestens betrachtet. Welche Fakten werden benötigt? Können diese gruppiert werden? Gibt es bereits Modelle, auf die man zurückgreifen kann? Das Team hat sich stark an den üblichen Recyclingprozess von alltäglichen Geräten orientiert.
- Das Logische Schema: Mit Hilfe eines Datenmodells wurden Anforderungen an das Schema entwickelt. Aus dem Datenmodell konnte nun beispielsweise gelesen werden: welche Datenstrukturen werden nun benötigt? Haben wir obligatorische beziehungsweise fakultative Daten? Welche Strukturen müssen in einer festen Reihenfolge auftreten? Welche Daten gehören in den Inhalt der Elemente und welche zu den Eigenschaften?
- Das Physische Schema: Nun erfolgte die reine Implementierung. Welche Daten werden als Elemente, Attribute,... deklariert?
- Die Nachbereitung und das Testen: Prüfung des Schemas auf Basis bereits vorliegender Dokumente. Einige Aspekte mussten geändert werden. Fehler oder Unklarheiten mussten beseitigt werden. Das Zurückspringen zu einer vorherigen Phase war vor allem auf Grund des neuen Datenmodells auch Teil des Projekts.

6.1 Aufbau und Struktur der XML Dateien

Für den Aufbau der XML Dateien wurde Anfangs eine große Datei mit dem Element "datenbank" als Container für alle Listen angelegt. Diese

Sammlung dient als Grundlage für die Views mit XSLT.

Diese Datei ist wie folgt aufgebaut:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE datenbank SYSTEM "datenbank.dtd">

<datenbank>
  <!--Liste der Geräte-->
  <geraeteListe>
    <!-- Geräte Samsung Galaxy S3-->
    <geraet geraetID="SGS3" geraetBezeichnung="Samsung Galaxy S3" zustand="gut" verschmutzungsgrad="akzeptabel" geraetekategorie="Smartp">
      <bestandteile>
        <bestandteil name="Glas" anteil="50" recyclinggrad="100"/>
        <bestandteil name="Kunststoff" anteil="30" recyclinggrad="80" />
        <bestandteil name="Metall" anteil="20" recyclinggrad="10"/>
      </bestandteile>
    </geraet>
  </geraeteListe>

  <!--Liste der Hersteller-->
  <herstellerListe>
    <hersteller herstellerID="Samsung">
      <adresse>
        <name>Samsung Electrics GmbH</name>
        <strasse>Am Kronberger Hang 6</strasse>
        <plz>65824</plz>
        <ort>Schwalbach am Taunus</ort>
        <telefon>0180 6 67267864</telefon>
      </adresse>
    </hersteller>
  </herstellerListe>

  <!--Liste der Kunden-->
  <kundenListe>
    <kunde kundenID="11112014_01" geraet="iPhone" geraetekategorie="Smartphone">
      <adresse>
        <name>Björn Tim Meßthaler</name>
        <plz>90498</plz>
        <telefon>09156556</telefon>
        <telefon>494164684</telefon>
      </adresse>
    </kunde>
    <kunde kundenID="11112014_05" geraet="Samsung" geraetekategorie="Smartphone">
      <adresse>
        <name>Max Muster-Mann</name>
        <strasse>Musterstr. 1</strasse>
        <plz>12345</plz>
        <ort>Musterstadt</ort>
        <telefon>0911 12 53 55</telefon>
      </adresse>
    </kunde>
    <kunde kundenID="11112014_10" geraet="Nokia" geraetekategorie="Handy">
      <adresse>
        <name>Robin Scherbatsky</name>
        <strasse>Klappenweg 45</strasse>
        <plz>25345</plz>
        <ort>Klappenort</ort>
        <telefon>0911 12 53 55</telefon>
        <telefon>0160 46848647</telefon>
      </adresse>
    </kunde>
  </kundenListe>

  <!--Liste der Recyclinghöfe-->
  <recyclinghofListe>
    <recyclinghof recyclinghofID="HofNord">
      <adresse>
        <name>Recyclinghof Nord</name>
        <strasse>Haeberleinstraße 7</strasse>
        <plz>90411</plz>
        <ort>Nürnberg</ort>
        <telefon>0911 / 36 76 05</telefon>
      </adresse>
      <oeffnungszeiten>Montag 9.30 Uhr - 18 Uhr</oeffnungszeiten>
      <oeffnungszeiten>Dienstag 9.30 Uhr - 18 Uhr</oeffnungszeiten>
      <oeffnungszeiten>Mittwoch 9.30 Uhr - 18 Uhr</oeffnungszeiten>
      <oeffnungszeiten>Donnerstag 9.30 Uhr - 18 Uhr</oeffnungszeiten>
      <oeffnungszeiten>Freitag 9.30 Uhr - 18 Uhr</oeffnungszeiten>
      <oeffnungszeiten>Samstag 9.30 Uhr - 15 Uhr</oeffnungszeiten>
      <oeffnungszeiten>Sonntag und Feiertag geschlossen</oeffnungszeiten>
      <spezialgebiet>smartphone</spezialgebiet>
      <spezialgebiet>Tablet</spezialgebiet>
    </recyclinghof>
    <recyclinghof recyclinghofID="HofMitte">
      <adresse>
        <name>Recyclinghof Mitte</name>
        <strasse>Am Pferdemarkt 23</strasse>
        <plz>90439</plz>
        <ort>Nürnberg</ort>
        <telefon>0911 / 80 138 04</telefon>
      </adresse>
      <oeffnungszeiten>Montag 12 Uhr - 18 Uhr</oeffnungszeiten>
      <oeffnungszeiten>Dienstag 12 Uhr - 18 Uhr</oeffnungszeiten>
```

```

<oeffnungszeiten>Mittwoch 9 Uhr - 18 Uhr</oeffnungszeiten>
<oeffnungszeiten>Donnerstag 9 Uhr - 18 Uhr</oeffnungszeiten>
<oeffnungszeiten>Freitag 12 Uhr - 18 Uhr</oeffnungszeiten>
<oeffnungszeiten>Samstag 8.00 Uhr - 12 Uhr</oeffnungszeiten>
<oeffnungszeiten>Sonntag und Feiertag geschlossen</oeffnungszeiten>
<spezialgebiet>smartphone</spezialgebiet>
</recyclinghof>
</recyclinghofListe>
</datenbank>

```

6.2 Validierung

Für die Validierung der XML-Struktur wurde das Commandline-Tool `xmllint` und der Standard Validator von W3C verwendet. Der Aufbau der entsprechenden XML Datei wird mit ihrer Document Type Definition abgeglichen und so überprüft, ob die Inhalte der Grammatik entsprechen. Dies ist wichtig, um die Datenstruktur zu testen und etwaige Fehler und Schwächen des Datenmodells zu erkennen und auszuschließen.

Das Commandline-Tool `xmllint` wird wie folgt verwendet:

```
xmllint -valid <Pfad zur XML Datei>
```

Im W3C Validator kann die XML Datei und die dazugehörige Document Type Definition direkt in das Textfeld eingegeben werden oder per Dateipfad das XML-Datei mit der beinhaltenden DTD ausgewählt werden.

Wenn Fehler auftreten gibt das Programm `xmllint` diese aus. Bei einer erfolgreichen Validierung wird die XML Datei ohne Formatierung ausgegeben. Falls es Warnungen gibt, gibt das Programm diese vor dem Inhalt, der zu prüfenden XML Datei aus.

Beispiel für einen "parser error":

```

datenbank.xml:11: parser error : internal error: xmlParseInternalSubset: error detected in Markup declaration
&produktListe;
^
datenbank.xml:11: parser error : DOCTYPE improperly terminated
&produktListe;
^
datenbank.xml:11: parser error : Start tag expected, '<' not found
&produktListe;
^

```

6.3 Testfälle

Um die XML Struktur auf Fehler und Schwachstellen zu Testen wurden Testfälle entworfen. Für die Tests wurden XML-Dateien mit fiktiven aber theoretisch möglichen Testdaten erstellt. Diese Test-Dateien wurden validiert und das Ergebnis ausgewertet.

Kriterien für die Tests waren:

- Testdaten werden so entworfen, wie sie auch in einer realen Anwendung vorkommen können.
- Die Ergebnisse der Tests müssen gründlich dokumentiert werden.
- Es muss die Ursache der Fehler, sowie die passende Lösung dokumentiert werden.
- Fehler, die sich aufgrund von Schwächen von XML nicht beheben lassen, müssen detailliert beschrieben werden und es muss ein Workaround dazu gefunden werden.

Folgende Testfälle wurden durchgeführt:

1. Beschreibung:

Das Einsetzen von Sonderzeichen in die KundenID wurde geprüft.

Ergebnis

```
<kunde kundenID ="11112014_01"></kunde>
```

Validierer hatte nichts zu meckern.

2. Beschreibung:

Verschiedensten Arten von Namen wurde in dem Adressblock ausprobiert. Doppelnamen, sowie Namen mit Sonderzeichen wie ö, ä und ü. Folgende Kunden wurden angelegt:

Ergebnis

```

<name>Björn Tim Meßthaler</name>
<name>Max Muster-Mann</name>
<name>Robin Scherbatsky</name>
<name>Lily Aldrin</name>
<name>Ted Mosby</name>

```

Validierer hatte nichts zu meckern, da das Encoding auf ISO-8859-1 gesetzt worden ist.

3. Beschreibung:

Elemente wie Telefon wurden mehr als einmal angelegt und der Inhalt des Elements Telefon wurde in unterschiedlichen Formationen implementiert, da es doch im alltäglichen Leben üblich ist, dass ein Kunde mehr als eine Telefonnummer besitzt (meist ist dies eine Festnetznummer und eine zusätzliche Mobiltelefonnummer).

Ergebnis

```
<telefon>0911 / 12 53 55</telefon>
<telefon>016046848647</telefon>
```

Validierer hatte nichts zu meckern.

4. Beschreibung:

Das Element Adresse wurde beim Testen der KundenListe etwas geändert. Da es durchaus möglich sein könnte, dass ein Kunde uns seine Wohnadresse nicht angeben möchte, muss es hier möglich sein dies zu unterstützen. Zusätzlich aber muss beachtet werden, dass mindestens ein Name, sowie eine Telefonnummer anzugeben ist, da bei Rückfragen es möglich sein muss den Kunden zu kontaktieren.

Ergebnis

Vorher	Nachher
<!ELEMENT adresse (name , strasse , plz , ort , telefon)>	<!ELEMENT adresse (name+ , strasse? , plz? , ort? , telefon+)>
<!ELEMENT name (#PCDATA)>	<!ELEMENT name (#PCDATA)>
<!ELEMENT strasse (#PCDATA)>	<!ELEMENT strasse (#PCDATA)>
<!ELEMENT plz (#PCDATA)>	<!ELEMENT plz (#PCDATA)>
<!ELEMENT ort (#PCDATA)>	<!ELEMENT ort (#PCDATA)>
<!ELEMENT telefon (#PCDATA)>	<!ELEMENT telefon (#PCDATA)>

5. Öffnungszeiten

Beschreibung:

Das Element Öffnungszeiten wurde ebenfalls noch einmal überarbeitet und geändert, um XPath Abfragen beziehungsweise der Umgang mit den Views für einzelne Öffnungszeiten an einem bestimmten Tag zu vereinfachen.

Ergebnis

Vorher	Nachher
<pre><oeffnungszeiten>Mo, Di, Mi, Do, Fr und Sa 9.30 Uhr - 18 Uhr</oeffnungszeiten> <oeffnungszeiten>Sonntag und an Feiertagen geschlossen</oeffnungszeiten></pre>	<pre><oeffnungszeiten>Montag 9.30 Uhr - 18 Uhr</oeffnungszeiten> <oeffnungszeiten>Dienstag 9.30 Uhr - 18 Uhr</oeffnungszeiten> <oeffnungszeiten>Mittwoch 9.30 Uhr - 18 Uhr</oeffnungszeiten> <oeffnungszeiten>Donnerstag 9.30 Uhr - 18 Uhr</oeffnungszeiten> <oeffnungszeiten>Freitag 9.30 Uhr - 18 Uhr</oeffnungszeiten> <oeffnungszeiten>Samstag 9.30 Uhr - 15 Uhr</oeffnungszeiten> <oeffnungszeiten>Sonntag und Feiertag geschlossen</oeffnungszeiten></pre>

6. Weitere Änderungen

Aufgrund unseres neuen Datenmodells mussten einige Anpassungen gemacht werden.

Folgende Änderungen wurden durchgeführt:

Ergebnis

Vorher	Nachher
Attribut war noch nicht angegeben.	<!ATTLIST geraet hersteller CDATA #REQUIRED>
Eine ID für die Geräteliste war vorher noch nicht vergeben.	<!ATTLIST geraeteListe geraeteListeID NMTOKEN #REQUIRED>
Die Attribute "geraet" und "geraetekategorie" wurde entfernt und durch "geraeteListe" ersetzt.	<kunde kundenID="4641653" geraeteListe="liste1">

<kunde kundenID="4641653" geraet="AI5" geraetekategorie="Smartphone">	
Spezialgebiet war vorher noch nicht vorhanden.	<!ELEMENT spezialgebiet (#PCDATA)>

7 XML Path Language

XPath ist eine vom W3C-Konsortium entwickelte Anfragesprache um Teile von XML-Dokumenten zu adressieren und auszuwerten (Angelehnt an Wikipedia, abgerufen am 01.01.2015). Sie arbeitet auf dem DOM eines XML-Dokuments und ermöglicht: eine Navigation, eine Wahl von Knotenmengen und Auswertungsoperationen.

7.1 Einarbeitung

Mit Hilfe von XPath-Ausdrücken können Teilmengen eines XML-Dokuments bestimmt werden. Dadurch werden Ausdrücke evaluiert und das Ergebnis der Evaluierung ist ein Wert mit Typ node-set, string, boolean oder number. Um eine Knotenmenge zu spezifizieren, gibt man einen Weg von einem anderen Knoten aus an (meistens der Wurzel). Sie sind untergliedert in mit "/" getrennten Schritte. Ein Schritt setzt sich zusammen aus:

Achse: :Knotentest[Prädikat]

Um sich etwas in die XML Path Language einzuarbeiten, wurde mit Hilfe des Notepad++ XML Tools "XPath Expression Evaluation" einige Abfragen durchgeführt. Die meisten XSLT-Views wurden mit dem selben Gedanken beziehungsweise dem selben Ziel wie die unten aufgezeigten XPath-Ausdrücke erzeugt.

Beispiele:

Mit Hilfe des XPath-Ausdrucks: `/datenbank/geraeteListe/child::*` konnten alle bisher abgegebenen Geräte angezeigt werden (siehe Abbildung 4). Es werden hier nun alle fünf Geräte, die wir bis dahin erhalten haben, angezeigt.

The screenshot shows an XML document with the following structure:

```

1 <?xml version="1.0" encoding="ISO-8859-1"?>
2 <?xml-stylesheet type="text/xsl" href="responsive-neu.xsl"?>
3 <!DOCTYPE datenbank SYSTEM "datenbank.dtd">
4
5 <datenbank>
6   <!-- Liste der Geräte -->
7   <geraeteListe>
8     <!-- Gerät iPhone akzeptabel -->
9     <geraet geraetID="AiP5" geraetBezeichnung="iPhone 5" zustand="neuwertig" verschmutzungsgrad="akzeptabel" geraetekategorie="Smartphone">
10      <bestandteile>
11        <bestandteil name="Glas" anteil="60" recyclinggrad="100"/>
12        <bestandteil name="Kunststoff" anteil="10" recyclinggrad="80"/>
13        <bestandteil name="Metall" anteil="30" recyclinggrad="10"/>
14      </bestandteile>
15    </geraet>
16
17    <!-- Gerät iPhone nicht verschmutzt -->
18    <geraet geraetID="AiP5" geraetBezeichnung="iPhone 5" zustand="defekt" verschmutzungsgrad="akzeptabel" geraetekategorie="Smartphone">
19      <bestandteile>
20        <bestandteil name="Glas" anteil="60" recyclinggrad="100"/>
21        <bestandteil name="Kunststoff" anteil="10" recyclinggrad="80"/>
22        <bestandteil name="Metall" anteil="30" recyclinggrad="10"/>
23      </bestandteile>
24    </geraet>
25
26    <!-- Geräte Samsung Galaxy S3 -->
27    <geraet geraetID="SGS3" geraetBezeichnung="Samsung Galaxy S3" zustand="gut" verschmutzungsgrad="akzeptabel" geraetekategorie="Smartphone">
28      <bestandteile>
29        <bestandteil name="Glas" anteil="50" recyclinggrad="100"/>
30        <bestandteil name="Kunststoff" anteil="30" recyclinggrad="80"/>
31        <bestandteil name="Metall" anteil="20" recyclinggrad="10"/>
32      </bestandteile>
33    </geraet>
34
35    <!-- Nokia 3110 -->

```

The XPath Expression Evaluation window shows the expression `/datenbank/geraeteListe/child::*` evaluated to a set of five nodes, each of type `geraet`. The values are:

Type	Name	Value
Node	geraet	geraetID="AiP5" geraetBezeichnung="iPhone 5" zustand="neuwertig" verschmutzungsgrad="akzeptabel" geraetekategorie="Smartphone"
Node	geraet	geraetID="AiP5" geraetBezeichnung="iPhone 5" zustand="defekt" verschmutzungsgrad="akzeptabel" geraetekategorie="Smartphone"
Node	geraet	geraetID="SGS3" geraetBezeichnung="Samsung Galaxy S3" zustand="gut" verschmutzungsgrad="akzeptabel" geraetekategorie="Smartphone"
Node	geraet	geraetID="N3110" geraetBezeichnung="Nokia 3110" zustand="defekt" verschmutzungsgrad="akzeptabel" geraetekategorie="Smartphone"
Node	geraet	geraetID="LenovoYogaPad2" geraetBezeichnung="Lenovo Yoga Pad 2" zustand="gut" verschmutzungsgrad="akzeptabel" geraetekategorie="Tablet"

Abbildung 4: Liste der bisher abgegebenen Geräte

Mit Hilfe des XPath-Ausdrucks: `/datenbank/kundenListe//name` konnten alle Kundennamen angezeigt werden (siehe Abbildung 5). Es werden hier nun alle fünf Kunden, die wir bis dahin versorgt hatten, angezeigt.



Abbildung 5: Liste der bisher bedienten Kunden

Mit Hilfe des XPath-Ausdrucks: `/datenbank/herstellerListe/hersteller` konnten alle Hersteller mit ihrer eindeutigen ID angezeigt werden (siehe [Abbildung 6](#)). Es werden hier nun alle fünf Hersteller, die wir bis dahin vermerkt hatten, angezeigt.

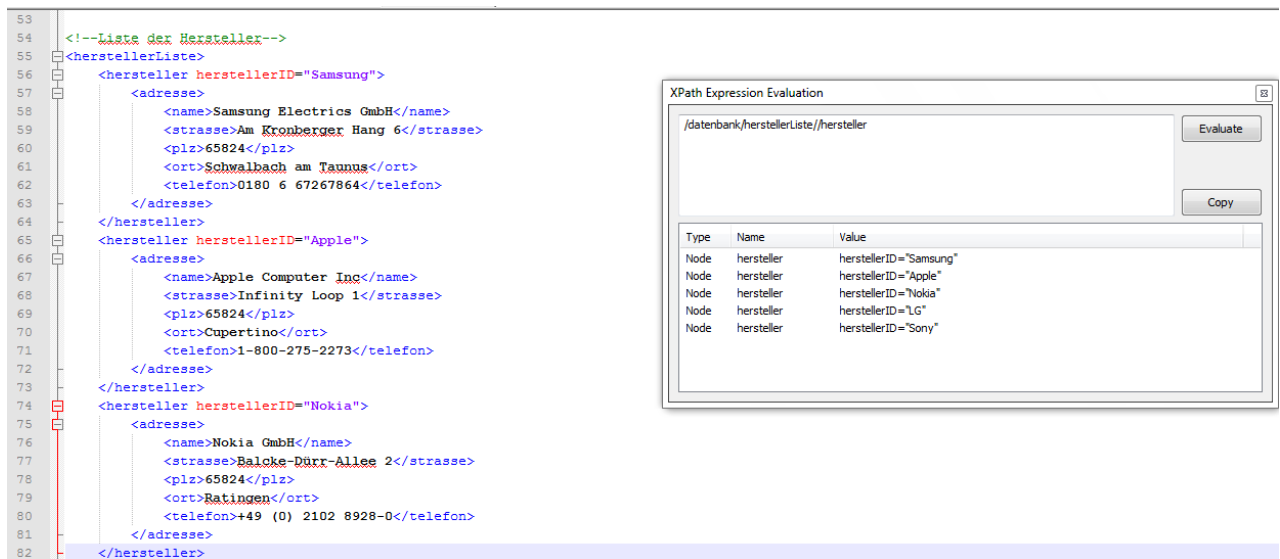


Abbildung 6: Liste der bisher vermerkten Hersteller

Mit Hilfe des XPath-Ausdrucks: `/kundenListe/child::*/adresse/plz` konnten alle Postleitzahlen unserer Kunden angegeben werden (siehe [Abbildung 7](#)).

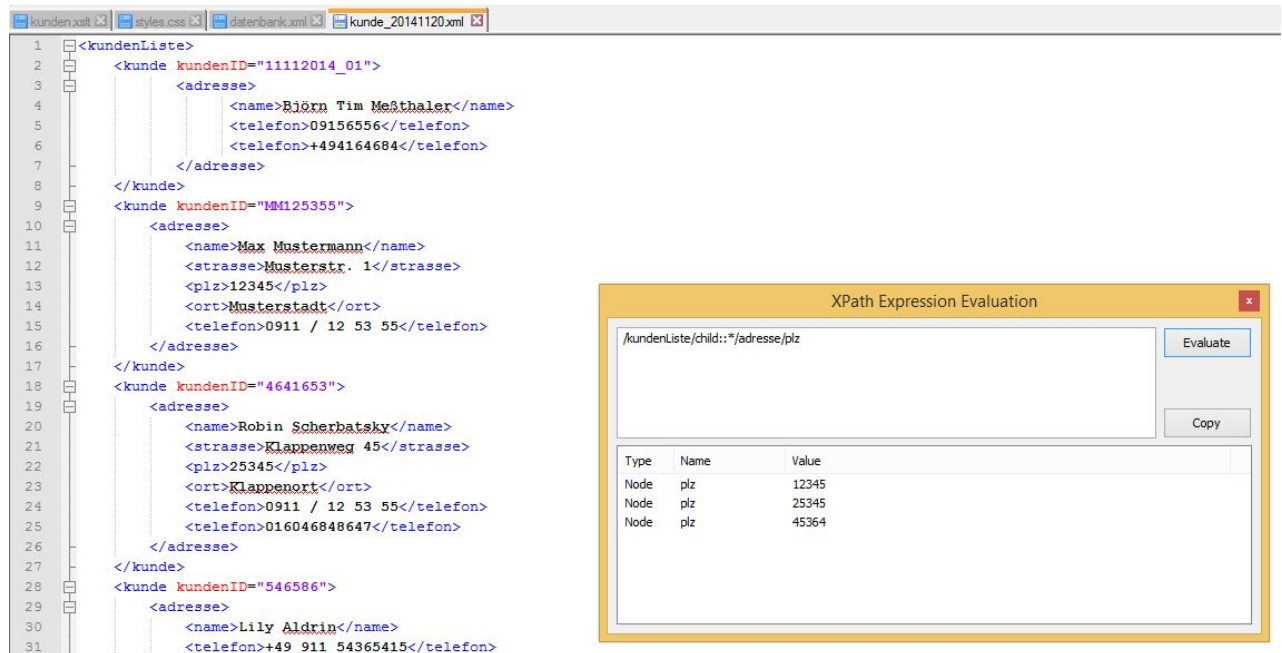


Abbildung 7: Liste der Postleitzahlen unserer Kunden

Mit Hilfe des XPath-Ausdrucks: `/datenbank/recyclinghofListe/recyclinghof[@=recyclinghofID="HofNord"]/oeffnungszeiten` konnten die Öffnungszeiten des Recyclinghofs Nord angezeigt werden (siehe Abbildung 8).

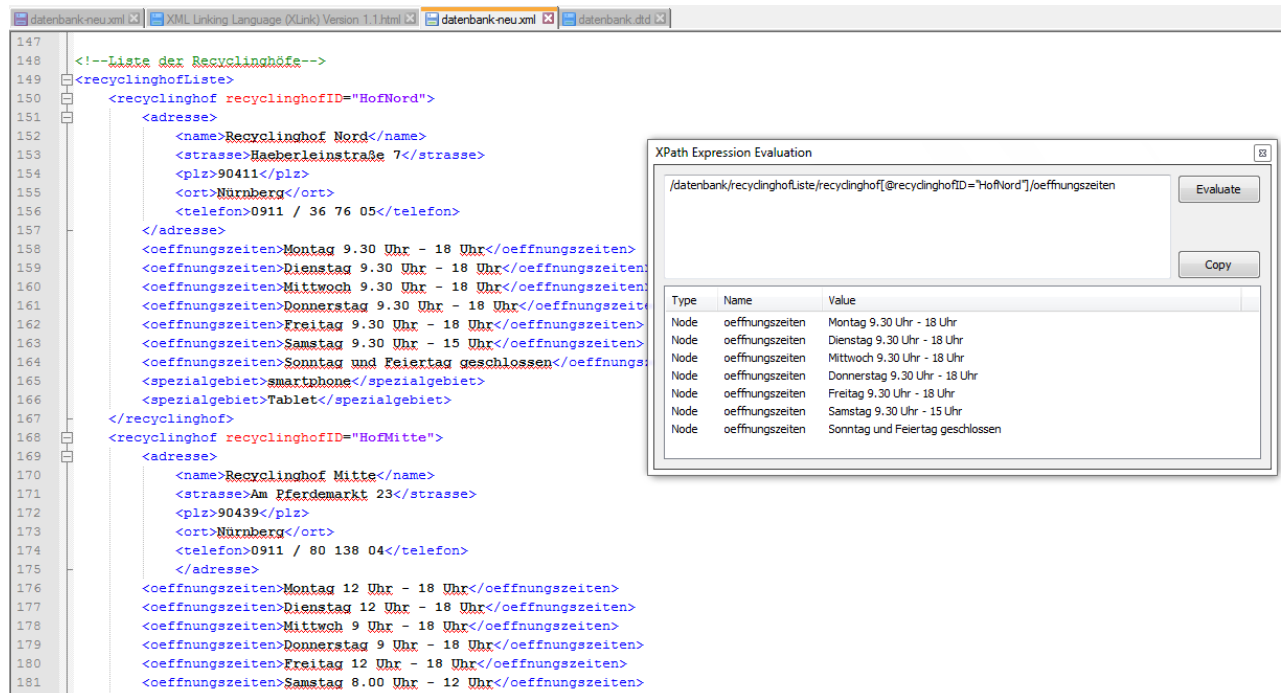


Abbildung 8: Anzeige der Öffnungszeiten des Recyclinghofs Nord

7.2 Document Order

Die Document Order beschreibt eine Ordnung über alle Knoten eines DOM-Dokuments. Das Wurzelement ist das erste Element der Ordnung. Die Elementknoten kommen vor ihren Kindern. Kinder werden nach der Folge der XML-Quelle geordnet. Attributordnungen sind implementierungsabhängig. Eine Document Order ist immer dann sehr vom Vorteil, wenn ein XML Dokument immer mehr an Größe erreicht. Momentan ist die Datenbank des Projektteams noch recht übersichtlich aber im Laufe der Zeit und auch durch Erweiterungen kann es hilfreich sein, eine Übersicht aller Knoten in diesem Dokument zu erhalten. Auf der Website: <http://xmlgrid.net/> wurde aus unserer Datenbank eine Baumstruktur aller Knoten erstellt. Diese Baumstruktur kann jederzeit erweitert werden.

Es folgt eine Baumstruktur, die die erste Ebene unserer Datenbank beschreibt. Zu Sehen ist das Rotelement "datenbank" und ihre darauffolgenden Kinder: gerateListe, herstellerListe, kundenListe und recyclinghofListe. Auch der Validierer dieser Website erkannte hier ein well-formed XML (siehe Abbildung 9).

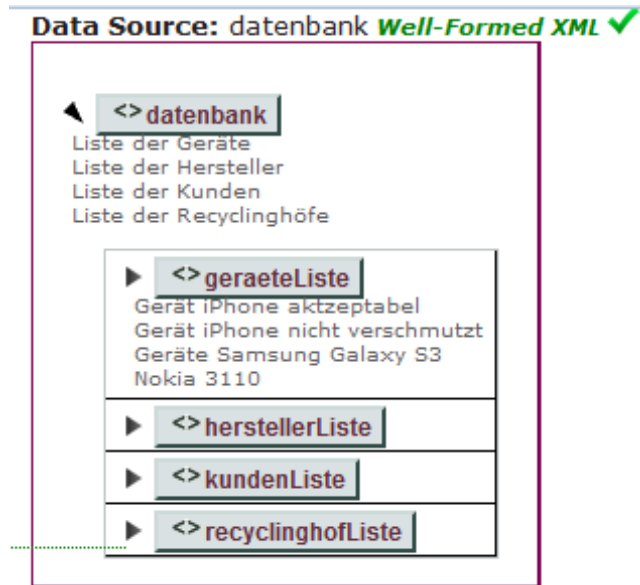


Abbildung 9: Wurzelement und die erste Ebene ihrer Kinder

In der nächsten Abbildung ([siehe Abbildung 10](#)) sind die weiteren Kinder der geraeteListe zu erkennen. Die eingetragenen Geräte werden hier in einer übersichtlichen Ansicht dargestellt. Zusätzlich sind auch kleine Angaben wie die Summe der Geräte(5) angegeben. Hier sind alle Attribute und Elemente eines Geräts in Spaltenansicht angezeigt. Auch die einzelnen Bestandteile sind aufgelistet.

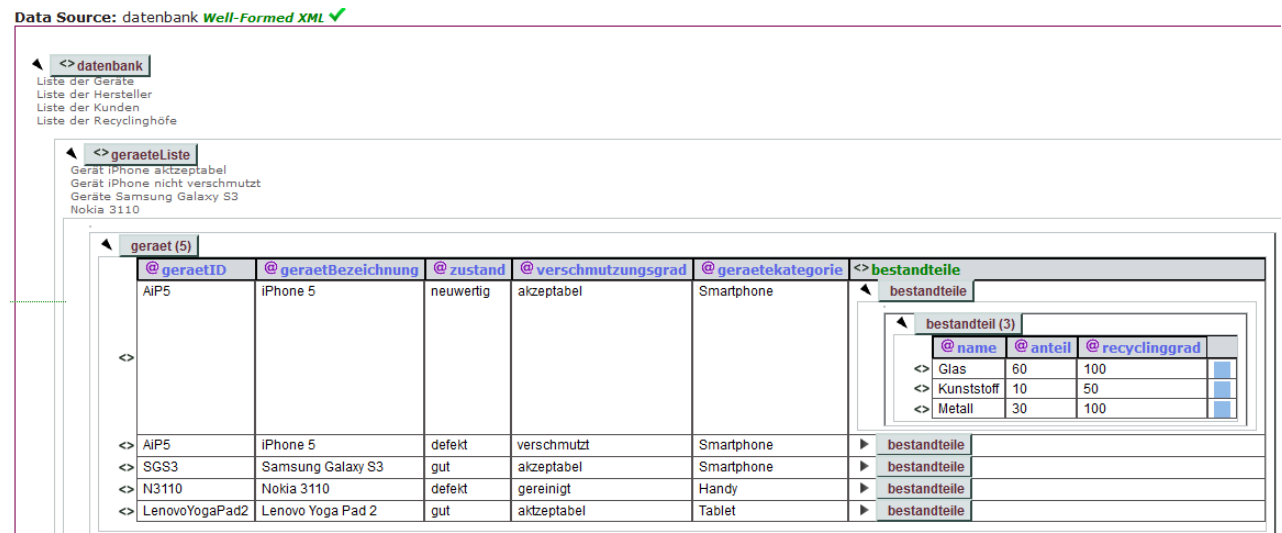


Abbildung 10: Kinder der Geräteliste

In der folgenden Abbildung ([siehe Abbildung 11](#)) sind nun die Kundenliste und die Herstellerliste als Baumstruktur abgebildet. Die Kinderelemente der Adresse werden ebenfalls in der Baumstruktur abgebildet.

<> herstellerListe

hersteller (5)

@ herstellerID	<> adresse										
Samsung	<div>adresse</div> <table> <tr><td><> name</td><td>Samsung Electrics GmbH</td></tr> <tr><td><> strasse</td><td>Am Kronberger Hang 6</td></tr> <tr><td><> plz</td><td>65824</td></tr> <tr><td><> ort</td><td>Schwalbach am Taunus</td></tr> <tr><td><> telefon</td><td>0180 6 67267864</td></tr> </table>	<> name	Samsung Electrics GmbH	<> strasse	Am Kronberger Hang 6	<> plz	65824	<> ort	Schwalbach am Taunus	<> telefon	0180 6 67267864
<> name	Samsung Electrics GmbH										
<> strasse	Am Kronberger Hang 6										
<> plz	65824										
<> ort	Schwalbach am Taunus										
<> telefon	0180 6 67267864										
<> Apple	> adresse										
<> Nokia	> adresse										
<> LG	> adresse										
<> Sony	> adresse										

<> kundenListe

kunde (5)

@ kundenID	@ geraet	@ geraetekategorie	<> adresse								
11112014_01	iPhone	Smartphone	<div>adresse</div> <table> <tr><td><> name</td><td>Björn Tim Meßthaler</td></tr> <tr> <td><> telefon (2)</td> <td> <table> <tr><td><></td><td>09156556</td></tr> <tr><td><></td><td>+494164684</td></tr> </table> </td> </tr> </table>	<> name	Björn Tim Meßthaler	<> telefon (2)	<table> <tr><td><></td><td>09156556</td></tr> <tr><td><></td><td>+494164684</td></tr> </table>	<>	09156556	<>	+494164684
<> name	Björn Tim Meßthaler										
<> telefon (2)	<table> <tr><td><></td><td>09156556</td></tr> <tr><td><></td><td>+494164684</td></tr> </table>	<>	09156556	<>	+494164684						
<>	09156556										
<>	+494164684										
<> MM125355	Samsung	Smartphone	> adresse								
<> 4641653	Nokia	Handy	> adresse								
<> 546586	Lenovo Yoga Pad 2	Tablet	> adresse								
<> 23654	iPhone 6Plus	Smartphone	> adresse								

Abbildung 11: Kinder der Kunden- und Herstellerliste

Die letzte Abbildung (siehe [Abbildung 12](#)) zeigt die Übersicht der Recyclingofliste. Die Kinderelemente Adresse, Öffnungszeiten und Spezialgebiete werden mitabgebildet.

<> recyclinghofListe			
recyclinghof (4)			
@recyclinghofID	<> adresse	<> oeffnungszeiten	<> spezialgebiet
HofNord	<div>adresse<div><div><> name</div>Recyclinghof Nord<div><> strasse</div>Haeberleinstraße 7<div><> plz</div>90411<div><> ort</div>Nürnberg<div><> telefon</div>0911 / 36 76 05</div></div>	<div>oeffnungszeiten (7)<div><div><></div>Montag 9.30 Uhr - 18 Uhr<div><></div>Dienstag 9.30 Uhr - 18 Uhr<div><></div>Mittwch 9.30 Uhr - 18 Uhr<div><></div>Donnerstag 9.30 Uhr - 18 Uhr<div><></div>Freitag 9.30 Uhr - 18 Uhr<div><></div>Samstag 9.30 Uhr - 15 Uhr<div><></div>Sonntag und Feiertag geschlossen</div></div>	<div>spezialgebiet (2)<div><div><></div>smartphone<div><></div>Tablet</div></div>
<> HofMitte	> adresse	> oeffnungszeiten (7)	smartphone
<> HofSued	> adresse	> oeffnungszeiten (7)	handy
<> HofOst	> adresse	> oeffnungszeiten (7)	> spezialgebiet (3)

Abbildung 12: Kinder der Recyclinghofliste

8 Entwicklung der Views mit XSLT

Um die Daten aus unserer XML-Datenstruktur benutzerfreundlich zu repräsentieren wurden Ansichten (Views) mit XSLT (Extensible Stylesheet Language Transformations) entwickelt. Mit XSLT ist es möglich mit einfachen Mitteln Daten aus einer oder mehreren XML-Dateien in einem Browser formatiert darzustellen.

Bei der Entwicklung wurde darauf geachtet, dass die Darstellung auf mobilen Geräten sowie Desktop PCs gut leserlich und passend ist.

Im Rahmen unseres Projekts wurden mehrere verschiedene Views erstellt, die unsere Daten darstellen. Diese Ansichten werden in den folgenden Abschnitten beschrieben.

8.1 Vorbereitung

8.1.1 CSS

Um ein möglichst übersichtliches XSL-Dokument zu erhalten haben wir so gut wie alle Styles in ein externes CSS-Dokument ausgelagert

```
&#x0000; charset "UTF-8";
```

```
/* CSS Document */
/* Cool */

html {
    font-family: Arial, Helvetica, sans-serif;
}

body {
    background-color: #EBE8E4;
    color: #222;
    font-family: "HelveticaNeue-Light", "Helvetica Neue Light", "Helvetica Neue", Helvetica, Arial, "Lucida Grande", sans-serif;
    font-weight: 300;
    font-size: 12px;
    margin: 20px;
}

a {
    color: #000;
    text-decoration: none;
}

a:hover {
    text-decoration: underline;
}

h1 {
    font-size: 15px;
    padding: 0;
    margin: 0;
    margin: 0px;
    padding: 0px;
    color: #333;
    text-transform: uppercase;
    /*border-bottom: solid 1px #CCC;*/
}

h2 {
    font-size: 12px;
    padding: 0;
    margin: 0;
    padding: 0px;
    color: #666;
    text-transform: uppercase;
    border-bottom: solid 1px #CCC;
}

h3 {
    font-size: 12px;
    padding: 0;
    margin: 0;
    margin: 15px 0px 10px 0px;
    padding-bottom: 5px;
    color: #999;
    text-transform: uppercase;
    border-bottom: dotted 1px #CCC;
}

nav {
    display: block;
    overflow: hidden;
    width: 100%;
    height: 40px;
}

nav ul {
    margin: 0;
    padding: 5px;
}

nav ul li {
    display: inline-block;
    list-style-type: none;
    margin-right: 10px;
}

#container {
    position: relative;
    margin: 0 auto;
    width: 100%;
}

#head {
    position: relative;
    float: left;
    width: 100%;
    height: 100px;
}

#links {
    position: relative;
    float: left;
    padding-left: 5px;
    width: 100%;
    min-height: 100px;
    border: solid 1px #CCC;
    background-color: #FEEFEFE;
```

```

        margin-right:10px;
        margin-bottom: 10px;
    }

    #rechts {
        position:relative;
        float:left;
        padding-left:5px;
        width: 100%;
        min-height: 100px;
        border: solid 1px #CCC;
        background-color: #FEEFEFE;
        margin-right:10px;
        margin-bottom: 10px;
    }

    @media only screen and (min-width: 768px) {

    nav {
        display: block;
        overflow: hidden;
        width: 90%;
    }

    nav ul {
        margin: 5px;
        padding: 5px;
    }

    nav ul li {
        display: inline-block;
        list-style-type: none;
        padding-right: 10px;
    }

    #container {
        position: relative;
        margin:0 auto;
        width: 768px;
    }

    #head {
        position:relative;
        float:left;
        width: 768px;
        height:100px;
    }

    #links {
        position:relative;
        float:left;
        padding:5px;
        width: 300px;
        min-height: 100px;
        border: solid 1px #CCC;
        background-color: #FEEFEFE;
        margin:5px;
    }

    #rechts {
        position:relative;
        float:left;
        padding:5px;
        width: 300px;
        min-height: 100px;
        border: solid 1px #CCC;
        background-color: #FEEFEFE;
        margin:5px;
    }

    p {
        margin:0px 0px 5px 0px;
        padding:0px;
    }

    ul {
        margin:5px;
        padding: 0px 0px 0px 10px;
        list-style-type:square;
    }

    li {
        margin:0px;
        padding: 0px;
    }
    
```

8.1.2 Encoding

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

8.1.3 Anlegen von Variablen

Zur leichten Verarbeitung der Datenbank haben wir Variablen für einzelnen Listen in der XSL-Datei angelegt. Im späteren Verlauf werden weitere Variablen angelegt

```
<xsl:variable name="geraete" select="/datenbank/geraeteListe" />
<xsl:variable name="hersteller" select="/datenbank/herstellerListe" />
<xsl:variable name="kunden" select="/datenbank/kundenListe" />
<xsl:variable name="recyclinghof" select="/datenbank/recyclinghofListe" />
```

8.1.4 Verweis auf die die XSL-Datei in der XML-Datei

Damit der Webbrowser aus der XML-Datei eine HTML-Seite erzeugt muss im ein Verweis auf die XSL-Datei im XML-Dokument angelegt werden.

```
<?xml-stylesheet type="text/xsl" href="geraeteliste.xsl"?>
```

8.2 Geräteliste

8.2.1 Berechnung des Recyclinggrades

Anzeige aller Geräte ([Abbildung 13](#)) im Bestand mit Angabe von Anteil und dem Recyclinggrad. Der Recyclinggrad wird für jedes Geräte berechnet.

8.2.2 Angaben in der Variable "geraete"

```
<geraet geraetID="LenovoYogaPad2" geraetBezeichnung="Lenovo Yoga Pad 2" zustand="gut" verschmutzungsgrad="akzeptabel"
  geraetekategorie="Tablet">
  <bestandteile>
    <bestandteil name="Glas" anteil="30" recyclinggrad="100"/>
    <bestandteil name="Kunststoff" anteil="50" recyclinggrad="80" />
    <bestandteil name="Metall" anteil="20" recyclinggrad="10"/>
  </bestandteile>
</geraet>
```

8.2.3 Codefragment zur Berechnung in der XML-Datei

Der Recyclinggrad eines Geräts wird aus der Summe des Recyclinggrads der Bestandteile berechnet und durch die Anzahl der Bestandteile dividiert.

```
<p>
<strong>Recyclinggrad: <xsl:value-of select="format-number(sum(bestandteile/bestandteil/@recyclinggrad)
div count(bestandteile/bestandteil), '###,###,##0.00')"/> %</strong>
</p>
```

8.2.4 Screenshot

Startseite Geräteliste Herstellerliste Kundeliste PLZ fehlt Recyclinghof exakt Recyclinghof in der Nähe
GERÄTE
IPHONE 5
Kategorie: Smartphone Zustand: neuwertig Verschmutzungsgrad: akzeptabel Bestandteile: <ul style="list-style-type: none"> ■ Glas (60%) ■ Kunststoff (10%) ■ Metall (30%) Recyclinggrad: 83.33 %
IPHONE 5
Kategorie: Smartphone Zustand: defekt Verschmutzungsgrad: verschmutzt Bestandteile: <ul style="list-style-type: none"> ■ Glas (60%) ■ Kunststoff (10%) ■ Metall (30%) Recyclinggrad: 83.33 %
SAMSUNG GALAXY S3
Kategorie: Smartphone Zustand: gut Verschmutzungsgrad: akzeptabel Bestandteile: <ul style="list-style-type: none"> ■ Glas (50%) ■ Kunststoff (30%) ■ Metall (20%) Recyclinggrad: 63.33 %
NOKIA 3110
Kategorie: Handy Zustand: defekt Verschmutzungsgrad: gereinigt Bestandteile: <ul style="list-style-type: none"> ■ Glas (10%) ■ Kunststoff (80%) ■ Metall (10%) Recyclinggrad: 63.33 %
LENOVO YOGA PAD 2
Kategorie: Tablet Zustand: gut Verschmutzungsgrad: akzeptabel Bestandteile: <ul style="list-style-type: none"> ■ Glas (30%) ■ Kunststoff (50%) ■ Metall (20%) Recyclinggrad: 63.33 %

Abbildung 13: Geräteliste

8.3 Herstellerübersicht

Ein Liste ([Abbildung 14](#)) aller in der Datenbank vorhandene Hersteller.

8.3.1 XML-Fragment

```

<hersteller herstellerID="Samsung">
  <adresse>
    <name>Samsung Electrics GmbH</name>
    <strasse>Am Kronberger Hang 6</strasse>
    <plz>65824</plz>
    <ort>Schwalbach am Taunus</ort>
    <telefon>0180 6 67267864</telefon>
  </adresse>
</hersteller>
    
```

8.3.2 XSLT-Fragment

```
<div id="geraete">
  <h1>Hersteller</h1>
  <xsl:for-each select="$hersteller/hersteller">
    <xsl:sort select="@herstellerID"/>
    <p>
      <h3><xsl:value-of select="position()" />. Hersteller</h3>
      <p><strong>Name: </strong><xsl:value-of select="adresse/name"/></p>
      <p><strong>Straße: </strong><xsl:value-of select="adresse/strasse"/></p>
      <p><strong>PLZ: <xsl:value-of select="adresse/plz"/>&#160;</strong><strong>Ort: </strong><xsl:value-of select="adresse/ort"/></p>
      <strong>Telefonnummern:</strong>
      <ul>
        <xsl:for-each select="adresse">
          <li><xsl:value-of select="telefon"/></li>
        </xsl:for-each>
      </ul>
    </p>
  </xsl:for-each>
</div>
```

8.3.3 Screenshot

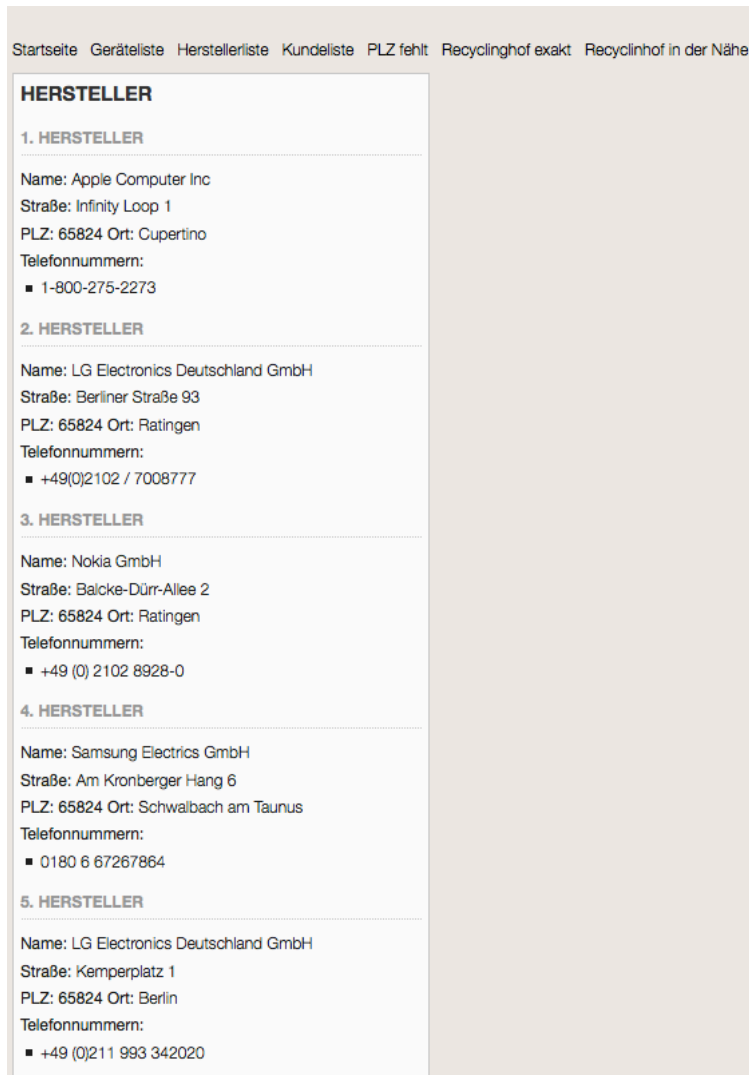


Abbildung 14: Herstellerliste

8.4 Kundenliste

8.4.1 Herausfiltern von nicht vorhandenen Daten

Eine vollständige Kundliste ([Abbildung 15](#)) aller im System gespeicherten Kunden.

Dadurch das wir in unserer XML-Datei für Kunden nur eine Telefonnummer und den Namen voraussetzen, bleiben manche Felder unausgefüllt diese werden über bestimmte Anweisungen und Abfragen nicht in dem View angezeigt.

8.4.2 Filtern mit dem if-Befehl

```
<xsl:if test="adresse/strasse">
```

```
<p>Straße: <xsl:value-of select="adresse/strasse" /></p>
</xsl:if>
```

8.4.2 Filtern mit dem choose-Befehl

Wegen dem höheren Schreibaufwand und der Mehrzeilen Code ist der if-Befehl vorzuziehen.

```
<xsl:choose>
  <xsl:when test="adresse/ort">
    <p>Ort: <xsl:value-of select="adresse/ort" /></p></xsl:when>
    <xsl:otherwise></xsl:otherwise>
  </xsl:choose>
```

Screenshot

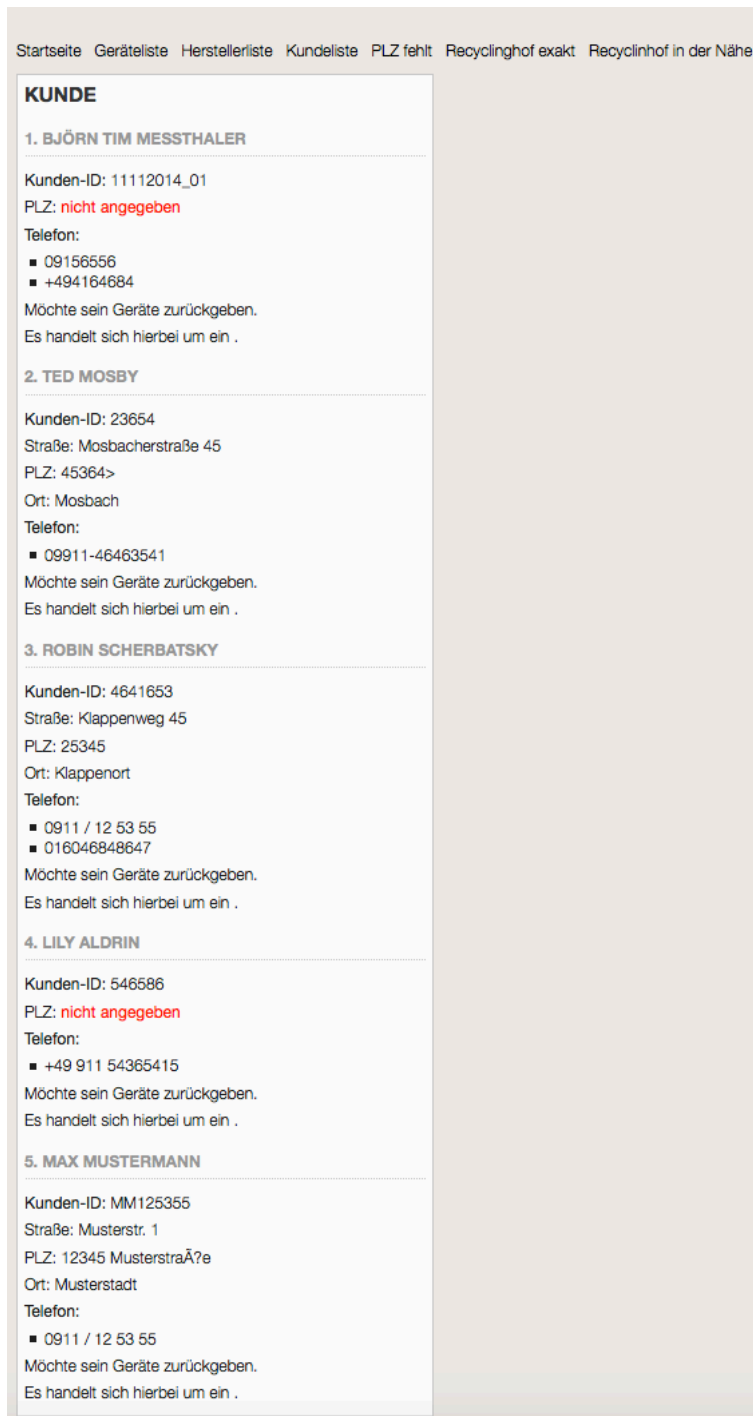


Abbildung 15: Kundenliste

8.5 Übersicht Kunden-Recyclinghof fehlende Postleitzahl

8.5.1 Anwendungsfall: Kunde kommt und möchte sein Gerät zurückgeben

Der Kunde kommt zu uns und möchte sein Gerät recycling lassen. Er möchte gerne wissen welcher Recyclinghof in seiner Nähe sein Gerät zurücknimmt

Leider hat er bei seiner Registrierung seine Postleitzahl vergessen einzugeben, um jedoch einen Recyclinghof in seiner näheren Umgebung zu finden ist die Postleitzahl zwingend notwendig.

Der Fehler wird mit durch einen rotmarkierten Text ([Abbildung 16](#)) ausgegeben.

Unser System liefert einen Fehler und bittet die Postleitzahl nachzuliefern.

8.5.2 XML Kunde "Lily"

```
<kunde kundenID="11112014_35" geraet="Lenovo Yoga Pad 2" geraetekategorie="Tablet">
  <adresse>
    <name>Lily Aldrin</name>
    <telefon>049 911 54365415</telefon>
    <telefon>049 175 129323</telefon>
  </adresse>
</kunde>
```

8.5.3 Anlegen einer neuen Variablen

Unser imaginäre Kunde Lily wird in der Variable "lily" gespeichert um eine leichtere Verarbeitung zu gewährleisten.

```
<xsl:variable name="lily" select="$kunden/kunde[@kundenID='11112014_35']" />
```

8.5.4 XSLT-Code

```
<div id="content">
<div id="links">
  <h1>Kunde</h1>
  <h3><xsl:value-of select="$lily/adresse/name" /></h3>
  <!-- Check ob eine Strasse eingegeben ist -->
  <xsl:if test="adresse/strasse">
    <p>Straße: <xsl:value-of select="text()" /></p>
  </xsl:if>

  <!-- Check ob eine PLZ eingegeben wurde -->
  <xsl:choose>
    <xsl:when test="adresse/plz">
      <p>PLZ: <xsl:value-of select="text()" /></p></xsl:when>
    <xsl:otherwise><p>PLZ: <strong style="color:red;">nicht angegeben</strong></p></xsl:otherwise>
  </xsl:choose>

  <!-- Check ob eine PLZ eingegeben wurde -->
  <xsl:choose>
    <xsl:when test="adresse/ort">
      <p>Ort: <xsl:value-of select="adresse/ort" /></p></xsl:when>
    </xsl:choose>

    <strong>Telefon:</strong>
    <ul>
      <xsl:for-each select="$lily/adresse/telefon">
        <li><xsl:value-of select="text()" /></li>
      </xsl:for-each>
    </ul>
    <p>Möchte sein Gerät <strong><xsl:value-of select="$lily/@geraet" /></strong> zurückgeben.</p>
    <p>Es handelt sich hierbei um ein <strong><xsl:value-of select="$lily/@geraetekategorie" /></strong>.</p>
  </div>

  <div id="rechts">
    <h1>Vorgeschlagene Recyclinghoefe</h1>
    <xsl:choose>
      <xsl:when test="$lily/adresse/plz">
        <p>PLZ: <xsl:value-of select="text()" /></p></xsl:when>
      <xsl:otherwise><p>Um einen Recyclinghof in ihrer Nähe ermitteln zu können benötigen wir eine gültige Postleitzahl!</p>
      </xsl:otherwise>
    </xsl:choose>
  </div>
</div>
```

8.5.5 Screenshot

[Startseite](#)
[Geräteliste](#)
[Herstellerliste](#)
[Kundeliste](#)
[PLZ fehlt](#)
[Recyclinghof exakt](#)
[Recyclinghof in der Nähe](#)

KUNDE

LILY ALDRIN

PLZ: nicht angegeben

Telefon:

 ■ +49 911 54365415

VORGESCHLAGENE RECYCLINGHOEFE

 Um einen Recyclinghof in ihrer Nähe ermitteln zu können benötigen wir eine gültige Postleitzahl!

Abbildung 16: Fehler bei fehlender PLZ

8.6 Anzeige welcher Recyclinghof passt genau zur PLZ des Kunden

Nachdem die Postleitzahl eingefügt wurde, können wir nun anhand der PLZ ermitteln welcher Recyclinghof sich in der Nähe des Kunde befindet.

In diesem Fall suchen wir nach genau dem Recyclinghof der zur Adresse von der Kundin passt ([siehe Abbildung 17](#)). Nicht schön aber für den Anfang okay.

8.6.1 XML

```
<kunde kundenID="11112014_35" geraet="Lenovo Yoga Pad 2" geraetekategorie="Tablet">
  <adresse>
    <name>Lily Aldrin</name>
    <plz>90411</plz>
    <telefon>049 911 54365415</telefon>
    <telefon>049 175 129323</telefon>
  </adresse>
</kunde>
```

8.6.2 XSLT

```
<div id="links">
  <h1>Kunde</h1>
  <h3><xsl:value-of select="$lily/adresse/name" /></h3>
  <!-- Check ob eine Strasse eingegeben ist -->
  <xsl:if test="$lily/adresse/strasse">
    <p>Straße: <xsl:value-of select="$lily/adresse/strasse" /></p>
  </xsl:if>

  <!-- Check ob eine PLZ eingegeben wurde -->
  <xsl:choose>
    <xsl:when test="$lily/adresse/plz">
      <p>PLZ: <xsl:value-of select="$lily/adresse/plz" /></p></xsl:when>
    <xsl:otherwise><p>PLZ: <strong style="color:red;">nicht angegeben</strong></p></xsl:otherwise>
  </xsl:choose>

  <!-- Check ob eine PLZ eingegeben wurde -->
  <xsl:choose>
    <xsl:when test="$lily/adresse/ort">
      <p>Ort: <xsl:value-of select="$lily/adresse/ort" /></p></xsl:when>
    </xsl:choose>

    <strong>Telefon:</strong>
    <ul>
      <xsl:for-each select="$lily/adresse/telefon">
        <li><xsl:value-of select="text()" /></li>
      </xsl:for-each>
    </ul>
    <p>Möchte sein Gerät <strong><xsl:value-of select="$lily/@geraet" /></strong> zurückgeben.</p>
    <p>Es handelt sich hierbei um ein <strong><xsl:value-of select="$lily/@geraetekategorie" /></strong>.</p>
  </div>
```

```
<div id="rechts">
  <h1>Vorgeschlagene Recyclinghoefe</h1>
  <xsl:for-each select="$recyclinghof/recyclinghof">
    <xsl:variable name="aktuellplz" select="adresse/plz" />
    <xsl:choose>
      <xsl:when test="$lily/adresse/plz = $aktuellplz">
        <h3><xsl:value-of select="adresse/name" /></h3>
        <p>Straße: <xsl:value-of select="adresse/strasse" /></p>
        <p>Ort: <xsl:value-of select="adresse/plz" /> &#160; <xsl:value-of select="adresse/ort" /></p>
        <p>Öffnungszeiten:</p>
        <ul>
          <xsl:for-each select="oeffnungszeiten">
            <li><xsl:value-of select="text()" /></li>
          </xsl:for-each>
        </ul>
        <p></p></xsl:when>
      <xsl:otherwise></xsl:otherwise>
    </xsl:choose>
  </xsl:for-each>
</div>
```

8.6.3 Screenshot

[Startseite](#)
[Geräteliste](#)
[Herstellerliste](#)
[Kundeliste](#)
[PLZ fehlt](#)
[Recyclinghof exakt](#)
[Recyclinghof in der Nähe](#)

KUNDE

LILY ALDRIN

Straße: Mosbacherstraße 45
 PLZ: 90411
 Ort: Nürnberg
 Telefon:
 ■ +49 911 54365415

VORGESCHLAGENE RECYCLINGHOEFE

RECYCLINGHOF NORD

Straße: Haaberleinstraße 7
 Ort: 90411 Nürnberg
 Öffnungszeiten:
 ■ Montag 9.30 Uhr - 18 Uhr
 ■ Dienstag 9.30 Uhr - 18 Uhr
 ■ Mittwoch 9.30 Uhr - 18 Uhr
 ■ Donnerstag 9.30 Uhr - 18 Uhr
 ■ Freitag 9.30 Uhr - 18 Uhr
 ■ Samstag 9.30 Uhr - 15 Uhr
 ■ Sonntag und Feiertag geschlossen

Abbildung 17: Einfache Ausgabe des passenden Recyclinghofs

8.7 Anzeige welcher Recyclinghof ist in der Nähe des Kunden

Nachdem die Postleitzahl eingefügt wurde, können wir nun anhand der PLZ ermitteln welcher Recyclinghof sich in der Nähe des Kunde befindet. Das lösen wir in dem wir die ersten drei Zahlen der Recyclinghöfe-Postleitzahl mit der Kunden-Postleitzahl vergleichen ([Abbildung 18](#)).

XSLT-Code

```

<div id="rechts">
  <h1>Vorgeschlagene Recyclinghoefe</h1>
  <xsl:for-each select="$recyclinghof/recyclinghof">
    <xsl:variable name="aktuellplz" select="adresse/plz" />
    <xsl:variable name="aktuellplz1" select="substring($aktuellplz, 1,3)" />
    <xsl:choose>
      <xsl:when test="starts-with($lily/adresse/plz,$aktuellplz1)">
        <h3><xsl:value-of select="adresse/name" /></h3>
        <p>Straße: <xsl:value-of select="adresse/strasse" /></p>
        <p>Ort: <xsl:value-of select="adresse/plz" />&#160;<xsl:value-of select="adresse/ort" /></p>
        <p>Öffnungszeiten:</p>
        <ul>
          <xsl:for-each select="oeffnungszeiten">
            <li><xsl:value-of select="text()" /></li>
          </xsl:for-each>
        </ul>
        <p></p></xsl:when>
        <xsl:otherwise></xsl:otherwise>
      </xsl:choose>
    </xsl:for-each>
  </div>

```

Screenshot

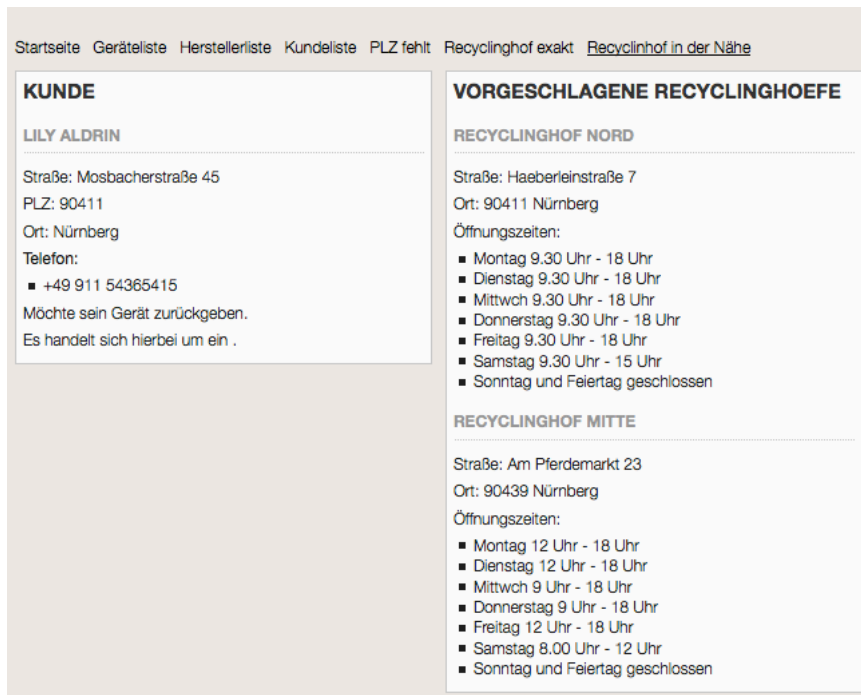


Abbildung 18: Fehler beim Chrome Browser

8.8 Aufbau der Anwendung

Die Anwendung besteht aus verschiedenen Views und einer HTML Startseite *index.html* (siehe Abbildung 19). Zum Starten der Anwendung muss die Seite *index.html* aufgerufen werden.



Abbildung 19: Startseite der Anwendung Smartphone Recycling

Die Anwendung hat folgende, zum Teil exemplarische, Ansichten:

- Geräteliste
- Herstellerliste
- Kundenliste
- PLZ fehlt
- Recyclinghof exakt
- Recyclinghof in der Nähe

Die einzelnen Views sind über das Menü erreichbar.

8.9 Probleme und Herausforderungen bei der Entwicklung der Views

Die Angabe der DTD in der XML hat im Chrome Browser und Safari Probleme verursacht. Die Daten werden nicht geladen und es wird eine Sicherheitsmeldung ausgegeben (siehe Abbildung 20). Unsere Anwendung ist momentan für Firefox optimiert.

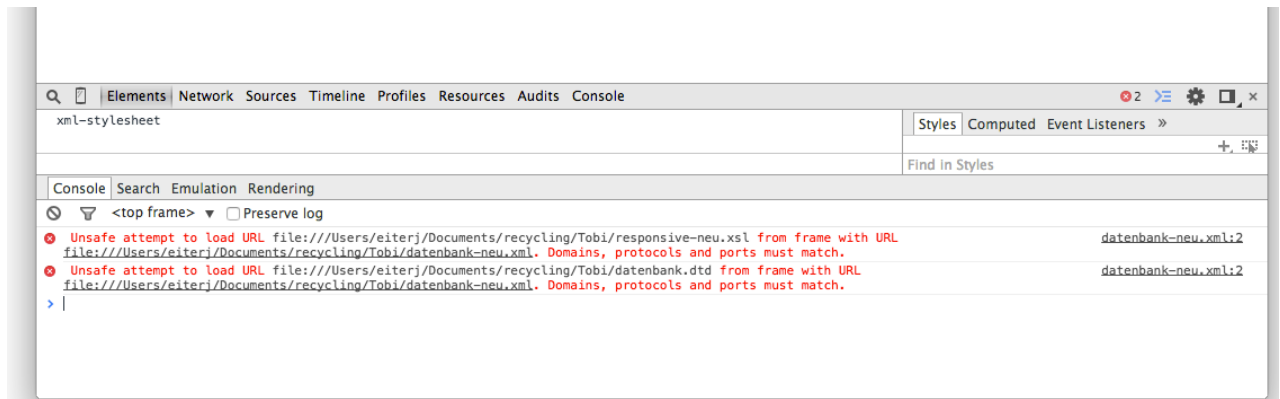


Abbildung 20: Fehler beim Chrome Browser

Ein weiteres Problem war die Validierung der einzelnen XML Dateien. Die Aufsplittung der Daten in einzelne XML Dateien hat bei der Validierung einen Fehler ergeben. Das Problem wurde durch eine Zentrale XML Datei mit dem Wurzelement *datenbank* erstellt.

Ein weiteres Problem das durch ein großes XML Dokument entstanden ist war die selektierung der XSL Stylesheets für die jeweilige View. Leider war es uns nicht möglich dieses Problem zu lösen. Ein möglicher Ansatz für die Lösung des Problems wäre eine Synchronisierung auf Dateiebene.

9 Mögliche Erweiterungen und Einsatzgebiete

Da dieses Projekt nur in einem begrenztem Rahmen durchgeführt werden konnte, gibt es natürlich noch mögliche Erweiterungen und Verbesserungen.

Für einen umgreifenderen Einsatzbereich könnte die Anwendung statt nur Smartphones, Tablets und Handys auch andere elektronische Geräte miteinbeziehen. Es wäre dazu auch möglich zum Beispiel auch alte Haushaltsgeräte wie Mixer, Waschmaschinen, Staubsauger etc. mit der Anwendung zu verwalten.

Des Weiteren könnte eine mobile Applikation in Form einer App entwickelt werden, in der der Benutzer alle seine Geräte verwalten kann und ihm Informationen zu den Bestandteilen und dem Recyclinggrad zur Verfügung stellt. Es würde auch leicht realisierbar sein, dem Nutzer Informationen über die Entsorgung und den passenden Recyclinghof zu seinem Gerät in der näheren Umgebung mitzuteilen.

Durch weitere Aufnahme von Daten zur Herstellung der Geräte, wie zum Beispiel CO2 Emmissionen, die bei der Produktion eines Stücks entstehen oder logistische Informationen, wie den Vertriebsweg usw. könnte auch die Nachhaltigkeit eines Gerätes errechnet werden. Der Nutzer hat im Grunde eine Art Info-Portal, in dem er sieht wie Nachhaltig sein Gerät produziert wurde und wo er es, wenn es defekt oder veraltet ist abgeben könnte.

Die Aufnahme der folgenden Daten wären durchaus denkbar:

- Informationen über die CO2 Emmissionen bei der Produktion und Logistik
- Daten über die Rohstoffgewinnung des Produkts. Unter welchen Umständen wurde ein Bestandteil gefördert?
- Produktionsdaten: Wo, Wie, Wann wurde das Gerät hergestellt? Könnte der Benutzer dadurch inländische Unternehmen unterstützen.
- Weitere Herstellerinformationen wie Unternehmensgröße, Umsatz usw.

Auch einfache Features würden der aktuellen Applikation einen Mehrwert bieten. Das Hinzufügen einer Suche, oder das Darstellen einer Karte, in der sich der nächstgelegene Recyclinghof befindet wären durchaus sinnvolle Erweiterungen. Eine Schnittstelle hierzu wäre Google Maps mit einzubeziehen.

10 Verwendetet Tools

Folgende Tools haben wir zur Umsetzung des Projekts verwendet:

- Dreamweaver
- Sublime 2
- Firefox
- GitHub
- Notepad++
- xmllint

Folgende Tools haben wir zur Dokumentation und Präsentation des Projekts verwendet:

- Keynote
- Dreamweaver

11 Fazit

Nach anfänglichen Startschwierigkeiten mit der Materie "Datenmodellierung in XML" konnten wir uns schnell für das Thema begeistern und einarbeiten. Durch die Projektarbeit haben wir viel zum Thema XML gelernt, wenn wir auch nur an der Oberfläche kratzen konnten. Wir konnten uns eine strukturierte Beschreibung von Daten mittels der Auszeichnungssprache XML und deren Einsatz sehr gut aneignen. Wir haben ein sehr gutes Verständnis der Modellierung von Daten mittels XML und ddazu assoziierten Technologien erreicht. Werkzeuge zur Modellierung und Nutzung von XML-basierten Datenstrukturen haben wir ebenfalls kennengelernt und angewendet.

Wir stellten fest, dass sich aus einem theoretischen Datenmodell noch lange keine vernünftigen Views entwickeln lassen und so mussten wir zu einem sehr späten Zeitpunkt im Projekt nochmals unser Datenmodell überdenken und ändern. Jedoch sind XML und XSLT gute und schnelle Wege Daten zu visualisieren auch wenn die Fehlersuche sich mit wenig Erfahrung als eher schwierig.

Leider mussten wir auch feststellen, dass XSLT 1.0 sehr eingeschränkt arbeitet, so hätten wir gerne eine komplexere Webseite mit mehreren Menüpunkten aus unserer XML-Datenbank erzeugt und komplexere Abfragen entwickelt. Auch die XML Dokumente werden bei größeren Datenmengen schnell unübersichtlich.

Außerdem konnten wir auf Grund unserer Gruppengröße das Thema JSON nicht mehr bearbeiten. Allerdings fanden wir eine Gruppengröße von drei Personen sehr angenehm, da wir keinen großen Reibungsverlust bei der Kommunikation hatten.

12 Aufgabenverteilung

Grundsätzlich hat im Projekt "Recycling" alle Teammitglieder in allen Bereichen seine Kenntnisse vertieft und weiter ausgeprägt. Jeder war im Entstehungsprozess und an der Entwicklung beteiligt. Bei einer Gruppengröße von 3 Personen ist uns eigentlich nichts anderes übrig geblieben, um die Anforderungen des Projekts zu erfüllen. Zudem wurde durch das gleichmäßige Verteilen von Aufgaben der Lerneffekt verstärkt, da jeder bei jedem Themengebiet mitgewirkt hat.

Jeder war auch umfassend an der Dokumentation des Projekts beteiligt.

A Anhang

A.1 Präsentation

Die Datei Präsentation.pdf befindet sich im Verzeichnis *Präsentation*.

A.2 Quelltext

Der Quellcode unseres Projekts befindet sich im Verzeichnis *Quelltext*.

Das Verzeichnis enthält folgende Dateien bzw. Unterverzeichnisse:

- index.html
- css
- geraeteliste
- herstellerliste
- kundenliste
- plzfehlt
- recyclinghofexakt
- recyclinghofnahe