Erstellung eines modularen Newsboards zur Sentimentanalyse

Felix Meyer Lukas Taake

Betreut durch Herrn Prof. Dr.-Ing. Carsten Gips

Projektbericht
eingereicht am Campus Minden
der FH Bielefeld, University of Applied Sciences

18. Januar 2017

FH Bielefeld
University of
Applied Sciences

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Mit der steigenden Zahl an Nachrichten aus immer mehr Kanälen steigt auch der Aufwand, diese zu bewerten. Als eine Möglichkeit bietet es sich deshalb an, Computer zur automatisierten Bewertung von Nachrichten einzusetzen. Anstatt einen Bewertungs-Algorithmus blind sämtliche Nachrichten eines Kanals oder einer Quelle bewerten zu lassen, können Nachrichten vorher auch thematisch gesammelt und gespeichert werden.

1.2 Projektziel

Das Ziel dieses Projekts ist die Konzeption und Erstellung eines Newsboards zum Speichern gesammelter Nachrichten und deren Bewertungen. Das eigentliche Sammeln und Bewerten der Nachrichten geschieht durch externe Module, die über eine Schnittstelle mit dem Newsboard kommunizieren.

Darüber hinaus soll das Newsboard auch eine grafische Oberfläche enthalten, mit der die enthaltenen Nachrichten und deren Bewertungen eingesehen werden können. Aus diesen Anforderungen kann das in Abbildung 1 dargestellte Minimalset an Anwendungsfällen abgeleitet werden.

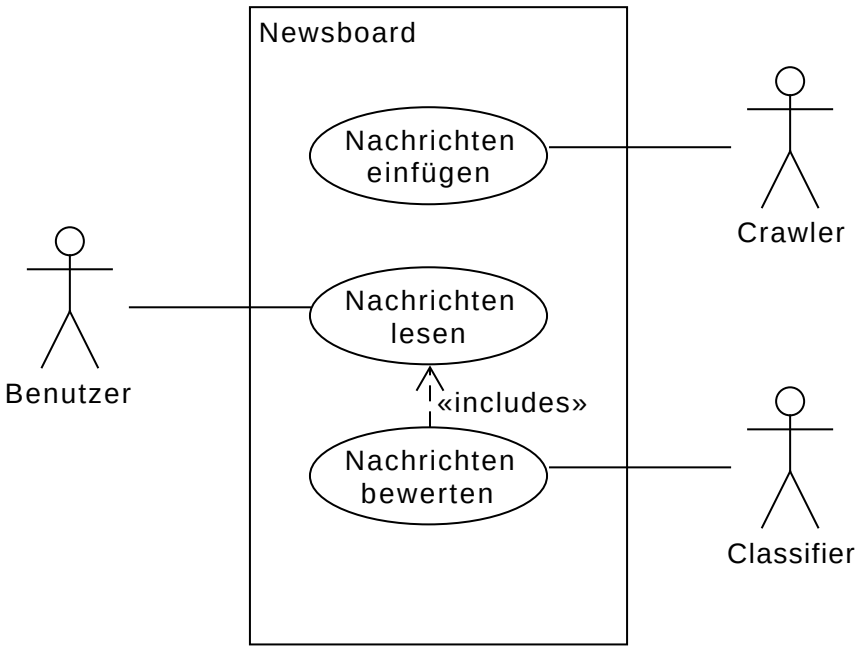

Abbildung 1: Anwendungsfälle

Das Newsboard soll verwendet werden, um Nachrichten zu sammeln und zu bewerten, welche die Fachhochschule Bielefeld betreffen. Darüber hinaus soll es als möglichst praxisnahe Möglichkeit dienen, im Modul Intelligente Systeme entwickelte Crawler- und Bewertungsmodule zu verwenden.

2 Konzept

Um die Funktionsweise des Newsboards aufzuzeigen, wird zuerst das zugrunde liegende Konzept erläutert. Dazu gehören neben Architekturentscheidungen auch das Datenmodell und die Schnittstelle zur Kommunikation mit den externen Modulen.

2.1 Architektur

Der erste richtungweisende Punkt des Konzepts ist die Architektur des Newsboards. Da eine Webseite zum Einsehen der bewerteten Nachrichten eine der Voraussetzungen ist, bietet sich eine klassische Client-Server-Architektur an. Dabei treten neben den Webbrowsern der Benutzer auch die Crawler- und Bewertungsmodule als Clients auf.

Um die Komplexität des Newsboads möglichst gering zu halten, sollte die Kommunikation mit den Modulen ebenfalls über HTTP stattfinden. Das bringt den Vorteil mit sich, dass es für viele Programmiersprachen bereits gute Client-Bibliotheken gibt und sich so keine unnötigen Einschränkungen ergeben.

2.2 Datenmodell

Bei der Analyse der zu persistierenden Daten wurde ebenfalls darauf geachtet, ein sowohl möglichst einfaches, aber auch flexibles Datenmodell zu entwickeln. Zu Projektbeginn wurden diverse NoSQL-Systeme mit verschiedenen relationalen Datenbanksystemen verglichen. So schien es durchaus plausibel, eine der verschiedenen NoSQL-Varianten einsetzen zu können, um Daten schemafrei und somit sehr flexibel ablegen zu können. Bei der Recherche dieser Systeme konnte jedoch keine Variante überzeugenden Mehrwert liefern. Deshalb setzt das Newsboard in der jetzigen Form auf ein relationales Modell zur Persistenz, da diese Systeme weiter verbreitet, ausgereift und dokumentiert sind. Darüber hinaus werden an der FH Bielefeld relationale Datenbanken als Pflichtfach gelehrt, was der Weiterentwicklung des Newsboards zu Gute kommen kann.

Das Entity-Relationship-Diagramm in Abbildung 2 gibt eine Übersicht über die im Newsboard vorhandenen Entitäten und ihren Attributen.

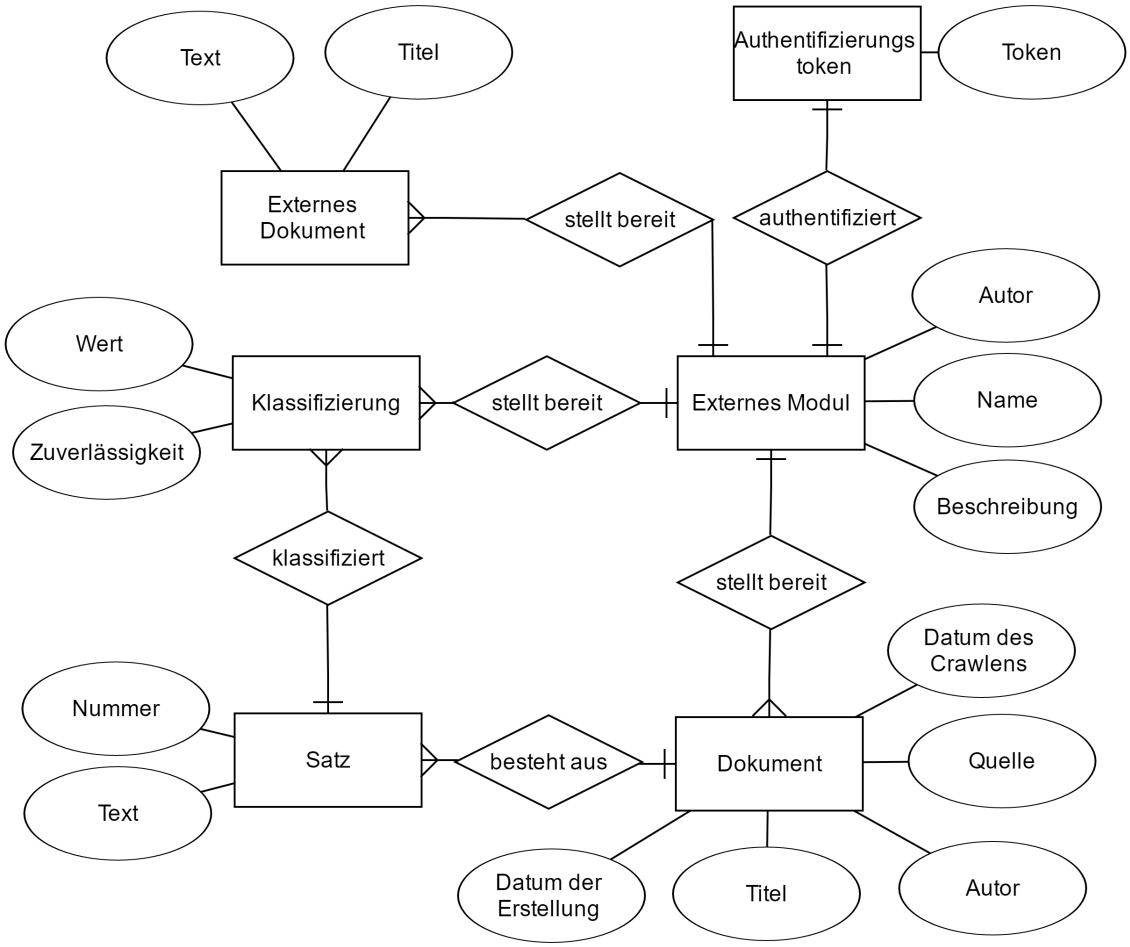

Abbildung 2: ERD des Newsboards

Wie aus dem ERD ersichtlich, besteht das Newsboard aus diesen nun näher beschriebenen Entitäten:

- **Externe Module**
  Die externen Systeme, die Daten für das Newsboard hochladen, werden in dieser Entität abgebildet. Von Interesse sind Informationen über den Autor des Moduls, den Namen sowie eine kurze Beschreibung.
- **Authentifizierungstoken**
  Authentifizierungstokens dienen zur Bestätigung der Identität von externen Modulen. Nur Module, die über ein gültiges Authentifizierungstoken verfügen, dürfen in der Lage sein, Daten für das Newsboard hochzuladen. Ein Token besteht aus einer zufälligen Zeichenkette.
- **Dokumente**
  Nachrichten und Texte, die im Newsboard dargestellt werden sollen oder zur Klassifizierung bereitstehen, werden als Dokumente im System abgelegt. Der Text wird in Sätze aufgeteilt, über die Dokumente werden einige Metainfor-

mationen gespeichert. Hierzu zählen zum Beispiel das Datum der Erstellung und des Crawlens, der Autor sowie der Titel und die Quelle des Dokumentes.

- **Externe Dokumente**
  Dokumente oder Nachrichten, die nicht klassifiziert, dem Gesamtsystem aber dennoch zur Verfügung stehen sollen, werden als externe Dokumente gespeichert. Der Inhalt dieser Dokumente wird als Plaintext abgelegt. Auch der Titel des Dokumentes ist von Interesse.

- **Sätze**
  Die eigentlichen Sätze, die in einem Dokument geschrieben sind, werden separat gespeichert. Sie sind es auch, die zur Klassifizierung von externen Modulen bereitstehen. Um ein Dokument wieder korrekt zusammenzusetzen, wird zu dem die Position eines jeden Satzes im Dokument als Nummer gespeichert.

- **Klassifizierungen**
  Die Klassifizierungen der Sätze werden in Form dieser Entität gespeichert. Neben den Informationen, zu welchem Satz eine Klassifizierung gehört, muss zudem das klassifizierende Modul abgelegt werden. Sie besteht aus zwei Werten: einem für die errechnete Klassifizierung und einem für die Zuversichtlichkeit des Moduls.

2.3 Schnittstelle

Die Hauptziele bei der Konzeption der Schnittstelle sind zum einen eine möglichst einfache Verwendung, zum anderen eine verlässliche Validierung der übertragenen Daten. Da die Schnittstelle auf HTTP aufbaut, bietet es sich an, mit einer REST-Schnittstelle die Möglichkeiten von HTTP zu nutzen und auf eine zusätzliche Zwischenschicht zu verzichten.

Als Datenformat für REST-Schnittstellen sind sowohl JSON, als auch XML etabliert. Da XML im Gegensatz zu JSON allerdings eine verlässliche Möglichkeit zur Validierung und Dokumentation durch ein Schema besitzt, ist es in diesem Fall die bessere Wahl. Der geringfügig höhere Speicherverbrauch, sowie die derzeit geringe Popularität von XML sollen hier keine Rolle spielen.

Bei der Konzeption der verwendeten REST-Ressourcen, sowie deren akzeptierten Methoden, bestehen ebenfalls zwei mögliche Szenarien. Das erste ist die Abbildung sämtlicher Datentypen des Datenmodells und das Erlauben aller CRUD-Operationen. Das zweite Szenario ist es, ausschließlich einige wenige Ressourcen zur Verfügung zu stellen, welche die geplanten Anwendungsfälle abbilden.

Letzteres verlangt komplexere Daten, die von der Schnittstelle verarbeitet werden müssen, andernfalls läge diese Komplexität jeweils in den einzelnen Clients und müsste mehrfach konzeptioniert und umgesetzt werden. Das führt letztendlich zu mehr unnützer Arbeit und gleichzeitig zur mehr möglichen Fehlerquellen. Da für diese Schnittstelle potentiell regelmäßig neue Clients entwickelt werden sollen, ist die zweite Variante als deutlich zweckmäßiger sowie allgemein eleganter anzusehen.

Die sich aus den Anwendungsfällen und konzeptionellen Vorüberlegungen resultierenden Aktionen, welche durch die REST-Schnittstelle ermöglicht werden, lauten wie folgt:

- Einfügen neuer Dokumente
- Lesen von Dokumenten
- Einfügen neuer Klassifizierungen für Dokumente

3 Implementierung

Nach dem Konzept wird nun dessen konkrete Umsetzung behandelt, um einen tieferen Einblick in die Funktionsweise des Newsboards zu geben.

3.1 Technische Basis

Umgesetzt wird das Newsboard in Java, sowohl wegen persönlicher Präferenzen der Autoren, als auch auf Grund der Tatsache, dass hauptsächlich Java als Programmiersprache an der FH Bielefeld gelehrt wird. So werden der Weiterentwicklung des Newsboards durch weitere Studenten keine vermeidbaren Hürden in den Weg gestellt. Darüber hinaus gibt es für Java eine große Zahl an ausgereiften und gut dokumentierten Bibliotheken sowie verlässliche Frameworks zum Build-Management. Verwendet wird das aktuelle Feature-Level von Java 8, da es keinerlei Altlasten gibt, auf die Rücksicht genommen werden müsste.

Als Basis für das Newsboard wird das Spring-Framework im Zusammenspiel mit Maven als Build-Management-Tool verwendet. Diese Kombination ist in der Praxis bewährt, ist flexibel für verschiedenste Anwendungsfälle, sowie ohne große Einarbeitungszeit gut zu benutzen. Außerdem besteht so keinerlei Bindung an eine bestimmte Entwicklungsumgebung.

Spring bietet außerdem eine sehr ausgereifte Inversion-of-Control bzw. Dependency Injection, wodurch eine lose Kopplung der einzelnen Komponenten untereinander gewährt wird[2]. In der Umsetzung von Spring werden dabei die einzelnen Klassen nur mit Annotationen versehen und haben ansonsten keine zwingenden Abhängigkeiten zum Spring Framework.

Darüber hinaus können neben konkreten Klassen auch Abhängigkeiten zu Interfaces injiziert werden, welche wiederum in ihrer Auflösung in konkrete Klassen durch die Konfiguration von Spring beeinflusst werden können. So kann zum Beispiel entweder eine Konfiguration für eine SQL-Datenbank, oder einen anderen Datenspeicher angelegt werden und in Abhängigkeit der aktivierten Konfiguration wird eine andere Implementierung für das Interface injiziert.

3.2 Datenmodell

Das Datenmodell des Newsboards wird mit dem relationalen Datenbankmanagementsystem MySQL abgebildet. Als weitverbreitetes Open-Source Produkt eignet es sich optimal, um im Newsboard eingesetzt zu werden. Basierend auf dem vom ERD beschriebenen Modell (Abb. 2) des Newsboards wurde ein physisches Modell für MySQL entwickelt (Abb. 3).

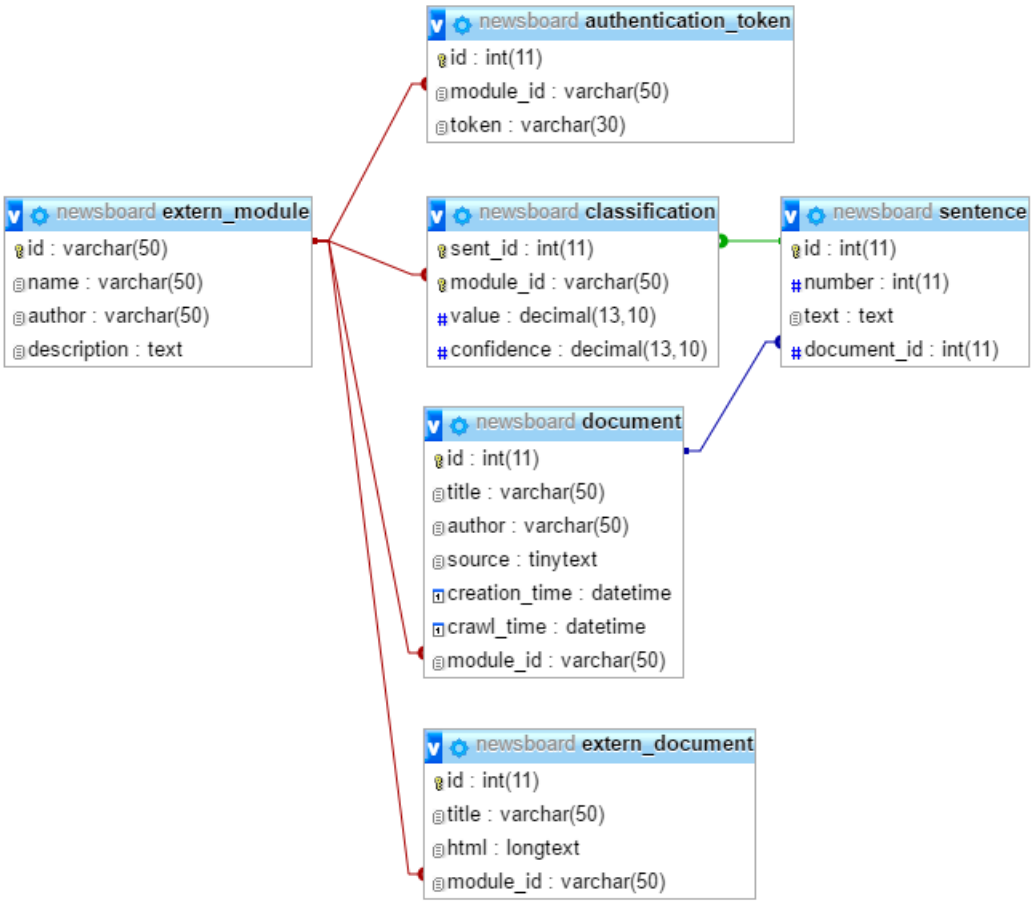

Abbildung 3: Physisches Modell des Newsboards

Das Modell wird in der Schnittstelle über acht Klassen abgebildet (Abb. 4). Hier zeigen sich Abweichungen zum physischen Modell. So ist die Klasse Document eine Containerklasse, die die Metainformationen und die Sätze des Dokuments speichert. Um die Metainformationen eines Dokumentes zu struktieren, ist für sie eine eigene Klasse vorhanden: DocumentMetaData. Für Sätze steht die Klasse Sentences bereit, die zudem eine Liste mit allen Klassifikationen eines Satzes beinhaltet. Die Klasse RawDocument dient rein zum Einlesen von Dokumenten, welche von Crawlern bereitgestellt werden. Sie besteht aus den Metainformationen und dem Text eines Dokumentes. Nach dem Einlesen wird ein RawDocument in ein Document überführt.

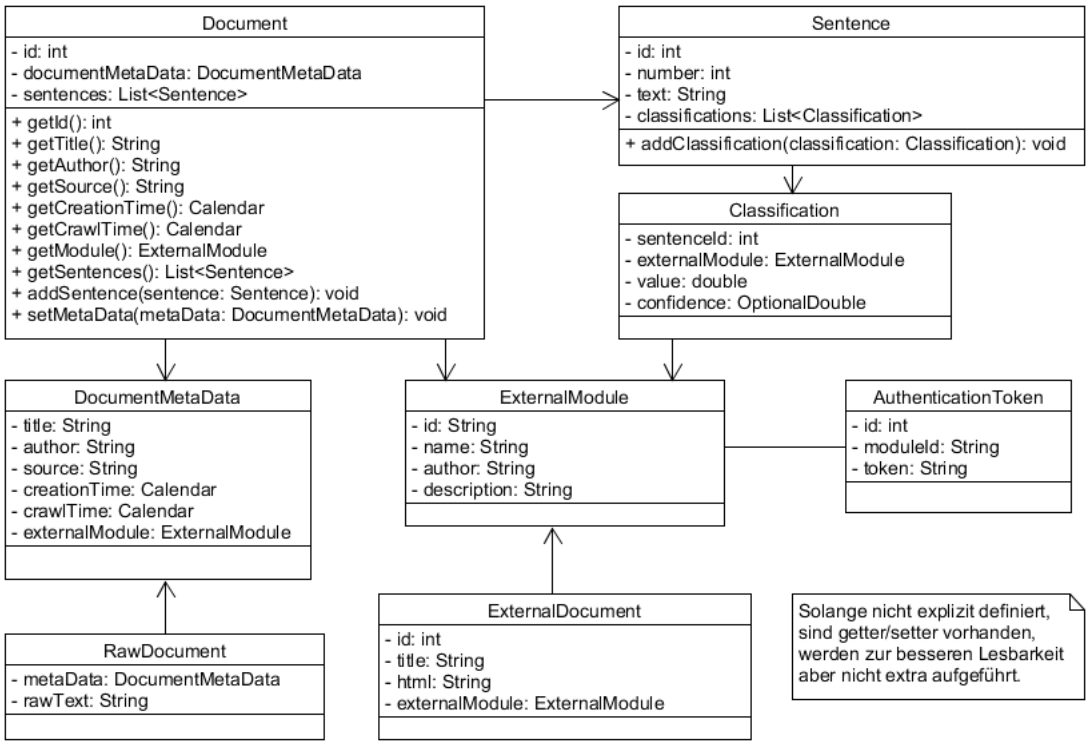

Abbildung 4: Diagramm der intern verwendeten Modellklassen

Für die Kommunikation zwischen der Schnittstelle und der Datenbank wird auf das Data-Access-Object Design Muster gesetzt, um eine hohe Modularität und Flexibilität zu gewährleisten [3]. Beim DAO Muster werden verschiedene Interfaces verwendet, um den Zugriff auf Datenbankoperationen zu maskieren, während die eigentliche Übertragungslogik pro Datenquelle eigens entwickelt werden kann. Das, und die Dependency-Injection von Spring, ermöglichen es, im Falle eines Datenbankwechsels von MySQL hin zu einem beliebigen anderem System, die Schnittstelle fast ohne Anpassungen weiter verwenden zu können.

3.3 Dokumentverarbeitung

Um die RawDocuments, welche von Crawlern über die REST-Schnittstelle eingefügt werden, in Document-Objekte zu überführen, muss der Text in einzelne Sätze aufgeteilt werden. Diesen Zweck erfüllt die Klasse RawDocumentProcessor. Sie nutzt die Apache openNLP Bibliothek, um den unverarbeiteten Text von RawDocuments in Sentences zu überführen.

Apache openNLP ist eine Sammlung verschiedener Werkzeuge, um mit Java einfach diverse NLP-Probleme zu lösen [4]. Es liefert neben unterschiedlichen Machine-Learning-Algorithmen auch fertige Modelle zur Verwendung mit, unter anderem zum Finden von einzelnen Sätzen. Diese Modelle sind in verschiedenen Sprachen

vorhanden, für das Newsboard wird eines zur Erkennung der deutschen Sprache genutzt.

OpenNLP stellt zur Satzerkennung das Interface SentenceDetector und die davon ableitende Klasse SentenceDetectorME zur Verfügung. SentenceDetectorME nutzt ein Maximum-Entropy-Modell zur Klassifizierung von Satzzeichen, die einen Satz beenden. Sie wird auch im RawDocumentProcessor eingesetzt. Über einen einfachen Methodenaufruf kann so ein kompletter Text verarbeitet werden. Auch die Kopplung zu openNLP bleibt sehr gering, da die Bibliothek nur hier Verwendung findet.

3.4 Schnittstelle

Zur Implementierung der REST-Endpoints der Schnittstelle werden die in Spring enthaltenen Controller-Features genutzt. Um damit einen Endpoint zu bedienen, reicht es aus, eine Methode innerhalb einer Controller-Klasse mit `RequestMapping` zu annotieren. Listing 1 zeigt beispielhaft, wie ein solcher Endpoint unter Angabe der Annotations-Parameter implementiert werden kann.

Der Rückgabewert der Methode wird so direkt als Antwort an den Client zurückgeliefert. Darüber hinaus können mit Hilfe der Dependency-Injection von Spring weitere Parameter definiert werden, wie die HTTP-Anfrage oder eine Antwort wie der im Beispiel.

```
1  @RestController
2  public class RestApiController {
3    [...]
4    @RequestMapping(path = "/document", method = RequestMethod.
      GET, produces = MediaType.APPLICATION_XML_VALUE)
5    public String listDocuments(HttpServletResponse response) {
6      [...]
7      return documents;
8    }
9    [...]
10 }
```

Listing 1: Implementierung eines REST-Endpoints mit Spring

Auf diese Weise werden jeweils die im Konzept aufgezählten Aktionen implementiert. Es ergeben sich zur wesentlichen Vereinfachung der Benutzung der Schnittstelle, sowie um möglichst effektiv arbeiten zu können, mehrere Endpoints zum Lesen der Dokumente. Somit bildet sich die folgende Liste an vorhandenen Kombinationen von Aktionen und Ressourcen:

- **GET /document** Lesen einer Liste der Metadaten aller Dokumente
- **GET /document/{id}** Lesen eines vollständigen, spezifizierten Dokuments inklusive einzelnen Sätzen und Klassifikationen
- **PUT /document** Einfügen eines neuen Dokuments durch einen Crawler

- **GET /unclassified** Lesen aller Dokumente, die vom aktuell authentifizierten Classifier noch nicht bewertet wurden
- **PUT /classify** Einfügen neuer Klassifizierungen durch einen Classifier

Um die Ressourcen der REST-Schnittstelle klar von der Web-Oberfläche zu trennen, sind die angegebenen Pfade relativ zum Verzeichnis `/rest`.

Da nicht jeder beliebige Nutzer neue Dokumente oder Klassifizierungen einfügen können soll, muss eine Art von Authentifizierung vorhanden sein. Es bietet sich an, auch das Lesen von unklassifizierten Dokumenten nur authentifizierten Nutzern zugänglich zu machen, da dies immer spezifisch für einen bestimmten Classifier ist. Umgesetzt wird dies mit einer HTTP-Basic Authentifizierung, die sehr einfach umzusetzen ist, aber für die vorgesehenen Einsatzzwecke ausreicht. Als Benutzername dient immer die ID des Crawlers oder Classifiers, die in der Datenbank hinterlegt ist. Als Passwort dient ein in der Datenbank hinterlegtes AuthenticationToken.

3.5 XML-Format

Über die REST-Endpoints hinaus ist bei der Schnittstelle noch die Verarbeitung der XML-Daten, sowie das Schema zur Validierung interessant. Da die zu erwartenden Datenmengen im Vorhinein nicht abschätzbar sind, sollte das Lesen und vor allem das Schreiben der XML-Daten möglichst wenig Speicher verbrauchen, um eventuelle Engpässe zu vermeiden. Beim Lesen ist eine Validierung gegen das Schema unbedingt notwendig, da nicht davon ausgegangen werden kann, dass die ankommenden Daten in jedem Fall valide sind.

Aufgrund der Speicheranforderungen ist ein herkömmlicher DOM-Parser in diesem Projekt nicht geeignet. Gelesen werden die XML-Daten stattdessen mit einem SAX-Parser[5]. Dieser ist weniger speicherbelastend als ein DOM-Parser und kann, anders als ein StAX-Parser, eine Validierung des Dokuments während des Einlesens durchführen. Verwendet wird die Standard Implementierung des JDK: `javax.xml.parsers.SAXParser`.

Da SAX-Parser üblicherweise nur lesen, aber nicht schreiben können, kommt dazu ein anderer Parser zum Einsatz. In diesem Fall ein StAX-Parser, da Validierung nicht zwingend zur Laufzeit erfolgen muss, sondern im Vorhinein im Zuge automatisierter Tests durchgeführt werden kann. Verwendet wird hier ebenfalls eine Implementierung, die Bestandteil des JDK ist: `javax.xml.stream.XMLStreamWriter`. Um die genauere Arbeitsweise beim Verarbeiten der XML-Daten zu beschreiben, wird an dieser Stelle das erstellte Schema beschrieben.

Das im Schema spezifizierte XML-Format erlaubt Dokumente verschiedenster Ausprägung. Das ist notwendig, damit einerseits nicht mehrere Schemata angelegt und gepflegt werden zu müssen, und andererseits die Anforderungen der verschiedenen REST-Endpoints erfüllt werden.

Das Schema ist so spezifiziert, dass jedes XML-Dokument eine Liste von Dokumenten im Sinne des Datenmodells enthalten kann. Dabei kann jedes Dokument

Metadaten enthalten (z.B. Autor, Quelle), die beim Einlesen neuer Dokumente und beim Ausgeben vorhandener Dokumente angegeben sein müssen. Dazu kommt entweder ein Element mit Rohtext beim Einfügen der Dokumente, oder eine Liste von Sätzen und Klassifikationen.

Die Sätze sind nur beim Ausgeben von Dokumenten wichtig, beim Einlesen neuer Klassifikationen sind sie, wie die Metadaten auch, optional und werden nicht beachtet. Wenn eine Liste von Sätzen vorhanden ist, muss sie mindestens einen Satz enthalten. Die Liste von Klassifikationen ist beim Ausgeben spezifizierter und unklassifizierter Dokumente von Belang, sowie natürlich beim Einlesen neuer Klassifikationen. Da z.B. bei frisch eingelesenen Dokumenten noch keine Klassifizierungen vorliegen können, darf die Liste auch leer sein.

Um das Arbeiten mit diesen Dokumenten deutlich zu vereinfachen, können XML-Dokumente weder in beliebiger Ausprägung geschrieben noch gelesen werden. Stattdessen können nur XML-Daten eingelesen werden, wie sie zum Einfügen neuer Dokumente oder Klassifizierungen notwendig sind. Geschrieben werden können nur Dokumentenausschnitte (intern Stub genannt), die nur Metadaten beinhalten, oder vollständige Dokument-Strukturen.

3.6 Web-Oberfläche

Um das Newsboard über die REST-Schnittstelle hinaus auch für Benutzer verfügbar zu machen, beinhaltet es eine einfache Web-Oberfläche. Diese ist im jetzigen Zustand nur im Stande, eine Liste von klassifizierten Dokumenten, oder eine Detailseite zu einem einzelnen Dokument anzuzeigen.

Auf der Übersichtsseite werden nur Titel, Inhalt und ein berechneter Durchschnittswert aller Klassifikationen zu einem Dokument angezeigt. Auf der Detailseite werden darüber hinaus noch weitere Metadaten, wie Quelle, Crawler, Einlesezeitpunkt und, wenn verfügbar, auch Autor und Veröffentlichungsdatum angezeigt.

Die Durchschnittsberechnung ergibt sich, indem zu jedem Satz einzeln ein Durchschnitt berechnet wird und anschließend das arithmetische Mittel aller Satzbewertungen gebildet wird. Die Satzbewertungen werden berechnet, indem ebenfalls das arithmetische Mittel aller vorliegenden Bewertungen gebildet wird. Falls vorhanden, wird dabei auch der Wert für die Zuversichtlichkeit mit einbezogen, indem er mit dem jeweiligen Klassifikationswert multipliziert wird.

Bei der Oberfläche handelt es sich um eine klassische, serverseitig generierte. Dabei wird die Thymeleaf-Template-Engine verwendet, für die es eine einfache Integration in Spring gibt. Sie verfolgt den sogenannten „Natural Templating“-Ansatz, bei dem die HTML-Templates auch in ihrer Rohfassung korrekt von einem Browser dargestellt werden können. Dadurch ist allerdings das Ausgabeformat auf HTML-Dialekte beschränkt.

Um die Oberfläche mit geringem zusätzlichen Aufwand optisch ansprechend zu gestalten, kommt das Bootstrap-Framework zum Einsatz. Es wird von Twitter entwi-

ckelt und beinhaltet viele Möglichkeiten, Layouts zu gestalten und einfache Inhalte ansehnlich darzustellen[1].

3.7 Testing

Um eine korrekte Funktion des Newsboards zu gewährleisten und möglichst früh Fehler zu erkennen, wird zu großen Teilen testgetrieben entwickelt. Als Testing-Framework kommt Spock zum Einsatz. Es bietet gegenüber JUnit einige Vorteile, wie das Strukturieren von Testfällen in given/when/then-Blöcke, oder das, durch die Programmierung in Groovy ermöglichte, Testen privater Methoden außerhalb der entsprechenden Klassen.

Zusätzlich kommt das JaCoCo-Maven-Plugin zum Einsatz, um die Testabdeckung des Newsboards zu messen. In der finalen Fassung liegt die Zeilenabdeckung bei 92% und die Entscheidungsabdeckung bei 76% für das gesamte Projekt. Für die Model-Klassen besteht darüber hinaus vollständige Zeilenabdeckung, sowie 89% Entscheidungsabdeckung. Dabei ist zu beachten, dass JaCoCo einige Entscheidungspfade mit einberechnet, die sich gegenseitig ausschließen, und entsprechend nicht abgedeckt werden können. Ein Beispiel dafür ist die oder-verknüpfte Abfrage auf Null und eine Klassenzugehörigkeit, die mehrfach in `equals`-Methoden vorkommt.

4 Schluss

Abschließend soll nun das Projektergebnis zusammengefasst und ein Ausblick auf mögliche Weiterentwicklungen gegeben werden.

4.1 Zusammenfassung

Durch die Verwendung von Spring, Java und MySQL als Grundlage konnte eine modulare und flexible Software entwickelt werden. Über die REST-Schnittstelle kann das System mit unterschiedlichsten Systemen interagieren. Ein umfangreiches Testen hat sichergestellt, dass die Funktionen des Newsboards darüber hinaus robust und einsatzfähig sind.

Die zur Satzerkennung eingesetzte Bibliothek openNLP ist zwar einfach zu verwenden, die Präzision hat jedoch noch Verbesserungspotential. Zum jetzigen Zeitpunkt gibt es Probleme bei der Erkennung einzelner Punkte in einem Satz, beispielsweise bei einer Abkürzung.

Leider konnte bis zum Projektabschluss keine Volltextsuche implementiert werden. Lediglich ein Prototyp, basierend auf Apache Lucene, wurde implementiert. Für die Suche wurde ein Index im RAM angelegt, welcher sich nicht für die Verwendung in Umgebungen mit großen Datenmengen eignet.

Die entwickelte Oberfläche ist zwar ausreichend, um die Funktion des Newsboards zu demonstrieren, kann allerdings noch nicht als vollständig bezeichnet werden. So fehlt zum Beispiel noch eine Administrationsfunktion, um neue Crawler oder Classifier hinzuzufügen. Aufgrund des sauberen Aufbaus der Anwendung sollte dies jedoch einfach nachträglich durch Erweitern oder Ersetzen der Oberfläche möglich sein.

Außerdem können dank der REST-Schnittstelle externe Anwendungen, wie Single-Page-Apps oder mobile Anwendungen auf das Newsboard zugreifen und alternative Oberflächen bereitstellen.

4.2 Ausblick

Im jetzigen Stand bietet das Newsboard viele Ansatzpunkte zur Entwicklung neuer Funktionalitäten. Eine Erweiterung der REST-Schnittstelle mit zusätzlichen Optionen, und auch eine überarbeitete und umfangreichere Oberfläche, könnten in späteren Studentenprojekten entwickelt werden können.

Auch eine Verwendung von weiteren Datenbanksystemen ist möglich. Mit den fortschreitenden Entwicklungen im Bereich der NoSQL-Datenbanken ist es womöglich sinnvoll, in absehbarer Zeit eine Alternative zum relationalen Modell einzusetzen.

Die Erkennung von Sätzen könnte ebenfalls in zukünftigen Projekten verbessert werden. Neben Optimierungen des eingesetzten Modells ist auch eine komplette Eigenentwicklung zur Satzextraktion denkbar.

Darüber hinaus ist die Entwicklung einer Volltextsuche in späteren Projekten sinnvoll. Sollte das Newsboard tatsächlich über einen längeren Zeitraum eingesetzt werden, können die Datenmengen schnell ansteigen. Eine performante Volltextsuche in Kombination mit einer umfangreichen Oberfläche würde einen großen Mehrwert für die Verwendung des Newsboards bedeuten.

Literatur

[1] *Bootstrap (Framework)*. https://de.wikipedia.org/wiki/Bootstrap_(Framework), 2016.

[2] Fowler, Martin: *Inversion of Control Containers and the Dependency Injection pattern*. https://martinfowler.com/articles/injection.html, 2004.

[3] Sun Microsystems, Inc.: *Core J2EE Patterns - Data Access Object*. http://www.oracle.com/technetwork/java/dataaccessobject-138824.html, 2002.

[4] The Apache Software Foundation: *Apache OpenNLP Developer Documentation*. https://opennlp.apache.org/documentation/1.7.0/manual/opennlp.html#opennlp, 2014.

[5] Ullenboom, Christian: *Java ist auch eine Insel*. http://openbook.rheinwerk-verlag.de/javainsel/javainsel_16_003.html, 2011.