SENTIMENTS OF BUNDESTAG

Graph-basiertes Informationssystem zur Analyse sozialer Interaktion im Deutschen Bundestag

23. Januar 2021

Deutscher Bundestag

Betreut von Prof. Dr. Thomas Hoppe
Informationssysteme
M.Sc. Angewandte Informatik
Hochschule für Technik und Wirtschaft
Treskowallee 8, 10318 Berlin, Deutschland

Inhaltsverzeichnis

Abbildungsverzeichnis **v**

Tabellenverzeichnis **6**

1 Vorwort **7**
1.1 Einleitung 8
1.2 Aufbau der Lösung 9

2 Crawler **11**
2.1 Einleitung 12
2.2 Anforderungen und Rahmenbedingungen 13
2.2.1 Anforderungen an dem Crawler 13
2.2.2 Rahmenbedingungen 13
2.3 Lösung und Konzepte 15
2.3.1 Standard Aufbau eines Crawlers 15
2.3.2 Crawler Mechanismus 15
2.3.3 Daten-Parser und Database-Modell 16
2.3.4 Gesamter Aufbau der Lösung 16
2.4 Implementierung und Bereitstellung der Lösung 18
2.4.1 Implementierung des Crawlers 18
2.4.2 Bereitstellung der Lösung 18

3 Kommunikationsmodell **21**
3.1 Einleitung 22
3.2 Grundlagen 23
3.3 Anforderungsanalyse und Konzept 24
3.4 Implementierung 25
3.5 Zusammenfassung und Ausblick 26

4 Sentiment Analyse **27**
4.1 Einleitung 28

4.2 Grundlagen 29
4.3 Anforderungsanalyse und Konzept 30
4.4 Implementierung 31
4.5 Zusammenfassung und Ausblick 32

5 Analyse der Interaktion zwischen Abgeordneten 33
5.1 Einleitung 34
5.2 Grundlagen 35
5.3 Anforderungsanalyse und Konzept 36
5.4 Implementierung 37
5.5 Zusammenfassung und Ausblick 38

6 Analyse der Interaktion zwischen Fraktionen 39
6.1 Einleitung 40
6.2 Grundlagen 41
6.3 Anforderungsanalyse und Konzept 42
6.4 Implementierung 43
6.5 Zusammenfassung und Ausblick 44

7 Graphauswertung 45
7.1 Einleitung 46
7.2 Grundlagen 47
7.3 Anforderungsanalyse und Konzept 48
7.4 Implementierung 49
7.5 Zusammenfassung und Ausblick 50

8 Benutzeroberfläche 51
8.1 Einleitung 52
8.2 Grundlagen 53
8.3 Anforderungsanalyse und Konzept 54
8.4 Implementierung 55
8.5 Zusammenfassung und Ausblick 56

Literaturverzeichnis viii

Glossar ix

Abbildungsverzeichnis

1.1 Aufbau der Lösung 9

2.1 Crawler: Standard Aufbau 15
2.2 Funktionsprinzip des unseres Crawlers 15
2.3 Page-Crawler für eine URL 16
2.4 Crawler: Komponentendiagramm 17

Tabellenverzeichnis

2.1 Gruppe 1 (Crawler) - Arbeitsaufteilung 11

Kapitel 1

Vorwort

1.1 Einleitung

„Der rauer Ton im Bundestag“ [1]. So betitelte die ZDF-Redaktion ihren Beitrag vom 13.11.2019 zur aktuellen Stimmung im Deutschen Bundestag. Dabei ist die Redaktion der Ansicht, dass der Ton seit dem Beitritt der AFD ins Bundestag rauer und aggressiver geworden sei. Sie stützt sich dabei auf zahlreichen Reden, Reaktionen und Kommentare von sowohl Abgeordneten dieser Partei als auch die anderer Parteien auf, dessen Inhalten, vermittelten und erzeugten Stimmungen, sowie Reaktionen darauf von Stenographen in Plenarprotokollen bei jeder Sitzung aufgezeichnet und dokumentiert werden. Diese Plenarprotokolle werden als Xml oder Zip Dateien gemeinsam mit allen Stammdaten auf die Webseite des Bundestags (Open Data [2]) veröffentlicht. Die in den Protokollen verzeichneten Interaktionen bilden Netzen zwischen einzelnen Personen und und Gruppen von Personen (Fraktionen, Parteien, etc.), die als Kommunikationsgraphen (Kommunikationsmodell) betrachtet werden können. In diesem Zusammenhang, im Rahmen des Moduls Informationssysteme vom Master-Studiengang Angewandte Informatik (AI) an der HTW Berlin und auf Vorschlag vom Prof. Dr. Thomas Hoppe haben wir uns zur Untersuchung der Aussage der ZDF-Redaktion vorgenommen ein Graph basiertes Informationssystem zur Analyse sozialer Interaktion im Deutschen Bundestag zu bauen. Ausgehend von den Plenarprotokollen und dem daraus abgeleiteten Kommunikationsmodell baut das System ein Sentiments-Graph auf, der die Interaktionen innerhalb des Bundestags abbildet, die Stimmung zwischen Abgeordneten und verschiedenen Gruppen analysiert (bzw. mißt) und grafisch darstellt. Zur Lösung dieser Aufgabe wurde das daraus resultierende Projekt in Teilprojekten unterteilt, mit denen sich sieben Gruppen von Studenten auseinander gesetzt haben. Auf die Liste dieser Teilprojekte sowie auf deren konkreten Zielsetzung wird im nächsten Abschnitt genauer eingegangen.

1.2 Aufbau der Lösung

Das Projekt für die Entwicklung eines Graph basierten Informationssystem zur Analyse sozialer Interaktion im Deutschen Bundestag benannt „Sentiments Of Bundestag“ besteht aus sieben Teilprojekten, die von 3 bis 4 Studenten bearbeitet werden.

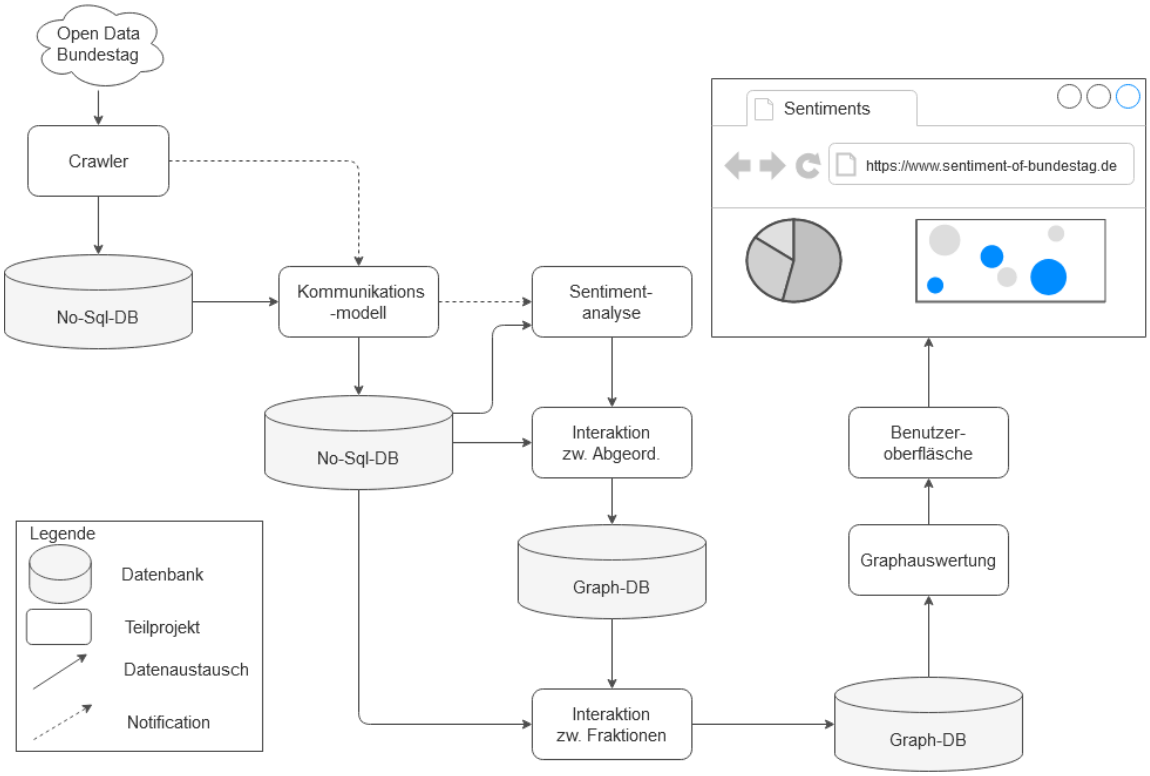

Abbildung 1.1: Aufbau der Lösung

Die auf der Abbildung 1.1 dargestellten Teilprojekten sind hier nun detaillierter aufgelistet:

- **Crawler**: Scannt regelmäßig die Open Data Webseite des Bundestags, sucht, parst und speichern neue Protokollen sowie Stammdaten der Abgeordneten in seiner No-Sql-Datenbank. Ziel ist es hier sicherzustellen, das die DB immer auf dem neusten Stand bleibt
- **Kommunikationsmodell**: Analysiert und erstellt aus den Protokollen von Gruppe 1 ein Kommunikationsmodell aus den Austauschen im Bundestag

- **Sentiment-Analyse**: Sentiment Analysis (Stimmungsanalyse). Ziele sind hier die Identifikation der Stimmung in den Äußerungen der Abgeordneten und die Verrechnung der Stimmung einer Äußerung zu positiver/negativer Bewertung der Beziehung
- **Interaktion zwischen Abgeordneten**: Aus dem Kommunikationsmodell von Gruppe 2 und die Stimmungsanalyse von Gruppe 3 werden hier Interaktionen zwischen einzelnen Personen (Abgeordneten, Präsident, Gäste, etc.) identifiziert. Erstellt wird hier ein gewichteter Sentiment-Graph zwischen Abgeordneten mit positiven/negativen Gewichtungen
- **Interaktion zwischen Fraktionen**: Aus dem Kommunikationsmodell von Gruppe 2 und den Sentiment-Graph zwischen Personen von Gruppe 4 werden hier Interaktionen zwischen Gruppen von Personen analysiert und in einen Sentiment-Graph zwischen Parteien. Der besteht aus einer Aggregation der Abgeordnetensentiments zu gewichteten Sentiment-Graph der Parteien (Fraktionen, Gruppen, etc.)
- **Graphauswertung**: Die Sentiment-Graphen von den Gruppen 4 und 5 werden hier anhand verschiedener Auswertungsmethoden analysiert und die Ergebnisse davon die nächste Gruppe (Benutzeroberfläche) zur Verfügung stellt.
- **Benutzeroberfläche**: Ziel ist hier die Realisierung einer interaktiven Benutzeroberfläche zur Darstellung der ausgewerteten Ergebnisse

Die einzelnen Teilprojekten werden in den nächsten Kapiteln in dieser Reihenfolge dokumentiert.

Kapitel 2

Crawler

Tabelle 2.1: Gruppe 1 (Crawler) - Arbeitsaufteilung

Aufgabe	Gruppenmietglieder	Anteil
Crawl-Manager	Boris Foko Kouti	ff
Crawl-Utilities	Marlon Daniel Kohlberger	ff
	Boris Foko Kouti	ff
DB-Manager	Marlon Daniel Kohlberger	ff
	Boris Foko Kouti	ff
Dokumenation	Arnauld Feussi	ff
	Marlon Daniel Kohlberger	ff
	Boris Foko Kouti	ff
Bereitstellung der Lösung	Boris Foko Kouti	ff

2.1 Einleitung

Die Umsetzung eines Graph-basierten Informationssystems zur Analyse sozialer Interaktion im Deutschen Bundestag (Sentiments-of-Bundestag) setzt voraus, dass aktuelle Informationen und Berichte über die Debatten im Bundestag zu jeder Zeit zur Verfügung stehen. Diese Informationen sowie Stammdaten von Abgeordneten und anderen Sitzungsteilnehmern werden regelmäßig auf die Seite des Bundestags veröffentlicht und sind für alle interessierten Anwender frei zugänglich [2]. Das Durchsuchen dieser Webseite, das Vergleichen, Herunterladen und Parsen der Protokollen sowie auch der Stammdaten lässt sich anhand eines Computerprogramms automatisieren. Ein solches Programm wird als Crawler bezeichnet. Im Allgemein besteht die Aufgabe eines Crawlers darin sich wiederholende Operationen (z. B. den Download oder die Indexierung einer Webseite) soweit wie möglich zu systematisieren und ohne menschlicher Eingriff regelmäßig laufen zu lassen. Ein Crawler funktioniert in der Regel wie ein Bot und kann sowohl zur Extraktion bestimmter Informationen aus eine oder mehreren Webseiten als auch zur kompletten ggf. leicht abgewandelten Replikation des Inhalts einer Webseite auf einer anderen Seite. Im zweiten Fall geht es um ein sogenannter Scraper. Zur Sammlung der vom Graph-basierten Informationssystems zur Analyse sozialer Interaktion benötigten Daten wird ein Crawler eingesetzt, der wie bei einem Scraper eine bestimmte Seite in einer festgelegten Frequenz herunterlädt, daraus Links für spezifische Datentypen extrahiert und abhängig davon, ob diese neu oder nicht sind, die entsprechenden Dateien downloadet, parst und in einer Datenbank speichern. Bevor dieser Crawler implementiert wird, wird es zunächst wichtig genaue technische Anforderungen, sowie eventuellen Schwierigkeiten (auch speziell für die Bundestag-Seite) zu identifizieren. Daraus wird ein Konzept zur Lösung dieser Aufgabe erarbeitet. Diese beide Punkte sowie die Implementierung des vorgeschlagenen Konzepts und die Bereitstellung der vollumfänglichen Lösung werden in den nächsten Abschnitten eingegangen.

2.2 Anforderungen und Rahmenbedingungen

Der zu entwickelnden Crawler soll in bestimmten Rahmenbedingungen einige Anforderungen erfüllen. Diese sowie die technischen und organisatorischen Rahmenbedingungen dazu werden in diesem Abschnitt aufgelistet.

2.2.1 Anforderungen an dem Crawler

Die Anforderungen an dem Crawler sind in [3] zusammengefasst und sehen vor, dass der Crawler kontinuierlich laufen sollte und dabei nach folgenden Dateien suchen:

- **Protokolle**: Ein Protokoll ist eine XML-Datei, die den gesamten Ablauf einer Plenarsitzung (Inhaltsverzeichnis, Sitzungsverlauf mit Sitzungsbeginn, Tagesordnungspunkte, Sitzungsende, Anlagen, Rednerliste und gewisse Konfigs) beinhaltet. Da sich die Formatierung der Protokollen in den Jahren stets verbessert hat, soll hier auf die Legislaturperioden geachtet werden:
  - 19. Legislaturperiode: diese sind sehr gut strukturiert und leicht zu parsen
  - 18. Legislaturperiode: schlechter formatiert als die 19. Legislaturperiode. Hierfür wird vom Prof. Dr. Thomas Hoppe eine mit besser formatierte Version zur Verfügung gestellt
- **Stammdaten der Abgeordneten**: Ist eine XML-Datei, die Informationen über Abgeordneten seit 1949 sammelt
- *Optional* **Namentliche Abstimmungen**: Liste aller namentlichen Abstimmungen im Bundestag seit dem 18.12.2009 in PDF-Format und seit dem 18.10.2012 auch in XLSX-Format [4]
- Es muss sichergestellt sein, dass alle Daten immer auf dem neusten Stand sind. Dafür soll die Frequenz des Crawlers sich an dem Sitzungskalender des Bundestags [5] orientieren

2.2.2 Rahmenbedingungen

Organisatorische Rahmenbedingungen

Für die Planung des gesamten Projekts ist in einem zwei Wochen-Takt ein Plenum für den Brainstorming und eventuelle Teams-übergreifende Abstimmungen festgelegt worden. Im Team-Crawler (Gruppe 1) ist folgende Planung abgemacht worden:

- Ziel bis zum 13.11.2020: Testdaten für andere Gruppen, erster lauffähiger Prototyp
- Ziel bis zum 27.11.2020: Vorläufiger Release und Integration-Test mit anderen Gruppen (Kommunikationsmodell)
- Ziel bis zum 04.12.2020: finale Version und Anfang der Dokumentation
- Ziel bis zum 25.02.2021: Fertigstellung der Dokumentation und Abgabe

Neben den organisatorischen Rahmenbedingungen sind technische Rahmenbedingungen zu betrachten bzw. sind Fehler (Probleme) zu vermeiden (zu lösen).

Technische Rahmenbedingungen: Probleme beim Crawlen

- Sperrung durch Server-Administrator (Server-Regel). Es soll vermieden werden, dass der ausführende Rechner wegen zu häufige Abfragen vom Server-Admin gesperrt wird und nur noch ein 500-Fehlercode erhält
- Ajax basierende Inhalte. Die Open Data Webseite verwendet Ajax für die Bereitstellung der Daten und lädt diese im Hintergrund. Ein einfacher Download der Seite genug da nicht. Außerdem nutzen die meisten Links, hinter denen Dateien stehen, Javascript (es ist z. B. der Fall bei Slides und verborgenen Regionen). Dies muss entsprechend gehandelt werden
- Mongo-DB-Konfiguration und Zurverfügungstellung der Daten für die Gruppe Kommunikationsmodell

2.3 Lösung und Konzepte

2.3.1 Standard Aufbau eines Crawlers

Ein Crawler besteht in der Regel aus zwei wichtigen Teilen: ein Downloader und ein Scheduler. Der Downloader ist mit dem Webserver verbunden und lädt die Webseite(n) sowie alle erwünschten Dateien herunter. Der Scheduler ist, wie der Name schon sagt, für die Planung zuständig und erteilt dem Downloader URLs nach internen Priorisierung-Mechanismen (als Liste FIFO und als Graph mit DFS und BFS [6]–[7]).

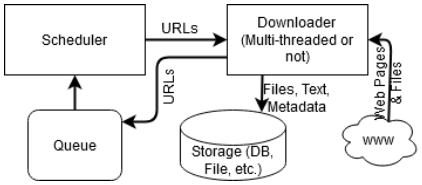

Abbildung 2.1: Crawler: Standard Aufbau

2.3.2 Crawler Mechanismus

Für unseren speziellen Fall, wird die Aufgaben von dem Scheduler vom Crawl-Manager übernommen, der sowohl für die Planung als auch für die Steuerung (Start, Pause, Stopp, Zustand) laufender Crawl-Aufgaben zuständig ist. Der Crawl-Manager erzeugt für jede URL ein Thread (Page-Crawler), dessen Aufgabe darin besteht die Seite herunterzuladen, mit seinem Page-Analyser nach relevanten Inhalten in der Seite zu suchen (URLs, Dateien, Action-Links: Javascript, etc.) und bei Bedarf weitere parallele Sub-Threads für den Download, das Parsen und die Speicherung der relevanten Dateiinhalte (Protokollen, Stammdaten, weitere Metadaten). Dieser Mechanismus wird durch die Abb. 2.2 und die Abb. 2.3 veranschaulicht.

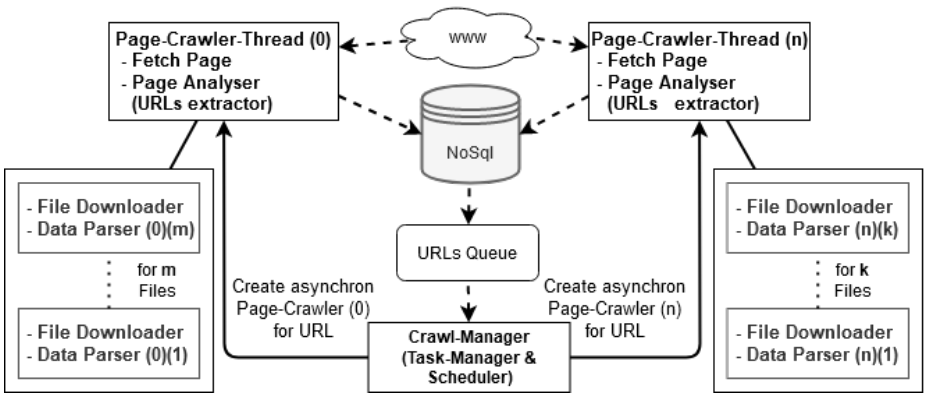

Abbildung 2.2: Funktionsprinzip des unseres Crawlers

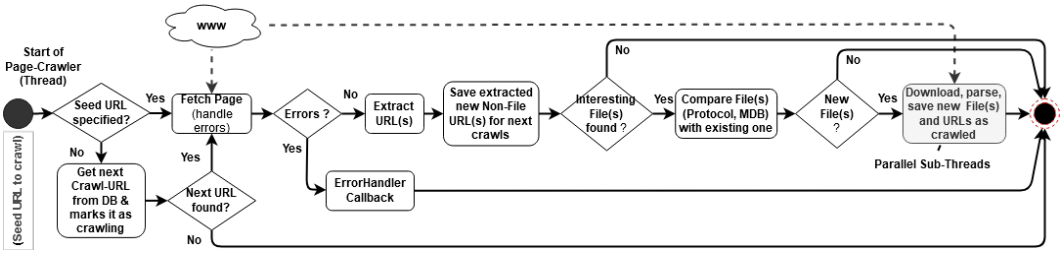

Abbildung 2.3: Page-Crawler für eine URL

Ein Crawl auf einer Seite (durch den Page-Crawl) kann in einer durch den Scheduler (Crawl-Manager) regelmäßig gemacht werden, in dem dieser nach einer bestimmten Zeit einen neuen Page-Crawl-Thread startet. Die Frequenz der Generierung dieses Threads orientiert sich an dem Sitzungskalender des Bundestags [5]. Dieser Kalender sieht vor, dass Sitzungen an bestimmten Wochentagen zwei bis vier Wochen pro Monat (abgesehen von August: Ferien) stattfinden. Darauf basierend ist die Entscheidung getroffen worden, die Frequenz auf einmal pro Tag von Montag bis Freitag (mit der Möglichkeit bei Bedarf den Crawler in der Ferienzeit zu stoppen) festzulegen. So lässt sich eine Sperrung wegen zu häufiger Abfragen verhindern. Bei Manchen Server (durch Regeln oder Server-Admin) kann aber ein sich wiederholender Abfrage-Muster (wie eine Abfrage jeden Tag um dieselbe Uhrzeit) auch zu einer Sperrung der IP (Rechner) führen. Aus diesem Grund wird zusätzlich zur Frequenz ein Zufallsfaktor verwendet. Ein Page-Crawl-Thread wird zwar von Montag bis Freitag um 23Uhr durch den Crawl-Manager gestartet, allerdings wartet dieser Thread ein zufällige Zeit t (mit $10 \leq t \leq 7200$) bis er den tatsächlichen Crawl durchführt. Während und nach dem Crawl-Prozess werden Daten (und Dateien) manipuliert, analysiert und gespeichert. Diese Vorgänge sowie die dafür verwendeten Datenstrukturen werden im nächsten Abschnitt beleuchtet.

2.3.3 Daten-Parser und Database-Modell

- Parser für die 19. Legislaturperiode
- Parser für die 18. Legislaturperiode
- ER-Diagramm und Beschreibung des DB-Modells

2.3.4 Gesamter Aufbau der Lösung

Die gesamte Lösung besteht aus vier Komponenten und einer Datenbank, wie auf die Abb. 2.4 dargestellt.

- **Crawl-Manager**: Taskmanager und Scheduler
- **Crawl-Utilities**: Liefert die für den Crawl-Prozess nötigen Funktionalitäten (Page-Fetcher, Page-Analyser, Downloader, Data-Parser)
- **DB-Manager**: Verwaltet den Zugriff auf die Datenbank
- **Rest-ServiceProvider**: Rest-API für die Steuerung des Crawl-Manager und den eingeschränkten Zugriff auf die DB-Daten
- **Datenbank**: NoSql (MongoDB) Datenbank für die Sicherung der Daten

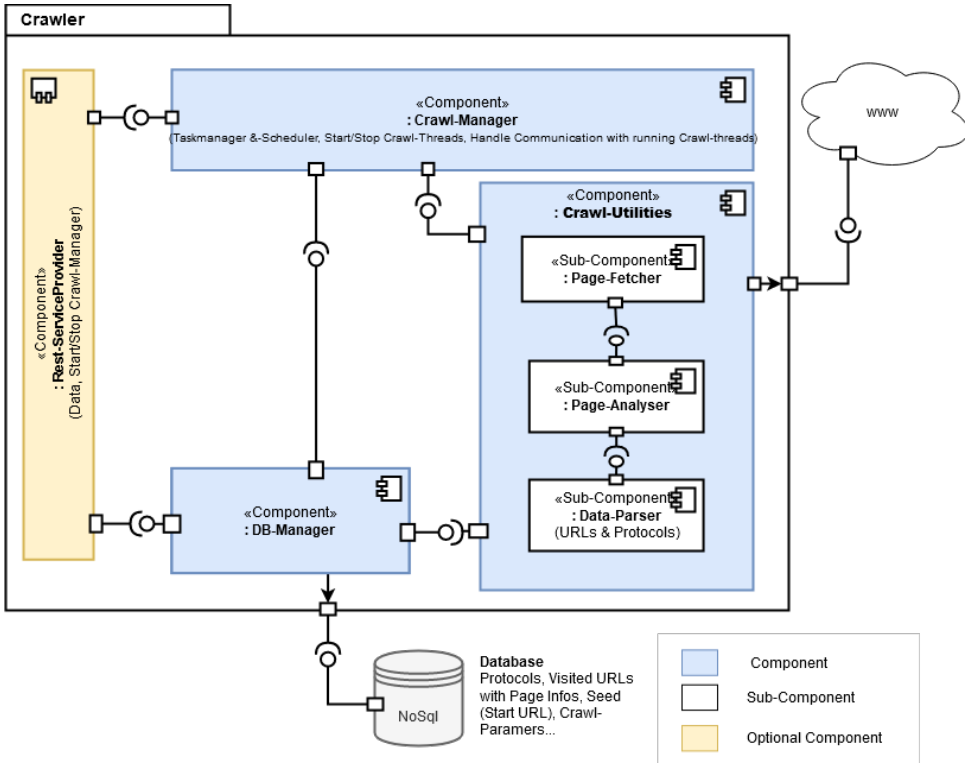

Abbildung 2.4: Crawler: Komponentendiagramm

Am Ende eines Crawl-Prozess, bei dem neue Protokolle oder Stammdaten geladen worden sind, wird die Gruppe-Kommunikationsmodell die Liste aller Protokollen sowie Stammdaten über deren Rest-API [8] als Json-Datei mitgeteilt. Die Gruppe-Kommunikationsmodell kann dann asynchron die entsprechenden Dateien aus unseren Datenbank holen.

2.4 Implementierung und Bereitstellung der Lösung

2.4.1 Implementierung des Crawlers

Für die Umsetzung des Crawler wurde die Entscheidung getroffen eine Web-Anwendung (Rest-Schnittstelle, Crawl-Prozess und Scheduler) zu bauen. Dafür wurde Spring Boot [9] als Java-basierender Framework für die Entwicklung von Web-Anwendungen gewählt. Damit lassen sich Konfigurationen (für Rest-APIs, Datenbanktreiber, etc.) mit Hilfe des Spring-Initializers vereinfachen. Außerdem verfügt Spring Boot über einen eigenen Task-Scheduler der für den Crawl-Manager verwendet werden kann. Die finale Lösung beinhaltet vier Packages: **crawler** mit den Funktionalitäten für den Crawl-Prozess und das Parsen, **models** und **repositories** für den DB-Manager und **web** mit den Controllers für die Rest-Schnittstelle und den Services für den Crawl-Manager. Das gesamte Projekt-Verzeichnis inkl. Code ist auf dem Github-Repository *Sentiments-of-Bundestag/Crawler* [10] zu finden. Dort kommt unter anderem für die Ausführung Javascript-basierende Inhalte für den Download bestimmter Teile (Protokoll-Slides) der Webseite die **HtmlUnit**-Bibliothek [11] zum Einsatz (siehe 2.2.2).

2.4.2 Bereitstellung der Lösung

Der Crawler wurde auf dem Virtual-Server der Gruppe 1 (141.45.146.161) an der HTW (nur im HTW-Netz sichtbar) als Ubuntu-Service bereitgestellt. Der Crawler-Service liefert eine Rest-Schnittstelle, die jedoch nur auf dem infosys1 Rechner aus Sicherheitsgründen zugänglich ist. Die Konfiguration des Crawler-Services, die unter */etc/systemd/system/crawler.service* angelegt wird, sieht wie folgt aus (Start des Crawlers: systemctl enable crawler):

```
1  [Unit]
2  Description=Crawler App
3  ...
4  User=local
5  [Service]
6  ExecStart=/usr/bin/java -jar ../Crawler-1.0-SNAPSHOT.jar
7  Restart=always
8  SyslogIdentifier=crawler
9  ...
10 [Install]
11 WantedBy=multi-user.target
```

Da für den Crawler eine MongoDB benötigt wird, soll diese auch bei der Bereitstellung konfiguriert werden. Dabei muss sichergestellt werden, dass

der zugriff auf die Datenbank nur mit den entsprechenden Zugangsdaten erfolgt. Also soll nach dem Hinzufügen eines neuen Admin-User die Security-Konfiguration von mongod unter *sudo nano /etc/mongod.conf* auf *authorization: enabled* umgestellt werden. Da die MongoDB von der Gruppe 2 (Kommunikationsmodell) verwendet wird, sollen dafür entsprechend die Firewall-Regeln angepasst werden. Der nachfolgende Auszug aus unserer Konfiguration-Datei (/root/Firewall.sh) zeigt wie dies beispielhaft erfolgen sollte.

```
...
#
# Erlaube zugang im 145.45.x.x port 27017 MongoDB
#
iptables -A INPUT -s 141.45.0.0/16 -p tcp -m tcp --dport
    27017 -j ACCEPT
...
```

Der bereitgestellten Crawler basiert auf Java 11 und ist somit eine Plattform-unabhängige Lösung, die auf allen Betriebssystemen angeboten werden kann. Die Steuerung des Crawlers erfolgt ausschließlich über die Rest-Schnittstelle mit folgenden Abfragen (hier mit curl in der Shell-Konsole):

```
# opendata: https://www.bundestag.de/services/opendata
# Start a default crawl to opendata
$ curl -X POST http://localhost:8080/task/default
# Start the default cron process to opendata: Mon - Fri, 23
$ curl -X POST http://localhost:8080/task/cron
# List all running tasks
$ curl -X GET http://localhost:8080/tasks
# Cancel planed task
$ curl -X POST http://localhost:8080/task/cancel/{task_id}
# Cancel all planed tasks
$ curl -X POST http://localhost:8080/tasks/cancel
```

Der Crawler auf dem infosys1-Server (infosys1.f4.htw-berlin.de) läuft seit dem 27.11.2020 mit ein Paar Unterbrechungen für Aktualisierungen problemlos und konnte zum Zeitpunkt der Verfassung dieser Dokumentation schon 204 Protokollen herunterladen, parsen und in der MongoDB speichern, Stammdaten von 4.086 Abgeordneten sammeln und dabei mehr als 500 Urls durchsuchen. Was den Datenaustausch mit anderen Gruppen angeht, konnte bis jetzt erfolgreich durch Gruppe 2 auf unsere MongoDB zugegriffen werden und Mitteilungen von neu gecrawlten Daten konnten fehlerfrei zugestellt werden.
Aus den vom Crawler gesammelten Daten wird nun durch Gruppe 2 ein Kommunikationsmodell gebaut, das von den anderen Gruppen für weitere Verarbeitungen und Analysen verwendet wird. Der Aufbau dieses Kommunikationsmodells wird nun von Gruppe 2 im nächsten Kapitel aufgegriffen.

Kapitel 3

Kommunikationsmodell

3.1 Einleitung

3.2 Grundlagen

3.3 Anforderungsanalyse und Konzept

3.4 Implementierung

3.5 Zusammenfassung und Ausblick

Kapitel 4

Sentiment Analyse

4.1 Einleitung

4.2 Grundlagen

4.3 Anforderungsanalyse und Konzept

4.4 Implementierung

4.5 Zusammenfassung und Ausblick

Kapitel 5

Analyse der Interaktion zwischen Abgeordneten

5.1 Einleitung

5.2 Grundlagen

5.3 Anforderungsanalyse und Konzept

5.4 Implementierung

5.5 Zusammenfassung und Ausblick

Kapitel 6

Analyse der Interaktion zwischen Fraktionen

6.1 Einleitung

6.2 Grundlagen

6.3 Anforderungsanalyse und Konzept

6.4 Implementierung

6.5 Zusammenfassung und Ausblick

Kapitel 7

Graphauswertung

7.1 Einleitung

7.2 Grundlagen

7.3 Anforderungsanalyse und Konzept

7.4 Implementierung

7.5 Zusammenfassung und Ausblick

Kapitel 8

Benutzeroberfläche

8.1 Einleitung

8.2 Grundlagen

8.3 Anforderungsanalyse und Konzept

8.4 Implementierung

8.5 Zusammenfassung und Ausblick

Literaturverzeichnis

[1] *Der raue Ton im Bundestag*. besucht am 10.01.2021, um 11Uhr. URL: https://www.zdf.de/nachrichten/heute-sendungen/videos/rauer-ton-im-bundestag-100.html#xtor=CS5-95.

[2] *Open Data*. besucht am 10.01.2021, um 11Uhr. URL: https://www.bundestag.de/services/opendata.

[3] Thomas Hoppe. „Informationssysteme: Übung“. In: *Informationssysteme*. 2020, S. 10–12. URL: https://moodle.htw-berlin.de/pluginfile.php/1015183/mod_resource/content/0/Planung%20%C3%9Cbung%20Teil%202.pdf.

[4] *Parlament*. besucht am 10.01.2021, um 11Uhr. URL: https://www.bundestag.de/parlament/plenum/abstimmung/liste.

[5] *Sitzungskalender*. besucht am 10.01.2021, um 11Uhr. URL: https://www.bundestag.de/parlament/plenum/sitzungskalender/sitzungskalender-196314.

[6] Thomas H. Cormen u. a. *Introduction to Algorithms, Third Edition*. 3rd. The MIT Press, 2009, S. 540–549. ISBN: 0262033844.

[7] Donald Knuth. „The Art of Computer Programming“. In: *The Art of Computer Programming: Second Edition*. Bd. 3. 2. Boston: Addison-Wesley Professional, 1998. Kap. Sorting and Searching, S. 73–168, 180–197, 392–478. ISBN: 978-0-201-89685-5.

[8] *Dokumentation CME*. besucht am 10.01.2021, um 11Uhr. URL: infosys2.f4.htw-berlin.de:9001/cme/doc/docs#/data/get_session_cme_data_session__session_id__get.

[9] *Spring Boot 2.4.2*. besucht am 10.01.2021, um 11Uhr. URL: https://spring.io/projects/spring-boot.

[10] *Crawler*. besucht am 10.01.2021, um 11Uhr. URL: https://github.com/Sentiments-of-Bundestag/Crawler.

[11] *HtmlUnit*. besucht am 10.01.2021, um 11Uhr. URL: `https://htmlunit.sourceforge.io/`.