Benutzerhandbuch
Version 1.0
Abgabedatum: 2016-05-30

Karsten Schreiblehner
Gruppe EMM-16

30. Mai 2016

Inhaltsverzeichnis

1 Allgemeines

Im Rahmen des Softwarepraktikum wird ein Programm entwickelt, das Studieninteressierten ermöglicht, Kurse von verschiedenen E-Learning-Portalen auf einer gemeinsamen webbasierten Plattform zu suchen.

Es soll eine Applikation entstehen, welche e-Learning-Kurse sächsischer Hochschulen darstellt. Die Kurse stammen aus den beiden Plattformen Moodle (LMS) und OPAL. Zielgruppe für die Plattform sind Studieninteressierte. Die Applikation soll eine Nutzeroberfläche bereitstellen, die in vorhandene Websites eingebunden wird.

Die angesprochene Oberfläche soll ansehnlich darstellen, wo welche Kurse angeboten werden. Der Nutzer soll die Möglichkeit haben, nach Kursen zu suchen. Die Suche soll intelligent die passendsten Kurse liefern, indem in allen Daten des Kurses nach Übereinstimmung gesucht wird. Dafür müssen die Informationen nach Wichtigkeit gerankt und vergleichbar gemacht werden. Besonders die Vereinigung der Daten von Moodle und OPAL spielt dabei eine Rolle.

Ergebnisse sollen übersichtlich und strukturiert nach Hochschule und/oder Studiengang dargestellt werden. Der Nutzer soll die Möglichkeit haben, zu einem Kurs Detailinformationen zu bekommen und ähnliche Dokumente ausfindig machen zu können. Die gesamte Bedienung erfolgt intuitiv ohne große Erklärungen. Inhaltsschwerpunkt sind die Ergebnisse.

Um eine solche Suche und Ergebnisanzeige zu realisieren, müssen die Metadaten der Kurse regelmäßig abgefragt und gecacht werden. Besonders wichtig ist die effiziente Speicherung und schnelle Durchsuchbarkeit der Inhalte. Das System soll dezentral auf den Webservern selbst laufen also ist eine ressourceneffiziente Programmierung Pflicht. Außerdem soll der Webadmin keine zusätzlichen Module oder Verwaltungsprozesse installieren müssen.

2 Produktübersicht

Die einzige für den Nutzer sichtbare Oberfläche ist das eingebundene Webinterface. In diesem kann er eine Suche in einem Suchfeld tätigen. Dort wird ihm auch erklärt, welche Möglichkeiten er zur Suche hat. Ein Crawler speichert eine Kopie der Daten der Portale lokal und übergibt sie in den Standardisierungsprozess. In diesem werden die Daten zunächst in das interne Datenmodell (Course) konvertiert. Für dieses Modell existiert ein Binding von und zu Triplen. Somit können diese in einem RDF Store gespeichert, als auch ausgelesen werden. Der RDF Store läuft lokal auf dem Dateisystem, als Apache Jena TDB Store. Der interne Zugriff auf die Daten erfolgt über die Apache Jena TDB API. Ein externer Zugriff kann zusätzlich über SPARQL erfolgen mittels des Fuseki Servers, der auf die gleichen Daten zurückgreift.

Als RDF-Framework bietet sich Apache Jena an, da sowohl volle SPARQL und modellbasierte Unterstützung für die die Daten besteht, als auch eine einfache Implementierung einer Volltextsuche möglich ist.

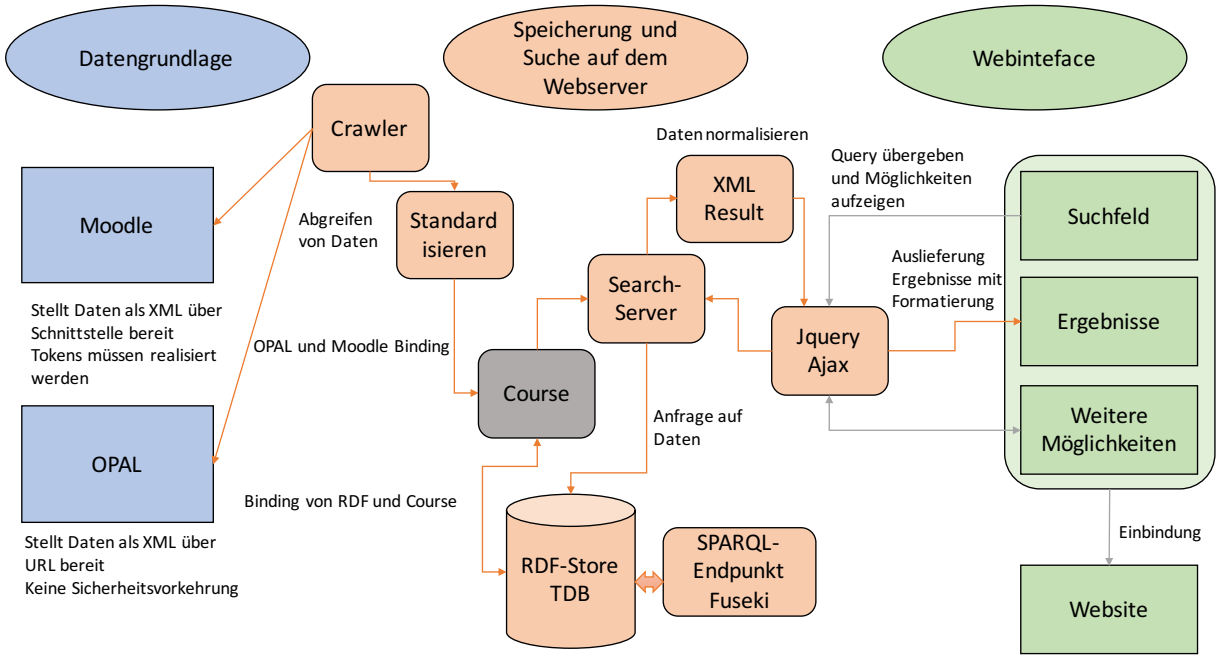

Abbildung 1: Übersicht des Projektentwurfs

Im Gesamtenwurf lässt sich besonders gut die Dreiteilung der Anwendung erkennen. Erstens die Crawler und Datensammlungsseite, Zweitens die interne Verarbeitung und Drittens das Frontend. Der Datenfluss ist von den Quelldaten in das interne Datenmodell Course, dann in RDF Tripel. Daraufhin kann bei einer Anfrage des Frontends dann von dem RDF-Store in das interne Datenmodell, in XML bis zur menschenlesbaren Darstellung unterschieden werden.

3 Installation

3.1 Installation der Anwendung

EMM16.jar in einen beliebigen Ordner verschieben.
Es werden grundsätzlich 3 Ordner benötigt. Einer für den TDB Store, einen für den Lucene Index und einen für die HTML Dokumente. Der einzige Ordner der vorher erstellt werden muss ist der HTML Ordner. Der HTML-Ordner muss vom Webserver aus zugänglich sein. In diesen kommen alle Dokumente aus der Installationsdatei.
Wenn kein Moodle als Datenquelle verwendet werden soll, bei Schritt 3 weiter lesen.

3.2 Installation des Moodle-Plugin

Falls Sie Ihr Moodle einbinden wollen, benötigen sie Administrator-Rechte.
Gehen Sie auf Website-Administration ⇒ Zusatzoptionen
und setzen sie ein Häkchen bei "Webservices"
Gehen Sie auf Website-Administration ⇒ Plugins ⇒ Webservices ⇒ Protokolle verwalten
und aktivieren sie "Protokoll REST"
Gehen Sie auf Website-Administration ⇒ Plugins ⇒ Plugin installieren
und installieren Sie das Plugin über die mitgelieferte "MoodlePlugin.zip" Datei
Gehen sie nun auf Website-Administration ⇒ Webservices ⇒ Externe Services
und klicken Sie auf „Berechtigte Personen"des Externen Services "EMM16 service"
Fügen Sie nun eine Person zu den berechtigten Personen hinzu
oder erstellen Sie zuvor einen Nutzer, der hinzugefügt werden soll
Gehen Sie nun auf Website-Administration ⇒ Plugins ⇒ Webservices ⇒ Tokens verwalten
und klicken Sie auf "Hinzufügen"
Wählen Sie den Nutzer aus, den Sie zuvor zu den berechtigten Personen hinzugefügt haben und den Service "EMM16 service"
Danach klicken Sie auf Änderungen speichern und die Installation des Plugin ist abgeschlossen

3.3 Erstellen der Konfigurationsdatei manuell

Die Konfigurationsdatei kann auch automatisch im Installationsprozess erstellt werden, siehe Punkt 1.5

Öffnen Sie die beiliegende config.ini und Tragen sie ein:
storeAddress=' ' (Die URI des RDF-Store)
Beispiel: storeAddress='http://test.de/emm16/'
storePort=' ' (Der Port auf dem RDF-Store ansprechbar sein soll)
Beispiel: storePort='1620'
storeName=' ' ()
Beispiel: storeName='elearning'
storeDirectory=' ' (Pfad, an dem der RDF-Store erstellt werden soll (relativ oder absolut))
Beispiel: storeDirectory='TDBStore/'
luceneDirectory=' ' (Pfad, an dem der Lucene-Index erstellt werden soll (relativ oder absolut))
Beispiel: luceneDirectory='lucene/'
htmlDirectory=' ' (Pfad, an dem die HTML-Dokumente liegen (relativ oder absolut))
Beispiel: htmlDirectory='HTML/'
source=';' (Zuerst ein eindeutiger Name der Quelle, danach die Adresse des XML-Quelle. Es dürfen beliebig viele Quellen existieren)
Beispiel: source='opal;https://bildungsportal.sachsen.de/opal/catalog.xml'

Wenn Sie kein Moodle als Datenquelle verwenden, fahren sie bei 5. fort
Erstellen der Konfigurationsdatei direkt
einmaliges Starten der jar mittel „java -jar EMM16.jar"
forwardToJava.php muss bereits im html order sein
alle Daten äquivalent zu manueller Anleitung bzw. Punkt 5

3.4 Moodle als Quelle eintragen

Erstellen sie eine Zeile
"source='<MoodleName>;<MoodleAdresse>/webservice/rest/
server.php?wstoken=<MoodleToken>&wsfunction=local_emm16_get_courses_categories' "

Dabei entspricht:
<MoodleName> einem eindeutigen Namen für dieses Moodle z.B. "moodle1"
<MoodleAdresse> der URL unter der Ihr Moodle erreichbar ist
z.b. "http://pcai042.informatik.uni-leipzig.de/~emm16/moodle/"
<MoodleToken> dem Token, das Sie in Schritt 2 erstellt haben

Beispiel: source='moodle1;http://pcai042.informatik.uni-leipzig.de/~emm16/
moodle/webservice/rest/server.php?wstoken=123456&wsfunction=local_emm16_get_courses_categories'

3.5 Beispiel unter Linuxsystemen

Listing 1: bash-Beispiel

```
1 #ins verzeichnis gehen, wo das programm installiert werden soll
2 cd /path/to/target
3 cp /projekt/EMM16.jar .
4 mkdir html
5 cp -r /projekt/html html/
6 #server starten
7 ./checkrunning.sh
8 #installationsprozess folgen
9 #bei fehler manuelle erstellung der config siehe punkt 1.3
```

Danach müssen noch die Dateien *checkrunning.sh* und *reloadData.sh* als Cronjob zum Server hinzugefügt werden.
checkrunning.sh prüft nur, ob das Programm läuft und startet es ggf. neu. Diese sollte in einem 20 minütigen Turnus ausgeführt werden.
reloadData.sh beendet die Programminstanz und lädt die Quelldaten neu ein. Diese sollte wöchentlich ausgeführt werden.
In der Website, wo das Suchinterface eingebunden werden soll, muss an der entsprechende Stelle im Quellcode die iaIndex.html inkludiert werden.

Listing 2: PHP-Beispiel

```
1 <?php
2 include('/path/to/target/projekt/html/iaIndex.html');
3 ?>
```

4 Kompilierungsanleitung für Eclipse

1. **Vorbedingungen**
   "Java JDK"
   "Eclipse IDE for Java Developers"

2. **Importieren**
   Öffnen Sie Eclipse und klicken sie auf File ⇒ Import ⇒ Maven ⇒ Existing Maven Projects
   und wählen Sie den Ordner des Projekts aus, dieser enthält die pom.xml.
   Klicken Sie auf Finish und das Projekt wird importiert.

3. **Kompilieren**
   Wählen Sie nun den Hauptordner Ihres Projektes aus und klicken Sie auf Run ⇒ Run As ⇒ Maven Build...
   und geben Sie bei "Goals:" "package" ein.
   Klicken Sie nun auf "Run"
   Es sollten nun alle benötigten Abhängigkeiten heruntergeladen und das Projekt kompiliert werden.

4. Die EMM16.jar befindet sich nun im Ordner "target"

5 Webinterface

Das Webinterface bietet dem Nutzer eine Suchzeile. In dieser kann er mit einer Fließtexteingabe nach Kursen suchen. Als Ergebnis werden ihm die passenden Kurse geordnet nach Treffenquote angezeigt. Jeder Kurs kann, um mehr Informationen zu erlangen, angeklickt werden. Daraufhin öffnet sich eine Detailansicht mit allen verfügbaren Informationen zum Kurs. Im Falle, dass die Daten aus dem OPAL Portal stammen, wird oben eine Einordnung in den Universitätsbaum angezeigt. Dieser ist wiederum verlinkt. Beim Klick auf eine Universität oder Untergruppe werden in einer Liste alle Kurse angezeigt, die diesem zugeordnet sind. Bei jedem Klick auf einen Kurs oder eine Universität erscheint links oben ein Zurückbutton. Es wird bewusst auf die Funktionalität des normalen Zurückbuttons verzichtet, da die Seite eingebunden wird.

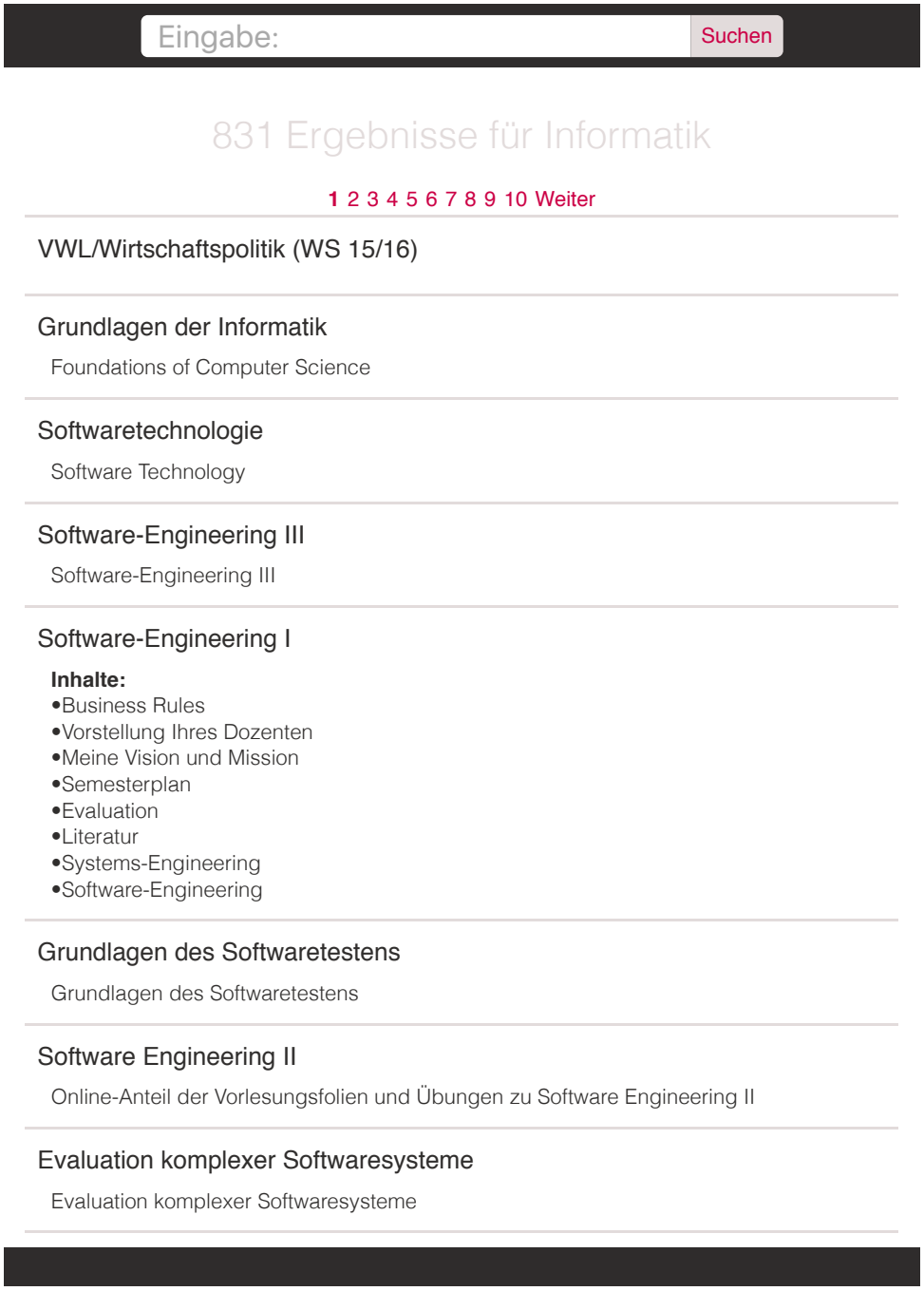

Abbildung 2: Skizze Webinterface bei ausgeführter Suchanfrage

6 Glossar

Blank Node: Repräsentiert eine Ressource, aber indiziert noch keine URI für eine Ressource. Verhält sich wie erstellte, aber nicht initialisierte Variable.

URI: Uniform Ressource Identifier ist ein Identifikator und besteht aus einer Zeichenfolge, die zur Identifizierung einer abstrakten oder physischen Ressource dient.

LOM: Learning Objects Metadata Standard zur Lernobjekten der IEEE. Standard

Dublin Core: Ein Standard für Metadaten. Core web site.

Literal: Ein String von Buchstaben

Object: Der Teil des Tripels, der den Wert des Predicate darstellt.

Predicate: Der Property-Teil des Tripels.

Property: Ist ein Attribut einer Ressource. Zum Beispiel dc.title (Dublin Core) oder rdf.type.

Resource: Eine Entität. Kann ein reales oder irreales Objekt sein. Sie sind durch ihre URI indentifiziert.

Statement: Ist ein Ast im RDF Modell, bestehende aus Subject, Predicate und Object.

Subject: Die Ressource, die Subjekt Teil des Tripels ist.

Triple: Eine Struktur, die Subject, Predicate und Object beinhaltet. Anderer Ausdruck für Statement.