Projekt-Dokumentation

Bachelorarbeit

„XLink-basierte personalisierte Grafikannotierung als Firefox-Plugin“

vorgelegt von
Sebastian Schwill

eingereicht bei
Prof. Dr. Wolfgang May
Databases and Information Systems Group
Georg-August-Universität Göttingen

eingereicht am
05. September 2008

Georg-August-Universität
Göttingen

Institut für Informatik
Databases and Information Systems Group
Prof. Dr. Wolfgang May

Bachelorarbeit
im Studiengang „Wirtschaftsinformatik“

XLink-basierte personalisierte Grafikannotierung als Firefox-Plugin

vorgelegt von
Sebastian Schwill

eingereicht bei
Prof. Dr. Wolfgang May
Databases and Information Systems Group
Georg-August-Universität Göttingen

eingereicht am
05. September 2008

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Göttingen, den 26. August 2008

Zusammenfassung

In der vorliegenden Bachelorarbeit wird eine prototypische Implementierung einer Anwendungsumgebung für die Problemstellung der personalisierten Grafikannotation vorgestellt. Hierbei werden das entwickelte Konzept, die tatsächliche Realisierung, sowie ein Handbuch zur Nutzung der Anwendung betrachtet.

Personalisierte Grafikannotation bedeutet im Zusammenhang mit dieser Arbeit, es einem Benutzer zu ermöglichen, Veränderungen an beliebigen Grafiken durchzuführen und diese in einer Datenbasis zu speichern.

Durch Nutzung eines Firefox-Plugins werden die hinterlegten Veränderungen automatisch beim erneuten Betrachten der Grafik sichtbar. Dadurch besitzt der Benutzer die Möglichkeit jede beliebige Grafik für sich selbst so anzupassen, dass Sie für ihn zusätzlichen Nutzen bietet.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Die vorliegende Arbeit stellt die schriftliche Ausarbeitung einer Bachelorarbeit zum Thema "XLink-basierte personalisierte Grafikannotierung als Firefox-Plugin" dar. Sie entstand als Abschluss eines Bachelorstudiums der Wirtschaftsinformatik. Vorangegangen sind dabei einige theoretische Überlegungen und die praktische Umsetzung einer Anwendung zum oben genannten Thema. Die daraus resultierenden Ergebnisse werden in dieser Arbeit vorgestellt und erläutert.

1.1 Voraussetzung

Um den Ausführungen dieser Arbeit folgen zu können, ist es nötig, dass der Leser mit den grundlegenden Strukturen von XML (Extended Markup Language) und den daran angrenzenden Themenbereichen vertraut ist. Ebenfalls sollten dem Leser Schlagworte wie PHP und Javascript nicht fremd sein. Der Abschnitt *2 Technische Grundlagen* soll lediglich einen ersten Eindruck der verwendeten Techniken vermitteln. Es wird dabei bewusst keine detaillierte Einführung gegeben.

1.2 Problemstellung und Zielsetzung

Diese Arbeit hat das Ziel ein Firefox-Plugin zu entwickeln, welches dem Benutzer die Möglichkeit bietet Grafiken zu individualisieren. Mit Hilfe des Plugins soll der Benutzer in die Lage versetzt werden eine beliebige, im Internet befindliche Grafik dem eigenen Nutzen anzupassen. Dabei können sowohl Inhalte entfernt, verändert, sowie neue hinzugefügt werden. Um die Grundgedanken zu verdeutlichen wird nachfolgend ein Beispiel erläutert.

Beispiel Stadtplanmarkierung:

Der Benutzer findet auf einer beliebigen Internetseite einen Stadtplan der Stadt, in die seine nächste Geschäftsreise geht. Diese Internetseite, speziell den Stadtplan, fügt er seinen Favoriten im Internetbrowser hinzu, um ihn zu einem späteren Zeitpunkt erneut aufrufen zu können. Nun sucht der Benutzer die Straße in der sich sein *Hotel* befindet heraus. Die Position des Hotels kann durch den Browser selbst jedoch nicht markiert werden. Beim nächsten Besuch müsste somit das Hotel erneut gesucht werden.

Durch das Plugin soll es dem Benutzer möglich sein, ohne eine veränderte Kopie der Grafik zu erzeugen, eine Veränderung an der Grafik vorzunehmen. Es sei angenommen, dass der Benutzer einen Pfeil mit der Beschriftung „*Hotel*“ zur Grafik hinzufügt. Das Plugin würde daraufhin lediglich die Veränderung, also den Pfeil mit zugehöriger Beschriftung und deren Position in der Grafik, sowie eine Referenz auf die zugehörige Grafik speichern.

Wird die Internetseite erneut durch den Benutzer besucht, so würde das Plugin in diesem Beispiel automatisch die originale Grafik um den von dem Benutzer erzeugten Pfeil mit der Beschriftung *Hotel* ergänzen. Eine erneute Suche des *Hotels* entfällt.

Dieses Beispiel kann schnell und leicht auf weitere Einsatzzwecke übertragen werden.

Um einen Lösungsansatz für dieses Problem zu finden muss ein Plugin erstellt werden, welches Grafiken erkennt, für die Modifikationen vorhanden sind. Des Weiteren wird eine Datenbasis benötigt, die ein Importieren und Exportieren von Modifikationen ermöglicht. Diese

Datenbasis sei als Linkbase bezeichnet. Zum Datenaustausch soll eine standardisierte XML-Datei dienen, die auf den Strukturen von XLink basiert.

1.3 Aufbau der Arbeit

Der erste Teil dieser Arbeit hat bereits einen gewissen Überblick über die Problemstellung und Zielsetzung dieser Arbeit gegeben.

Im zweiten Teil werden die einzelnen Technologien, die für eine mögliche Realisierung einer Anwendung nötig sind, kurz dargestellt. Es wird nur eine kurze Beschreibung gegeben, die es dem Leser nicht ermöglicht, die Technologien selbst anzuwenden. Der Leser ist darauf angewiesen, sich in die passende Fachliteratur einzulesen. Ihm soll lediglich ein Eindruck vermittelt werden, welche Grundgedanken sich hinter der jeweiligen Technologie verbergen.

Im dritten Abschnitt wird ein Konzept für eine Anwendung vorgestellt, welches das Problem teilweise oder vollständig abdeckt. Zusätzlich werden Überlegungen angestellt, deren Resultat die Auswahl der einzusetzenden Technologien bedingen.

Ein Überblick über die Architektur und die Zusammensetzung und Verteilung der Komponenten wird im vierten Abschnitt gegeben.

Anschließend erhält der Leser im fünften Abschnitt einen groben Überblick der gewählten Umsetzung. Auf vollständigen Quelltext wird bewusst verzichtet. Nur ausgewählte Auszüge sollen Betrachtung finden. Des Weiteren werden Grenzen aufgezeigt, die bei der Realisierung mit den Technologien sichtbar wurden.

Ein ausführliches Handbuch zu der Anwendung wird im sechsten Abschnitt vorgestellt. Hierbei werden die Installation und Konfiguration, sowie die Handhabung der einzelnen Funktionalitäten betrachtet. Zum Abschluss dieses Abschnittes soll mittels eines kleinen Fallbeispiels die Funktion der Anwendung demonstriert werden.

Am Ende dieser Arbeit wird eine Schlussbetrachtung vorgenommen.

An dieser Stelle sei darauf verwiesen, dass sämtliche Daten auf der beiliegenden CD-Rom zu finden sind. Dazu zählen Quellcode wie auch Archive, mit denen eine eigene Installation auf dem eigenen Webserver möglich ist.

2 Technische Grundlagen

Dieser Abschnitt bietet einen kurzen Einblick über die verschiedenen Technologien, die in dieser Arbeit eine wichtige Rolle spielen.

2.1 DOM (Document Object Model)

Mit dem Document Object Model (DOM) hat das World Wide Web Consortium (W3C) einen mächtigen Standard geschaffen, der eine neutrale Programmierschnittstelle mit lesendem und schreibendem Zugriff auf wohlgeformte XML-Dokumente schafft. Darüberhinaus ist der Standard plattform- und programmiersprachenunabhängig. Im DOM wird jeder Bestandteil eines XML-Dokumentes als ein Knoten aufgefasst. Dabei werden das Dokument selbst, seine Elemente und dessen Attribute und deren textuelle Inhalte als entsprechende Objekte repräsentiert.[1]

```
1  <table>
2      <thead>
3          <tr>
4              <th>Nr.</th>
5              <th>Grafik</th>
6          </tr>
7      </thead>
8      <tbody>
9          <tr>
10             <td>1</td>
11             <td>Stadtplan</td>
12         </tr>
13     </tbody>
14 </table>
```

Abbildung 1: Ausschnitt einer XML-Syntax

Diese Objekte sind den zugehörigen Knotenklassen zugeordnet und untereinander verknüpft. Die Verknüpfung geschieht im einfachsten Fall über eine simple Eltern-Kind-Beziehung. Dadurch lässt sich ein XML-Dokument mittels DOM als eine Art Baum darstellen. Anhand dieser Baumstruktur kann durch das gesamte Dokument navigiert werden und jedes beliebige Objekt eindeutig adressiert werden.[2]

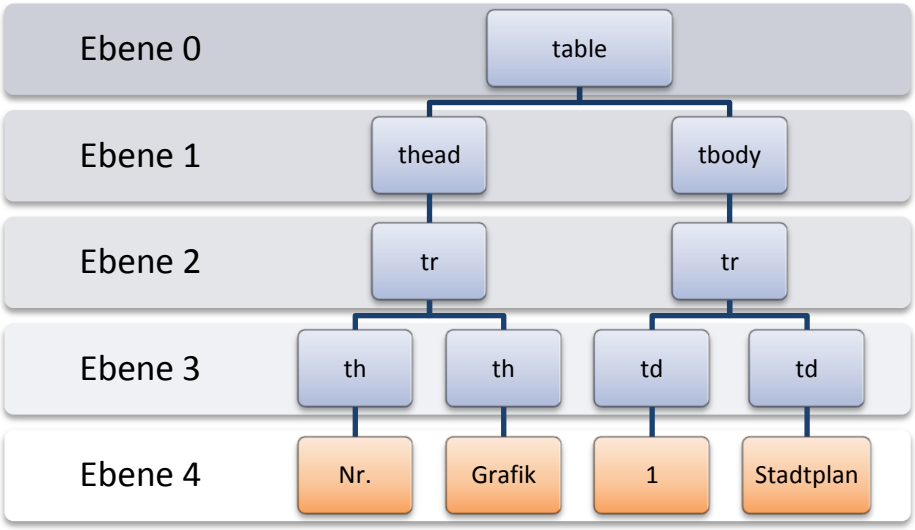

Abbildung 2: DOM-Baum des XML-Ausschnitts

[1] Vgl. (Argerich, et al., 2002, S. 183 ff.)
[2] Vgl. (Argerich, et al., 2002, S. 184 ff.)

Abbildung 1 zeigt einen Ausschnitt aus einer XML-Datei. Der Kontext beschreibt eine Tabellenstruktur ähnlich der, die in HTML-Dokumenten verwendet wird. Dabei stellt der Knoten *table* das Wurzelelement dar. Es besitzt zwei Kindelemente, *thead* und *tbody*. Diese Beziehungen lassen sich beliebig weiterführen - in diesem Beispiel bis zu den innersten Elementen, den Texten. Diese Struktur kann mittels DOM in eine Baumstruktur überführt werden. Abbildung 2 zeigt die zugehörige Baumstruktur der in Abbildung 1 dargestellten Tabellenstruktur. Die bläulichen Kästen repräsentieren Knotenelemente, die orangen repräsentieren die Textknoten. Ebenfalls zeigt die Abbildung, dass obiges XML-Dokument durch einen Baum mit 5 Ebenen repräsentiert werden kann. Soll nun der Textknoten *Stadtplan* adressiert werden, so wäre dieses: das erste Kind – vom zweiten Kind – vom ersten Kind – vom zweiten Kind – des Vaters.

2.2 SVG (Scalable Vector Graphics)

SVG (Scalable Vector Graphics) ist ein Standard zur Beschreibung von zweidimensionalen Vektorgrafiken auf der Basis von XML-Syntax. Somit stellt eine Baumstruktur, die aus ineinander verschachtelten Elementen und den zugehörigen Attributen besteht, die Repräsentation einer Grafik dar. Dadurch ist es dem Betrachter ebenfalls möglich die Grafik in einem Texteditor zu öffnen und den reinen XML-Code selbst zu verändern.

Nach Empfehlung der W3C unterstützen heute fast alle gängigen Browser SVG-Grafiken. Die einzige Ausnahme stellt dabei der Internet Explorer von Microsoft dar, der ein zusätzliches Plugin benötigt, um SVG-Grafiken darzustellen.

Auch dynamische Grafiken können mittels SVG realisiert werden. Hierfür ist es möglich Skripte einzubinden. Diese Skripte können den DOM der Grafik manipulieren oder auf Interaktionen mit dem Betrachter reagieren. Bei den Skripten handelt es sich um *ECMAScripte*. Dies ist eine standardisierte Variante von Javascript.[3]

Ein Einsatzgebiet für SVG könnte die Bildbeschriftung sein. Beispielsweise ist ein Foto von einer Landschaft oder ein Stadtplan vorhanden. Darin soll ein Hotel eingezeichnet werden. Normalerweise kommt nun Bildbearbeitungssoftware zum Einsatz. Es wird eine Kopie von dem Bild mit den Zusatzinformationen erzeugt. Genau hier zeigt SVG seine Stärken.

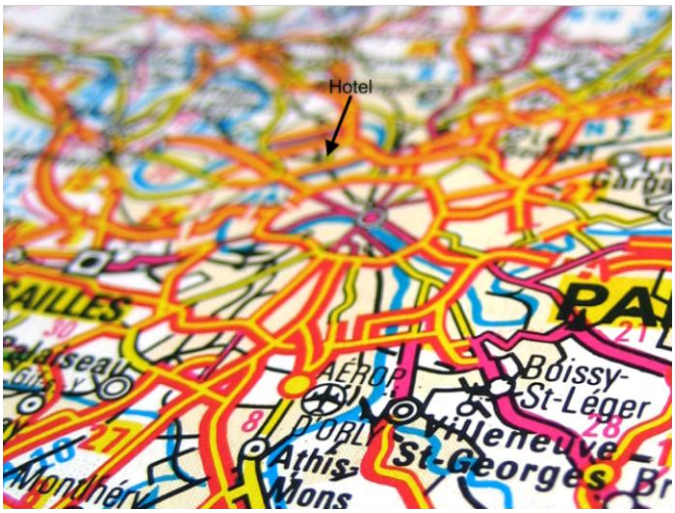

Abbildung 3: SVG-Grafik, Stadtplan mit Pfeil

[3] Vgl. (SVG Working Group, 2004, S. und Unterseiten)

Abbildung 3 zeigt eine SVG-Grafik, wo ein Stadtplan per Link eingebunden wurde. Darüber wurde ein Pfeil gesetzt mit der Beschriftung *Hotel*. Vorteilhaft hierbei ist, dass die Grafik des Stadtplanes nicht doppelt abgespeichert werden muss.

```
1  <?xml version="1.0" encoding="utf-8"?>
2
3  <!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN" "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
4  <svg version="1.1" id="Ebene_1" xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink"
5       width="640px" height="480px" viewBox="0 0 640 480">
6
7      <image width="897" height="600" xlink:href="paris.jpg">
8          <title>Paris-Karte</title>
9      </image>
10
11     <line fill="none" stroke="#000000" stroke-width="3" x1="333.5" y1="85.5" x2="313.5" y2="135.5"/>
12
13     <polygon stroke="#000000" stroke-width="3" points="310.512,129.092 319.908,132.512 311.79,140.198 "/>
14
15     <text transform="matrix(1 0 0 1 311.4995 82.5)" font-family="'ArialMT'" font-size="18">Hotel</text>
16 </svg>
```

Abbildung 4: SVG-Grafik, Stadtplan mit Pfeil, im Texteditor

Abbildung 4 stellt die in einem Texteditor geöffnete SVG-Grafik aus Abbildung 3 dar. Hier wird der Aufbau nach XML-Syntax gut sichtbar. In Zeile 7 wird durch Referenzieren auf das Bild der Landkarte (paris.jpg) verwiesen und diese damit eingebunden. Zeilen 11 – 13 fügen eine Linie und ein Polygon hinzu, die zusammen einen Pfeil ergeben. In Zeile 15 wird anschließend der Schriftzug *Hotel* über dem Pfeil positioniert.

Das obige Beispiel gibt einen kurzen Eindruck, welcher Nutzen durch die Verwendung von SVG generiert werden kann. Nachfolgend soll die XML-Struktur von SVG genauer aufgezeigt werden. Abbildung 5 zeigt eine Grafik, die sich die Gruppierung von Objekten bzw. deren Verschachtelung, zu Nutze macht.

Abbildung 5: SVG-Grafik, Gruppierung von verschobenen Quadraten

Die Grafik beinhaltet jeweils 4 orange und 4 blaue Quadrate. Diese sind wie ein verschobenes Schachbrett angeordnet. Hierbei wurden die Quadrate ihrer Farbe nach gruppiert. Dieses bietet zwei Vorteile. Zum einen kann die Gruppe der Quadrate verschoben werden, ohne dass jedes einzelne Quadrat verschoben werden muss. Zum anderen ist es so möglich, die Gruppen als eine Art Ebene zu bertachten und somit durch einfaches Verschieben der Gruppe entweder die orangen oder die blauen Quadrate in den Vordergrund zu bringen.

Der zur Grafik gehörende Quellcode ist in Abbildung 6 dargestellt. Hier ist in Zeile 7 bis 11 die Gruppe der blauen Quadrate zu sehen. Die Zeilen 12 bis 17 zeigen den Quelltext der Gruppierung der orangen Quadrate. Die Verschachtelung geschieht wie in XML üblich.

Hierbei gestaltet sich die reguläre Anzeigereihenfolge wie folgt:

- Ein Knoten wird optisch unterhalb eines Knotens angezeigt, wenn dieser im Sinne von *priviousSibling* vor den zweiten Knoten positioniert ist.
- Eine weitere Möglichkeit, um einen Knoten unterhalb eines Anderen zu Positionieren, stellt die Verschachtelung dar. Hierbei wird ein *Elternknoten* optisch unterhalb seiner *Kindknoten* angezeigt.

Zu erwähnen ist, dass dieses Verhalten durch die Benutzung von Stylesheets beeinflusst werden kann. Die Benutzung von Stylesheets soll an dieser Stelle nicht weiter erläutert werden.

```
1  <!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN" "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
2
3  <svg version="1.1"  xmlns="http://www.w3.org/2000/svg" viewBox="0 0 1000 1000">
4
5      <g>
6          <g>
7              <rect x="75.451" y="9" fill="#009EE0" width="66.667" height="66.667"/>
8              <rect x="208.805" y="9" fill="#009EE0" width="66.667" height="66.667"/>
9              <rect x="9" y="75.637" fill="#009EE0" width="66.667" height="66.667"/>
10             <rect x="142.168" y="75.637" fill="#009EE0" width="66.667" height="66.667"/>
11         </g>
12         <g>
13             <rect fill="#F29400" width="66.667" height="66.667"/>
14             <rect x="133.168" fill="#F29400" width="66.667" height="66.667"/>
15             <rect x="66.451" y="66.637" fill="#F29400" width="66.667" height="66.667"/>
16             <rect x="199.805" y="66.637" fill="#F29400" width="66.667" height="66.667"/>
17         </g>
18     </g>
19
20 </svg>
```

Abbildung 6: SVG-Quelltext zur Gruppierung

Die Gruppierung, die in Zeile 5 geöffnet und in 18 geschlossen wird, fasst die orangen und blauen Quadrate zu einer adressierbaren Gruppe zusammen. Eine Verschiebung dieser Gruppierung über geänderte Positionsangaben, bewirkt eine Verschiebung aller innenliegenden Grafikelemente. Dabei wird nicht jedes einzelne Element verschoben, sondern nur das Gruppenelement an sich. Dadurch verändert sich der Bezugsrahmen der Positionsangaben der innenliegenden Elemente. Diese beziehen sich nun nicht weiter auf die gesamte Grafik, sondern nur noch auf die Gruppe, in der sie sich befinden.

2.3 XML Path Language (XPath)

Um Teile eines XML-Dokumentes zu adressieren wurde vom W3C die XML Path Language (XPath) als Abfragesprache entwickelt. XPath dient als Grundlage für andere Standards, die in Zusammenhang mit XML stehen, wie XSLT, XPointer und XQuery. Es wird eine kompakte Syntax verwendet, die verschieden der von XML ist. Dadurch ist es möglich XPath-Ausdrücke innerhalb eines XML-Dokumentes zu verwenden, ohne dass diese mit der XML-Syntax kollidieren. XPath operiert nicht auf der äußerlichen Syntax eines XML-Dokumentes, sondern auf seiner abstrakten, logischen Struktur. Die Syntax gleicht der Pfad-Notation einer URL und ermöglicht es hierarchisch durch ein XML-Dokument zu navigieren.

In 2.1 wurde eine verbale Vater-Kind-Navigation zu dem Textknoten Stadtplan aufgeführt. Diese soll nun erneut aufgegriffen und mittels XPath umgesetzt werden. Bei XPath werden die Knotennamen zur Navigation benutzt. Nun liegt nahe, dass */table/tbody/tr/td* der XPath-Ausdruck für das angeführte Problem sei. Der gesuchte Textknoten wird zwar durch den Ausdruck selektiert, jedoch werden ebenfalls alle weiteren td-Elemente, die unterhalb eines

tr-Elements, das unterhalb eines tbody-Elements, welches wiederum unter einem table-Element liegt, selektiert. Somit würde dieser XPath-Ausdruck auch den Textknoten mit dem Inhalt *1* selektieren. Der XPath-Ausdruck muss noch um eine Positionsangabe ergänzt werden. Der Ausdruck */table/tbody/tr/td [position()=2]* würde genau den gesuchten Knoten adressieren, da es sich um den zweiten td-Knoten unterhalb … handelt.

2.4 Hypertext Preprocessor (PHP)

PHP (Hypertext Preprocessor) ist eine Erweiterung für Webserver, die es ermöglicht, dynamische Webanwendungen mit verhältnismäßig wenig Aufwand zu erstellen. PHP ist eine Skriptsprache, die eine an C bzw. C++ angelehnte Syntax besitzt. Durch eine breite Datenbankunterstützung, die Einbindung von Internet-Protokollen, sowie die Verfügbarkeit zusätzlicher Funktionsbibliotheken zeichnet sich PHP aus.

Der Quellcode von PHP wird serverseitig verarbeitet. Somit wird der Quellcode nicht direkt an den Browser übermittelt, sondern an einen Interpreter auf dem Webserver. Der Browser bekommt daraufhin nur die Ausgabe des Interpreters zu sehen. Von Vorteil ist, dass der Quelltext für den Benutzer der Internetseite nicht einsehbar ist. Ein weiterer Vorteil von PHP besteht darin, dass der Browser keine speziellen Voraussetzungen erfüllen muss, um PHP-Seiten anzeigen zu können. Somit werden die Ressourcen des Clients nicht unnötig strapaziert, da die Verarbeitung und die Kommunikation mit einer Datenbank komplett auf dem Server ablaufen.

Der größte Nachteil von PHP besteht darin, dass der Webserver für jeden http-Request den Quelltext in Bytecode umwandeln muss. Standartmäßig besitzt PHP keinen Bytecode-Caching. Es existieren jedoch Erweiterungen, die diese Funktionalität nachrüsten.[4]

2.5 Javascript/ Asynchronous JavaScript and XML (AJAX)

Javascript ist eine clientseitige interpretierte Skriptsprache. Sie wurde ursprünglich von Netscape entwickelt. Daraufhin zog Microsoft mit dem Internet Explorer nach. Jeder moderne Webbrowser liefert einen Javascript-Interpreter, der jedoch teilweise einen unterschiedlichen Sprachumfang unterstützt. Das liegt an der getrennten Entwicklung durch verschiedene Browserhersteller. Bis heute ist Javascript nicht komplett standardisiert.

Javascript wird entweder direkt oder über einen Verweis auf eine gesonderte Skriptdatei in ein HTML-Dokument eingebettet. Die Verbindung von Javascript mit HTML und CSS wird durch das Schlagwort "DHTML", dynamisches HTML, beschrieben. Heutzutage wird Javascript überwiegend dazu eingesetzt, um die Usability einer Webseite zu verbessern.

In Javascript sind Objekte, Funktionen und Prozeduren bekannt, jedoch keine Klassen. Jedes Objekt wird vom Basisobjekttypen "Object" abgeleitet. Auch das Konzept der Vererbung wird über die Eigenschaft "prototype" unterstützt. Javascript stellt somit eine prototypbasierte objektorientierte Skriptsprache dar.

Ein Begriff der häufig im Zusammenhang mit Javascript fällt ist AJAX (Asynchrones Javascript und XML). AJAX stellt ein Konzept dar, die dem asynchron ablaufende Funktionen im Hintergrund (XMLhttpRequest) mittels Javascript Anfragen an einen Server senden und durch die empfangenen Antworten weitere Änderungen im HTML-Dokument vorgenommen werden.

[4] Vgl. (Schmid & Cartus, 2001, S. 22 ff.)

Diese Änderungen können ganz verschiedener Natur sein. Es können kleine Änderungen eines Formulars sein. Aber auch ein komplettes HTML-Dokument kann eine mögliche Antwort darstellen.

Die browsereigenen Ajax-Funktionen sind Erweiterungen, die über Javascript angesprochen und genutzt werden. Sie gehören damit nicht zum Javascript-Standard. Diese Funktionen können direkt verwendet werden. Jedoch erleichtert das Verwenden eines Javascript-Frameworks die Nutzung der Funktionen, da diese eine einheitliche Schnittstelle bereitstellen. Damit muss selbst nicht mehr zwischen den einzelnen Browserversionen unterschieden werden. Als mögliches Framework sei JQuery genannt. Für nähere Informationen zu diesem Framework sei auf www.jquery.com verwiesen.[5]

Abbildung 7 zeigt das klassische Modell einer Webanwendung. Hierbei findet ein ständiger Wechsel zwischen Client (Benutzeraktivität) und Serververarbeitung statt. Die Kommunikation verläuft synchron. Immer wenn der Server eine Anfrage verarbeitet und zurücksendet, hat der Client Leerlauf und kann keine neue Aktion auslösen. Dementsprechend auch umgekehrt. Solange der Client keine Anfrage abschickt, hat der Server Leerlauf.

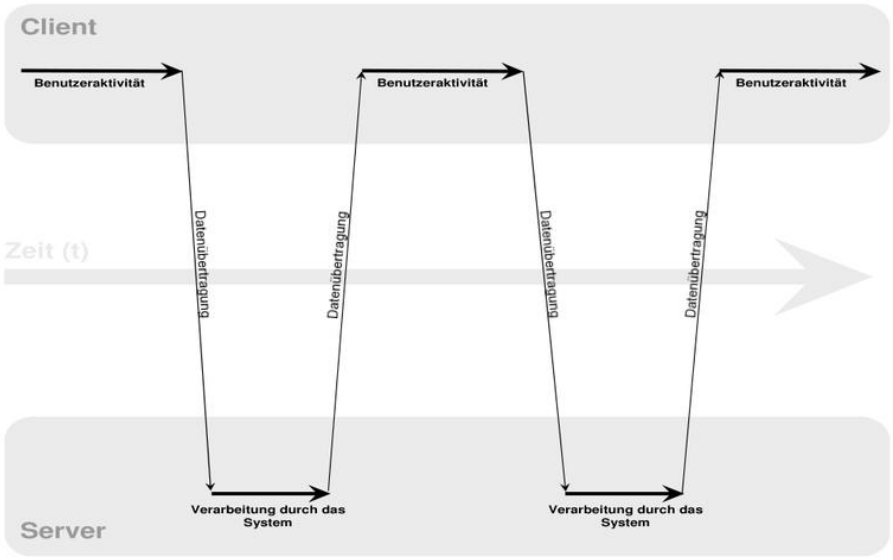

Abbildung 7: Klassisches Modell einer Web-Anwendung (synchrone Datenübertragung)[6]

Dem gegenüber steht das AJAX-Modell, welches in Abbildung 8 dargestellt ist. Es fällt auf, dass clientseitig eine Trennung in Benutzeroberfläche und AJAX-Engine vorgenommen wurde. Somit muss der Benutzer nicht auf die Antwort zu seiner Anfrage warten. Es ist so möglich mehrere Interaktionen auszulösen, die nacheinander vom Server verarbeitet werden. Sobald eine Verarbeitung vollständig ist, wird das Ergebnis dem Client angezeigt. Durch die asynchrone Kommunikation wird die punktuelle Serverlast minimiert und auf eine stetige Last verteilt. Ebenfalls erhöht sich die Usability und ähnelt immer mehr der einer traditionellen Desktopanwendung.

[5] Vgl. (Bergmann & Bormann, 2005, S. 53 ff. und S. 84 ff.)
[6] (Freie Universität Berlin, 2007)

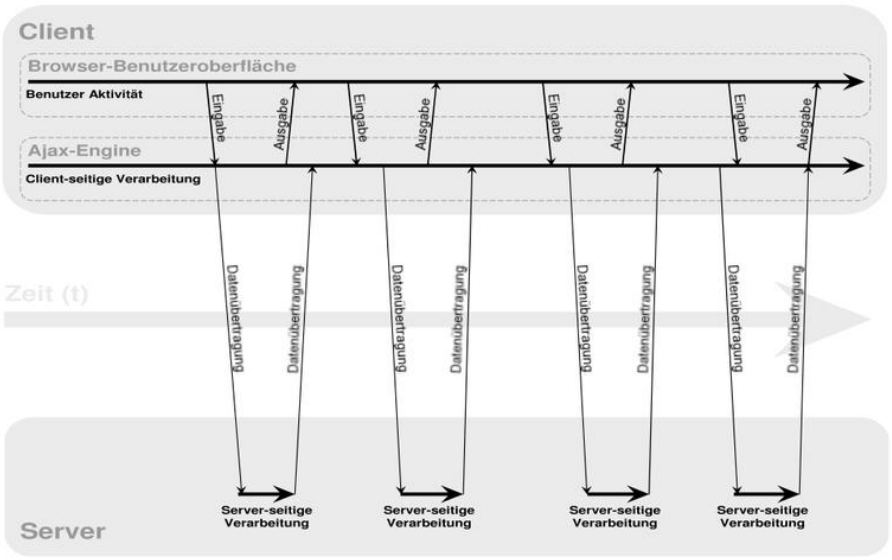

Abbildung 8: Ajax-Modell einer Web-Anwendung (asynchrone Datenübertragung)[7]

2.6 Firefox – Greasemonkey

Greasemonkey ist eine Firefox-Extension, die es erlaubt Skripte auszuführen. Die Skripte manipulieren die angezeigte Internetseite.

Greasemonkey wird überwiegend dazu verwendet die Übersichtlichkeit von Internetseiten zu steigern und auf Ihren Benutzer anzupassen. Ein weiteres Einsatzgebiet von Greasemonkey ist die Vernetzung von Internetseiten. Es ist möglich mittels Greasemonkey den Inhalt einer anderen Internetseite in die aktuelle einzubinden. Dieses bietet dem Benutzer zusätzlichen Nutzen, da er nicht zwischen mehreren Seiten wechseln muss.

Die oben beschriebenen Einsatzgebiete absolviert Greasemonkey nicht von alleine. Es werden sogenannte *user scripts* benötigt. Diese Skripte sind in Javascript geschrieben. Dasselbe Javascript, welches auf Internetseiten eingesetzt wird. Der Vorteil bei *user script* besteht darin, dass ein Javascript auf mehrere Websites angewendet werden kann. Ebenfalls muss es nicht vom Eigentümer der Internetseite eingebunden werden. Greasemonkey stellt spezielle API-Funktionen zur Verfügung, die dazu führen, dass *user scripts* weitaus mächtiger sind als normale Javascripts.[8]

Weiterführende Informationen zu Greasemonkey sind auf der zugehörigen Webseite unter http://www.greasespot.net/ zu finden.

[7] (Freie Universität Berlin, 2007)
[8] Vlg. (Pilgrim, 2006, S. XIX ff.)

3 Konzept

In diesem Abschnitt soll die Anwendung konzeptuell dargestellt werden. Es wird dargelegt, welche Mittel für die Umsetzung eingesetzt wurden. Auf eventuelle Alternativen soll nicht näher eingegangen werden, da dieses den Rahmen dieser Arbeit überschreiten würde. Die weitere Gliederung ähnelt der eines Pflichtenheftes.[9]

3.1 Konzeptuelle Modellierung

Bevor genauer auf das Konzept eingegangen wird, soll zunächst die Interaktion der einzelnen Objekte näher erläutert werden.

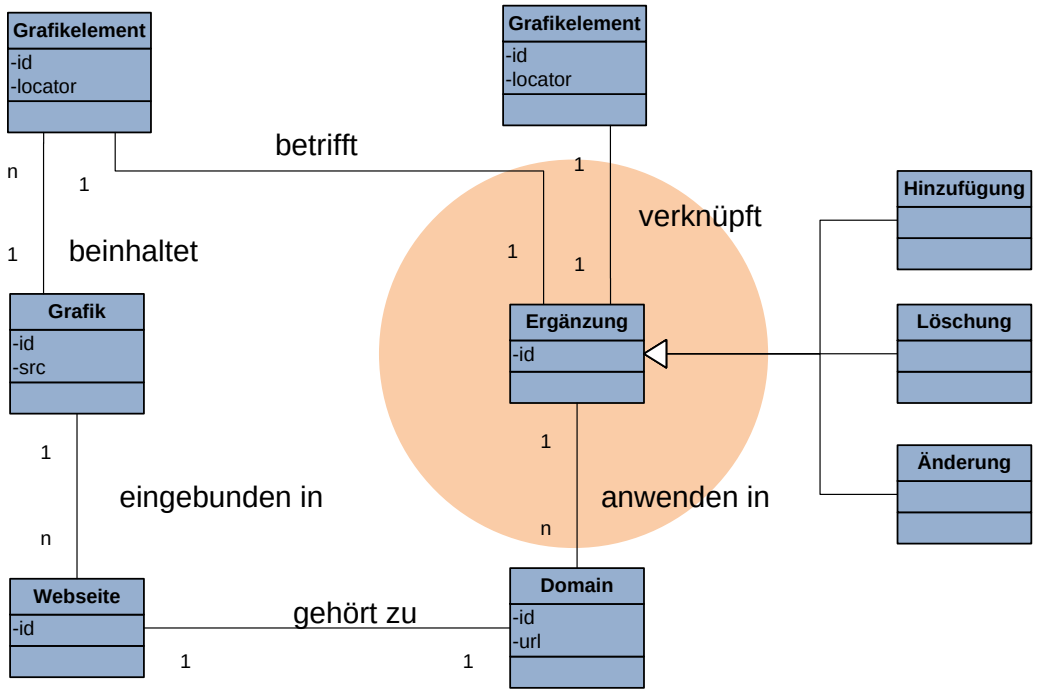

Abbildung 9: Konzeptuelle Modellierung als UML-Diagramm

Abbildung 9 stellt die *Ergänzungen* als zentrales Objekt der Anwendung dar. Eine weitere Spezifizierung der Ergänzungen ist nun nötig. Ergänzungen können in Form von *Hinzufügungen*, *Löschungen* oder *Änderungen* vorliegen. Dabei wird eine Ergänzung in einer oder mehreren *Domains* angewendet. Zu einer Domain gehört eine *Webseite*, in die eine *Grafik* eingebunden ist. Eine Grafik an sich kann in ihre einzelnen *Grafikelemente* aufgeschlüsselt werden. Der Bezugspunkt einer Ergänzung stellt ein Grafikelement dar. Dieses wird durch die gewählte Aktion mit einem zusätzlichen Grafikelement einer anderen Grafik verknüpft.

3.2 Zielbestimmungen

Zunächst sollen die Zielvorgaben betrachtet werden. Dabei soll eine Differenzierung in Muss- und Kannkriterien erfolgen.

3.2.1 Musskriterien

Dieser Abschnitt beschreibt alle Funktionalitäten von inRelation die für einen prototypischen Einsatz unabdingbar sind.

[9] Vgl. (Baur, 2008)

3.2.1.1 Linkbase

Die Linkbase spielt die zentrale Rolle in dieser Anwendung. Jedoch muss bei der Linkbase zwischen zwei Dingen unterschieden werden. Zum einen wird die Sammlung aller Modifikationen von allen Benutzern der Anwendung als Linkbase bezeichnet. Zum anderen werden die XML-Dateien, die Modifikationen nach einem XLink ähnlichen Schema enthalten, als Linkbase bezeichnet. In diesem Abschnitt soll nur auf die XML-Datei eingegangen werden. Die Linkbase als globaler Speicher- und Sammelort wird im Abschnitt der Webplattform erneut aufgegriffen.

Eine Linkbase soll eine XML-Datei darstellen. Diese kann zum Sichern, Transportieren und Austauschen von Modifikationen dienen. Es soll ein Schema geschaffen werden, nach dem eine solche Linkbase aufgebaut ist.

Sollen Modifikationen an eine andere Person weitergegeben werden, kann es nötig sein eine Linkbase-Datei zu verwenden. Zwei mögliche Gründe kommen in Betracht. Zum einen, dass die andere Person nicht im selben inRelation-System arbeitet, zum anderen können dem Benutzer selbst die Rechte fehlen, um seine Modifikationen für bestimmte Domains als öffentlich zu markieren. In beiden Fällen kann der Benutzer seine Modifikationen in eine Linkbase-Datei exportieren und diese der anderen Person zuschicken. Somit ist die Voraussetzung geschaffen, die Linkbase-Datei bei sich zu importieren.

3.2.1.2 Webplattform (serverseitig)

Die Webplattform stellt eine globale Linkbase dar, in der mehrere Benutzer gleichzeitig arbeiten können. Es soll gesichert sein, dass jeder Benutzer seine Modifikationen eintragen kann.

Grundsätzlich müssen zwei *Benutzergruppen* geschaffen werden. Zum einen der *Administrator*, der in der Lage ist, weitere Benutzer für die Webplattform zu registrieren und diesen die Rechte zuzuteilen, für bestimmte Domains öffentliche Modifikationen in das System einzutragen. Zum anderen der normale *Benutzer*, dieser kann Modifikationen in das System eintragen und seine eigenen und die öffentlichen Modifikationen von anderen Benutzern abrufen. Auch die Möglichkeit sein *Passwort zu ändern* sollte jedem zur gegeben werden.

Jeder Benutzer kann die eigenen und die ihm öffentlich zugänglichen Modifikationen in eine Linkbase-Datei *exportieren*, sowie Einträge über eine solche *importieren*. Ebenfalls ist der Benutzer dazu befähigt, seine private Linkbase *zurückzusetzen* und damit alle seine Einträge zu löschen.

Eine Kernfunktion dieser Anwendung stellt die *Vererbung von Modifikationen* dar. Dies bedeutet, wenn eine Modifikation mit einer Domain verknüpft ist, so ist diese automatisch für alle Unterverzeichnisse dieser Domain zugänglich. Die Usability für den Benutzer wird hierdurch erheblich gesteigert. Es ist nicht nötig, eine Modifikation für alle Unterverzeichnisse zu definieren.

3.2.1.3 Greasemonkey-Script (clientseitig)

Die Schaffung eines Plugins für den Firefox-Browser stellt einen zentralen Punkt der gesamten Anwendung dar. Mit aktiviertem Plugin werden automatisch alle originalen Grafiken um die Modifikationen, die der Benutzer in seiner Linkbase hinterlegt hat, ergänzt. Der Benutzer

erhält somit die Möglichkeit, die Grafiken auf beliebigen Internetseiten zu verändern und somit an seine Bedürfnisse anzupassen.

3.2.2 Kannkriterien

Nachfolgend werden Funktionen beschrieben, die einen zusätzlichen Nutzen für die Anwendung generieren, jedoch für die Funktionsfähigkeit der Anwendung nicht unverzichtbar sind. Eine Realisierung ist dennoch erstrebenswert.

Speziell auf der *Webplattform* steigern zusätzliche Funktionen die Benutzerfreundlichkeit.

Durch eine *Übersicht aller Modifikationen* erhält der Benutzer einen besseren Überblick über seine Modifikationen und deren Verknüpfungen zu den einzelnen Domains. Eine Übersicht könnte eine Auflistung von allen Originalgrafiken darstellen. Zusätzlich sieht der Benutzer, für welche Domains Modifikationen hinterlegt sind. Eine Betrachtung der modifizierten Grafiken für eine bestimmte Domain wäre ebenfalls erstrebenswert.

Eine weitere Zusatzfunktion stellt ein *grafischer Editor* dar. Er bietet dem Benutzer eine grafische Unterstützung, um Modifikationen einfach und übersichtlich, also mit wenigen Klicks durchzuführen und zur Linkbase hinzuzufügen. Dies beinhaltet sowohl die *Aktionsauswahl* als auch die Unterstützung der *Domainverknüpfung*. Gerade komplexere Abläufe, wie die *Adressierung von einzelnen Grafikelementen* in der Originalgrafik bzw. aus der Grafik, aus der etwas kopiert werden soll, werden vereinfacht. Auch ein kleines *Zeichenprogramm*, welches ein Repertoire an Grundformen von Objekten beinhaltet, erleichtert dem Benutzer die Arbeit. Durch einfaches Zusammenklicken von Formen und Texten kann ein Benutzer, ganz ohne genauere Kenntnisse über SVG-Grafiken, eine Grafik mit einfachen Objekten ergänzen.

Eine Installationsroutine würde es dem Administrator erleichtern, die Anwendung auf seinem Server zu installieren und an seine Serverumgebung anzupassen. Sie stellt ebenfalls eine erstrebenswerte Funktionalität dar.

3.3 Anwendungseinsatz

Nachdem die Zielbestimmungen der Anwendung dargelegt wurden, soll nun auf einen möglichen Anwendungsbereich eingegangen werden. Anschließend werden die Betriebsbedingungen, die für einen zweckmäßigen Einsatz nötig sind, betrachtet.

3.3.1 Anwendungsbereiche

Nachdem in Abschnitt 1.2 Problemstellung und Zielsetzung dargelegt wurden, soll die Grundidee nun weiter verfeinert und für einen komplexeren Anwendungsbereich erweitert werden.

Zunächst soll der Grundgedanke der Anwendung aufgegriffen werden. Es soll ein Firefox-Plugin entstehen, das dem Benutzer ermöglicht, Modifikationen für bestimmte Grafiken zu hinterlegen. Das Plugin soll selbstständig prüfen, ob Modifikationen vorhanden sind und ggf. die Modifikationen an der Grafik durchführen. Es ist durchaus wahrscheinlich, dass dieselbe Grafik auf zwei unterschiedlichen Seiten eingebunden ist und in verschiedenem Kontext gebraucht wird. Aus diesem Grund soll es dem Benutzer möglich sein, seine Modifikationen an bestimmte Domains zu binden. Die Sammlung von Modifikationen wird, wie in 1.2 erwähnt, als Linkbase bezeichnet.

Im Zeitalter des Web 2.0 liegt der Fokus immer mehr auf Netzwerkeffekten. Aus diesem Grund sollen Netzeffekte für die Anwendung berücksichtigt werden. Mehrere Nutzer sollen zusammen eine Linkbase nutzen. Dabei können ausgewählte Modifikationen für andere zugänglich gemacht werden.

Nachfolgend wird der Nutzen der angesprochenen Netzeffekte unter Einbeziehung eines Beispiels verdeutlicht. Hierzu sei angenommen, dass eine Universität einen Raumplan für alle Universitätsgebäude besitzt. Dieser wird von der Universität auf ihrer Internet-Seite öffentlich zur Verfügung gestellt. Ein Universitätsmitarbeiter trägt die Bezeichnungen der einzelnen Institute in die Linkbase ein und kennzeichnet diese als öffentlich. Wird dieselbe Linkbase ebenfalls durch einen Institutsmitarbeiter benutzt, so stehen diesem bereits die eingetragenen Bezeichnungen aller Institute zur Verfügung. Dieser Mitarbeiter trägt zusätzlich die Namen und Positionen der einzelnen Lehrstühle in die Grafik ein. Einem dritten Benutzer, der einem Lehrstuhl angehört, ist es möglich, zusätzlich die Namen aller Mitarbeiter seines Lehrstuhls in die entsprechende Räume eintragen.

Mit den zuvor getroffenen Annahmen soll nun die ursprüngliche Grafik des Raumplanes auf den jeweiligen Internetseiten betrachtet werden. Wird die Grafik auf der Internet-Seite der Universität eingebunden, so ist für den Benutzer der Linkbase ein Raumplan mit allen Instituten sichtbar. Besucht der Benutzer hingegen die Internetseite eines Lehrstuhls, wird ihm der Raumplan mit allen Institutsbezeichnungen wie auf der Universitäts-Seite angezeigt. Zusätzlich enthält die Grafik die Namen aller Mitarbeiter des Lehrstuhls.

Ein Vorteil, der sich aus diesem Scenario ergibt, ist die Wartungsfreundlichkeit des Raumplanes. Nicht jedes Institut bzw. jeder Lehrstuhl muss eine veränderte Kopie des Raumplanes anlegen. Wird der Raumplan von der Universität verändert, weil beispielsweise ein neues Gebäude hinzugekommen ist, so bleiben alle Modifikationen der Institute und Lehrstühle vorhanden. Das zusätzliche Gebäude ist automatisch in den Grafiken der Institute und Lehrstühle integriert.

Zusammenfassend lässt sich festhalten: Es soll eine Anwendung geschaffen werden, in die mehrere Benutzer Modifikationen an Grafiken eintragen können. Diese Modifikationen können entweder als privat oder öffentlich gekennzeichnet werden und enthalten eine Verknüpfung mit bestimmten Domains. Des Weiteren findet eine Vererbung von Modifikationen an Unterseiten statt. Damit diese Modifikationen für den Anwender sichtbar und nutzbar sind, muss ein Plugin in seinem Browser installiert und aktiviert sein.

3.3.2 Betriebsbedingungen

Die inRelation-Anwendung grenzt sich bezüglich der Betriebsbedingungen nicht wesentlich von anderen Internetdiensten bzw. -anwendungen ab. Jedoch ist die Anwendung nur für registrierte („eingeladene“) Benutzer nutzbar. Zusätzlich wird die eigentliche Webplattform durch ein zusätzliches Browser-Plugin ergänzt.

Die Anwendung muss 24 Stunden am Tag und 365 Tage im Jahr zugänglich, nutzbar und dabei möglichst wartungsfrei sein. Die Sicherung der Datenbank wird jedoch manuell vom Administrator der Anwendung durchgeführt.

3.4 Anwendungsumgebung

Die Anwendung ist weitestgehend unabhängig von der Plattform, sofern nachfolgende Anforderungen an die Produktumgebung erfüllt sind.

3.4.1 Software

Bei der Software muss zwischen Anforderungen an den Client und Server unterschieden werden. Als Client wird der User der Anwendung bzw. dessen Computer bezeichnet.

3.4.1.1 Client

Die Anforderungen, die an den Client gestellt werden, um die Anwendung nutzen zu können, sind relativ gering. Auf Seite der Software wird lediglich der *Firefox-Browser* in Version 2.0.0.12 oder neuer, mit eingeschaltetem Javascript, benötigt. Des Weiteren muss das *Greasemonkey-Plugin* im Firefox installiert und aktiviert sein. Zusätzlich ist das *inRelation-Greasemonkey-Script* nötig, das ebenfalls installiert und aktiviert sein muss.

Der Vorteil in der Umsetzung des benötigten Plugins mittels Greasemonkey liegt darin, dass der Entwicklungsaufwand wesentlich geringer ist. Es muss lediglich ein Skript entwickelt werden, das Grafiken in einer Internetseite aufspürt und anschließend, sofern vorhanden, der Grafik die Modifikationen des Benutzers hinzufügt. Der gesamte Grundprogrammieraufwand für ein neues Plugin, um dieses in den Firefox-Browser integrieren zu können, fällt weg. Ein weiterer Vorteil besteht darin, dass Greasemonkey API-Schnittstellen bereitstellt, die wichtige Funktionalitäten ohne großen Programmieraufwand nutzbar machen. Der größte Nachteil, den Greasemonkey mit sich bringt, ist die etwas verschlechterte Performance des Skriptes. Jedoch kann diese für eine prototypische Implementierung vernachlässigt werden. Somit haben die Vorteile bei der Nutzung von Greasemonkey die Nachteile relativiert.

3.4.1.2 Server

Die Anforderungen an den Server sind etwas höher. Es wird eine *MySQL-Datenbank* benötigt. Des Weiteren muss der Webserver *PHP 4* unterstützen. Am besten in Version 4.4.8. Von PHP ist die Standardinstallation vollkommen ausreichend. Lediglich das *domxml* Modul muss nachträglich aktiviert werden.

Auf die Installation eines geeigneten Webservers und die Aktivierung des domxml PHP-Moduls wird in dieser Arbeit nicht eingegangen. Hierzu sei auf die einschlägige Fachliteratur verwiesen.

Es wurde bewusst PHP für die Implementierung gewählt. Zum einen stellt PHP eine relativ einfache und mächtige Möglichkeit dar, Anwendungen im Internet umzusetzen. Hinzu kommt, dass alle benötigten Technologien wie z.B. DOM in PHP standardmäßig enthalten sind. Nicht zu unterschätzen ist die Verbreitung von PHP. Heutzutage ist bei fast jedem Webserver bzw. Webhostingpaket bereits PHP integriert. Somit ließe sich die Anwendung ohne großen Aufwand auf einem anderen Webserver installieren und betreiben.

PHP wird komplett serverseitig verarbeitet. Gerade bei komplexeren Interaktionen zwischen Server und Client kann dieses zu großen Zeitverzögerungen führen. Es müsste für jede Benutzereingabe eine komplette HTML-Seite als Serverausgabe erzeugt werden. Diese Seite müsste dann an die Anfrage des Clients jeweils angepasst werden. Um die Zeitverzögerungen zu minimieren schaffen Technologien wie AJAX Abhilfe. Hier werden nur Teile der Seite an die Benutzerinteraktion angepasst. Zusätzlich findet eine Auslagerung bestimmter Interaktionen

zur Clientseite hin statt. Dabei werden jedoch nur Funktionen zum Client hin verschoben, die im Wesentlichen die Usability für den Benutzer steigern. Die eigentliche Prozesslogik bleibt weiterhin serverseitig und wird mittels PHP umgesetzt.

Die Entscheidung, die globale Linkbase mittels einer MySQL-Datenbank abzubilden, ist zum einen darin begründet, dass PHP standardmäßig sehr gut mit dieser Datenbank zusammenarbeitet. Des Weiteren gehört MySQL heutzutage ebenfalls meist zum Standard, der auf jedem Webserver verfügbar ist. Es wurde bewusst auf die Verwendung einer XML-Datei als globale Linkbase verzichtet, da im Multiuser-Betrieb mit einer relativ großen Linkbase gerechnet wird. Die Performance würde davon sehr schnell negativ beeinflusst werden.

3.4.2 Hardware

Aus Gründen der Vollständigkeit werden die nötigen Hardwarevoraussetzungen kurz dargelegt.

3.4.2.1 Client

Clientseitig stellt der *Internetanschluss* die einzige Voraussetzung dar, die der Rechner zusätzlich zu den Softwarevoraussetzungen erfüllen muss. Die Geschwindigkeit der Internetverbindung besitzt direkten Einfluss auf die Arbeitsgeschwindigkeit in der Anwendung für den Client.

3.4.2.2 Server

Hardwareseitig ist beim Server ebenfalls eine *schnelle Internetverbindung* Grundvoraussetzung. Hinzu kommt, dass der Rechner die Ansprüche der o.g. *Server-Software* erfüllt. Für den Multiuserbetrieb sollte der Server ebenfalls über *ausreichend Rechen- und Festplattenkapazität* verfügen.

3.4.3 Orgware

In den Bereich der Orgware fällt die *ständige Verfügbarkeit der Internetverbindung*, die unabdingbar für den reibungslosen Betrieb der Anwendung ist. Zu den Aufgaben des Administrators zählt die *Basiskonfiguration* durchzuführen, sowie die Erstellung regelmäßiger Backups. Für den Benutzer der Anwendung dient das *Handbuch* in Abschnitt 6 als Grundvoraussetzung.

3.5 Anwendungsfunktionen

Auf die Anwendungsfunktionen soll an dieser Stelle nicht weiter eingegangen werden. Es sei hierzu auf das Handbuch in Abschnitt 6 verwiesen. Dort werden alle Funktionalitäten ausführlich vorgestellt. Ausgewählte Funktionen werden aus technischer Sicht im Abschnitt 5 Realisierung zusätzlich näher betrachtet.

3.6 Anwendungsdaten

Die Daten, die durch die inRelation-Anwendung gespeichert werden, beschränken sich auf die Benutzerdaten zum einen und die Daten der Modifikationen zum anderen.

3.6.1 Benutzerdaten

Es werden zu jedem Benutzer der Vor- und Zuname gespeichert. Zusätzlich ist bei jedem Benutzer eine eindeutige Emailadresse hinterlegt. Das heißt, es kann zu jeder Email-Adresse nur genau ein Benutzerkonto angelegt werden. Für die Identifikation des Benutzers werden ein Benutzername und ein Passwort gespeichert. Der Benutzername ist ebenfalls eindeutig innerhalb einer Installation der Anwendung. Um zu verhindern, dass jeder Benutzer das Recht

besitzt weitere Benutzer in die Anwendung aufzunehmen, ist dieser einer Benutzergruppe mit bestimmten Zugriffsrechten, zugeordnet. Zu guter Letzt können durch den Administrator, dem Benutzer bestimmte Domains zugeordnet werden, für die er seine Modifikationen als öffentlich kennzeichnen kann.

3.6.2 Modifikationsdaten

Den zweiten Bereich, über den Daten gespeichert werden, stellen die Modifikationen dar. Die Modifikation wird über ihre Aktion beschrieben. Zusätzlich wird zu der Aktion die zugehörige Grafikquelle hinterlegt, sowie das genaue Element in der Grafik, auf welches sich die Modifikation bezieht. Hinzu kommt die Zuordnung zu dem Benutzer, der die Modifikation eingestellt hat. Auch die Verknüpfung zu bestimmten Domains wird berücksichtigt. In diesem Zusammenhang kann ebenfalls hinterlegt werden, ob die Modifikation öffentlich sichtbar sein darf oder nicht. Je nachdem, ob es sich um ein *update*, *delete* oder *insert* handelt, werden weitere Angaben nötig. Bei einem update werden die zu ändernden Attribute und ihre neuen Attributwerte abgelegt. Für ein *delete* sind keine zusätzlichen Angaben nötig. Bei einem *insert* gibt es zwei verschiedene Arten der zusätzlich nötigen Daten. Es wird entweder direkter SVG-Quellcode oder eine Referenz auf ein Element in einer beliebigen anderen SVG-Grafik, das kopiert und eingefügt werden soll, hinterlegt.

Aus Gründen der Nachvollziehbarkeit werden zusätzlich noch die IP-Adresse des Benutzers, das Datum sowie die Uhrzeit des Einfügens gespeichert.

3.7 Anwendungsleistungen

Die Anwendungsleistungen gliedern sich in drei Bereiche.

Einen Bereich stellt die *Akkumulation* dar. Der Benutzer bekommt Rückmeldungen vom System, sobald eine Eingabe fehlerhaft war. Zusätzlich erhält er eine Auflistung aller aufgetretenen Fehler, die im Zusammenhang mit seiner Interaktion stehen.

Den zweiten Bereich spiegelt die *Toleranz* wieder. Bei fehlerhaften Eingaben muss der Benutzer die Möglichkeit besitzen, seine vorherige Eingabe zu berichtigen, ohne alle Eingaben erneut eingeben zu müssen.

Den dritten Bereich bildet die *Vertraulichkeit*. Die Anwendung muss gewährleisten, dass alle Prozesse richtig ablaufen. Kein Benutzer darf die Möglichkeit haben das System absichtlich oder unabsichtlich zu manipulieren bzw. auszutricksen. Bewusste und unbewusste Manipulationen müssen verhindert werden. Die Prozesse werden deshalb auf ihre Richtigkeit und Plausibilität hin überprüft.

3.8 Benutzeroberfläche

Auf eine Darstellung der Benutzeroberfläche wird an dieser Stelle verzichtet. Hierzu wird auf den Abschnitt *6 Handbuch* verweisen. Dort werden die Benutzeroberflächen der Webplattform und des Browser-Plugins dargestellt.

4 Architektur

Nachdem das Konzept der Anwendung dargestellt wurde, soll nun deren Architektur näher betrachtet werden.

4.1 Komponenten- und Aufgabenverteilung

Bei der Architektur lassen sich drei Bereiche unterscheiden, über die die Komponenten verteilt sind. Dies sind der *Client*, der *Server* und die *Linkbase*.

Die zentrale Komponente auf Seite des Servers stellt die *Datenbank* dar. Hier werden alle Modifikationen hinterlegt. Diese Modifikationen können aus dem Bereich der Linkbase über eine *Linkbasedatei* importiert werden. Die *Konfigurationsdatei* beinhaltet alle Parameter die durch den Administrator angepasst werden können. Unter anderem sind dort die Zugangsdaten für die Datenbank hinterlegt. Um dem Benutzer einen Zugriff für die Verwaltung seiner Modifikationen auf die Datenbank zu gewähren, bildet das *Content-Management-System* die Schnittstelle zwischen dem *Userinterface* und der Datenbank. Das Userinterface unterstützt den Benutzer beim Hinzufügen von neuen Modifikationen zur Datenbank.

Die eigentliche Manipulation der Grafik übernimmt die Komponente der *Grafikmanipulation*. Diese vereint die originale Grafik mit den Modifikationen des Benutzers und gibt die modifizierte Grafik aus.

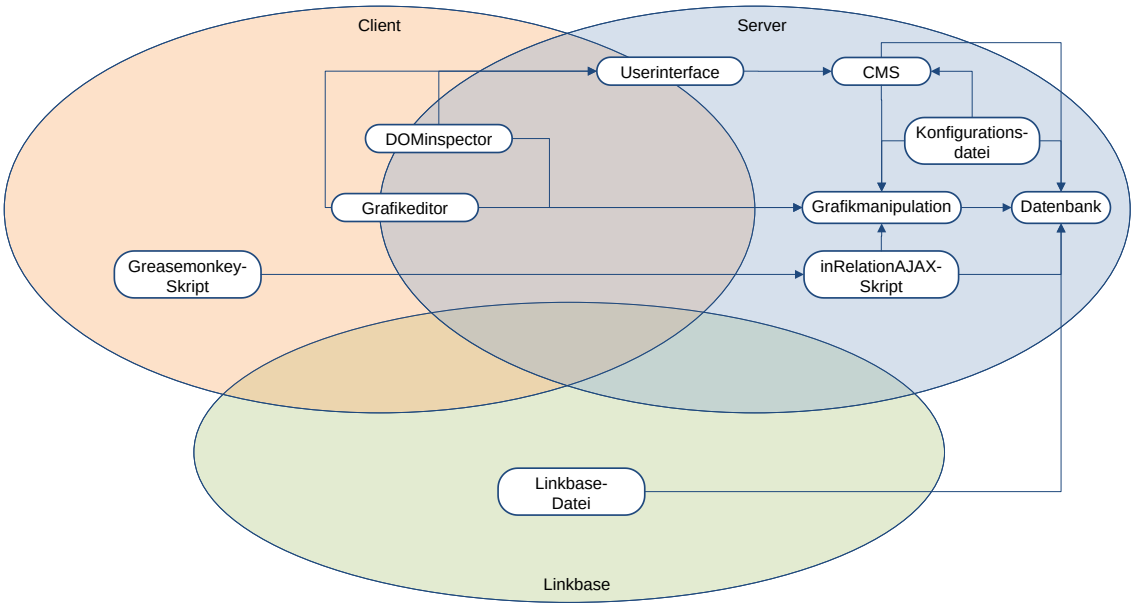

Abbildung 10: Komponentenarchitektur

Auch die Komponenten, die das Userinterface unterstützen, greifen auf die Grafikmanipulation zu, um dem Benutzer eine visuelle Unterstützung durch die modifizierte Grafik zu bieten. Dabei sind die Funktionalitäten des *DOMinspectors* denen der Firefoxerweiterung „DOMinspector“ nachempfunden. Aus diesem Grund wurde auch der Name übernommen. Mit Hilfe des DOMinspectors ist es dem Benutzer möglich, grafikunterstützt, bestimmte Elemente in einer Grafik zu identifizieren und zu adressieren. Die zweite Komponente, die der Unterstützung des Userinterfaces dient, stellt der *Grafikeditor* dar. Hier können einfache Grafikobjekte zu einer Grafik durch einfaches Klicken hinzugefügt werden.

Eigentlich stellen sowohl das Userinterface, der DOMinspector sowie der Grafikeditor serverseitige Komponenten dar. Jedoch wird hier eine Verlagerung in Richtung Clientseite vorgenommen. Der Grund hierfür ist eine erhebliche Verbesserung der Usability für den Benutzer. Wie in Abbildung 10 zu sehen ist, sind der DOMinspector und der Grafikeditor recht weit in Richtung Client verschoben. Dennoch besitzen sie serverseitige Anteile.

Das *Greasemonkey-Skript* läuft hingegen fast ausschließlich clientseitig ab. Jedoch wird ein „serverseitiger Kommunikationspartner“ benötigt. Diesen Partner stellt das *inRelationAJAX-Skript* dar. Seine Aufgabe besteht darin zu überprüfen, ob für eine bestimmte Grafik, die in einer bestimmten Domain eingebunden ist, Modifikationen vom Benutzer hinterlegt sind und somit ggf. eine Manipulation der originalen Grafik nötig ist.

4.2 Client-Server-Kommunikation

Nachdem zuvor die Komponentenverteilung der Anwendung dargestellt wurde, soll nun die Client-Server-Kommunikation erläutert werden. Abbildung 11 zeigt ein Sequenzdiagramm, in dem die Kommunikation zwischen den einzelnen Teilbereichen der Anwendung bei dem Aufruf einer Internetseite abgebildet ist. Das Diagramm ist an ein UML-Sequenzdiagramm angelehnt. Es werden nur die grundlegenden Kommunikationen der Teilbereiche dargestellt.

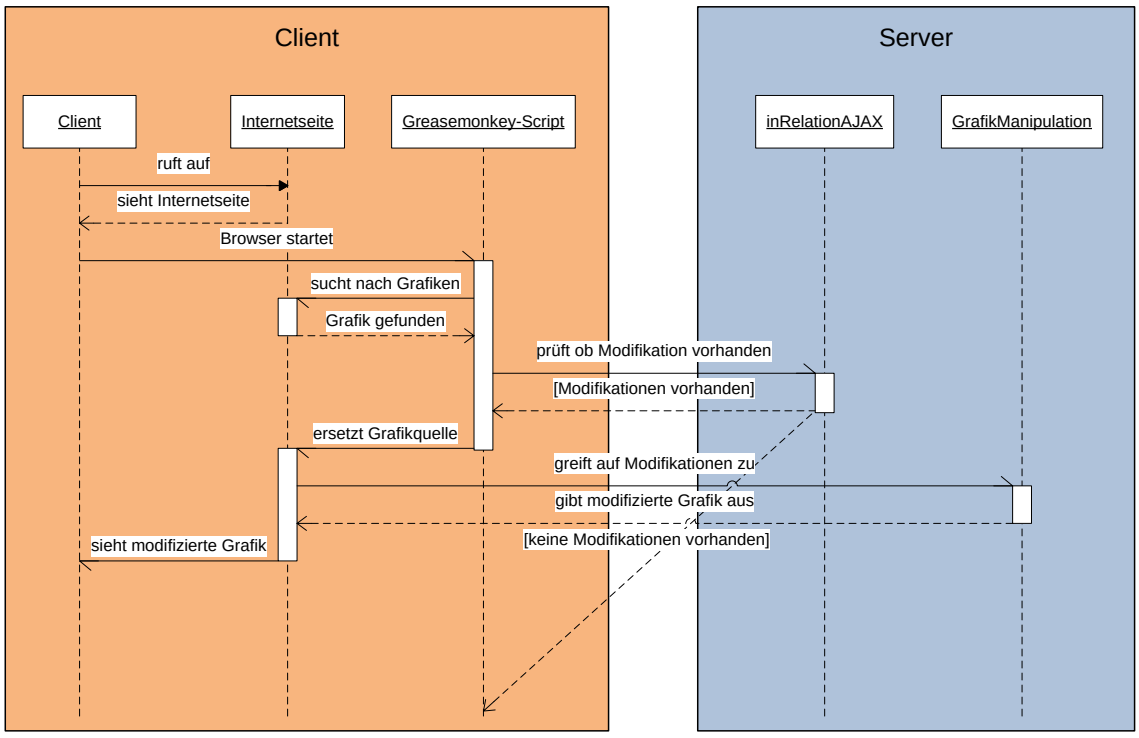

Abbildung 11: Sequenzdiagramm der Client-Server-Kommunikation

Horizontal sind die einzelnen Akteure dargestellt. Die Vertikale stellt eine Zeitachse dar, welche von oben beginnend nach unten verläuft.

Die einzelnen Akteure sind jeweils ihrer Umgebung zugeordnet, in der sie ausgeführt werden. Clientseitig sind der Client selbst, die Internetseite, sowie das Greasemonkey-Skript zu finden. Auf der Seite des Servers sind der inRelationAJAX-Prozess und die Grafikmanipulation angeordnet. inRelationAJAX repräsentiert das Skript zur Überprüfung auf vorhandene Modifikationen. In der Abbildung werden synchrone Kommunikationen durch Pfeile dargestellt, die eine ununterbrochene Linie und eine ausgefüllte Pfeilspitze besitzen.

Antworten sind gekennzeichnet durch eine gestrichelte Linie und eine Doppelpfeilspitze. Pfeile die eine ausgefüllte Linie besitzen und deren Spitze eine einseitige Linie darstellen, stellen asynchrone Anfragen dar. Bedingungen stehen in eckigen Klammern.

Auslösendes Ereignis ist der Aufruf einer Internetseite durch den Client. Nachdem diese Internetseite geladen ist und im Browser des Clients angezeigt wird, startet der Browser die Verarbeitung des Greasemonkey-Skripts. Dies geschieht asynchron. Die Verarbeitung läuft im Hintergrund.

Das Skript durchsucht die Internetseite nach Grafiken. Sollte eine Grafik gefunden werden, löst diese eine Anfrage an das serverseitig laufende inRelationAJAX-Script aus. Die Anfrage prüft, ob Modifikationen für die gefundene Grafik und die Domain, in der diese eingebunden ist, vorhanden sind. Sollten keine Modifikationen vorhanden sein, so terminiert die Überprüfung für diese Grafik. Sind hingegen dem Benutzer zugängliche Modifikationen im System hinterlegt, wird dieses dem Greasemonkey-Skript gemeldet. Daraufhin wird die Grafikquelle der Internetseite durch das Skript zur Grafikmanipulation ersetzt. Dieses manipuliert die ursprüngliche Grafik und gibt diese aus. Im letzten Schritt wird dadurch die manipulierte Grafik für den Benutzer sichtbar.

5 Realisierung

In diesem Abschnitt soll die Realisierung der Anwendung in Ausschnitten betrachtet werden. Auf die detaillierte technische Realisierung wird an dieser Stelle verzichtet. Der gesamte Quellcode ist auf der beiliegenden CD verfügbar.

5.1 Linkbase

Das realisierte XML-Schema dient als Standard für die Linkbase-Datei. Der Standard beinhaltet fünf Elemente. Zum einen die drei Aktionstypen *insert*, *update* und *delete*. Zum anderen existieren die Elemente *data* und *dataref*, mit denen die Aktionen näher beschrieben werden.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <inrelation xmlns:xlink="http://www.w3.org/1999/xlink"
3              xmlns:svg="http://www.w3.org/2000/svg"
4              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5              xsi:noNamespaceSchemaLocation="http://www.inrelation.de/xsd/basic/version1.0/inrelation.xsd"
6              xlink:type="extended">
7
8      <dataref xlink:label="L1" xlink:type="locator" xlink:href="http://beispiel.inrelation.de/logo.svg#xpointer(/*/child::*[position()=4])"/>
9
10     <data xlink:label="R1" xlink:type="resource">
11         <g xmlns="http://www.w3.org/2000/svg">
12             <text>Hier wird Text hinzugefügt</text>
13         </g>
14     </data>
15
16     <data xlink:label="U1" xlink:type="resource">
17         <atomic select="@x" value="400"/>
18         <atomic select="@y" value="400"/>
19     </data>
20
21     <update xlink:type="arc" xlink:from="L1" xlink:to="U1" public="0" domain="http://beispiel.inrelation.de http://www.uni-goettingen.de/informatik/"/>
22
23     <insert xlink:type="arc" xlink:from="L1" xlink:to="L1" public="0" domain="http://beispiel.inrelation.de"/>
24     <insert xlink:type="arc" xlink:from="L1" xlink:to="R1" public="0" domain="http://beispiel.inrelation.de"/>
25
26     <delete xlink:type="arc" xlink:from="L1" xlink:to="NULL" public="0" domain="http://beispiel.inrelation.de"/>
27
28 </inrelation>
```

Abbildung 12: Struktur-Beispiel einer Linkbase-Datei

Abbildung 12 zeigt ein strukturelles Beispiel einer Linkbase-Datei. Hierbei ist die vollständige Sinnfreiheit des abgebildeten Beispiels anzunehmen. Es stellt lediglich die Handhabung der verschiedenen Elemente dar.

- Zeile 1 beschreibt die Tatsache, dass es sich beim vorliegenden Dokument um ein XML-Dokument der Version 1.0 mit Zeichen nach dem UTF-8 Standard handelt.
- Die Zeilen 2 bis 6 beschreiben den Wurzelknoten. Hier werden alle benötigten Namespace-Angeben gesetzt.
- Zeile 8 stellt das Element *dataref* dar. Dieses beschreibt einen Locator, eine Referenz auf eine SVG-Datei. Diese Referenz beinhaltet ebenfalls die Adressierung eines Elementes innerhalt des Dokumentes.
- Die Zeilen 10 bis 14 zeigen ein *data*-Element, welches hier als Ressource dient. Das *g*-Element umschließt den SVG-Quellcode.
- Zeilen 16 bis 19 beschreiben ebenfalls ein *data*-Element. Dieses beinhaltet hingegen Informationen für das Update eines Elementes. Die einzelnen Updateparameter sind über die *atomic*-Elemente realisiert.
- Zeile 21 stellt einen update-Eintrag dar. Dieser bezieht sich auf das in dem Locator mit der Bezeichnung *L1* referenzierte Dokument. Die Updateinformationen sind durch die Ressource mit der Bezeichnung *U1* beschrieben. Des Weiteren ist dieser Eintrag nicht öffentlich und bezieht sich auf die Domains *http://beispiel.inrelation.de* und *http://www.uni-goettingen.de/informatik/*.
- Die Zeilen 23 und 24 stellen zwei *insert*-Elemente dar. Sie unterscheiden sich lediglich im *xlink:to*. Das *insert* der Zeile 23 dupliziert das Element, das in dem dataref *L1*

referenziert wird. Das *insert* in Zeil 24 hingegen fügt SVG-Quellcode dem eigentlichen Dokument hinzu. Der Quellcode stammt aus der Ressource *R1*.

- In Zeile 24 ist ein *delete*-Element dargestellt. Dieses besitzt als Besonderheit ein leeres *xlink:to*, welches durch den Wert *NULL* beschrieben wird.

5.2 Webplattform (serverseitig)

Nachdem die Grundstruktur der Linkbase dargestellt wurde, soll im Nachfolgenden auf die Webplattform genauer eingegangen werden. Dazu werden erst einige allgemeine Dinge betrachtet. Im Anschluss werden einzelne Funktionsbereiche herausgegriffen und näher betrachtet.

5.2.1 Allgemeines

Der nachfolgende Text betrachtet grob Einzelbereiche wie: das Datenbankmodell, die Verzeichnisstruktur, die Klassen der Anwendung, die Konfigurationsdatei sowie den Aufbau des Mini-CMS.

5.2.1.1 Datenbankmodell

Abbildung 13 zeigt das Datenbankmodell hinter inRelation.

- Die einzelnen Rechtecke repräsentieren die Tabellen der Datenbank, mit den jeweiligen Tabellenspalten.
- Die Schlüsselattribute der Tabellen sind durch kleine gelbe Schlüssel markiert.
- Die roten Rauten markieren die Fremdschlüssel.
- Die schwarz-gestrichelten Linien stellen die Beziehungen zwischen den Datenbanktabellen dar.
- Die Besonderheit stellen die orangen Linien dar. Dies sind ebenfalls Beziehungen zwischen den Tabellen. Jedoch handelt es sich hierbei um *1 zu n* Beziehungen zwischen den Tabellen.

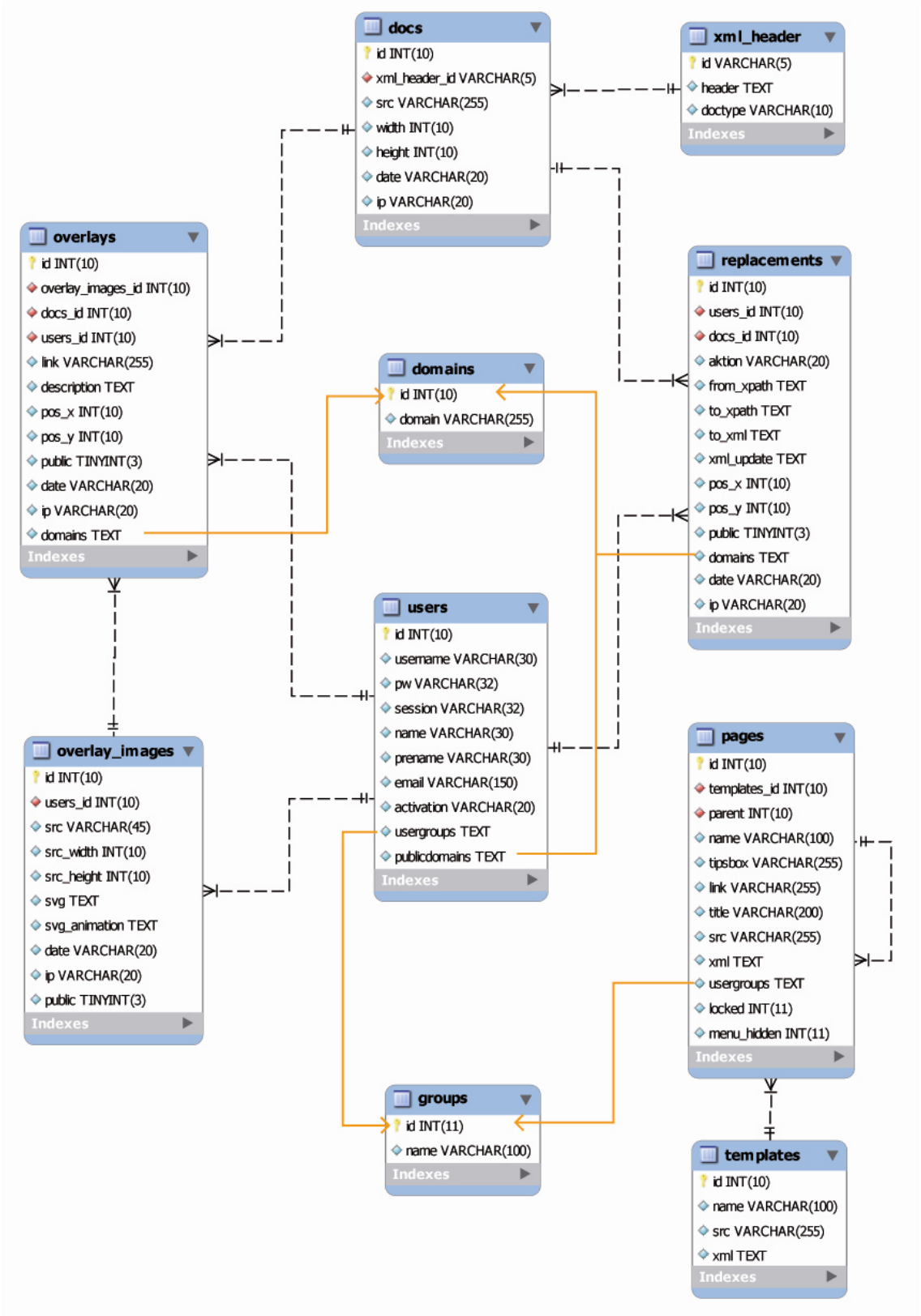

Abbildung 13: Datenbankmodell

5.2.1.2 Verzeichnisstruktur

Tabelle 1 stellt eine Auflistung der Verzeichnisstruktur dar. Dabei wird zu jedem Verzeichnis eine kurze Beschreibung aufgelistet.

Verzeichnis	Beschreibung
/	Dieses Verzeichnis enthält die Konfigurationsdatei sowie andere wichtige Moduldateien. Auch der DOMinspector, Grafikeditor und das Mini-CMS befindet sich in diesem Ordner. Diese Begrifflichkeiten werden an späterer Stelle noch ausführlicher erläutert.
/classes/	Alle Klassen, die in der Anwendung vorhanden sind, befinden sich in diesem Ordner.
/install/	Das Verzeichnis *install* enthält das Installationsscript und die SQL-Datei zur Initialisierung der Datenbank.
/resources/	Hier sind alle Unterverzeichnisse vorhanden die zu den Ressourcen zählen.
/resources/css/	Alle CSS-Dateien, die in dieser Anwendung verwendet werden, liegen in diesem Verzeichnis.
/resources/images/	Dieser Ordner beinhaltet alle Bilder, die für die Webplattform benötigt werden.
/resources/js/	Das Verzeichnis *js* stellt eine Sammlung von Javascript-Dateien dar, die für diese Anwendung benötigt werden.
/resources/pages/	Die Einzelseiten der Webanwendung werden in diesem Ordner abgelegt.
/resources/templates/	Hier sind alle Templates gespeichert, die in der Webplattform verwendet werden.

Tabelle 1: Verzeichnisstruktur-Übersicht

5.2.1.3 Klassen

Die nachfolgende Tabelle stellt die Klassen dar. Dabei wird zu jeder Klasse eine kurze Beschreibung angeführt.

Klasse	Beschreibung
db.class.php	Die Klasse stellt Funktionen zur Kommunikation mit der Datenbank zur Verfügung. Auch komplexere Funktionen wie beispielsweise das Einfügen von Datensätzen unter der Bedingung, dass diese noch nicht vorhanden sind, sind implementiert.
groups.class.php	Die Funktionen dieser Klasse können dazu verwendet werden, um eine String-Liste von Benutzergruppen in ein Array umzuwandeln. Die umgekehrte Umwandlung ist ebenfalls möglich.
image.class.php	Ein Bild kann durch diese Klasse als Objekt dargestellt werden.
linkbase.class.php	Diese Klasse dient zum Importieren einer Linkbase-Datei in die Datenbank und zum Exportieren aus der Datenbank.
manipulation.class.php	In dieser Datei sind alle Klassen und Funktionen gekapselt, die zur Modifikation einer Grafik nötig sind. Dabei werden die Modifikationen direkt aus der Datenbank gelesen.
menu.class.php	Diese Klasse erzeugt das Navigationsmenu, das auf der Webplattform eingesetzt wird. Die Einträge werden aus der Datenbank gelesen.
menuitem.class.php	Diese Klasse stellt einen einzelnen Menupunkt als Objekt einer mehrfach verketteten Liste dar.
page.class.php	Durch diese Klasse wird im Mini-CMS eine Seite aus ihren „Einzelteilen“ zusammengesetzt und ausgegeben.
user.class.php	Die Repräsentation eines Users als Objekt geschieht durch diese Klasse.

xml.class.php	Dient zur Kapselung von Funktionen, die den Umgang mit XML-Dateien erleichtern.
xsddatatypes.class.php	Diese Klasse bildet das XML-Schema einer Linkbase-Datei ab.

Tabelle 2: Klassenübersicht

5.2.1.4 Konfigurationsdatei

Die Datei *inrelation.config.php* stellt die Konfigurationsdatei der Anwendung dar. Es besteht die Möglichkeit einige Einstellungen zu verändern. Hierzu gehören die Zugangsdaten für die Datenbank, die Pfadangaben der Installation auf dem Server, sowie einige Parameternamen.

Die Konfigurationsdatei ist in Form eines bidirektionalen Arrays aufgebaut.

5.2.1.5 Aufbau Mini-CMS

Der Webplattform liegt ein kleines Content-Management-System zu Grunde. Das CMS dient zur Trennung von Inhalt und Layout. Diese Trennung wird durch Aufspaltung einer Seite in ein Template und den Seiteninhalt realisiert. Hierdurch gestaltet sich der Seitenaufruf wie folgt:

- Der Client ruft über die *index.php* mit Übergabe des Parameters *id* eine Seite auf.
- Durch eine Datenbankabfrage wird geprüft, ob der Benutzer eine Zugriffsberechtigung besitzt.
- Liegt diese vor, werden der Pfad zur Template-Datei und der Pfad zur Content-Datei aus der Datenbank abgefragt.
- Im nächsten Schritt wird die Template-Datei geöffnet. Die enthaltenen Platzhalter werden ersetzt. Der Platzhalter für den Content wird durch das Verarbeiten der Content-Datei mit Inhalt gefüllt.
- Nach der Ausgabe ist die zusammengesetzte Seite als Ganzes für den Client sichtbar.

Der obige Ablauf ist bewusst vereinfacht dargestellt. Er bietet lediglich einen groben Überblick. Für die genaue technische Umsetzung sei auf die Dateien */index.php* und */classes/page.class.php* verwiesen.

5.2.2 Funktionsbereiche

Nachfolgend soll die Implementierung einiger grundlegenden Funktionen bzw. Module genauer beschrieben werden.

5.2.2.1 Grafikmanipulation

Bei der Erläuterung des Ablaufs im Modul der Grafikmanipulation wird nicht auf technische Details eingegangen. Hierzu sei auf die Datei */classes/manipulation.class.php* verwiesen. Lediglich der strukturelle Ablauf wird betrachtet.

- Nachdem das Manipulationsmodul unter Angabe der URL zur Grafik aufgerufen wurde, beginnt die Manipulation.
- Zuerst wird mit einer Datenbankabfrage überprüft, ob zugängliche Modifikationen für diese Grafik hinterlegt sind. Zugänglich bedeutet, dass es sich entweder um die eigenen Modifikationen des Benutzers oder um öffentliche Modifikationen anderer Benutzer handelt, die das Veröffentlichungsrecht für die aktuelle Domain besitzen.
- Nachfolgend wird auf die Originalgrafik zugegriffen und diese in XML-ASCII-Serialisierung als Variable gespeichert.
- Das XML-Dokument wird mit DOM geparst und erhält somit den kompletten Funktionsumfang des DOM-Konzepts.

- Alle dem Benutzer zugänglichen Modifikationen werden an dieser Stelle aus der Datenbank ausgelesen.
- Der nächste Schritt dient der Identifikation des zur Manipulation zugehörigen Knotens im Dokument. Dem Knoten wird ein temporäres Attribut hinzugefügt. Als Wert dieses Attributes wird eine fortlaufende Nummer vergeben. Parallel werden die Daten für die Manipulation in einem Array zwischengespeichert. Die fortlaufende Nummer wird hierbei mit zu diesem Datensatz vermerkt. Dies geschieht nacheinander für alle Modifikationen. Nachdem alle Modifikationen in das Array überführt wurden, ist es durchaus wahrscheinlich, dass sich zwei Modifikationen auf denselben Knoten beziehen. Zu beiden Modifikationen wird in dem Fall dieselbe Nummer mitgespeichert. Dadurch bleibt die Korrektheit des Ablaufes gewährleistet.
- Anschließend werden alle Modifikationen aus dem Array auf den entsprechenden Knoten mit dem zuvor gekennzeichneten Attribut, dass die richtige Nummer besitzt, angewendet. Nachdem der Vorgang abgeschlossen ist, besitzen die temporären Attribute keine Bedeutung mehr und werden entfernt.

  Nachfolgendes Beispiel verdeutlicht, wieso der Umweg über ein temporäres Attribut für das korrekte durchführen aller Modifikationen nötig ist. Sei *abcd* eine Zeichenkette. In diese Zeichenkette soll ein *e* an der zweiten Position und ein *f* an der dritten Position eingefügt werden (Die Zählung beginnt an der nullten Position). Die Positionsangaben sollen sich auf die ursprüngliche Zeichenkette beziehen. Somit müsste sich am Ende die Zeichenkette *abecfd* ergeben. Zurück zur ursprünglichen Zeichenkette. Nach dem Einfügen des *e*'s an der zweiten Stelle ergibt sich folgende Zeichenkette: *abecd*. Anschließend soll das *f* an der dritten Stelle eingefügt werden. Dadurch ergibt sich: *abefcd*. Beim Vergleichen dieser Zeichenkette mit der eigentlich gemeinten fällt auf, dass beide nicht identisch sind. Durch das Einfügen des ersten Buchstabens ändern sich die eigentlichen Positionen der anderen nachfolgenden Zeichen.

- Um die Skalierbarkeit der Grafik herzustellen, werden die Breiten- und Höhenangaben aus dem SVG-Tag entfernt. Sollte das Attribut *viewBox* im SVG-Tag nicht vorhanden sein, wird dieses unter Benutzung der alten Höhen- und Breitenangaben erzeugt. Als neue Breiten und Höhenangabe wird der Wert *100%* eingefügt. Damit passt sich die Grafik automatisch an Ihre Umgebung an und ist von außen skalierbar. Dieses wäre nicht der Fall, wenn eine feste Pixel-Angabe als Breite bzw. Höhe angegeben ist.
- Im nächsten Schritt werden die Headerinformationen für eine SVG-Grafik erzeugt und ausgegeben.
- Die DOM-XML-Struktur der modifizierten Grafik muss als ASCII-XML-Repräsentation serialisiert werden, damit diese später ausgegeben werden kann.
- Existieren in der Grafik Verlinkungen auf externe Dateien, ist es erforderlich, diese so anzupassen, dass sie weiterhin funktionsfähig sind. Ein regulärer Ausdruck löst diese Anpassung.
- Abschließend wird der erzeugte SVG-Quelltext ausgegeben. Der Browser des Benutzers interpretiert die Ausgabe dieses Skriptes als SVG-Grafik und zeigt diese an.

5.2.2.2 DOMinspector

Der Name DOMinspector ist an die Erweiterung des Firefox-Browsers angelehnt. Dieser bietet dem Benutzer eine ähnliche Funktionalität.

Für die Implementierung des DOMinspectors wurde eine clientseitige Verarbeitung gewählt. Da viele Benutzerinteraktionen notwendig sind, erhöht sich hierdurch die Usability für den Benutzer. Hierzu wird das JQuery-Javascript-Framework eingesetzt.

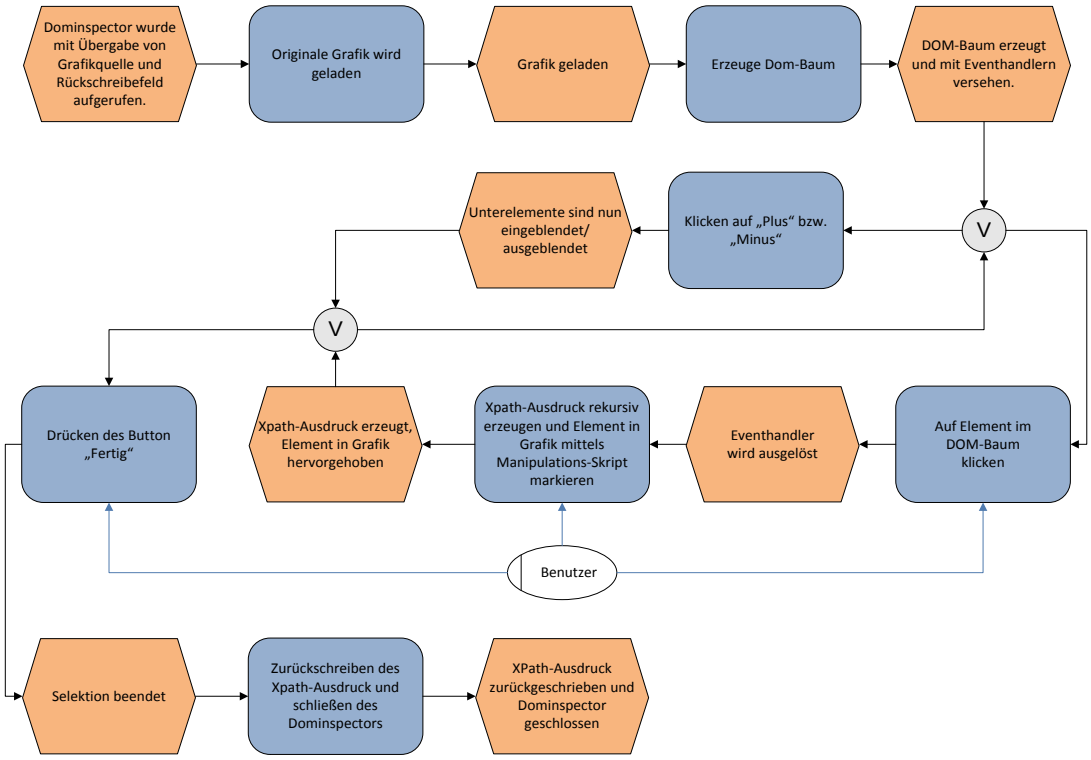

Abbildung 14: Vereinfachte EPK des DOMinspectors

Abbildung 14 stellt die ereignisgesteuerte Prozesskette dar, die dem DOMinspector zu Grunde liegt.

- Der DOMinspector wird mit zwei Übergabeparametern aufgerufen und in einem Popup geöffnet. Zum einen die URL der Grafik und zum anderen der Name des Feldes, in das der generierte XPath-Ausdruck zurückgeschrieben werden soll.
- Daraufhin wird mittels einer AJAX-Anfrage die Grafik als XML-Dokument geladen und dem Benutzer angezeigt.
- Ein DOM-Baum wird aus der XML-Struktur der Grafik erzeugt und mit Eventhandlern versehen.
- Durch Klicken auf eines der *Plus* bzw. *Minus*-Zeichen können die Unterelemente im DOM-Baum ein- bzw. ausgeblendet werden.
- Das Klicken auf ein Element im DOM-Baum löst einen Eventhandler aus. Dieser generiert mittels Rekursion den XPath-Ausdruck und schreibt diesen in das dafür vorgesehene Feld. Zusätzlich wird das Manipulationsskript aufgerufen und das selektierte Element durch dieses optisch hervorgehoben.
- Die Selektion kann beliebig oft wiederholt werden, bis das gewünschte Element ausgewählt wurde.
- Durch das Klicken auf *Fertig* beendet der Benutzer die Selektion.
- Der XPath-Ausdruck wird in das Fenster, das den DOMinspector aufgerufen hat, zurückgeschrieben. Anschließend schließt sich das Fenster des DOMinspectors.

Die genaue technische Umsetzung soll nicht weiter erläutert werden. Hierzu sei auf die Datei *dominspector.php* verwiesen.

5.2.2.3 Grafikeditor

Der Grafikeditor ist ebenfalls überwiegend mittels Javascript umgesetzt. Somit besteht auch hier eine fast ausschließliche clientseitige Verarbeitung. Dieses wurde erneut aus Gründen der Usability gewählt.

Es werden das JQuery-Javascript-Framework sowie das SVG-Plugin zu JQuery verwendet.

Der Aufbau des Grafikeditors ist etwas komplexer. Abbildung 15 zeigt den Ebenen-Aufbau des Grafikeditors. Die Zeichenfläche besteht aus drei Ebenen.

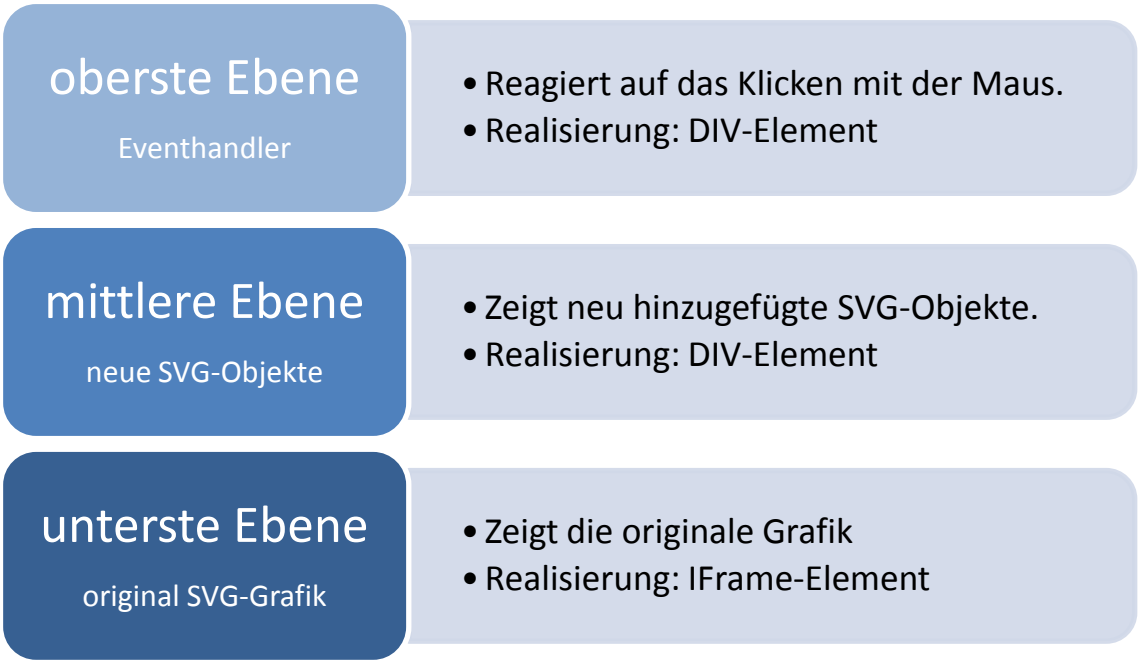

Abbildung 15: Ebenen im Grafikeditor

Die unterste Ebene ist ein IFrame, in dem die Originalgrafik eingebunden ist. Genau darüber liegt ein DIV-Element, das die mittlere Ebene darstellt. In dieses DIV werden alle hinzugefügten SVG-Objekte eingefügt. Dadurch erhält der Benutzer den Eindruck, als wäre sie direkt in die Grafik eingebunden. Die oberste Ebene ist ebenfalls mit einem DIV-Element realisiert. Dieses reagiert auf das Klicken mit der Maus. Die oberste Ebene ist zur Positionsbestimmung für das Hinzufügen von Objekten zuständig.

Durch die absolute Positionierung der drei Ebenen liegen sie genau übereinander und sind exakt gleich groß.

Abbildung 16 zeigt die vereinfachte EPK des Grafikeditors.

- Der Grafikeditor wird mit zwei Übergabeparametern aufgerufen und in einem Popup geöffnet. Zum einen die URL der Grafik und zum anderen der Name des Feldes, in das der generierte SVG-Quellcode zurückgeschrieben werden soll.
- Die originale SVG-Grafik wird geladen, indem sie als Quelle in den IFrame der untersten Ebene eingebunden wird.

- Nach dem Einbinden der Grafik werden die Grafikeigenschaften ermittelt. Hierzu gehören Höhen, Breiten, Offset-Werte und Skalierungsfaktoren.

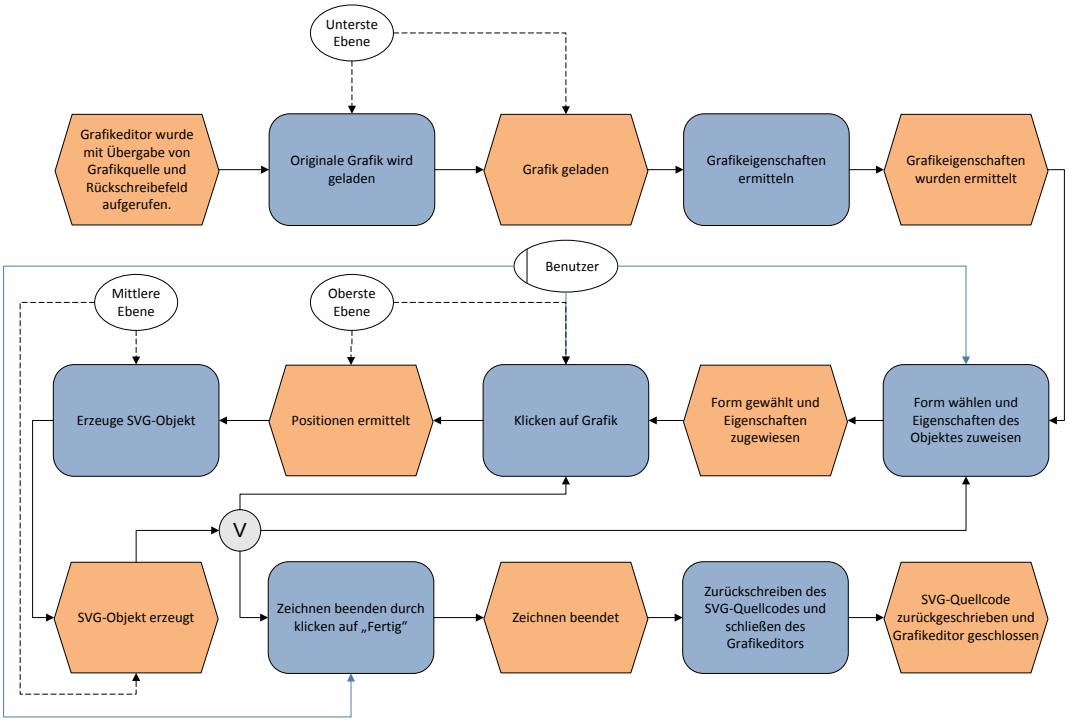

Abbildung 16: Vereinfachte EPK des Grafikeditors

- Der Grafikeditor ist fertig geladen und bereit für den Einsatz. Nun kann eine Form für das zu erzeugende Objekt gewählt werden und die anderen Parameter entsprechend verändert werden.
- Durch das Klicken auf der obersten Ebene werden die Positionen ermittelt, die für die Objekterzeugung nötig sind. Eine genauere Beschreibung, wie Objekte erzeugt werden, ist im Abschnitt 6.2.4.4.2 Grafikeditor zu finden.
- Konnten vom Grafikeditor alle benötigten Positionsangaben bestimmt werden, wird das SVG-Objekt auf der mittleren Ebene erzeugt.
- Nachdem das Objekt erzeugt wurde, besitzt der Benutzer die Möglichkeit weitere Objekte zu erzeugen, ggf. auch eine andere Form auszuwählen. Alternativ kann der Benutzer durch Klicken auf *Fertig* die Eingabe beenden.
- Dadurch wird der SVG-Quellcode in das Fenster, durch das der Grafikeditor aufgerufen wurde, zurückgeschrieben. Anschließend schließt sich das Fenster des Grafikeditors.

Abschließend ist zu erwähnen, dass die Positions- und Größenangaben des erzeugten SVG-Quellcodes für das Einbinden in die Originalgrafik mit dem Skalierungsfaktor skaliert werden.

Die genaue technische Umsetzung wird nicht näher erläutert. Hierzu sei auf die Datei *drawer.php* verwiesen.

5.3 Greasemonkey-Script (clientseitig)

Dem Greasemonkey-Skript müssen die Zugangsdaten bzw. Benutzerdaten für die inRelation-Installation zur Verfügung stehen. Der einfachste Weg Konfigurationsdaten zu speichern besteht darin, die vorgefertigten API-Funktionen von Greasemonkey zu nutzen und damit die

Konfigurationseinstellung in der Konfigurationsdatei von Firefox zu hinterlegen. Die dazu verwendeten Funktionen sind *GM_setValue()* und *GM_getValue()*.

Der genaue Funktionsablauf des Skriptes gestaltet sich wie folgt:

- Die Konfigurationen werden aus der Firefox-Konfigurationsdatei geladen.
- Google-Analytics-Skripte werden aus der Webseite entfernt. Somit können die modifizierten Grafiken nicht von Google aufgespürt werden. Da bei diesem Prototyp der Benutzername und das Passwort vom Benutzer unverschlüsselt übertragen werden, ist diese Handlung aus Sicherheitsgründen unabdingbar.
- Der Konfigurations-Button mit dem Konfigurations-Fenster wird erzeugt und in die Webseite integriert.
- Sollte der inRelation-Button bereits zu einem früheren Zeitpunkt durch den Benutzer verschoben worden sein, wird dieser an die zu letzt bekannte Position gesetzt.
- Es werden alle SVG-Grafiken identifiziert, die mittels eines Object-Tags in die HTML-Datei integriert sind. Für jede gefundene Grafik wird mittels einer AJAX-Anfrage an den inRelation-Server überprüft, ob Modifikationen vorhanden sind. Sollten Modifikationen hinterlegt sein, wird die originale Grafikquelle durch das Manipulations-Script ersetzt. Die AJAX-Anfragen werden mittels der *GM_xmlhttpRequest()* API-Funktion von Greasemonkey umgesetzt.
- Im nächsten Schritt werden alle SVG-Grafiken gesucht, die mittels eines Embed-Tag eingebunden sind. Die Prozedur ist hier für jede gefundene Grafik die gleiche wie zuvor.
- Im letzten Schritt werden alle restlichen Bilder aufgespürt, die mittels eines herkömmlichen IMG-Tags eingebunden sind. Auch hier wird auf Modifikationen geprüft und ggf. der IMG-Tag durch einen Embed-Tag, mit dem Manipulationsskript als Quelle, ersetzt.

Der vorgestellte Funktionsablauf ist bei jeder Internetseite derselbe, sofern das Greasemonkey-Skript aktiv ist.

Für die Konfiguration wurde ein Konfigurationsfeld erzeugt. Mit dem Speichern werden die neuen Einträge in die Firefox-Konfigurationsdatei geschrieben. Zusätzlich wird der Benutzer an der Webplattform angemeldet, um ihm eine neue Session zuzuordnen.

Das Konfigurationsfeld ist in Abbildung 17 in Abschnitt 6.1.2 des Handbuchs abgebildet. Dort ist auch eine Anleitung zur Bedienung abgedruckt.

6 Handbuch

6.1 Installation und Konfiguration

Im folgenden Abschnitt wird die schrittweise Installation und Konfiguration von inRelation erklärt.

6.1.1 Webplattform (serverseitig)

Als Erstes wird die aktuelle Version von inRelation benötigt. Auf der Internetseite http://www.inrelation.de kann diese, als ZIP-Archiv, heruntergeladen werden.

Nach erfolgreichem Entpacken des ZIP-Archives muss die Konfigurationsdatei (*inrelation.config.php*) von inRelation an die Serverumgebung angepasst werden. Es müssen die Zugangsdaten für die Datenbank, sowie die Pfadangaben für die Serverumgebung angepasst werden.

Sind die Konfigurationseinträge richtig gesetzt, kann die eigentliche Installation von inRelation durchgeführt werden. Hierfür steht unter http://*Domain*/*inRelationPfad*/install/index.php ein Installationsskript zur Verfügung. Durch Klicken auf den Button *inRelation installieren* wird die Installation durchgeführt. Die Datenbank wird initialisiert. Ist die Installation erfolgreich abgelaufen, werden die Zugangsdaten für den Administrator angezeigt.

Die Grund-Installation und –Konfiguration von inRelation ist erfolgreich abgeschlossen.

6.1.2 Greasemonkey-Script (clientseitig)

Nachdem die Plattform erfolgreich eingerichtet wurde, kann das Greasemonkey-Skript installiert werden. Dazu ist zuvor die Installation des Firefox-Plugins Greasemonkey nötig. Das Plugin kann auf http://www.greasespot.net/ heruntergeladen werden.

Abbildung 17: Konfigurationsfeld des Greasemonkey-Skripts

Nach erfolgreicher Installation und Aktivierung von Greasemonkey kann das inRelation-Skript hinzugefügt werden. Da Greasemonkey aktiviert ist, reicht zur Installation ein Klick auf

folgende URL: http://www.inrelation.de/download/inrelation.user.js . Greasemonkey erkennt das Skript automatisch und durch Klicken auf *Installieren* wird das Skript automatisch installiert.

Das Skript in nun erfolgreich installiert. Zur Konfiguration des Scripts wird eine beliebige Webseite im Firefox geladen. Das Skript erzeugt einen zusätzlichen Button in der oberen Ecke der Webseite. Durch Klicken auf das Plus-Zeichen in der rechten oberen Ecke des inRelation-Buttons erscheint ein Konfigurationsfeld.

Hier müssen die Domain zur inRelation-Installation sowie der Benutzername und das dazugehörige Passwort eingetragen werden. Durch Klicken auf *Speichern* ist die Konfiguration abgeschlossen.

6.2 Webplattform (serverseitig)

In diesem Abschnitt werden die einzelnen Funktionen der Webplattform näher erläutert. Als Grundlage dient die Installation der Plattform auf http://www.inrelation.de .

6.2.1 Login

Wird die Domain der Webplattform aufgerufen, erscheint ein Loginfeld. Hier kann sich ein Benutzer mit seinem Usernamen und Passwort anmelden.

Abbildung 18: Startseite der inRelation-Webplattform

6.2.2 Passwort ändern

Nach erfolgreichem Login steht dem Benutzer, unter dem Menupunkt *Profil > Passwort ändern* die Funktionalität bereit sein Passwort zu ändern. In dem ersten Feld muss das Passwort eingegeben werden. In dem zweiten muss dasselbe Passwort zur Kontrolle erneut eingegeben werden.

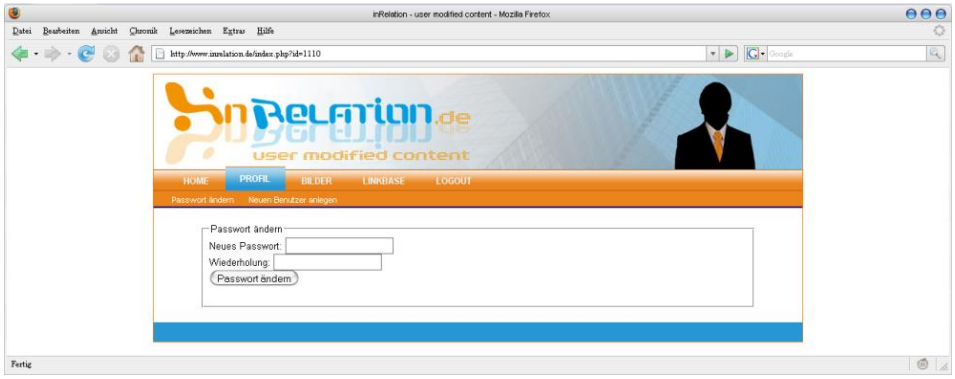

Abbildung 19: Passwort ändern der inRelation-Webplattform

6.2.3 Benutzer anlegen

Die Funktionalität einen neuen Benutzer anzulegen steht nur Benutzern der Webplattform zur Verfügung, die Administrations-Rechte besitzen. Um einen Benutzer erfolgreich anzulegen müssen alle Felder ausgefüllt werden. Die meisten der Felder sind selbsterklärend.

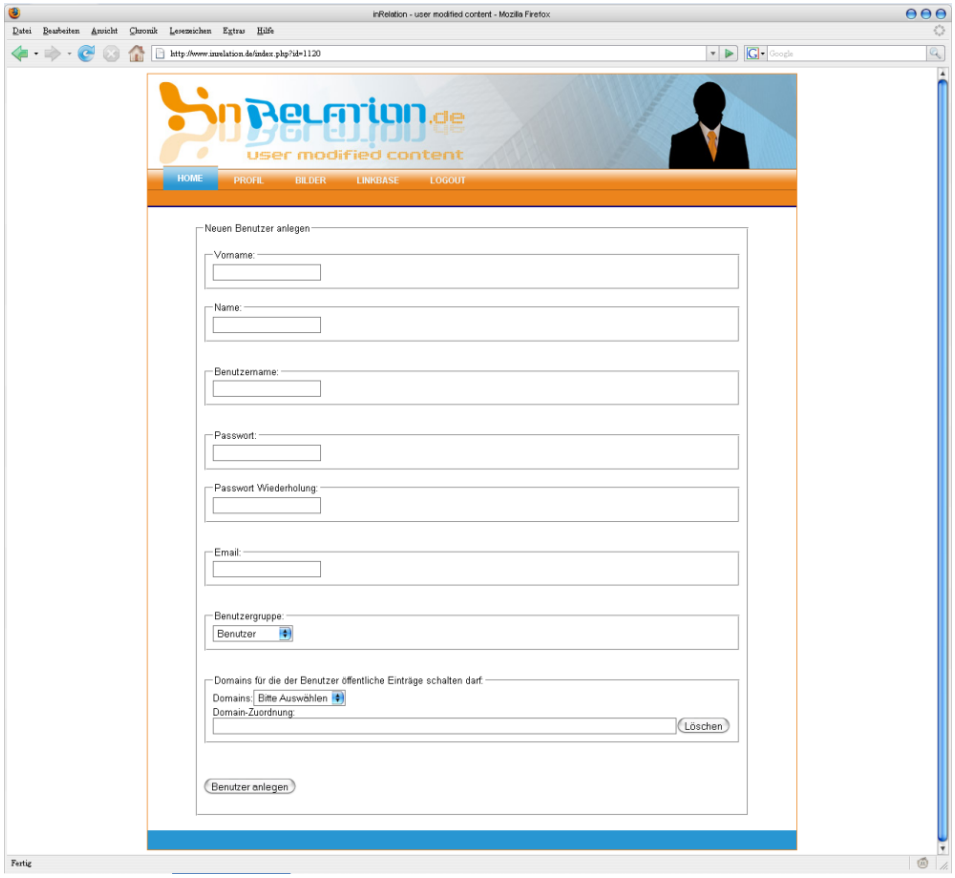

Abbildung 20: Neuen Benutzer anlegen der inRelation-Webplattform

Es soll das Feld *Domain-Zuordnung* herausgegriffen und genauer erklärt werden. Hier werden die Domains ausgewählt, für die der neue Benutzer das Recht erhalten soll, öffentlich sichtbare Einträge zur Linkbase hinzuzufügen. Sollten bereits Domains in der Linkbase benutzt werden, stehen diese im Drop-Down-Menu zum Auswählen zur Verfügung. Durch das aufeinanderfolgende Auswählen von Einträgen aus dem Drop-Down-Menu können mehrere Domains für einen Benutzer freigeschaltet werden. Sollte die gewünschte Domain nicht im Drop-Down-Menu vorhanden sein, so kann diese auch manuell in das Textfeld eingetragen

werden. Generell stellt das Textfeld eine Liste von Domains dar, die mittels eines Leerzeichens getrennt sind.

6.2.4 Linkbase

Im Abschnitt Linkbase sollen alle Funktionalitäten vorgestellt werden, die direkten Einfluss auf die Linkbase besitzen.

6.2.4.1 Übersicht der Manipulationen

Unter dem Menupunkt *Bilder > Übersicht* wird eine Liste von Bildern angezeigt, für die Modifikationen vorhanden sind.

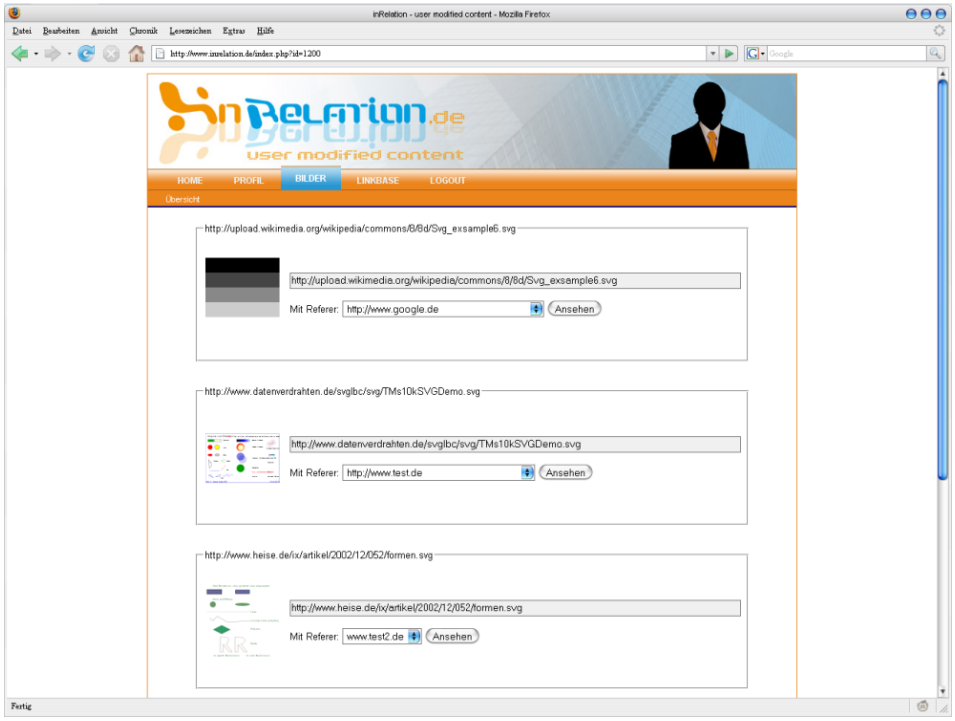

Abbildung 21: Bilder Übersicht der inRelation-Webplattform

Abbildung 21 zeigt die Auflistung von drei Bildern, die in der Test-Plattform vorhanden sind. Für jedes Bild wird die Original-Grafik links klein angezeigt. Rechts daneben befindet sich die URL zu der Original-Grafik. In dem Drop-Down-Menu darunter stehen alle URLs für die Manipulationen vorhanden sind.

Sollte die erste Grafik auf http://www.google.de oder einer Unterseite eingebunden sein, so sind dazu Modifikationen durch den Benutzer hinterlegt worden.

Durch Klicken auf *Ansehen* wird das Bild mit allen vorhandenen Modifikationen für den ausgewählten Referer angezeigt. Ein Referer stellt die Domain dar, für welche die Grafik betrachtet werden soll. Über diese Funktion erhält der Benutzer einen Überblick darüber, für welche Bilder und welche Domains in der Linkbase Modifikationen hinterlegt sind. Diese Modifikationen können sowohl vom Benutzer selbst angelegt worden sein, wie auch öffentliche Einträge von anderen Benutzern der Webplattform darstellen. Anzumerken ist, dass für öffentliche Einträge der einstellende Benutzer für die entsprechende Domain freigeschaltet sein muss.

6.2.4.2 Importieren

Mit dem Menupunkt *Linkbase > Importieren* besitzt der Benutzer die Möglichkeit eine Linkbase-Datei, die valide mit dem inRelation-XML-Schema ist, zu importieren. Diese Linkbase-Datei kann entweder vom Benutzer per Hand erzeugt worden sein, oder stellt einen Export aus einer inRelation-Linkbase dar.

Zum Importieren wird der Inhalt der Linkbase-Datei in das Textfeld kopiert und mit dem Klick auf den Button *Linkbase importieren* zur gesamten Linkbase hinzugefügt. Dabei werden nur Einträge berücksichtigt, die noch nicht in der Linkbase vorhanden sind.

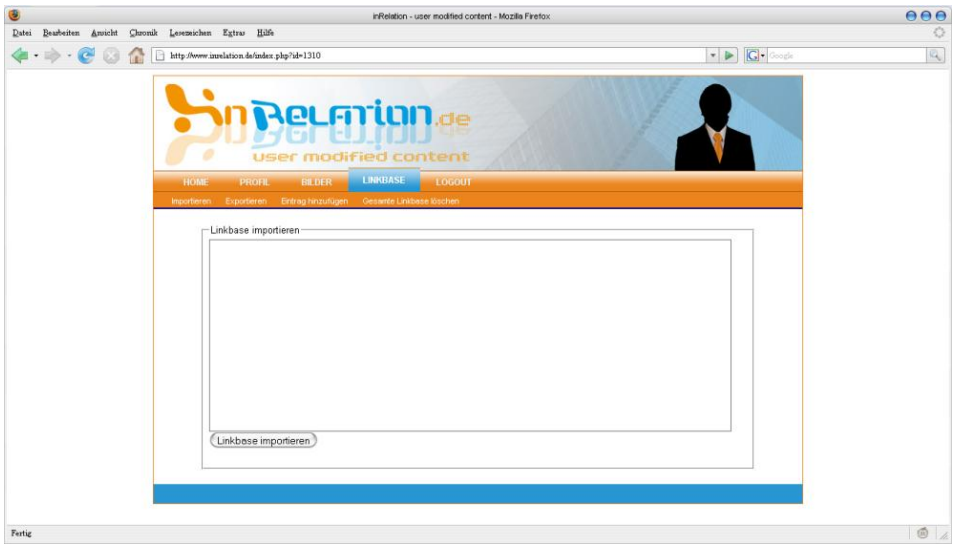

Abbildung 22: Linkbase importieren der inRelation-Webplattform

6.2.4.3 Exportieren

Dem Benutzer steht mit dem Menupunkt *Linkbase > Exportieren* die Möglichkeit offen, seine Linkbase zu exportieren. Die kann als Backup dienen oder auch um sie in einer anderen inRelation-Installation zu importieren. Beim Exportieren hat der Benutzer zwei Möglichkeiten. Zum einen können nur die eigenen Einträge exportiert werden und zum anderen ist es ebenfalls möglich auch alle öffentlichen Einträge von anderen Benutzern für den Export mit zu berücksichtigen. Hierfür muss im Drop-Down-Menu *Ja* ausgewählt werden.

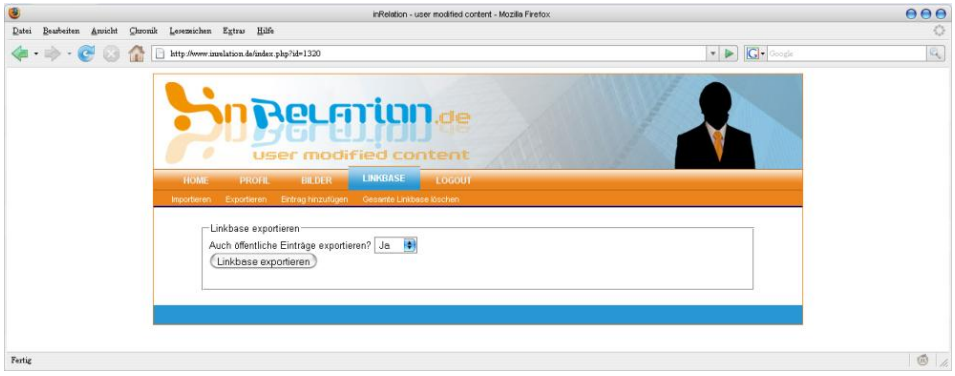

Abbildung 23: Linkbase exportieren der inRelation-Webplattform

Nach dem Klicken auf *Linkbase exportieren* wird ein Textfeld mit der XML-Linkbase-Datei angezeigt. Nun kann der Inhalt des Textfeldes kopiert und gespeichert werden.

6.2.4.4 Eintrag hinzufügen

Unter dem Menupunkt *Linkbase > Eintrag hinzufügen* steht dem Benutzer die Möglichkeit zur Verfügung, eine Modifikation mit Hilfe eines Editors zur Linkbase hinzuzufügen. In Abbildung 24 ist der Editor abgebildet. Im oberen Bereich, der *Aktionsauswahl*, hat der Benutzer die Möglichkeit auszuwählen, welche Art einer Modifikation er durchführen möchte.

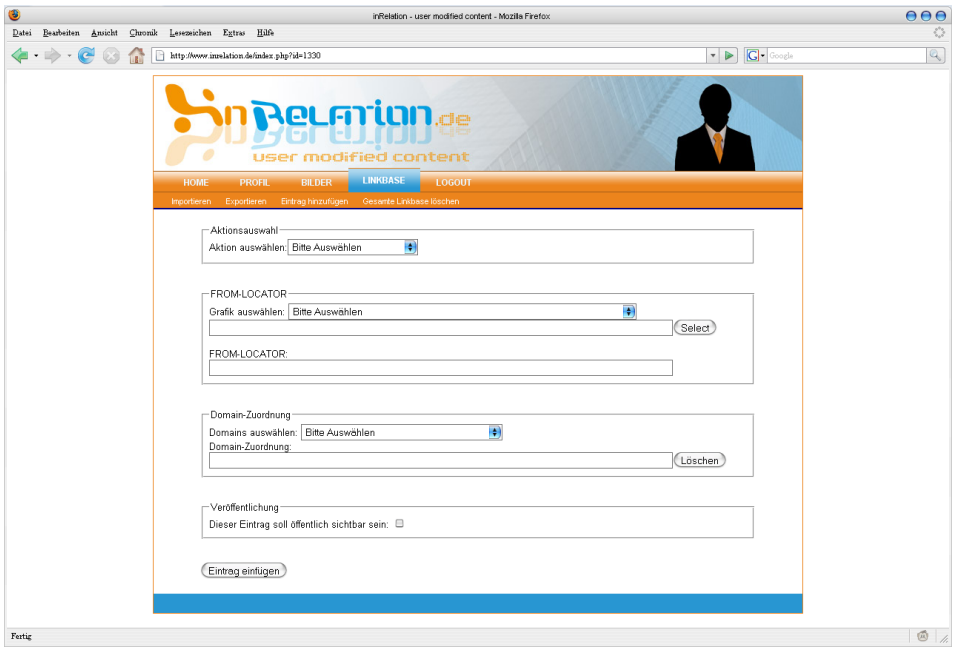

Abbildung 24: Eintrag hinzufügen der inRelation-Webplattform

Es stehen dem Benutzer drei Aktionen zur Verfügung.

- *insert >* Etwas in eine Grafik einfügen
- *delete >* Etwas aus einer Grafik löschen
- *update >* Etwas in einer Grafik verändern

Abbildung 25 zeigt das ausgeklappte Drop-Down-Menu mit den Auswahlmöglichkeiten. Es ist zu sehen, dass bei *insert* dem Benutzer drei verschiedene Möglichkeiten zur Verfügung stehen. Der Benutzer hat die Möglichkeit mit *insert (aus anderer Grafik)* ein Grafikelement aus einer beliebigen anderen SVG-Grafik in die zu modifizierende Grafik einzufügen. Mit *insert (SVG_Quellcode)* ist es möglich, direkten SVG-Quellcode in die zu modifizierende Grafik einzufügen. Der Quellcode kann hierbei beispielsweise selbst erzeugt oder aus einem Grafikprogramm kopiert werden. Die dritte Variante, *insert (Grafikeditor)*, stellt einen kleinen Grafikeditor dar, der es ermöglicht Grundformen zu erzeugen und in der Grafik zu platzieren. Genaueres hierzu ist in *6.2.4.4.2 Grafikeditor* zu finden.

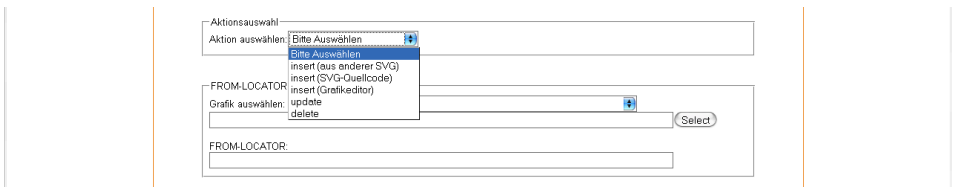

Abbildung 25: Aktionsauswahl beim Eintrag hinzufügen

Nachdem der Benutzer eine Auswahl der Aktion getroffen hat, muss die zu modifizierende Grafik ausgewählt werden. Dieses geschieht mittels des Funktionsbereiches *From-Locator*. Der

Benutzer hat die Möglichkeit über das Drop-Down-Menu eine Grafik auszuwählen, die bereits im System verwendet wird. Ist die Grafik hier nicht vorhanden, kann die URL zur Grafik in das Textfeld unterhalb des Drop-Down-Menus eingetragen werden. Durch Klicken auf *Select* wird ein Popup geöffnet und es erscheint der *DOMinspector*. Eine ausführliche Beschreibung zum DOMinspector ist in Abschnitt 6.2.4.4.1 zu finden. Der Funktionsbereich From-Locator ist in Abbildung 26 dargestellt.

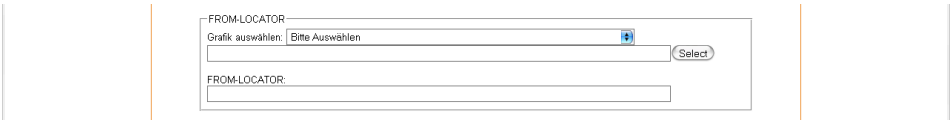

Abbildung 26: Funktionsbereich From-Locator beim Eintrag hinzufügen

Der nächste Funktionsbereich unterscheidet sich je nach ausgewählter Aktion. Ist als Aktion *delete* ausgewählt, so ist dieser Funktionsbereich nicht vorhanden.

Abbildung 27 zeigt den Funktionsbereich für die Aktion *insert (aus anderer SVG)*. Hier hat der Benutzer, wie beim From-Locator, die Möglichkeit eine Grafik auszuwählen und mittels des DOMinspectors ein beliebiges Element zu adressieren. Dieses Element wird daraufhin in die zu modifizierende Grafik eingefügt. Natürlich kann die Grafik aus der das Element entnommen wird auch identisch mit der zu modifizierenden sein. Dies hat dann zur Folge, dass ein Element dupliziert wird.

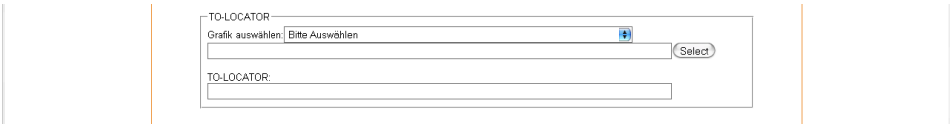

Abbildung 27: Funktionsbereich To-Locator bei „insert (aus anderer SVG)“ beim Eintrag hinzufügen

Ist als Aktion *insert (SVG-Quellcode)* ausgewählt, so ist hier als Funktionsbereich *SVG-Quellcode* sichtbar. Dieser wird in Abbildung 28 dargestellt. Hier hat der Benutzer die Möglichkeit, den SVG-Quellcode, der in die zu modifizierende Grafik eingefügt werden soll, per Hand einzugeben. Den Quellcode kann sich der Benutzer auch zuvor durch ein vektorbasiertes Grafikprogramm erzeugen lassen und anschließend in dieses Textfeld kopieren.

Abbildung 28: Funktionsbereich SVG-Quellcode bei „insert (SVG-Quellcode)" beim Eintrag hinzufügen

Dem Benutzer steht als dritte Möglichkeit ein Grafikeditor zur Verfügung, um etwas zu einer Grafik hinzuzufügen. Hierfür muss unter Aktionsauswahl *insert (Grafikeditor)* ausgewählt werden. Es erscheint, wie in Abbildung 29 abgebildet, der Funktionsbereich *SVG-Grafikeditor*.

Hier wird dem Benutzer ein Textfeld und darüber ein Button mit der Aufschrift *Grafikeditor starten* angezeigt. Durch das Klicken auf diesen Button wird in einem Popup der Grafikeditor gestartet. Die genaue Funktionsweise des Grafikeditors ist in *6.2.4.4.2 Grafikeditor* zu finden. Nachdem der Benutzer die gewünschten Grafikelemente mit dem Grafikeditor erzeugt hat, werden diese in dem Textfeld angezeigt. Nun besteht noch die Möglichkeit, per Hand nachträgliche Veränderungen am erzeugten SVG-Quellcode vorzunehmen.

Abbildung 29: Funktionsbereich SVG-Grafikeditor bei „insert (Grafikeditor)" beim Eintrag hinzufügen

Wurde als Aktion *update* ausgewählt, erscheint der Funktionsbereich Update-Attribute. Dieser Funktionsbereich ist in Abbildung 30 dargestellt. Hier werden die Attribute und Ihre neuen Werte angegeben. Ein Update bewirkt die Änderung eines vorhandenen bzw. das Hinzufügen eines noch nicht vorhandenen Attributes. Hierzu werden, in dem Textfeld pro Zeile, ein Attribut mit zugehörigem Attributwert, getrennt durch zwei Doppelpunkte, angegeben.

Abbildung 30: Funktionsbereich Update-Attribute bei "update" beim Eintrag hinzufügen

Der Funktionsbereich *Domain-Zuordnung* ist, unabhängig von der ausgewählten Aktion, immer sichtbar. Dieser ist in Abbildung 31 dargestellt. Hier besitzt der Benutzer die Möglichkeit seine Manipulation bestimmten Domains zuzuordnen. Dazu können Domains aus dem Drop-Down-Menu ausgewählt werden. Alternativ können auch per Hand Domains in das Textfeld darunter eingetragen oder ergänzt werden. Das Textfeld beinhaltet eine Liste von Domains, die mittels eines Leerzeichens getrennt werden.

Abbildung 31: Funktionsbereich Domain-Zuordnung beim Eintrag hinzufügen

Abbildung 32 zeigt den letzten Funktionsbereich, der dem Benutzer unter der Überschrift *Veröffentlichung* zur Verfügung steht. Hier besitzt dieser die Möglichkeit, seine Manipulation als öffentlich zu kennzeichnen, so dass sie auch für andere Nutzer der Anwendung sichtbar sind. Dabei ist zu beachten, dass die Kennzeichnung *öffentlich* nur umgesetzt werden kann, wenn der Benutzer die Rechte besitzt für eine der ausgewählten Domains öffentliche Modifikationen einzustellen. Dies bedeutet, wenn der Benutzer eine Modifikation mit vier Domains verknüpfen will, wovon er für zwei die Berechtigung für einen öffentlichen Eintrag besitzt, dann wird der Eintrag nur für diese beiden Domains öffentlich sichtbar sein.

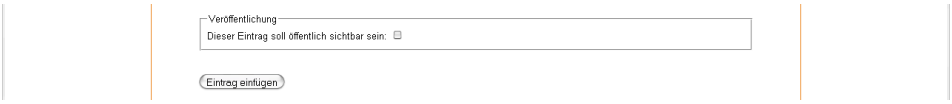

Abbildung 32: Funktionsbereich Veröffentlichung beim Eintrag hinzufügen

6.2.4.4.1 DOMinspector

Wie im vorherigen Abschnitt bereits erwähnt, besteht die Aufgabe des DOMinspectors darin, ein bestimmtes Element in einer SVG-Grafik oder allgemein in einem XML-Dokument auszuwählen. In Abbildung 33 ist der DOMinspector für das inRelation-Logo abgebildet. Im Funktionsbereich *DOM-Baum* ist die Baumstruktur der SVG-Grafik dargestellt. Durch das Klicken auf ein *Minus-Symbol* werden die Unterelemente ausgeblendet. Dem entsprechend bewirkt das Klicken auf ein *Plus-Symbol* das Einblenden der Unterelemente. Die einzelnen Knoten der SVG-Grafik werden in Form eines *Links* dargestellt. Durch Klicken auf einen dieser Links wird das entsprechende Element in der Grafik rechts durch eine orange Kontur hervorgehoben. Ebenfalls wird der XPath-Ausdruck für dieses Element im Funktionsfeld *XPath* angezeigt. In Abbildung 34 ist das *R* im inRelation-Logo durch Klicken auf das entsprechende Element ausgewählt worden. Das Element wurde zur Verdeutlichung im DOM-Baum blau hinterlegt. Des Weiteren ist der XPath Ausdruck zu diesem Element sichtbar. Beim Betrachten des Logos selbst fällt die orange Kontur um das *R* auf.

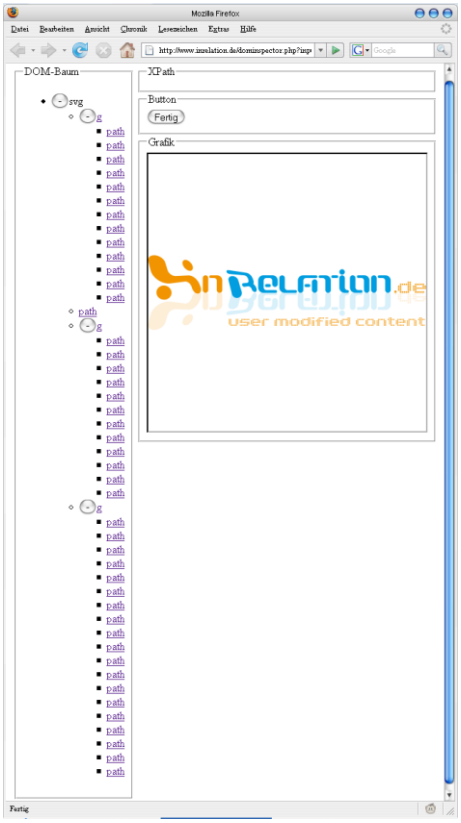

Abbildung 33: DOMinspector

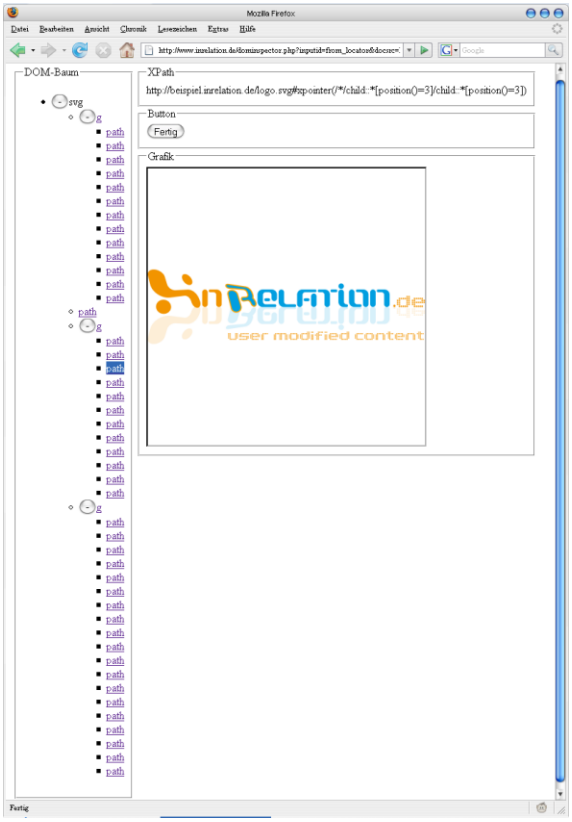

Abbildung 34: DOMinspector mit aktiver Auswahl

Wollte der Benutzer eigentlich ein anderes Element in der Grafik adressieren, kann er erneut auf einen Link im DOM-Baum klicken, um ein anderes Element hervorzuheben.

Nachdem das richtige Element identifiziert ist, wird die Selektion durch einen Klick auf den Button *Fertig* beendet.

6.2.4.4.2 Grafikeditor

Der Grafikeditor stellt ein Werkzeug dar, um einfache Grafik- und Textelemente zu einer Grafik hinzuzufügen. Abbildung 35 zeigt den Aufbau des Grafikeditors.

Zentral wird im Grafikeditor die Grafik angezeigt, zu der Grafikelemente oder Texte hinzugefügt werden sollen. Rechts daneben befindet sich der Funktionsbereich *Grafikeigenschaften*. Hier werden dem Benutzer Informationen zu der Grafik angezeigt. Die Eigenschaften *DIV-Breite* und *–Höhe* beschreiben die aktuelle Größe, in der die Grafik angezeigt wird. *Bild-Breite* und *–Höhe* geben die Originalgröße der Grafik an. Da das Feld, in dem die Grafik angezeigt wird, quadratisch ist, muss die Grafik an die Feldgröße angepasst werden. Hierzu wird die Grafik proportional skaliert. Das heißt, entweder begrenzt die Höhe der Grafik die Anzeige oder die Breite. Bei dem abgebildeten inRelation-Logo wird die Grafik durch die Breite begrenzt. Somit ist oberhalb und unterhalb der Grafik im Feld Leerraum vorhanden. Der Leerraum oberhalb der Grafik wird als *Bild-Offset-Y* angegeben. Entsprechend ist auch das *Bild-Offset-X* zu verstehen. Der letzte Wert, der angezeigt wird, ist die *Skalierung*. Diese beschreibt den Faktor, um welchen die Originalgrafik größer ist als die abgebildete Grafik.

Unterhalb des Funktionsbereiches der Grafikeigenschaften ist der Funktionsbereich *Button* angesiedelt. Hier steht dem Benutzer der Button *Fertig* zur Verfügung, mit dem die

Grafikbearbeitung abgeschlossen wird. Des Weiteren erscheinen hier, bei bestimmten Formen, weitere Button mit Zusatzfunktionalitäten.

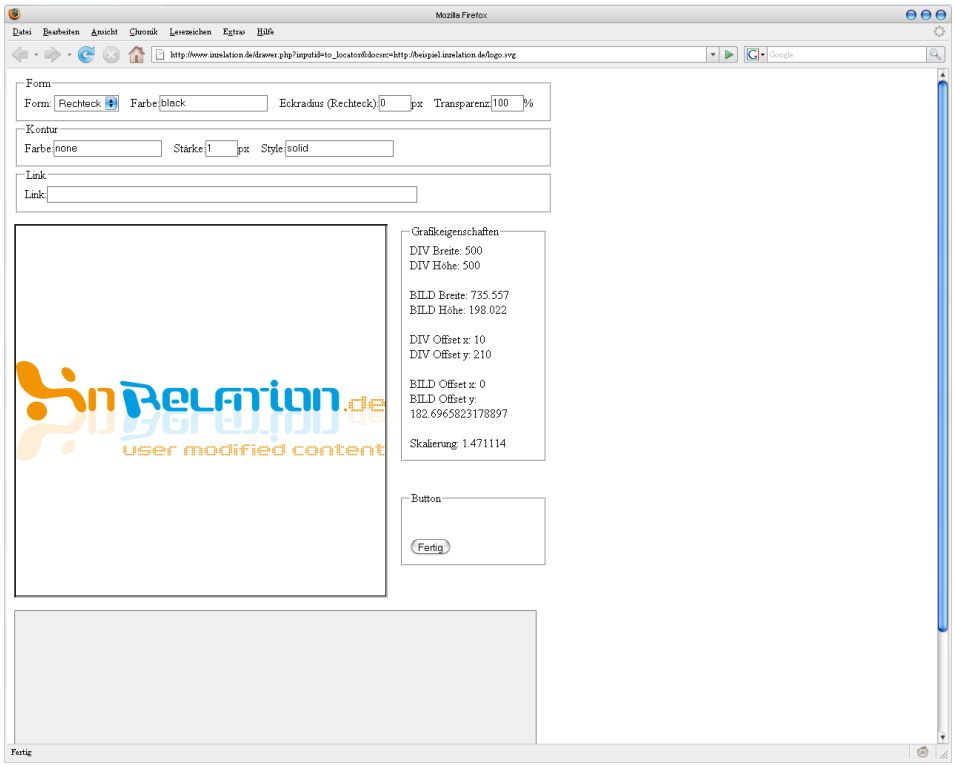

Abbildung 35: Grafikeditor

Ganz unten wird im Grafikeditor ein grau hinterlegtes Textfeld angezeigt. Dieses Textfeld zeigt den SVG-Quellcode, der zu den Objekten gehört, die vom Benutzer zur Grafik hinzugefügt wurden.

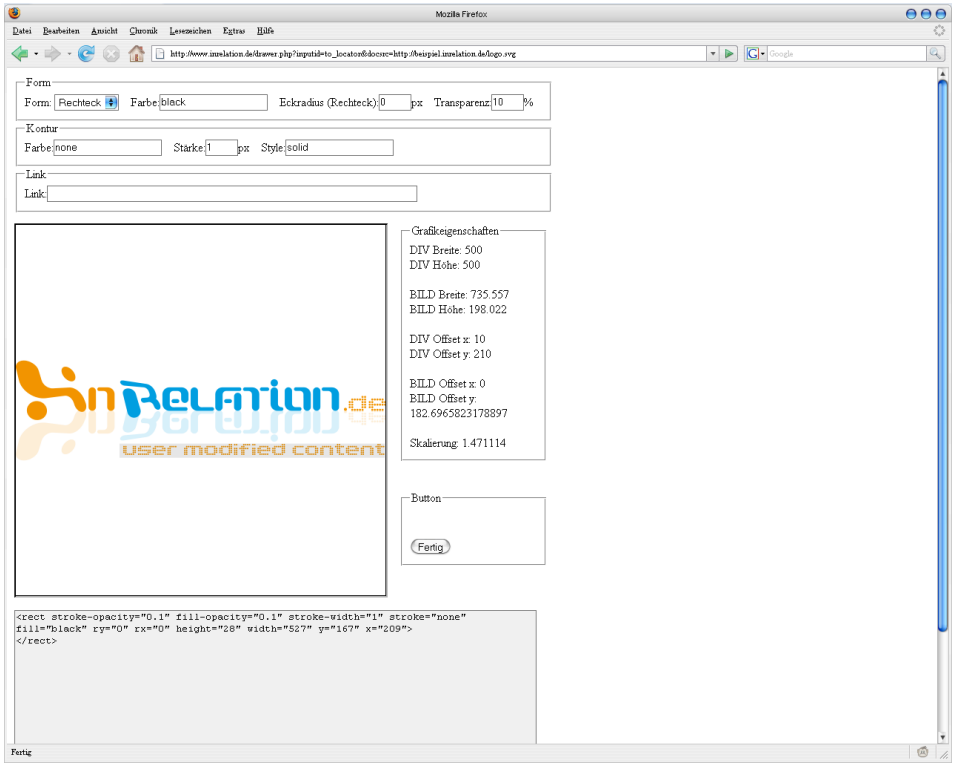

Abbildung 36: Grafikeditor mit eingefügtem Rechteck

In Abbildung 36 ist beispielsweise ein schwarzes Rechteck mit 10% Transparenz über den Schriftzug *user modified content* gelegt. Zum einen wird das Rechteck in der Grafik angezeigt und zum anderen ist der SVG-Quellcode sichtbar. Im oberen Bereich des Grafikeditors sind die eigentlichen Funktionen angesiedelt. Der Funktionsbereich *Form* beinhaltet ein Drop-Down-Menu, in dem die gewünschte Objektform ausgewählt werden kann. Es stehen dem Benutzer die Formen: Rechteck, Kreis, Ellipse, Linie Text, Polyline und Polygon zur Verfügung. Die Handhabung dieser Formen wird etwas später in diesem Abschnitt erläutert. Des Weiteren ist in diesem Funktionsbereich noch die Farbe für das Objekt zu wählen. Auch ein Eckradius kann ausgewählt werden. Damit ist es möglich bei einem Rechteck die Ecken abzurunden. Zuletzt kann die Transparenz des Objektes angegeben werden. 100% steht hierbei für volle Deckkraft.

Der zweite Funktionsbereich fasst die Eigenschaften der *Kontur* zusammen. Hier wird die Farbe, Stärke und der Stil der Kontur angegeben. Wird als Farbe *none* angegeben, so besitzt das Objekt keine Kontur.

Mit dem dritten Funktionsbereich *Link* besitzt der Benutzer die Möglichkeit, das Objekt als Link zu einer URL oder Datei im Internet zu gestalten. Dies geschieht durch einfaches Eintragen einer URL in das Textfeld.

Abbildung 37: Grafikeditor Funktionsbereich „Text“

Der Grafikeditor besitzt noch einen vierten Funktionsbereich, der in Abbildung 37 dargestellt ist. Der Funktionsbereich *Text* ist nur sichtbar, wenn als Form Text ausgewählt ist. Hier kann die Schriftgröße und –art festgelegt werden. Außerdem steht dem Benutzer ein Textfeld für die Eingabe des Textes zur Verfügung.

Nachdem eine Form ausgewählt ist und die Objekteigenschaften angepasst sind, kann die Form erzeugt werden. Nachfolgend wird kurz erklärt, wie die einzelnen Formen erzeugt werden. Dabei werden grundsätzlich die Positionen und Größen durch das Klicken auf die Grafik festgelegt.

Bei einem *Rechteck* repräsentiert die Position, an der die linke Maustaste gedrückt wird, die obere-linke Ecke. Die Position des Loslassens der Maustaste wird als untere-rechte Ecke des Rechtecks interpretiert.

Bei einem *Kreis* stellt das Drücken der linken Maustaste den Mittelpunkt des Kreises dar. Beim Wiederloslassen wird der Abstand zum zuvor gesetzten Mittelpunkt ermittelt und als Radius des Kreises verwendet.

Eine *Ellipse* hingegen wird zwischen den Positionen des Drückens und Loslassens der linken Maustaste generiert.

Das Linienwerkzeug erzeugt eine *Linie* zwischen der Position des Drückens und der Position des Loslassens der linken Maustaste. Hierbei ist zu beachten, dass die Farbe der Linie über ihre Kontur bestimmt wird.

Um einen *Text* hinzuzufügen wird die Position des Textes durch das Drücken der linken Maustaste bestimmt. Diese Position stellt die obere-linke Ecke des Textes dar.

Eine *Polyline* stellt Linien dar, wo das Ende der einen Linie den Anfang der anderen darstellt. Um eine Polyline zu erzeugen werden nur die Positionen des Drückens der Maustaste als Positionsangaben verwendet. Durch Bewegen der Maus und Drücken der Maustaste an unterschiedlichen Positionen wird die Polyline sichtbar. Um das Objekt fertigzustellen ist ein Klick auf den Button *Pfad fertig* nötig, der im Funktionsbereich *Button* erscheint. Dieser ist in Abbildung 38 zu sehen.

Abbildung 38: Grafikeditor, Polyline, „Pfad-Fertig"-Button

Ein *Polygon* ist ein Vieleck wo ebenfalls, wie bei der Polyline, jeder Klick als Eckpunkt aufgefasst wird. Um das Polygon-Objekt abzuschließen ist ein Klick auf den Button *Form fertig* nötig, der ebenfalls im Funktionsbereich *Button* erscheint. Hierdurch wird die Form des Polygons geschlossen. Das bedeutet, es wird die erste Eckposition mit der letzten Eckposition verbunden und anschließend das Polygon erzeugt.

6.2.4.5 Gesamte Linkbase löschen

Der Benutzer besitzt die Möglichkeit alle seine Einträge aus der Linkbase zu entfernen. Diese Funktion steht unter dem Menupunkt *Linkbase > Gesamte Linkbase löschen* bereit. Es werden alle Einträge gelöscht. Sie können nicht wieder herstellt werden. Zu beachten ist ebenfalls, dass diese Funktionalität auch Auswirkungen auf Einträge hat, die als öffentlich eingestellt wurden. Diese Einträge werden ebenfalls gelöscht und stehen somit auch allen anderen Benutzern ebenfalls nicht mehr zur Verfügung.

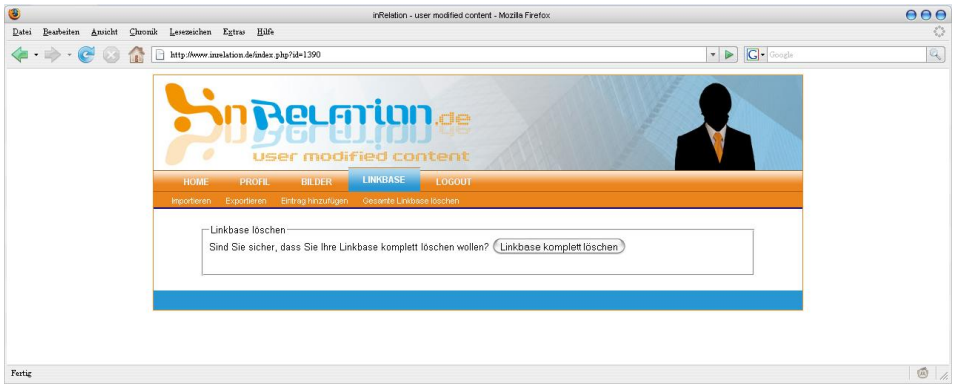

Abbildung 39: Gesamte Linkbase löschen der inRelation-Webplattform

6.2.5 Logout

Mit dem *Logout* kann sich der Benutzer von der Plattform abmelden. Dieses sollte aus Sicherheitsgründen immer genutzt werden. Wird das Abmelden vergessen, so wird der Benutzer automatisch nach einer Leerlaufzeit von (standardmäßig) 15 Minuten abgemeldet.

6.3 Beispiel

In diesem Abschnitt soll die Funktionsweise von inRelation anhand eines Beispiels veranschaulicht werden. Hierzu wurde eine fiktive Seite erzeugt und ins Internet gestellt. Die Seite ist über die Domain http://beispiel.inrelation.de zugänglich. Des Weiteren wird eine Unterseite erzeugt die über die URL http://beispiel.inrelation.de/unterseite/ erreichbar ist. Die zweite Seite soll die Vererbung innerhalb von inRelation demonstrieren.

6.3.1 Original SVG

Bevor eine Manipulation angelegt und die Seiten anschließend erneut mit eingeschaltetem Greasemonkey-Skript betrachtet werden, sollen zuvor die Originalseiten Betrachtung finden.

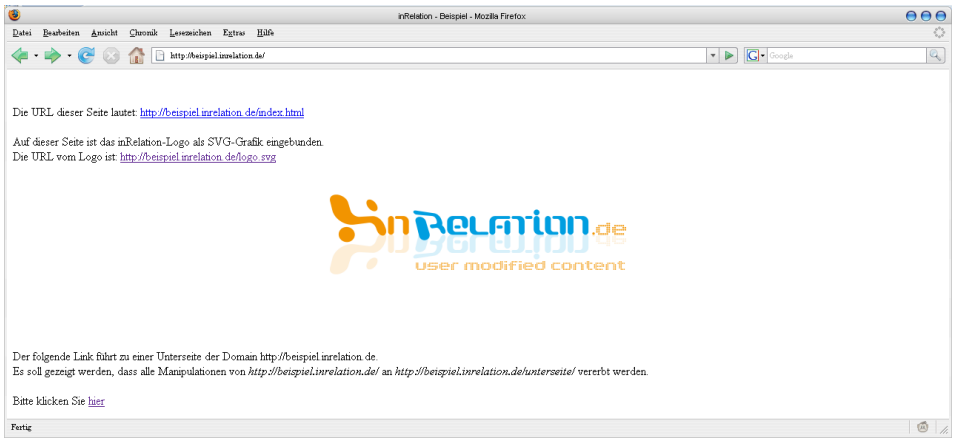

Abbildung 40: Original SVG in der Beispiel-Seite

Abbildung 40 und Abbildung 41 zeigen die originale Seite und die originale Unterseite bei ausgeschaltetem Greasemonkey-Skript. An dem grauen Greasemonkey-Logo im unteren rechten Bereich des Browserfensters ist zu erkennen, dass Greasemonkey deaktiviert ist.

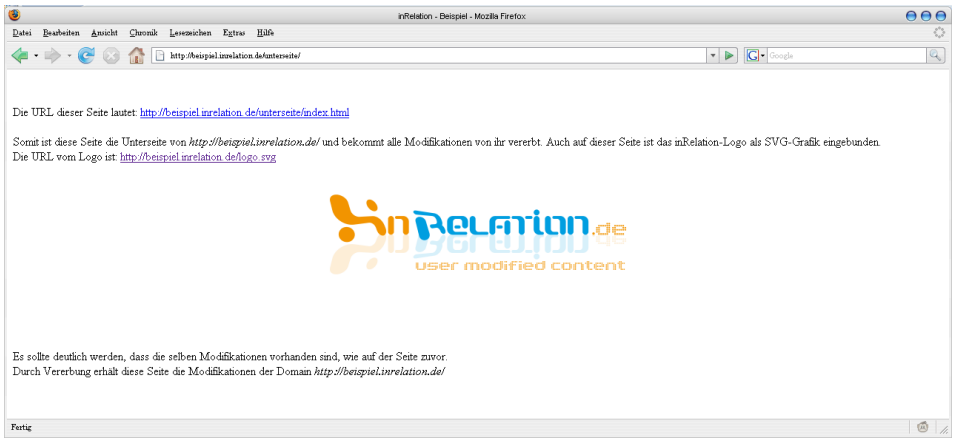

Abbildung 41: Original SVG in der Beispiel-Unter-Seite

6.3.2 Eintrag hinzufügen

Nun wird ein Eintrag zur Linkbase hinzugefügt. Dieser Eintrag soll den Schriftzug *user modified content* entfernen und für die Domain http://www.beispiel.inrelation.de gültig sein.

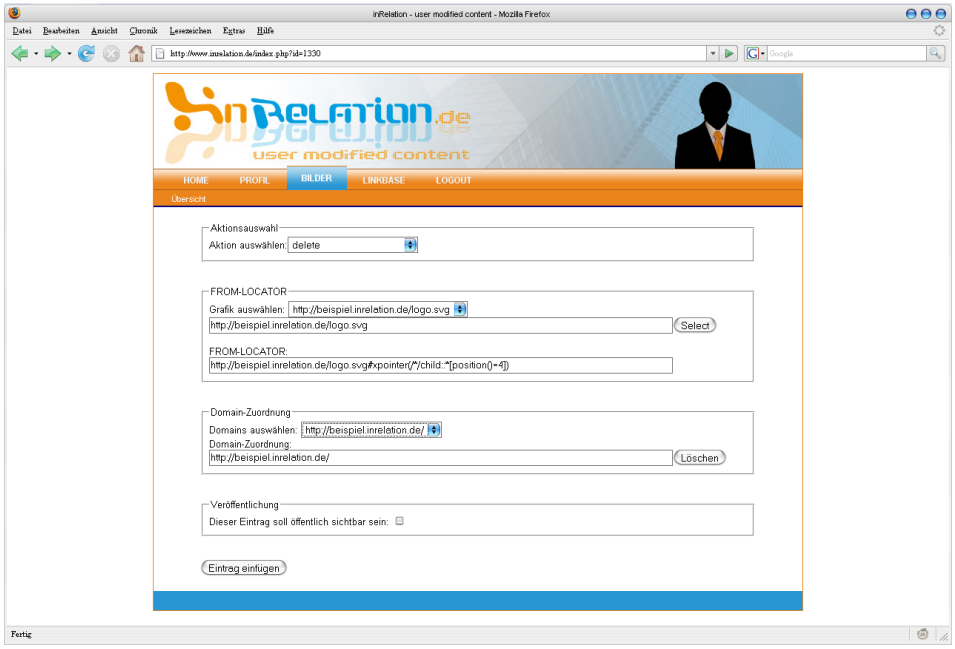

Abbildung 42: Eintrag zur Linkbase hinzufügen für die Beispiel-Seite

Abbildung 42 zeigt das ausgefüllte Formular, um den oben genannten Schriftzug aus der SVG-Grafik für die genannte Domain zu entfernen.

6.3.3 Modifiziertes SVG

Nachdem der Eintrag für die Manipulation der SVG-Grafik auf der Beispielseite zur Linkbase hinzugefügt wurde, kann nun das Greasemonkey-Skript aktiviert werden. Die Aktivierung geschieht durch Klicken auf das Greasemonkey-Logo unten rechts im Browserfenster. Wird die Domain http://www.beispiel.inrelation.de erneut aufgerufen, ist die Ansicht wie in Abbildung 43. Der inRelation-Button ist nun sichtbar und das originale SVG-Logo wurde durch das manipulierte Logo ohne Schriftzug ersetzt.

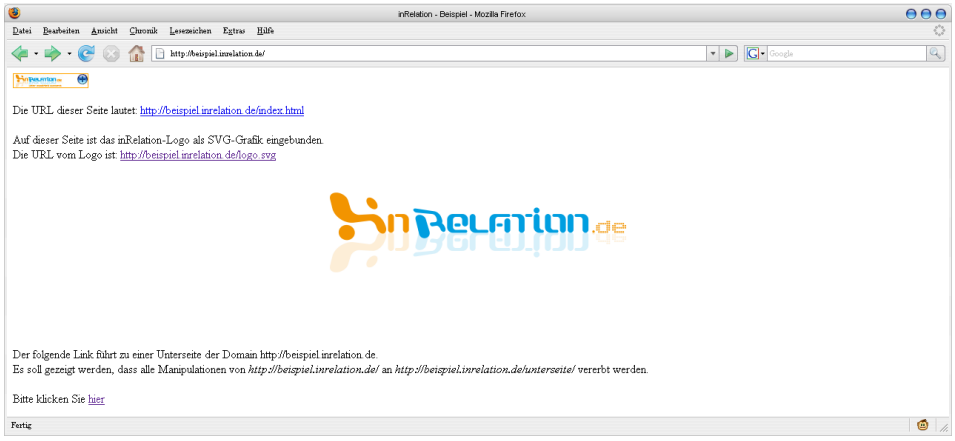

Abbildung 43: Modifiziertes SVG in der Beispiel-Seite

6.3.4 Vererbung

Die Betrachtung der Beispiel-Unter-Seite wurde bewusst in einen neuen Abschnitt ausgelagert, um die Vererbung in inRelation zu verdeutlichen. Es wurde für das SVG-Logo in 6.3.2 nur ein Eintrag hinzugefügt. Dieser Eintrag wurde mit der Domain http://www.beispiel.inrelation.de verknüpft. Abbildung 44 zeigt die Beispiel-Unter-Seite bei eingeschaltetem Greasemonkey-

Skript. Beim Betrachten des Logos fällt auf, dass auf dieser Seite ebenfalls das modifizierte SVG-Logo ohne Schriftzug angezeigt wird.

Abbildung 44: Modifiziertes SVG in der Beispiel-Unter-Seite

Vererbung ist das Stichwort. Alle Modifikationen die einer Domain zugeordnet sind, sind ebenfalls allen Ihren Unterseiten zugeordnet.

7 Schlussbetrachtung

Abschließend soll betrachtet werden, wie gut der realisierte Prototyp das ursprüngliche Problem löst. Das in 6.3 dargestellt Beispiel zeigt, dass die realisierte Systemlandschaft für einfache Internetseiten mit einer überschaubaren Anzahl von eingebundenen Grafiken eine Lösung, bzw. einen prototypischen Lösungsansatz des ursprünglichen Problems der personalisierten Grafikannotation darstellt.

Es wurde jedoch Abstand von einer ausschließlichen Implementierung mittels eines Firefox-Plugin genommen. Durch die Verteilung der Anwendung auf Client und Server konnte der Nutzen mittels Nutzung von Netzeffekten durch den Multiuser-Betrieb gesteigert werden.

Bei der Implementierung stand die Effizienz der Programmierung nicht an oberster Stelle, da lediglich eine prototypische Umsetzung erfolgt ist. Dies hat zur Folge, dass die Anwendung bei komplexeren Internetseiten, in denen viele Grafiken eingebunden sind, schnell an ihre Grenzen gerät. Werden mit eingeschaltetem Plugin Internetseiten wie www.spiegel.de betrachtet, so kommt es zu einem Javascript Timeout. Hier werden vom Plugin zu viele Bilder in der Internetseite erkannt. Für jedes Bild wird eine AJAX-Anfrage an die inRelation-Webplattform ausgelöst, die zur Überprüfung auf vorhandene Modifikationen dient. Dadurch sind zu viele Verbindungen gleichzeitig zur Webplattform geöffnet, sodass der Browser überfordert wird. Dadurch wird die erlaubte Laufzeit für ein Skript überschritten. Das Skript wird beendet, bevor es vollständig und korrekt terminieren kann.

Um diese Anwendung generell, auch für komplexere Internetseiten, einsatzfähig zu gestalten, ist weiterer Entwicklungsaufwand erforderlich. Es müssen einige Implementierungen optimiert werden und der Kommunikationsaufwand zwischen Greasemonkey-Skript und Webplattform optimiert werden. Auch der Ersatz des Greasemonkey-Skripts durch ein eigenständiges Plugin sei angeraten. Jedoch kann zu diesem Zeitpunkt und nach den Erfahrungen, die aus der Realisierung dieser prototypischen Umsetzung entstanden sind, nicht entschieden werden, ob es mit den gewählten technischen Technologien überhaupt möglich ist, die Anwendung soweit zu optimieren, dass sie auch für komplexere Internetseiten einsetzbar ist. Dies liegt unter anderem daran, dass schon bei dieser Realisierung teilweise die Grenzen der Technologien erreicht wurden. Ebenfalls wurden bei den Technologien einige Anomalien entdeckt, die auf eventuelle Fehler im Handwerkszeug schließen lassen.

Literaturverzeichnis

Argerich, L., Egervari, K., Anton, M., Lea, C., Killian, C., Hubbard, C., et al. (2002). *Professional PHP4 XML.* Birmingham: Wrox Press Ltd.

Baur, S. K. (13. Januar 2008). *Pflichtenheft: Brettspiel Beispiel*. Abgerufen am 09. Juni 2008 von http://www.stefan-baur.de/downloads/Pflichtenheft.pdf

Bergmann, O., & Bormann, C. (2005). *AJAX - Frische Ansätze für das Web-Design.* Berlin: SPC TEIA Lehrbuch Verlag.

Freie Universität Berlin. (14. Januar 2007). *AjaxParts in Java Web Parts*. Abgerufen am 09. Juli 2008 von https://www.inf.fu-berlin.de/w/VNBI/AjaxParts

Pilgrim, M. (2006). *Greasemonkey Hacks.* United States of America: O'Reilly.

Schmid, E., & Cartus, C. (2001). *php4.* München: Markt+Technik Verlag.

SVG Working Group. (29. Oktober 2004). *Scalable Vector Graphics (SVG) - About SVG*. Abgerufen am 05. Juli 2008 von http://www.w3.org/Graphics/SVG/About