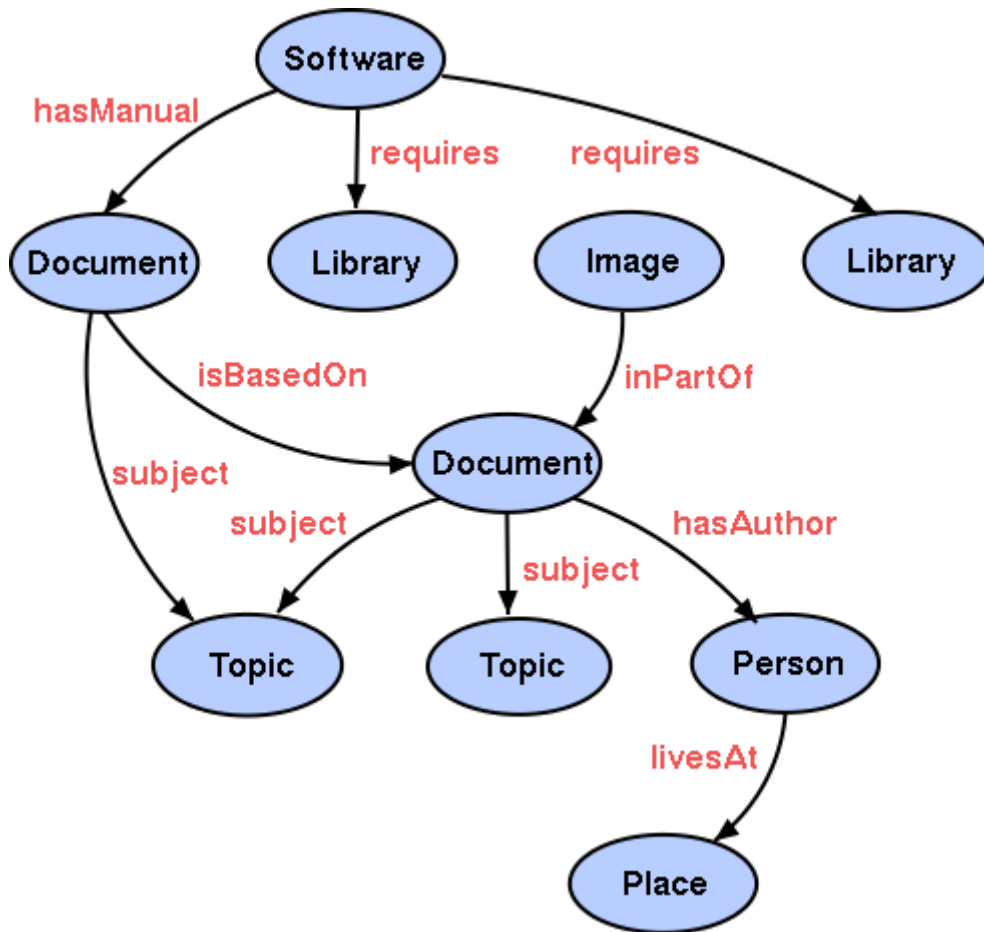


Semantic Web



1 Revision History

Version	Datum	Autor	Bemerkung
0.1	17.09.10	Moos	Initiale Struktur und erster Teil Requirements Engineering, technisches Design
0.2	05.10.10	Moos	Erweiterung RE, Unterteilung in Use cases und NFR
0.3	15.10.10	Moos	Titelblatt, GUI Entwurf
0.4	15.10.10	Lahn	ERM Entwurf
0.5	10.11.10	Moos	Relationales Modell eingefügt, kleine Anpassungen der Usecases
0.6	28.12.10	Moos	Sektion Applikationsaufbau

Inhaltsverzeichnis

- 1 Revision History.....2
- 2 Requirements Engineering3
 - 2.1 – Use cases.....3
 - 2.1.1 Verwaltung von Begriffen.....3
 - 2.1.2 Verwaltung von Beziehungstypen.....3
 - 2.1.3 Erfassen und Ändern von Beziehungen.....3
 - 2.1.4 Locking der einzelnen Einträge.....3
 - 2.1.5 History.....3
 - 2.1.6 Accounts und Rollen.....4
 - 2.2 Non-functional requirements.....4
 - 2.2.1 Darstellung.....4
 - 2.2.2 Mehrbenutzerfähigkeit.....4
- 3 Technisches Design.....5
 - 3.1 Entity-Relationship-Modell.....5
 - 3.2 Relationales Datenmodell.....6
 - 3.2.1 Diagramm.....6
 - 3.2.2 Erläuterung.....6
- 4 GUI Entwurf.....7
 - 4.1 Ablauf.....7
 - 4.1.1 Login.....7
 - 4.1.2 Begriff hinzufügen.....10
 - 4.1.3 Begriff bearbeiten.....10
 - 4.1.4 Begriffsübersicht.....11
 - 4.1.5 History.....12
- 5 Applikationsaufbau.....13
 - 5.1 Technologien.....13
 - 5.2 Struktur.....13
 - 5.2.1 Views.....13
 - 5.2.2 Struts Framework.....13
 - 5.2.3 Hibernate Persistenz.....13

2 Requirements Engineering

2.1 – Use cases

Die Grundlage für das Requirements Engineering stellt das abgegebene Dokument [„Aufgabenstellung Semesterarbeit "Cooperative Modeling" dar.](#)

Die Implementierungsdetails darunter sind per Definition frei. Alle gemachten Entscheide sollten der Übersicht halber hier kurz erwähnt werden.

2.1.1 Verwaltung von Begriffen

Es müssen Begriffe verwaltet werden können. Diese Funktionalität ist eine der grundlegenden Bausteine der Applikation.

2.1.2 Verwaltung von Beziehungstypen

Es müssen verschiedene Beziehungstypen erfasst werden können. Dies ist den Admin-Accounts vorbehalten.

2.1.3 Erfassen und Ändern von Beziehungen

Die definierten Begriffe werden untereinander verknüpft. Dies geschieht über die gerichteten Begriffstypen aus 2.1.2.

2.1.4 Locking der einzelnen Einträge

Wenn ein Benutzer einen entsprechenden Eintrag ändern will und die Maske aufruft, werden der Eintrag und alle Beziehungen damit gesperrt. Diese Sperre wird aufgehoben, wenn dieselbe Session eine andere Seite öffnet resp. die Änderungen abschliesst.

Für den Fall, dass der Benutzer seinen Browser ohne Logout schliesst, wird die Sperre nach 2 Minuten wieder aufgehoben.

Die anderen Benutzer sehen ein gesperrtes Item mit einer anderen Farbe und bekommen eine Warnung wenn sie dieses ändern wollen.

2.1.5 History

Es soll eine History geführt werden, in welcher die Änderungen von Begriffen und Relationen kurz kommentiert werden muss. Dies dient der Absprache mit den anderen Teammitgliedern und soll ein separates Tool (Forum / Chat) unnötig machen.

Die History-Funktion wird auch der Fokus dieser Arbeit darstellen. Es ist sicher, dass die Änderungen an einem Objekt textuell nachvollziehbar sind. Wir werden uns auch darauf konzentrieren, das Datenbankmodell so zu erstellen, dass zu einem späteren Zeitpunkt ein „read-only rollback“ gemacht werden kann. Die gesamte Ontologie könnte damit auf den früheren Stand gebracht werden.

Eine weitere Ausbaumöglichkeit wäre noch, RSS Feeds zu generieren. Diese könnten Benutzer über Änderungen an ihren Objekten informieren.

2.1.6 Accounts und Rollen

Accounts müssen verwaltet werden können. Zudem werden Benutzer in eigentliche User und Admins unterteilt, wobei Admins die gesamte Userverwaltung erledigen.

Zudem ist den Admins vorbehalten, neue Beziehungstypen zu erfassen.

2.2 Non-functional requirements

2.2.1 Darstellung

Die Darstellung der Ontologien und deren Beziehungen soll möglichst anwenderfreundlich sein. Eine Möglichkeit stellt die Benutzung von dynamisch generierten Vektorgrafiken dar. Der Aufwand dafür muss noch abgeklärt werden.

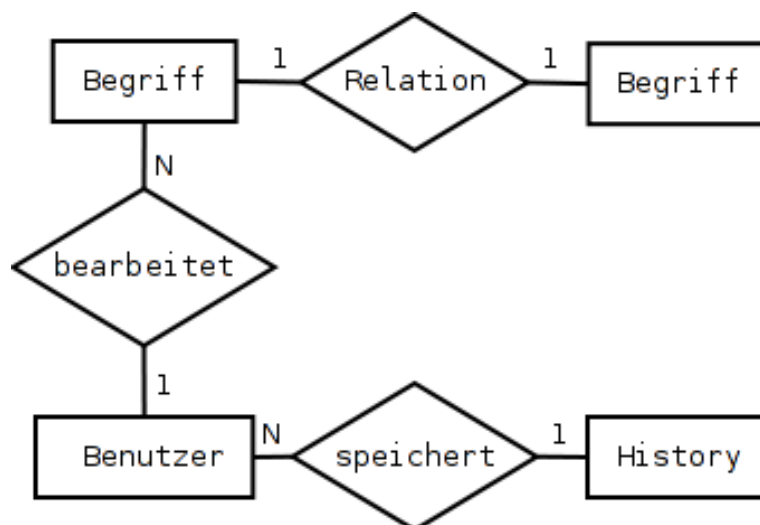
Es soll nicht zu viel Zeit in dieses Problem investiert werden.

2.2.2 Mehrbenutzerfähigkeit

Das Tool muss von mehreren Benutzern gleichzeitig bearbeitet werden können. Dabei darf es nicht zu inkonsistenten Daten kommen.

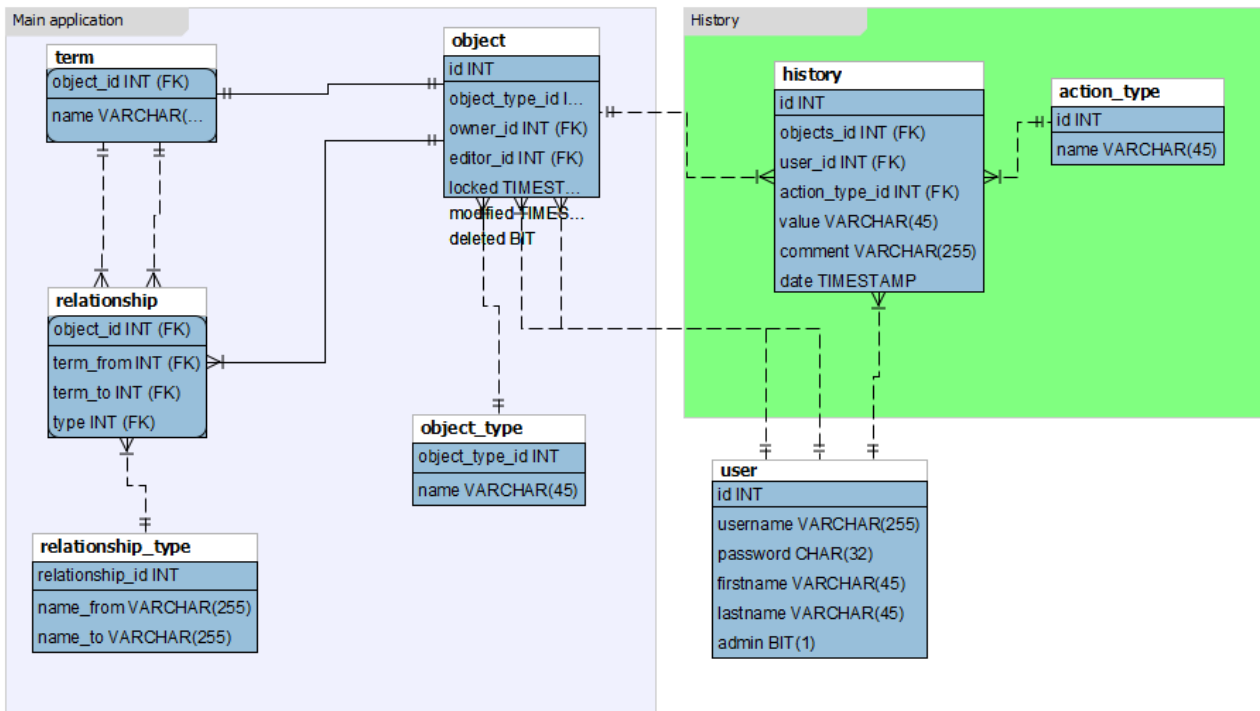
3 Technisches Design

3.1 Entity-Relationship-Modell



3.2 Relationales Datenmodell

3.2.1 Diagramm



3.2.2 Erläuterung

Das relationale Modell setzt sich aus dem Verwaltungsteil der Ontologien, sowie dem History Teil zusammen.

Der Ontologie-Teil ist um die zentrale 'objects' Tabelle aufgebaut. Objects ist eine Generalisierung von term resp. relationship_type. Der Hauptgrund für diese Entscheidung ist die Vereinfachung des History-Teils. So muss dieser nur eine History eines einzigen Objekts führen.

Die Tabelle Relationship ist die eigentliche Beziehung zweier Terme.

Der History-Teil ist relativ selbst erklärend, die Haupttabelle zeichnet alle gemachten Änderungen auf. Zudem wird eine Tabelle action geführt, welche alle möglichen Aktionen (create, rename, delete etc. führen soll). Dies könnte später das angesprochene read-only feedback ermöglichen.

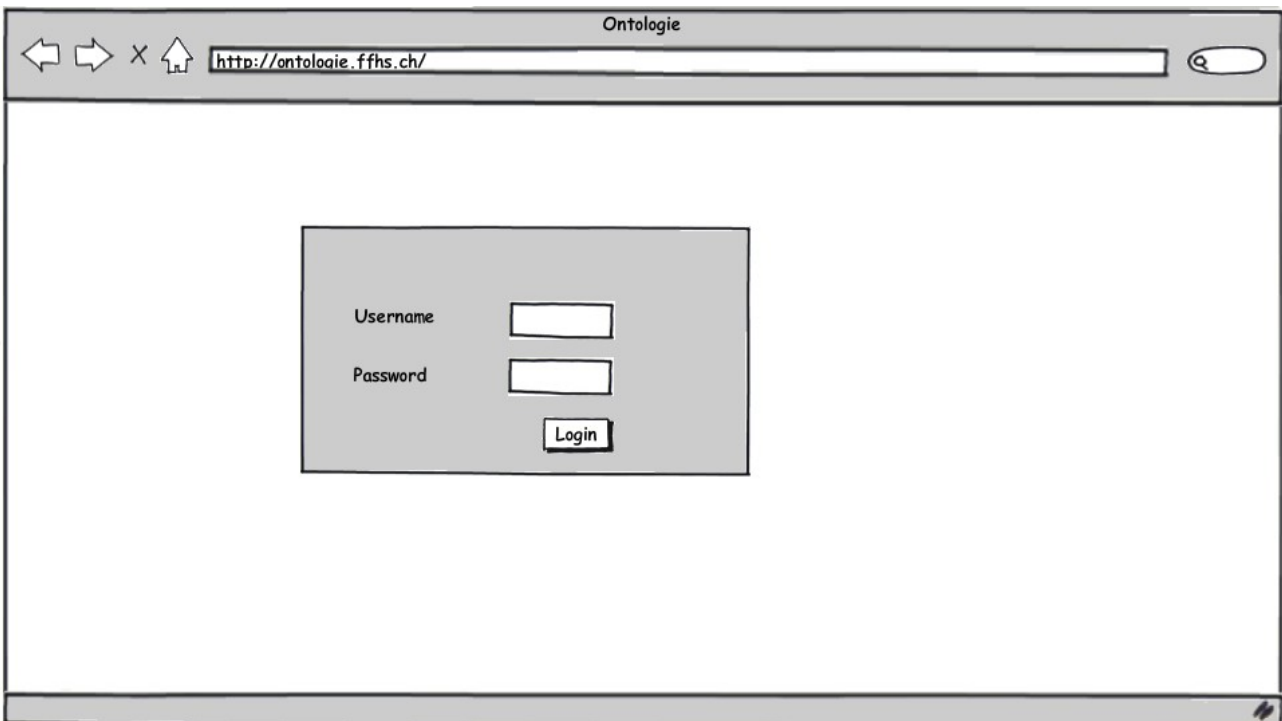
4 GUI Entwurf

Der GUI Entwurf soll kurz darstellen, wie die Applikation aussehen könnte. Sie dient vor allem als Grundlage, dem Kunden einen ersten Eindruck liefern zu können.

Als Tool haben wir uns für Balsamiq Mockups entschieden, welches schnelle und simple Entwürfe ermöglicht.

4.1 Ablauf

4.1.1 Login

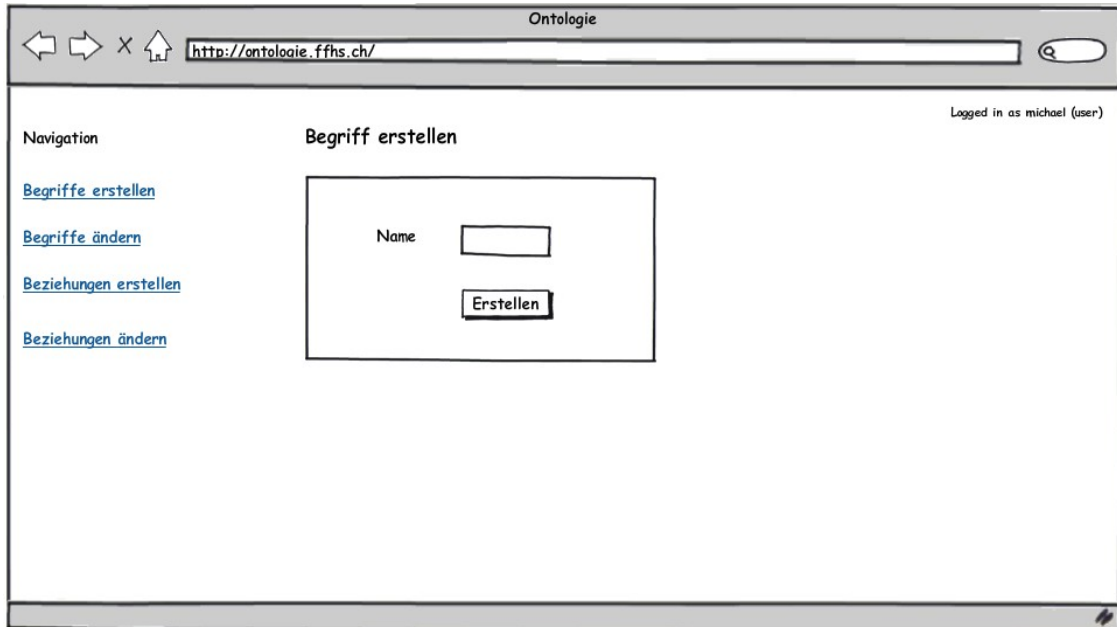


In einem ersten Schritt kann sich der Benutzer einloggen.

Danach erscheint die eigentliche Applikation, links die Navigation.

4.1.2 Begriff hinzufügen

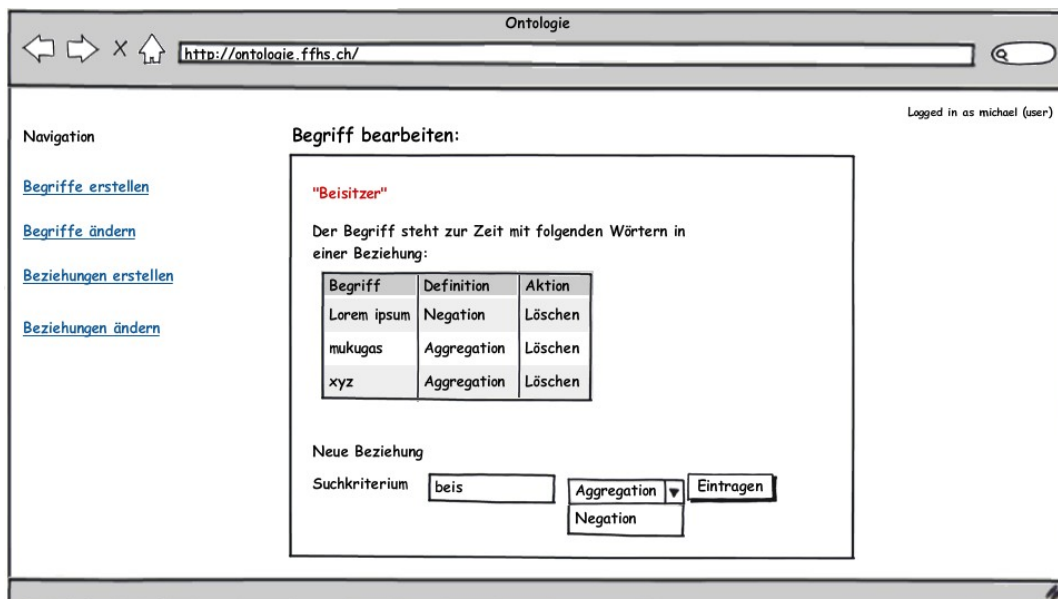
Die erste Funktion nach dem Login (im Benutzerbereich) dient dazu, Begriffe hinzuzufügen.



Hierbei muss lediglich der Name des neuen Begriffs eingegeben werden. Im nächsten Fenster können Beziehungen zu anderen Wörtern aufgebaut werden.

4.1.3 Begriff bearbeiten

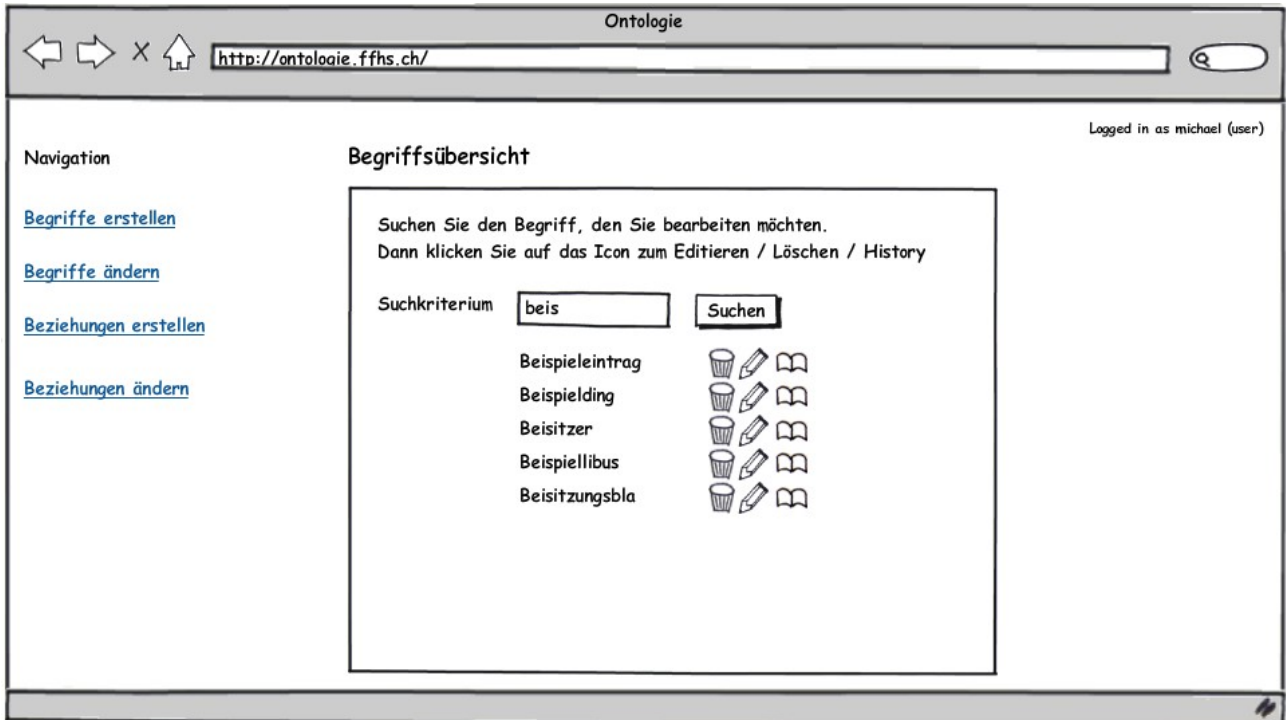
In diesem Formular erscheinen bereits bestehende Beziehungen zu anderen Begriffen – unten können neue hinzugefügt werden.



4.1.4 Begriffsübersicht

Wenn Begriffe geändert werden sollen, wird zuerst in einer Übersicht der entsprechende Eintrag ausgewählt. Dieser kann hier direkt gelöscht werden. Über die Editierfunktion kommt man auf die Bearbeitungs-Funktion zurück (siehe Hinzufügen).

Die letzte hier verfügbare Funktion ist die History. Diese wird im folgenden Abschnitt beschrieben.



4.1.5 History

Die History dient dazu, Änderungen zu kommentieren. So wird jede Änderung an einem Begriff systematisch geloggt. Es ist auf jeden Fall möglich, Kommentare zu erfassen, damit andere Mitarbeiter wissen, warum etwas geändert wurde (analog bsp. SVN Commits).

The screenshot shows a web browser window titled 'Ontologie' with the address bar containing 'http://ontologie.ffhs.ch/'. The page content is divided into a navigation sidebar on the left and a main content area. The sidebar contains links for 'Begriffe erstellen', 'Begriffe ändern', 'Beziehungen erstellen', and 'Beziehungen ändern'. The main content area is titled 'History für Begriff' and displays the history for the concept 'Beisitzer'. It includes a table with columns for 'Datum', 'Name', 'Kommentar', and 'Person'.

Navigation

- [Begriffe erstellen](#)
- [Begriffe ändern](#)
- [Beziehungen erstellen](#)
- [Beziehungen ändern](#)

History für Begriff

Logged in as michael (user)

"Beisitzer"

Der Begriff steht zur Zeit mit folgenden Wörtern in einer Beziehung:

Datum	Name	Kommentar	Person
12.10.10	Beisitzer	-	michael
10.10.10	Beisitzerr	-	michael
8.10.10	Beisitzerr	Synonym 'Zuschauer' hinzugefügt	thomas
6.10.10	Beisitzerr	Begriff erstellt	hansli

Die Beziehungen funktionieren analog zu den Begriffen. Es wird deshalb in diesem Rahmen darauf verzichtet, diese noch explizit aufzuführen.

5 Applikationsaufbau

5.1 Technologien

Die Applikation läuft auf Basis MySQL 5.1 mit einem Tomcat als Webserver für die jsp Seiten. Als Model-2 Framework wird Struts 2 verwendet, der Persistenzlayer wird mit Hibernate realisiert.

Für das Template-System wird Sitemesh verwendet. Dieses rendert die entsprechenden Seiten ins Design und verknüpft CSS etc.

Im nachfolgenden Absatz werden die einzelnen Konfigurationen näher beschrieben.

5.2 Struktur

Die Applikation ist nach dem Model-2 Prinzip aufgebaut.

5.2.1 Views

Die JSP Views liegen im WebContent Ordner. Diese werden mit Hilfe von Sitemesh gerendert. Das Configuration-File dafür liegt unter WebContent/WEB-INF/decorators.xml.

Dieses legt fest, dass das Template aus dem Folder WebContent/decorators/mainTemplate.jsp für alle Seiten verwendet wird.

Die JSP Seiten sind unterteilt in den /admin/ und /user/ Bereich. Entsprechend abhängig von diesem Pfad wird auch die Navigation für /admin/ resp /user/ geladen.

5.2.2 Struts Framework

Das Struts-Config File liegt im /src/struts.xml. Diese Konfiguration ist zentral und definiert die verwendeten Actions und Views für eine Aktion. Sie ist ebenfalls in ein User und Admin Package unterteilt, weil es dabei nur minime Überschneidungen gibt und die Applikation übersichtlicher macht.

Alle Struts Actions liegen im Java Package `ch.ffhs.webE.action`.

5.2.3 Hibernate Persistenz

Die Hibernate Konfiguration liegt im /src/hibernate.cfg.xml. Dort werden die Verbindungsparameter für die Datenbankschicht definiert. Ebenso wird auf die Domain Klassen verwiesen, welche entsprechend im Package `ch.ffhs.webE.domain` zu finden sind. Diese sind mit den hibernate Tools von JBoss erstellt worden.

Alle für den hibernate Zugriff verwendeten DAOs liegen im Package `ch.ffhs.webE.dao`.