Bern University
of Applied Sciences

Projekt 2

Elephant Search – Semantische Suche für Kinder

Berner Fachhochschule BFH, Technik und Informatik
Prof. Dr. Jürgen Eckerle

Mira Günzburger (gunzm1), Sven Osterwalder (ostes2)

Bern, 11.01.2014

Bern University of Applied Sciences

Inhalt

1 Einleitung

Bei diesem Dokument handelt es sich um das Abschlussdokument des Moduls 7302 „Projekt 2“ der Berner Fachhochschule von Sven Osterwalder und Mira Günzburger. Zusätzlich existiert ein Anforderungsdokument, welches die Aufgaben des Projektes genau beschreibt. Der Schwerpunkt dieser Dokumentation liegt bei der Beschreibung der verschiedenen Komponenten sowie deren theoretischen Grundlagen. Die Dokumentation bildet die Grundlage für die Bachelor Thesis.

Der Einfachheit halber wird in diesem Dokument - sofern nicht anders erwähnt - immer die männliche Form von Substantiven verwendet. Dabei ist die weibliche Form jedoch zu gleichen Teilen gemeint.

1.1 Modulbeschreibung

Dieses Modul hat zum Ziel, die Studierenden anhand von konkreten Projektaufträgen mit der Praxis der Projektführung vertraut zu machen. Zu diesem Zweck realisieren sie in kleinen Gruppen weitgehend selbständig je ein vollständiges Informatikprojekt, welches üblicherweise aus dem Gebiet des gewählten Schwerpunktes stammt. Die Studierenden übernehmen (Teil-)Projektleitungsaufgaben.

Bei diesem Projekt ist zu beachten, dass das Ziel nicht die Erstellung eines fertigen Produktes ist, sondern dass es sich um eine Recherchen-Arbeit handelt, welche schlussendlich als Grobkonzept für die Bachelor Thesis dienen soll.

2 Pflichtenheft

2.1 Projektbeschreibung

Semantische Suchmaschinen sind Werkzeuge, die in der Lage sind, auf Fragen mit Hilfe einer Datenbank oder des Internets Antworten zu generieren. Solche Werkzeuge können insbesondere dann eine sehr wertvolle Unterstützung für den menschlichen Experten sein, wenn unter extremer Zeitnot komplexe Entscheidungen getroffen werden müssen, wie beispielsweise in der medizinischen Diagnostik. Die Firma IBM hat vor nicht allzu langer Zeit für eine Überraschung gesorgt, als sie die Leistungsfähigkeit von „Watson“ im Quiz Jeopardy demonstriert hat. In diesem Quiz, wo schwierige, oft zweideutig formulierte Fragen aus beliebigen Bereichen unter Zeitdruck beantwortet werden müssen, konnte sich Watson überlegen gegenüber zwei bisher sehr erfolgreichen menschlichen Champions durchsetzen.

In dieser Projektarbeit soll ein Werkzeug für die semantische Suche in einer Wissensdatenbank implementiert werden. Bei der Gestaltung der Benutzerschnittstelle ist zu berücksichtigen, dass die vorgesehene Zielgruppe dieses Werkzeuges Kinder sind. Die Realisierung soll mittels Java unter Verwendung von RDF und Apache Stanbol erfolgen.

2.2 Allgemeine Anforderungen

Die allgemeinen Anforderungen werden im Detail im Dokument [1] beschrieben. Nachfolgend eine kurze Übersicht der Anforderungen:

- Bereitstellung der Arbeitsumgebung
- Recherchen zu allen Komponenten
- Festlegen der zu verwendenden Technologien
- Gegebenenfalls Erstellen von Prototypen
- Verifikation oder Falsifikation der geplanten Umsetzung aufgrund der gewonnenen Einsichten
    - Gegebenenfalls Anpassung des Konzepts

[1] Requirements_V3

3 Terminplan

Realisierungszeitplan in Arbeitswochen (Vom 16.09.2013 Bis 17.01.2014):

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1															
2															
3															
		4													
				5											
	6														
	7														
									8						
										9					
												10			
														11	
															12

Illustration 1: Terminplan

1 Bereitstellung Arbeitsumgebung

2 Arbeitsprotokoll

3 Anforderungen

4 Einarbeitung in Ruby On Rails

5 Einarbeitung in Three.js

6 Einarbeitung in Stanbol

7 Einarbeitung Semantische Suche und Logig

8 Erstellen der Prototypen

9 Dokumentation

10 Präsentationsvorbereitung

11 Präsentation

12 Abgabe

4 Vorarbeiten

Da das Thema sowie die Konstellation dieser Projektarbeit bereits relativ früh (im Frühjahr 2013) feststand und sich Herr Prof. Dr. Eckerle dazu bereit erklärte die Arbeit zu betreuen, fanden im Vorfeld bereits einige Vorarbeiten betreffend den eingesetzten Komponenten bzw. Technologien statt.

4.1 Semantische Suche

Während den Recherchen wurden so genannte „Question Answering"-Systeme gefunden, welche ein eigenes Forschungsgebiet zu sein scheinen. Diese setzen im Prinzip bereits das Ziel der Bachelor Thesis um, jedoch zeigten sich die Ergebnisse (Antworten) dieser Systeme als nicht befriedigend. Zudem sind die Systeme nur auf Englisch verfügbar, nicht kindergerecht oder noch in Entwicklung. Es sei hier auf[2], [3] und [4] verwiesen.

Aufgrund der durchgeführten Recherchen wurde eine eigene Implementation eines QA Systems in Betracht gezogen. Diese könnte gem. (Hirschman, et al., 2001) wie folgt aufgebaut werden:

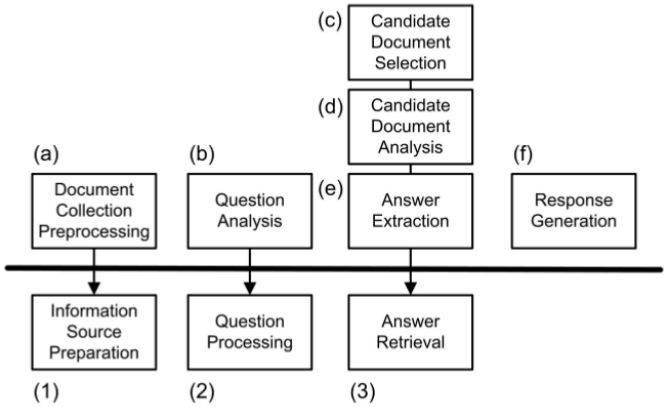

Illustration 2: Möglicher Aufbau eines QAS

Aufgrund der hohen Komplexität solcher Systeme (das START QAS wird beispielsweise seit Dezember 1993 am MIT entwickelt) wurde diese Idee jedoch verworfen. Nach weiterer Analyse und aufgrund einer Präsentation einer Habilitations-Arbeit, welche Sven Osterwalder besuchte, wurde schliesslich Apache Stanbol näher betrachtet und als geeignete Komponente ausgewählt. Dies, da das System pro Komponente mehrere Varianten bietet (welche sich beliebig erweitern lassen) und da es sehr flexibel konfigurierbar ist.

4.2 User Interface

Zur Umsetzung von Webapplikationen gibt es viele unterschiedliche Möglichkeiten und Technologien. Ein gängiges Pattern zur Umsetzung von Webapplikationen ist das MVC Pattern, zudem ist es ein integraler Bestandteil des Studiums an der Berner Fachhochschule.

[2] http://start.csail.mit.edu/index.php
[3] http://en.wikipedia.org/wiki/Question_answering
[4] http://www.qanus.com/

Unter diesen Gesichtspunkten haben wir uns entschieden, dieses für das User Interface zu verwenden. Eine ausgereifte und doch nicht alltägliche Umsetzung dieses Patterns bietet das Framework Ruby on Rails.

Die JavaScript Bibliothek Three.js lässt sich gut integrieren und bietet dem Programmierer die Möglichkeit mit sinnvollem Aufwand eine ansprechende und, in unserem Fall kinderfreundliche, dreidimensionale Oberfläche zu erschaffen.

Aus den oben genannten Gründen sowie der Motivation etwas Neues ausprobieren zu wollen, haben wir uns für diese zwei Technologien entschieden.

5 Komponenten

Dieses Kapitel beschreibt die einzelnen Komponenten, welche schlussendlich bei der eigentlichen Arbeit zum Einsatz kommen sollen. Die Architektur der Applikation ist wie folgt vorgesehen:

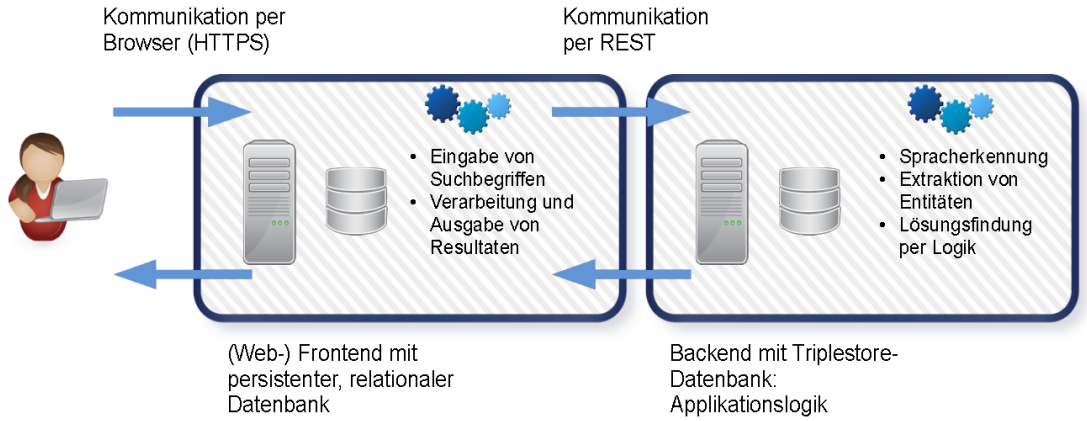

Illustration 3: Komponenten

Grundsätzlich soll dem Benutzer eine möglichst schlicht gehaltene Schnittstelle zur Eingabe von Fragen geboten werden, wobei Benutzer in erster Linie Kinder sein sollen. Die Frage wird vom Frontend (Web-Applikation basierend auf Ruby On Rails und Three.js) dann an das Backend (Apache Stanbol) weitergeleitet.

Im Backend wird die eingegebene Frage analysiert und Objekte extrahiert. Es wird dann versucht per Prädikatenlogik ein entsprechendes Resultat abzuleiten. Sofern das Resultat in der Datenbank existiert, wird der Inhalt an das Frontend weitergeben, wo dieser dargestellt wird.

5.1 Semantische Suche

Ein Kernstück der Applikation bildet die semantische Suche. Sie soll es Benutzern ermöglichen Fragen in natürliche Sprache einzugeben. Doch was bedeutet Semantik eigentlich?

Der Begriff Semantik kommt ursprünglich vom altgriechischen Wort *sēmaínein*, was übersetzt “bezeichnen” oder “zum Zeichen gehörig” bedeutet. Semantik nennt man die Theorie oder Wissenschaft der Bedeutung der Zeichen. (WDE1, 2013)

Gem. (Norvig, et al., 2010) basiert die semantische Suche bzw. semantische Systeme schlussendlich auf der Repräsentation von Wissen. In den Anfängen der künstlichen Intelligenz sprach man von der Repräsentation von Problemen bzw. der Repräsentation von Wissen. Daraus entwickelten sich Expertensysteme, welche dem Wissen von menschlichen Experten entsprachen oder dieses sogar übertrafen, allerdings nur in einem ganz bestimmten Feld unter bestimmten Voraussetzungen.

Mit der Zeit entwickelte die Forschung eine standardisierte Darstellung des Wissens in Form von Formalismen und Ontologien um die Erstellung neuer Expertensystem zu optimieren.

Ontologien

Seit Menschengedenken klassifiziert und kategorisiert der Mensch. Bereits Aristoteles legte grossen Wert auf die Schemata der Klassifizierung und Kategorisierung. In Organon, der Sammlung der logischen Schriften Aristoteles, führte er im Kapitel *Kategorien* ein System ein, welches wir heute als Ontologie bezeichnen würden. (Norvig, et al., 2010)

5.1.1 Bei Ontologien in der Informatik handelt es sich „ ... meist um sprachlich gefasste und formal geordnete Darstellungen einer Menge von Begrifflichkeiten und der zwischen ihnen bestehenden Beziehungen in einem bestimmten Gegenstandsbereich. Sie werden dazu genutzt, „Wissen“ in digitalisierter und formaler Form zwischen Anwendungsprogrammen und Diensten auszutauschen. Wissen umfasst dabei sowohl Allgemeinwissen als auch Wissen über sehr spezielle Themengebiete und Vorgänge.“ (WDE3, 2013) (Gruber, 1993)

Im semantischen Web werden Ontologien dazu genutzt Daten zu integrieren. Dies kann z.B. hilfreich sein wenn Doppeldeutigkeiten bestehen oder wenn zusätzliches Wissen benötigt wird um neue Beziehungen zu entdecken. Sie können aber auch dazu genutzt werden um Wissen zu organisieren (z.B. bei grossen Sammlungen/Kollektionen).

5.1.1.1 Technologien / Ontologie-Sprachen

Um Ontologien zu beschreiben, existieren W3C-Standards bzw. –Empfehlungen für die entsprechenden Formalismen. Die gängigsten Formalismen sind nachfolgend aufgeführt.

RDF

RDF steht für Resource Description Framework. Wie es der Name bereits nahe legt, liegt der Zweck von RDF darin, Aussagen über (Web-) Ressourcen zu machen. RDF basiert auf der XML-Syntax und nutzt die folgenden, grundlegen Konzepte:

- **Graph-basiertes Datenmodell**
  Die Grundlage des Graphen bildet die Tripel-Datenstruktur. Diese besteht aus einem Subjekt, einem Prädikat und einem Objekt. Eine Menge von Tripel bildet einen RDF-Graphen. Subjekt und Objekt sind hierbei Knoten, während das Prädikat, welches immer gerichtet gegen das Objekt zeigt, die Beziehung darstellt.
- **URI-basiertes Vokabular**
  Hierbei handelt es sich um eine sogenannte URI-Referenz. Diese ist eine Unicode-Zeichenkette, welche keine Kontrollzeichen enthält und eine valide URI Zeichenfolge produzieren muss.
- **Datentypen**
  Werden in RDF benutzt um Werte, wie z.B. ganze Zahlen, Fliesskommazahlen und Daten (sing. Datum), abzubilden. Ein Datentyp besteht dabei aus einem Wertebereich (z.B. *{W, F}*), einem lexikalischen Bereich (z.B. *{„0“, „1“, „wahr“, „falsch“}*) und einer Verknüpfung der beiden Bereiche (z.B. *{<“wahr“, W>, <“1“, W>, <“falsch“, F>, <“0“, F>}*).
- **Buchstabensymbole (Literale)**
  Werden benutzt um Werte, wie z.B. Nummern und Daten (sing. Datum), anhand deren lexikalischer Darstellung zu identifizieren. Dabei kann ein Buchstabensymbol das Objekt einer RDF-Aussage darstellen, nicht aber das Subjekt oder Prädikat. Dem Beispiel der Beschreibung von Datentypen folgend, könnten Buchstabensymbole z.B. *<xsd:boolean, "wahr">* oder *<xsd:boolean, "0">* sein.

- **Abbilden einfacher Tatsachen**
  Beschreibt eine Beziehung zwischen zwei Dingen. Dabei wird, wie bereits oben erwähnt, die Triple-Datenstruktur verwendet.
- **Folgebeziehungen**
  Ermöglichen eine Art Bedeutung von Aussagen mittels Inferenz. Man spricht davon, dass eine Aussage A eine andere Aussage B „enthält“, wenn Folgendes gilt:
  Alle möglichen Kombinationen der „(RDF-)Welt“ führen dazu, dass A wahr ist, so ist dadurch auch B wahr.

Nachfolgend ein Beispiel der RDF-Syntax:

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:dc="http://purl.org/dc/elements/1.1/" >

        <rdf:Description rdf:about="http://www.idsoftware.org/Company">
                <dc:isFounder >John Carmack</dc: isFounder >
        </rdf:Description>
</rdf:RDF>
```

Hierbei wäre nun *Company* das Subjekt, *John Carmack* das Objekt und *dc:isFounder* die Relation. Dies

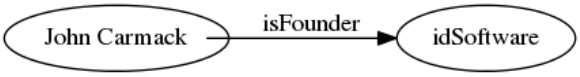

Illustration 4: Beispiel eines RDF-Graphen

würde als Graph wie folgt aussehen:

SKOS – Simple Knowledge Organization System

Bei SKOS handelt es sich um ein RDF-Vokabular für semi-formale Wissensanordnungssysteme (KOS, Knowledge Organization Systems). Dies können z.B. Thesauri, Taxonomien oder Klassifikationsschemata sein. SKOS basiert dabei auf dem Resource Description Framework (RDF). SKOS kann als eine Art Brücke zwischen den rigorosen logischen Formalismen von Ontologie-Sprachen (wie z.B. OWL) und den web-basierten Kollaborations-Applikationen verstanden werden.

SKOS basiert auf den folgenden Elementen:

- **Konzepte**
  Das fundamentale Element von SKOS. Dies sind „Denkeinheiten“, also Ideen, Bedeutungen oder Objekte und Ereignisse. Z.B*.: <http://www.foo.org/tiere > rdf:type skos:Concept.*
- **Beschriftungen**
  Stellen die Referenz von Konzepten in der natürlichen Sprache dar. Z.B.:

  ex:tiere rdf:type skos:Concept;
  skos:prefLabel "animals"@en;
  skos:prefLabel "animaux"@fr.

- **Semantische Beziehungen**
  Verbindet Konzepte miteinander, Operationen sind hierbei *skos:broader*, *skos:narrower* und *skos:related*.
- **Beschreibende Notizen**
  Dienen der zusätzlichen Dokumentation z.B. um etwas genauer zu präzisieren. Z.B.

  ex:microwaveFrequencies skos:scopeNote
  "Used for frequencies between 1GHz to 300Ghz"@en.

Bei OWL bzw. OWL2 handelt es sich um eine Art Metasprache, welche sich auf einem höheren Abstraktionsniveau als RDF befindet. Zudem dient RDF, wie bereits erwähnt, vor allem der Darstellung von Daten. OWL bietet daher eine Möglichkeit zur formalen Beschreibung von Bedeutungen.

OWL ist in folgende Unterarten unterteilt:

- **OWL Lite**
  Bietet primär eine Hierarchie zur Klassifikation sowie einfache Bedingungen, so wird zwar z.B. Kardinalität geboten, jedoch nur mit den Werten 0 und 1.
- **OWL DL**
  Bietet das Maximum an Ausdrucksmöglichkeiten unter Einhaltung der Vollständigkeit (alle Folgebeziehungen werden berücksichtigt und miteinbezogen) und der Entscheidbarkeit (alle Berechnungen enden nach endlicher Zeit).
- **OWL Full**
  Bietet das Maximum an Ausdrucksmöglichkeiten und die syntaktische Freiheit von RDF aber ohne Garantien betreffend Vollständigkeit und Entscheidbarkeit.

Auf eine detailliertere Beschreibung von OWL bzw. OWL2 wird verzichtet, da dies den Rahmen dieser Projektarbeit sprengen würde. Es sei jedoch auf [5] sowie [6] verwiesen.

Die nachfolgende Grafik bietet eine Übersicht der Struktur von OWL2:

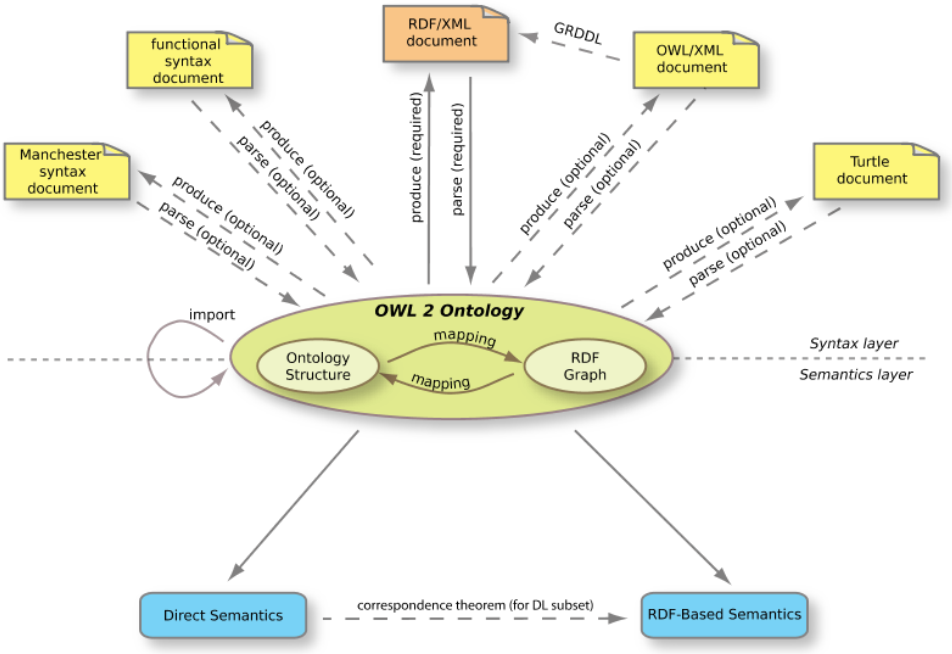

Illustration 5: Struktur von OWL2

[5] http://www.w3.org/TR/2004/REC-owl-guide-20040210/
[6] http://www.w3.org/TR/owl2-overview

Prädikatenlogik

Der Mensch, so scheint es, hat ein Wissen über gewisse Fakten und dieses Wissen hilft ihm Dinge zu tun. Dies zeigt, dass der Mensch nicht rein nach Reflexen handelt sondern aufgrund von Entscheidungen bzw. Schlussfolgerungen, welche auf der internen Darstellung von Wissen basieren. (Norvig, et al., 2010)

5.1.2 Um solche Fakten festzuhalten und somit Systeme zu repräsentieren, bietet sich als grundlegende Sprache beispielsweise Aussagenlogik an, welche zudem die Möglichkeit der Inferenz (Schlussfolgerung) bietet. Um allerdings komplexe Systeme zu repräsentieren, wie etwa die menschliche Sprache, reicht dies nicht aus.

Um Wissen komplexer Systeme entsprechend zu repräsentieren bietet sich die so genannte Prädikatenlogik an. Diese basiert, wie es der Name bereits sagt, auf Prädikaten, also Satzaussagen. (DUD1, 2013)

Prädikatenlogik basiert auf Objekten sowie deren Relationen und hat den folgenden Aufbau:

- **Prädikatsymbole**
  Stehen für Relationen, z.B. $Brother(\quad)$
- **Terme**
  Im wesentlichen Objekte, z.B. $Richard$*oder* $John$
- **Atomare Sätze**
  Verbinden Prädikatsymbole und Terme, z.B. $Brother(Richard, John)$
- **Komplexe Sätze**
  Mit Logik kombinierte atomare Sätze, z.B. $King(Richard) \vee King(John)$*oder* $\neg King(Richard) \Rightarrow King(John)$ *(Entweder ist Richard oder John König; Wenn Richard nicht König ist, so ist John König).*
- **Quantoren**
  Ermöglichen die Anwendung von Regeln auf Objektklassen anstatt einzelne Objekte zu benennen, z.B. $\forall x King(x) \Rightarrow Person(x)$*(alle Könige sind Menschen)*

5.2 Stanbol

Bei Stanbol handelt es sich um es sich um ein System bestehend aus diversen Software-Komponenten für die semantische Verwaltung von Inhalten, welche mittels REST untereinander kommunizieren. (STAN1, 2013)

Es wurde ursprünglich 2008 von der Salzburg Research Forschungsgesellschaft mbH ins Leben gerufen. Das Projekt startete dann 2009 mit Unterstützung der europäischen Union mit der Entwicklung. 2012 wurde es von der Apache Software Foundation übernommen. (STAN1, 2013)

Es besteht im Wesentlichen aus folgenden Komponenten:

- Anreicherung des Inhaltes (Content-Enhancement)
- Schlussfolgerungen (Reasoning)
- Wissensmodellen (Knowledge models)
- Persistenz (Persistence)

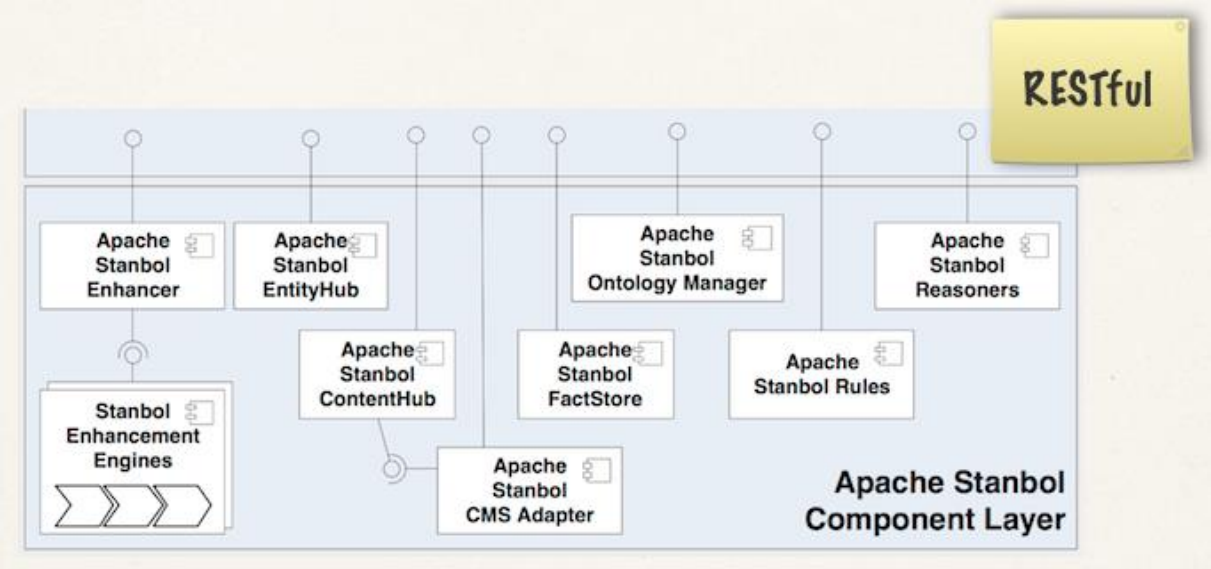

Illustration 6: Komponenten von Apache Stanbol

Enhancement

Bei der Anreicherung des Inhaltes geht es im Wesentlichen um die sogenannte Feature Extraction, also das Extrahieren von Objekten aus Text. Der zu verarbeitende Inhalt wird mittels einer Kette (der sog. Enhancement Chain)von Softwarekomponenten verarbeitet und die extrahierten Objekte in einem internen Objekt (ContentItem) gespeichert.

5.2.1

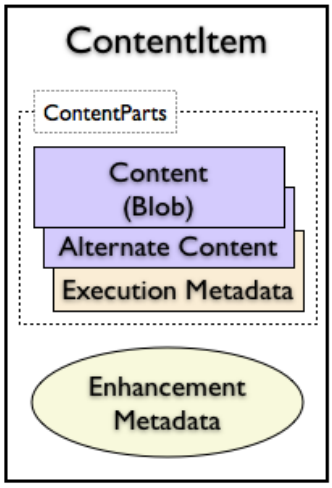

Illustration 7: ContenItem

Bei den Komponenten zur Anreicherung handelt es sich um folgende Kategorien:

- Vorverarbeitung (Preprocessing)
- Verarbeitung der natürlichen Sprache (Natural Language Processing)
- Vernetzung/Vorschläge (Linking/Suggestions)
- Nachbearbeitung / Andere (Postprocessing / Other)

5.2.2 Die gesamte, detaillierte Liste der Komponenten ist unter (STAN2, 2013) ersichtlich.

Rules

Um schliesslich aus den extrahierten Objekten Informationen gewinnen zu können, werden auf Prädikatenlogik basierende Regeln angewendet (siehe 10).

Regeln können innerhalb von Apache Stanbol als Rezepte gespeichert werden, diese sind nichts anderes als eine Zusammenfassung von Regeln, welche eine ähnliche Objektkategorie betreffen.

Zur Definition von Regeln werden aktuell Jena-Regeln oder SPARQL-CONSTRUCT unterstützt.

Nachfolgend, als Beispiel, die Regel $hasUncle(x, y)$, welche prüft ob das Objekt $y$der Onkel des Objektes $x$ist:

```
uncleRule[
        has(<http://www.foo.org/myont.owl#hasParent>, ?x, ?z)
        has(<http://www.foo.org/myont.owl#hasSibling>, ?z, ?y)
                ->
                has(<http://www.foo.org/myont.owl#hasUncle>, ?x, ?y)
]
```

Dies bedeutet, dass wenn gilt:

$$hasParent(x, z) \wedge hasSibling(z, y)$$

dies zu:

$$hasUncle(x, y)$$

führt. Also wenn z ein Elternteil von x ist und y der Bruder von z ist, dann gilt, dass y der Onkel von x ist.

Reasoners

5.2.3 Als Reasoner wird in Apache Stanbol die Komponente bezeichnet, welche eine automatische Schlussfolgerung basierend auf Ontologien und Regeln erlaubt.

Aktuell werden die Sprachen RDF, OWL, OWLMini und HermiT[7] unterstützt. Es handelt sich dabei um eine Art Metasprachen, welche bereits Objekte (und somit Ontologien) und Regeln beinhaltet (siehe auch 5.1.1.1).

5.2.4

Ontologies

Apache Stanbol bietet die Möglichkeit Ontologien zu speichern, zu bearbeiten und einzelne Ontologien miteinander zu vernetzen. Zur Eingabe wird aktuell nur die Metasprache OWL2[8]

5.2.5 unterstützt.

Entity Hub

Bei der Komponente Entity Hub handelt es sich um eine Komponente, welche Informationen zu Entitäten bzw. Objekten einer spezifischen Wissens-Domäne zur Verfügung stellt.

5.2.6 Dabei bietet sich die Möglichkeit Entitäten unter verschiedenen wissensspezifischen Domänen (so genannten Sites) abzulegen.

Content Hub

Die Komponente Content Hub stellt einen Speicher für textbasierte Dokumente zur Verfügung.

[7] http://hermit-reasoner.com/
[8] http://www.w3.org/TR/owl-overview/

Factstore

Um Prädikate, also schlussendlich Relationen, zu speichern, bietet sich die Komponente Factstore an. Diese speichert die Beziehung zwischen zwei oder mehr Entitäten anhand deren URI. Dies wird als Fakt bezeichnet.

Um Fakten zu speichern, wird ein so genanntes Fakten-Schema verwendet. Ein Fakten-Schema, z.B. 5.2.7 für den Fakt $worksFor(\ \ )$, sieht wie folgt aus:

```
{
    "@context": {
        "#types":{
            "person": "http://schema.org/Person",
            "organization": "http://schema.org/Organization"
        }
    }
}
```

Dies heisst, dass der Fakt $worksFor(\ \ )$ nur die Entitäten Personen und Organisationen betrifft. Das Schema würde in diesem Beispiel unter *http://factschema.org/worksFor* gespeichert.

Der eigentliche Fakt, in diesem Beispiel, dass die Person John Carmack für die Firma Id Software arbeitet, wird dann wie folgt abgelegt:

```
{
    "@profile": "http://factschema.org/worksFor",
    "person": {"@iri": "http://www.idsoftware.com/john_carmack" },
    "organization": {"@iri": "http://www.idsoftware.com"}
}
```

5.3 User Interface

Frontend

Das Frontend soll simpel gehalten werden, in erster Linie Kindergerecht. Die Idee ist, dass es einfach und intuitiv zu bedienen ist. Ein weiterer Schwerpunkt ist, dass es für Kinder ansprechbar sein soll. Sie sollen Spass daran haben, die Applikation zu benutzten.

5.3.1 Im Laufe unserer Arbeit am Projekt 2 ist uns bewusst geworden, dass die Fragen nicht so einfach zu formulieren sein werden, wie wir dies angedacht und in den Anforderungen nieder geschrieben haben. Die Fragen können in natürlicher Sprache bzw. Form eingegeben werden, allerdings Beschränkt sich das System auf die Beantwortung simpler Fragen (wie z.B. “Was ist eine Eule?”).

Trotzdem bleiben wir dabei, dass zumindest die Darstellung weiterhin freundlich und ansprechend sein soll.

Die ganze Applikation ist webbasiert. Der Hauptakteur ist ein Vogel oder ein Elefant, der die Benutzer durch ihre Anwendung führt. Das Tier wird den Benutzer auch dabei unterstützen, die Fragen in der richtigen Form zu stellen und die durch die Technik bedingten, Einschränkungen einzuhalten.

Um das Frontend kinderfreundlich zu halten, wird eine dreidimensionale Landschaft dargestellt, welche sich gegebenenfalls den Jahreszeiten anpasst. Das Tier ist freundlich und hilfsbereit und soll die Kinder animieren, ihre Fragen zu stellen.

Die Weboberfläche wird mit HTML5 und CSS mittels Ruby on Rails umgesetzt, die gesamte 3D-Darstellung mit Three.js.

Management Interface

Das Management Interface der Applikation dient zur Verwaltung der Wissensdatenbank. Es soll es einem Administrator ermöglichen Entitäten und Regeln hinzuzufügen und so die Suchmöglichkeiten zu erweitern. Ausserdem sollen nicht korrekt beantwortet Fragen im Management Interface angezeigt werden, damit die Regeln gegebenenfalls angepasst werden können umso die Fragen richtig beantworten zu können.

5.3.2

Das Management Interface und seine Struktur muss den Regeln von Stanbol angepasst werden. Es werden also Entitäten mit Eigenschaften, Relationen und Regeln abgebildet.

Da Stanbol seine Informationen in einem speziellen Format ablegt, ist eine Synchronisation der vom Management Interface verwendeten Datenbank und der Stanbol-spezifischen Speicherart notwendig. Dies wird per REST-Schnittstelle realisiert.

Beim Management Interface handelt es sich um eine Weboberfläche, welche, wie das Frontend, in Ruby on Rails umgesetzt wird. Als Datenspeicher wird die Datenbanksoftware SQLite3[9].

SQLite3 hat den Vorteil, dass es nur aus einer einzigen Datei besteht, welche nur mehrere Kilobyte gross ist. Zudem benötigt es keine serverseitigen Dienste und bietet praktisch alle Funktionen der grösseren Datenbankmanagementsysteme wie z.B. MySQL oder PostgreSQL unter Verwendung der SQL-Syntax. Auch in Sachen Geschwindigkeit scheint Sqlite3 praktisch gleichauf oder sogar schneller zu sein, siehe (SQL1, 2014)und (SQL2, 2014).

5.3.3 Three.js

Three.js ist eine leichtgewichtige browserübergreifende[10] JavaScript-Bibliothek zum Erstellen und Anzeigen von 3D-Computer-Grafiken in einem Webbrowser. Sie ist Open Source und wird auf GitHub zur Verfügung gestellt.

Three.js-Scripts werden in Verbindung mit HTML5 Canvas-Elementen, SVG oder WebGL[11] genutzt.

Das erste Release auf GitHub wurde von Ricardo Cabello im April 2010 vorgenommen. Die Anfänge der Bibliothek können aber bis in die frühen 2000er zurückverfolgt werden.

Um Three.js zu verwenden muss nur die entsprechende JavaScript-Bibliothek hinzugefügt werden. Dank WebGL braucht es keine zusätzlichen Browser-Plug-Ins. Three.js läuft in allen Browsern, welche WebGL unterstützten.

Damit in Three.js überhaupt etwas dargestellt werden kann, braucht es eine Szene, eine Kamera, einen Renderer und ein Objekt oder eine geometrische Form, die der Szene hinzugefügt wird.

[9] http://www.sqlite.org/
[10] http://de.wikipedia.org/wiki/Cross-Browser
[11] http://de.wikipedia.org/wiki/WebGL

5.3.3.1 Wichtigste Eigenschaften von Three.js

- **Szene**
  Enthält die gesamte Animation. Der Szene werden Objekte während der Laufzeit hinzugefügt und entfernt.
- **Kamera**
  Es gibt viele unterschiedliche Arten von Kameras, diese haben die Attribute Blickfeld, Bildformat, Nähe und Weite. Durch die Kamera werden also die Perspektive sowie die Projektion festgelegt.
- **Renderer**
  Auch hier gibt es unterschiedliche Arten. Der Renderer dient zur eigentlichen Bilddarstellung, also der Definition des Farbtons eines jeden Pixels auf dem Bildschirm.
- **Licht**
  Erschafft das Ambiente, Spots, Schatten usw.
- **Materialien**
  Setzt die Beschaffenheit und die Erscheinung der Objekte.
- **Math**
  Three.js enthält eine eigene Mathematik Bibliothek.
- **Geometrische** Formen
  Werden den Szenen hinzugefügt.
- **Objekte**
  Sind geometrische Formen, Abwandlungen davon und Materialien.

Ruby on Rails (MVC Framework basierend auf Ruby)

5.3.4.1 Ruby

Ruby ist eine dynamische, objektorientierte Programmiersprache. Sie wurde Mitte der 90er-Jahre von Yukihiro Matsumoto in Japan entwickelt.

5.3.4 Die Syntax von Ruby besteht aus den besten Teilen von Perl, Smalltalk sowie Eiffel und Lisp. Ruby beinhaltet Funktionen, Objektorientierung und Grammatik. Es hat dynamische Typen und eine automatische Speicherverwaltung (Memory-Management). Die Autoren des Buches „programming-ruby-1.9“ beschrieben Ruby folgendermassen: (Thomas, et al., 2012)

“When we discovered Ruby, we realized that we’d found what we’d been looking for. More than any other language with which we have worked, Ruby stays out of your way. You can concentrate on solving the problem at hand, instead of struggling with compiler and language issues. That’s how it can help you become a better programmer: by giving you the chance to spend your time creating solutions for your users, not for the compiler.”

5.3.4.2 Rails

Rails ist ein in Ruby geschriebenes Framework zur Entwicklung von Web Applikationen. Es unterstützt den Entwickler bei einer sauberen Programmierung. Vor allem erlaubt es mit wenig selber geschriebenem Code vieles zu erreichen.

Rails ist eine eigensinnige Software mit einer eigenen Philosophie. Rails geht davon aus, dass es einen besten Weg gibt, etwas zu tun. Um Rails sinnvoll zu nutzen, ist es erforderlich sich an diesen Weg bzw. an diese Philosophie zu halten. So verfolgt das Framework die Prinzipien „DRY“ (don’t repeat yourself), „Konvention vor Konfiguration“ und REST als beste Patterns für eine Web Applikation.

Bei Rails handelt es sich um ein MVC Framework. Es hat also die Model-, View- und Controller-Architektur im Zentrum. Dies ermöglicht eine Trennung der Businesslogik von der Benutzerschnittstelle und unterstützt den Grundsatz des DRY-Prinzips.

5.3.4.3 REST

REST steht für „Representational State Transfer“ und bildet die Grundlage von REST-basierten Architekturen.

Rails implementiert REST anhand der folgenden zwei Hauptprinzipien:

- Nutzung von REST-Identifikatoren, wie z.B. URLs, um Ressourcen zu repräsentieren
- Transferieren der Repräsentation von Ressourcen zwischen den einzelnen Komponenten eines Systems

Der folgende http-Request:

DELETE /photos/17

spricht z.B. die Löschung einer Foto-Ressource mit dem eindeutigen Identifikator 17 an.

6 Prototypen

Wir arbeiten auf zwei unterschiedlichen Betriebssystemen: Auf Microsoft Windows und auf Linux. Es war uns wichtig unsere Arbeitsumgebung zu vereinheitlichen. Deshalb haben wir uns entschieden mit dem Programm Vagrant[12] zu arbeiten

Bei Vagrant handelt es sich um eine Automationslösung für Umgebungen auf virtuellen Maschinen, basierend auf Oracle Virtual Box. Dabei kann mit so genannten Basis-Boxen (vorgefertigte virtuelle Maschinen) und einer Konfigurationsdatei (Vagrantfile) mit relativ wenig Aufwand eine Arbeitsumgebung in einer virtuellen Maschine geschaffen werden. Dies hat den Vorteil, dass das Ganze betriebssystemunabhängig ist und sich der stabile Grundzustand jeweils mit nur einem Befehl wieder herstellen lässt.

Dabei sind wir auf einige Start-Schwierigkeiten gestossen. Vor allem in Bezug auf Berechtigungsfragen auf dem Microsoft Windows Betriebssystem. Diese konnten wir jedoch im Verlaufe des Projektes lösen.

6.1 Stanbol

Im Rahmen dieser Projektarbeit wurde ein einfacher Prototyp erstellt, welcher Entitäten aus dem Text

> *John Carmack is the founder of Id Software.*

extrahiert. Als Entitäten werden hierbei eine Person (John Carmack), eine Organisation (Id Software) und die Sprache (Englisch) gefunden. Dabei wurden keine eigenen Entitäten erstellt sondern die Datenbank von DBpedia[13] verwendet.

Theoretisch würden die gefunden Entitäten dann an die Reasoner-Komponente weitergeleitet und es würde versucht eine Schlussfolgerung daraus zu ziehen.

Im Rahmen dieser Projektarbeit wurde jedoch manuell eine RDF-Datei[14] mit den entsprechenden Entitäten, Ontologien und Relationen erstellt. Aus dieser wurde dann mittels dem Jena OWL Reasoner die Schlussfolgerung gewonnen[15], dass John Carmack Mitarbeiter bei der Firma Id Software ist.

Die Frontend-Integration sowie die Ausgabe als Graph wurden hierbei jedoch vernachlässigt, es geht rein um einen Nachweis der Machbarkeit.

[12] http://www.vagrantup.com/
[13] http://dbpedia.org/About
[14] idsoftware.xml
[15] Idsoftware_enriched.xml

6.2 Userinterface

Bei dem Prototypen des Userinterfaces geht es in erster Linie darum, die verschiedenen, in der Bachelor Thesis voraussichtlich benötigten, Technologien und Funktionalitäten zusammen zu führen und zu testen. Ausserdem dient er auch der Einarbeitung in die Programmiersprache Ruby sowie in das Framework Ruby on Rails.

Frontend

In diesem Prototypen wurde Three.js in die Web-Applikation integriert. Ausserdem konnten anhand eines simplen Beispiels die Grundlagen von Three.js ausprobiert werden.

6.2.1

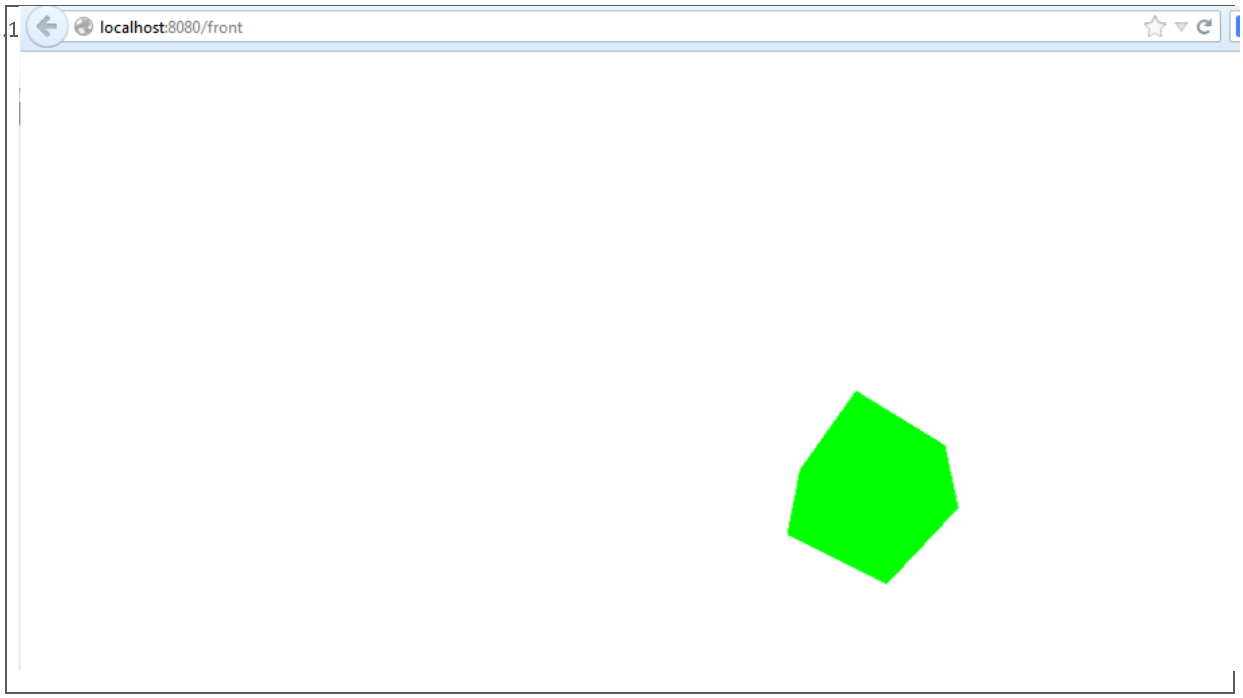

Illustration 8: Frontend

6.2.1.1 Probleme:

Three.js ist eine JavaScript-Bibliothek. Die aktuelle Version von Ruby on Rails nutzt aber in der Regel CoffeeScript[16] zur Einbindung von JavaScript (sog. Assets). Dies erforderte eine Umwandlung des in der Three.js Dokumentation vorgegebenen JavaScript-Codes in CoffeeScript.

[16] http://coffeescript.org/

Management Interface

Der Prototyp des Management Interfaces zeigt eine mögliche Verwaltungsoberfläche. Das Desgin wurde mittels Bootstrap (ein Front-End Framework von Twitter)[17] umgesetzt.

Eine für unsere weitere Arbeit sehr wichtige Funktionalität ist die Nutzung von RESTful Services da die Kommunikation mit Apache Stanbol durch REST abgewickelt wird. Aus diesem Grund greifen wir als
6.2.2 Beispiel auf einen frei verfügbaren REST-Dienst[18] zu. Die Nutzung des REST-Dienstes verarbeiten wir mit dem Ruby-Gem REST Client[19].

Ausserdem möchten wir zum Verwalten der falsch beantworteten Fragen ein editierbares Grid oder eine Tabelle verwenden. Dazu haben wir die Bibliothek editable.js verwendet.

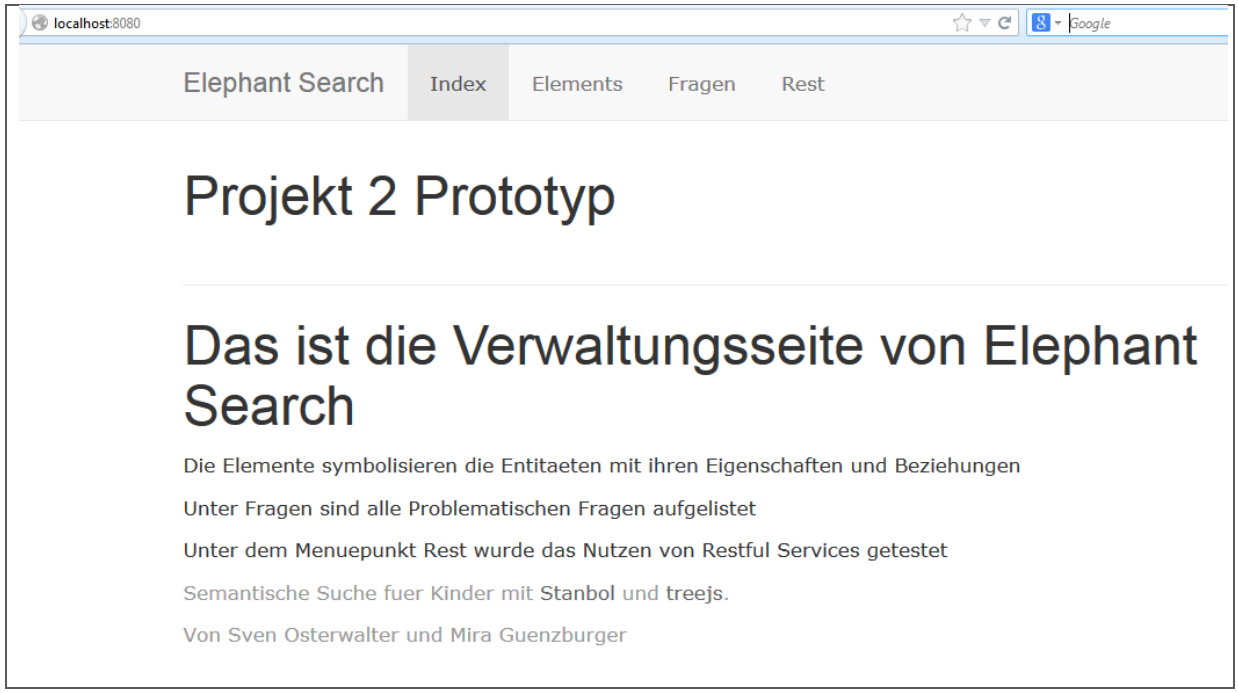

Illustration 9: Prototyp Backend

6.2.2.1 Probleme

Am Anfang des Semesters haben wir voller Elan und sehr viel Optimismus mit unserem Projekt gestartet. Eine der Hauptschwierigkeiten war dann allerdings, dass wir lange gar nicht wussten was überhaupt verwaltet werden muss. So mussten wir die Verwaltungsoberfläche einige Male umbauen.

Bei der Einbindung des RESTful Services sind wir auf einige Schwierigkeiten gestossen. Aber eigentlich nur, weil unser Beispiel-Server eine etwas unübersichtliche, verschachtelte Struktur hat.

[17] http://getbootstrap.com/
[18] http://www.thomas-bayer.com/sqlrest
[19] https://github.com/rest-client/rest-client

7 Fazit

Die Projektidee entstand aufgrund einer spontanen Idee im Januar 2013. Wir wollten das Technische mit dem Sozialen verbinden. Technik soll als Hilfsmittel für den Menschen verstanden werden – sie soll sich dem Menschen anpassen, nicht umgekehrt. Das Thema Kinder kam auf, da Sven Osterwalder im Vorfeld für infoklick.ch – Kinder- und Jugendförderung Schweiz tätig war.

Im Nachhinein gesehen sind wir das Projekt relativ optimistisch angegangen, davon ausgehend, dass die Technologie so weit fortgeschritten ist, dass eine generische semantische Suche möglich ist.

Nach einer ersten Recherche fiel der Entscheid relativ schnell auf Apache Stanbol, da dies das aktuell am meisten ausgereifte Produkt zu sein scheint. Mit der Zeit zeigte sich, dass das Produkt relativ umfangreich und komplex ist, sicherlich auch da es aus dem wissenschaftlichen Umfeld stammt und relativ stark seitens der europäischen Union gefördert wurde. Es zeigte sich aber, dass eine generische semantische Suche nicht ohne weiteres möglich ist. Expertensysteme oder auch Frage-Antwort-Systeme sind nach wie vor stark vom vorausgesetzten Grundwissen abhängig. Und generische Wissensdatenbanken liessen sich während des Projektzeitraumes nicht in Erfahrung bringen.

Schliesslich lässt sich die gewünschte Funktionalität dennoch umsetzen, allerdings nicht so generisch wie gedacht. Es muss für alle Fragen, die beantwortet werden können sollen, eine entsprechende Repräsentation erstellt werden. Dies umfasst mehrere Faktoren:

- Das Extrahieren von Objekten muss verlässlich sein und die gewünschten Resultate liefern
- Erstellung und/oder Einbindung von:
    - Entitäten
    - Ontologien der entsprechenden Gebiete
    - Fakten

Während der Arbeit an diesem Projekt hatten wir viele Höhen und Tiefen. Nach einem sehr optimistischen Einstieg kam sehr bald das grosse Erwachen. Es ist uns aber gelungen, uns Stück für Stück aus die Unklarheiten zu beseitigen und uns einen Überblick über die Komponenten zu verschaffen.

8 Ausblick Bachelor Thesis

Aufgrund der gemachten Erfahrungen und gewonnenen Erkenntnissen erachten wir das Projekt im Rahmen der Bachelor Thesis als durchaus durchführbar.

Allerdings muss der Rahmen der Thesis genau abgesteckt werden. Dies heisst, eine genaue Definition der unter 7 genannten Punkte, sowie welche Fragen genau beantwortet werden können sollen und auch wie diese beantwortet können werden sollen.

Dies führt zu einem beträchtlichen Mehraufwand im Bereich der semantischen Suche, was wiederum zu einer Anpassung des Konzepts führt. So wird die Arbeit betreffend dem Frontend eher geringer, damit es uns möglich ist das Backend in einem sinnvollen Umfang zu implementieren.

9 Anhang

9.1 Quellenverzeichnis

DUD1. 2013. Duden. [Online] 12 2013. http://www.duden.de/rechtschreibung/Praedikat.

Gruber, Thomas R. 1993. *A Translation Approach toPortable Ontology Specifications.* 1993.

Hirschman, L. and Gaizauskas, R. 2001. *Natural language question answering: the view from here.* s.l. : Cambridge University Press, 2001.

Norvig, Peter und Russel, Stuart. 2010. *Artifical Intelligence - A Modern Approach.* 2010.

SQL1. 2014. sqlite. [Online] 01 2014. www.sqlite.org/speed.html.

SQL2. 2014. wiki. [Online] 01 2014. http://wiki.hsr.ch/Datenbanken/files/Weber_Morier_Paper.pdf.

STAN1. 2013. Apache Stanbol. [Online] 12 2013. http://stanbol.apache.org/.

STAN2. 2013. Apache Stanbol Enhancement Engines. [Online] 12 2013. http://stanbol.apache.org/docs/trunk/components/enhancer/engines/list.html.

Thomas, Dave, Fowler, Chad und Hunt, Andy. 2012. Programming Ruby 1.9. *The Pragmatic Programmers' Guide.* 2012.

WDE1. 2013. Wikipedia. [Online] 12 2013. http://de.wikipedia.org/wiki/Ontologie_%28Informatik%29.

WDE3. 2013. Wikipedia. [Online] 12 2013. http://de.wikipedia.org/wiki/Semantik.

9.2 Bildverzeichnis

9.3 Glossar

Begriff	Erklärung
Jena	Apache Jena; Semantisches open-source Framework
MVC	Model View Controller
OWL	Web Ontology Language; Beschreiben von Ontologien anhand formaler Beschreibungssprachen
RDF	Resource Description Framework; Formulierung von log. Aussagen im Internet
Ruby On Rails	MVC Web-Framework basierend auf Ruby
SPARQL	Sparql Protocol And RDF Query Language; Graph-basierte Abfragesprache für RDF
SWRL	Semantic Web Rule Language
URI	Unified Resource Identifier; Einheitlicher Bezeichner für Ressourcen