Bachelorarbeit am Institut für Informatik der Freien Universität Berlin

Human-Centered Computing (HCC)

Weiterentwicklung einer Extension zur Anwendung des semantischen Webs

Daniel Seidel

Betreuerin und Erstgutachterin: Prof. Dr. C. Müller-Birn

Zweitgutachter: Prof. Dr. L. Prechelt

Berlin, 02. Mai 2018

Eidesstattliche Erklärung

Ich versichere hiermit an Eides Statt, dass diese Arbeit von niemand anderem als meiner Person verfasst worden ist. Alle verwendeten Hilfsmittel wie Berichte, Bücher, Internetseiten oder ähnliches sind im Literaturverzeichnis angegeben, Zitate aus fremden Arbeiten sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungskommission vorgelegt und auch nicht veröffentlicht.

Berlin, den 9. September 2019

Daniel Seidel

Danksagung
Mein Dank gilt Prof. Dr. C. Müller-Birn für die Betreuung dieser Arbeit. Erwähnen möchte ich auch die freundlichen Personen welche mich über den Hervorheben möchte ich die folgenden fleißigen Testpersonen, Gesprächs- partner und Gesprächspartnerinnen, die GoHyper ausprobiert und mir da- rüber ergiebige Rückmeldungen gegeben haben: Henning Möldner, Alex- ander Naydenov, Edith Pahlke-Gebske, Sonja Peschutter, Sabine Redlich, Elisabeth Sassi, Nico Schlömer, Marco Seelbinder, Daniel Seidel, Lyudmila Vaseva und Falko E. Vedder. Ihre Anregungen haben die Realisierung des Projekts vorangetrieben. Hardyna Vedder hat mir durch inspirierende Gespräche sowie ihre Testberichte bei der Entwicklung geholfen. Ich möchte Andrne Gaul für sein überaus hilfreiches Feedback und seine aufmunternde Unterstützung danken.

Zusammenfassung

In dieser Arbeit wird, aufbauend auf der Bachelorarbeit von Jennifer Gebske, die weitere Entwicklung von GoHyper dokumentiert. GoHyper ist eine Chrome Extension, welche ursprünglich als persönliches Recherchewerkzeug fungieren sollte. Dabei wird GoHyper mit einem aktuelleren Framework sowie einer Anbindung zum Backend von Neonion neu realisiert.
Neonion ist eine Webseite, welche im Rahmen eines Forschungsprojektes von Prof. Dr. C. Müller-Birn zur Untersuchung des Semantic Webs entwickelt wurde. Durch diese Weiterentwicklung von GoHyper wird aus dem ursprünglichen Recherchewerkzeug eine erste Version eines User Interfaces zur Verwendung des Semantic Webs.
Mit GoHyper können Annotationen, Kommentare und Highlights gesetzt, gelöscht und manipuliert werden. Die REST-basierte API, mit welcher GoHyper mit dem Backend von Neonion kommuniziert, wird mittels Swagger entwickelt und ist neben der weiteren Entwicklung von GoHyper ebenfalls in einem GIT Repository nachzuverfolgen.
Der Fokus der Arbeit liegt in der funktionsorientierten Implementierung von GoHyper, welches ursprünglich mit der ersten Version von AngularJS verwirklicht wurde und nun mittels React und Redux ein Update erhalten soll. Dabei sollen die Annotationen im Backend von Neonion hinterlegt werden.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

1.1 Thema und Kontext

Durch die starke Vernetzung der heutigen Zeit kommen immer mehr Informationen in kürzerer Zeit zusammen. Durch Enzyklopädien wie Wikipedia wird dieses Wissen in eingeschränktem Rahmen aufbereitet und gebündelt. Dennoch finden sich viele weitere Informationen und neu erforschtes Wissen im World Wide Web (Web). Dieses Wissen unabhängig voneinander zu bündeln und aufzuarbeiten, ist nicht nur für die Wissenschaft von Vorteil, sondern wird auch in der Lehre und im allgemeinen Interesse immer wichtiger.
Schüler und Studenten können sich so ein fachspezisches Informationsnetz erstellen und Forscher ihre Paper durch Verlinkungen ergänzen, aufwerten, durch das Zusammentragen neue Forschungsergebnisse generieren oder auch widerlegen. Um diesen Bedürfnissen gerecht zu werden, ist es wichtig, dass die Informationen nicht auf einer Quelle basieren, sondern nach Möglichkeit das gesamte Web als Quelle dienen kann. Eine gute und schnelle Übersicht der Informationen ist dabei existenziell.
Zu diesem Zweck wurde eine Extension für den Internetbrowser Chrome entwickelt. Die Extension läuft unter dem Namen GoHyper[1] und ist ein Recherchewerkzeug mit dem Textstellen, sogenannte Quotes, erstellt und mit Hyperlinks zu anderen Webseiten versehen werden können. Die Version 1.0 von GoHyper wurde von Jennifer Gebske entwickelt und im Rahmen Ihrer Bachelorarbeit[2] dokumentiert [Geb]. Der Funktionsumfang erlaubt es, besagte Textstellen mit Highlights, Kommentaren, Hyperlinks (Uniform Resource Locator (URL)) und Tags zu versehen. Dabei speichert die Extension die Daten derzeit lokal. Zudem ist eine Synchronisation der Daten während der ersten Entwicklung von GoHyper nicht vorgesehen gewesen. Der Umstand, dass die Daten lokal hinterlegt sind, soll nun geändert werden. Dabei verschwinden die Hyperlinks und die Tags und werden durch Annotationen [MBKB+15] ersetzt. Diese Annotationen sollen mittels einer clientseitigen Application programming interface (API), welche im Rahmen dieser Arbeit zu entwickeln ist, im Backend von einer Webseite names Neonion hinterlegt werden. Diese Webseite ist Teil eines Forschungsprojekt von Prof. Dr. C. Müller-Birn[3]. In diesem Forschungprojekt wird sich näher mit den Problemen rund um das Thema des Semantic Web befasst. Im Rahmen dessen wurde die genannte Webseite entwickelt, in welcher Forscher wissenschaftliche Paper hochladen und diese mit Kommenta-

[1] `https://chrome.google.com/webstore/detail/gohyper/bemkdkdpdcepkncpclmcphgaddaameff`
[2] `https://www.inf.fu-berlin.de/inst/ag-se/theses/Gebske16-GoHyper.pdf`
[3] `https://www.clmb.de/`

ren, Highlights und Annotationen versehen können. Um den Funktionsumfang nicht mehr nur auf eine einzelne Webseite zu beschränken, sondern im gesamten Web verwenden zu können, wird GoHyper nun entsprechend manipuliert und angepasst. Die Weiterentwicklung von GoHyper wird funktionsorientiert entwickelt. Der Quellcode der clientseitigen API [Cli] ist auf GitHub, ebenso wie GoHyper[Bac] unter der Lizenz GNU GPLv3 zu finden.

1.2 Zielsetzung dieser Arbeit

Zum Ende der Arbeit sollen zwei verschiedene Dinge erreicht werden. Der ersten Aspekt beleuchtet die clientseitige API entwickelt, welche mit dem Backend von Neonion (BvN) zusammenarbeitet. Da die serverseitige API sich während des Schreibes dieser Arbeit noch im Aufbau befindet, wird an einigen Stellen mit Stubs gearbeitet werden. Der Fokus wird dabei auf den Annotationen liegen. Im Laufe der Arbeit werden die Quotes, welche derzeit Highlights, Kommentare, Hyperlinks und Tags in sich vereinen, aufgesplittet. Letztlich sollen drei differenzierte Funktionen für den User bereitstehen:

- das Erzeugen und Löschen von Highlights,
- das Setzten, Manipulieren und Löschen von Kommentaren und
- die Erstellung, Bearbeitung und Löschung von Annotationen.

Eine genaue Beschreibung der Aufsplittung folgt in Kapitel 3. Die Kommentare und die Highlights werden hierbei in der neuen Version von GoHyper übernommen, erhalten jedoch ihre Funktionen aus der Version 1.0 von GoHyper bei und werden nicht mittels der API im BvN hinterlegt.
Der zweite Aspekt betrifft die Entwicklung von GoHyper selber. GoHyper ist eine Erweiterung, welche unter dem Internetbrowser Chrome läuft. Die Erweiterungen unter Chrome heißen Extensions, unter Firefox Add-ons, unter dem Internetbrowser Edge schlicht Erweiterungen und unter anderen Internetbrowsern werden sie abermals anders betitelt. Da GoHyper speziell für Chrome entwickelt wurde, wird im folgenden nur noch von Extensions berichten. Mit Extensions sind Programme gemeint, welche bei der Vanilla-Version eines Programmes nicht mit installiert werden, vom User jedoch nachträglich für neue Funktionen, Grafiken oder zusätzliche Daten hinzugefügt werden können. Die Vanilla-Version eines Programmes ist die Originalversion ohne jegliche Modifikationen oder zusätzliche Erweiterungen.
GoHyper verwendet in der Version 1.0 das Framework AngularJS mit der Version 1.6 [Anga]. Inzwischen existiert eine neuere Version 5.2.10 von AngularJS [Angb], welche nicht kompatibel mit der früheren Version ist. Ein einfaches Update von AngularJS 1.6 auf die Version 5.2.10 ist nicht möglich. Somit ist das zweite Ziel dieser Arbeit die Dokumentation auf ein aktuelles Framework. Da es nicht Aufgabe dieser Arbeit ist, die entsprechenden Annotationen, Kommentare oder Highlights auf stetig manipulierten Internetseiten wieder zu finden,

werden zum Testen von GoHyper speziell die Internetseiten von Open Access Publisher, wie z. B. Elsevier [els] verwendet. Sobald ein Wissenschaftler seine Paper veröffentlicht hat, kann an dieser Stelle davon ausgegangen werden, dass die Seite nicht mehr manipuliert wird.

1.3 Aufbau

Im zweiten Kapitel werden die Frameworks beschrieben, warum diese verwendet werden und was deren genaue Aufgabe ist. Dabei werden mehrere Frameworks gegenübergestellt und deren Vor- und Nachteile herausgearbeitet.
Das dritte Kapitel befasst sich mit den Unterschieden zwischen den beiden GoHyper Versionen 1.0 und 2.0. In diesem Kapitel werden auch die Stärken und Schwächen der jeweiligen Versionen genannt. Anschließend werden auch bereits existierenden Extensions mit der Version 2.0 verglichen.
Im vierten Kapitel wird auf die eigentliche Umsetzung eingegangen. Wie GoHyper in der Version 2.0 verwirklicht wird, wo die Unterschiede im Detail zu Version 1.0 sind, ob die Unterschiede wirklich zu einander in Relation gesetzt werden können und welche Aspekte aus der früheren Version übernommen werden. Auch wird hier auf die Entwicklung der API eingegangen, wo diese im Workflow von GoHyper ihren Platz hat und wie mit den Responses umgegangen wird.
Das fünfte Kapitel rundet die Arbeit mit einem entsprechenden Fazit ab und stellt mögliche Weiterentwicklungen vor.

2 Verwendete Hilfsmittel

2.1 API Unterstützung

Da beim BvN mit der Entwicklung einer API begonnen wurde, stand bereits im Vorfeld dieser Arbeit fest, dass clientseitig ebenfalls eine Representational State Transfer (REST) API entwickelt werden muss. Um eine besonders effiziente Entwicklung einer REST API zu gewährleisten, wurden sich RAML[RAM], API Blueprint[Blu] und Swagger[Swa] näher angeschaut. Alle drei sind Beschreibungssprachen für eine REST basierte API.
RAML (RESTful API Modeling Language) ist eine YAML[YAM] basierte Beschreibungssprache für REST APIs. Vorteile von RAML sind, dass es durch die Verwendung von YAML eine gute Lesbarkeit besitzt und einen besonderen Fokus auf die Zusammenarbeit in einem Team legt. Durch das gute Interface ist es leicht zu erlernen. Leider existiert für neuere Versionen ein schlechter Support. Da außerdem im Rahmen dieser Arbeit nicht in einem Team gearbeitet wird, wurde sich gegen die Nutzung von RAML entschieden.
API Blueprint ist ebenfalls leicht zu erlernen und simpel zu schreiben. Leider hat es, im Vergleich zu den beiden anderen Optionen, sonst keine herausragenden Eigeschaften. Die Vorteile, im Vergleich zu den anderen Beschreibungssprachen, fallen ziemlich gering aus, weswegen sich gegen API Blueprint entschieden wurde. Die letzte Option ist Swagger, mit den Vorteilen einer große Community und der umfangreichenden Dokumentation. Es gibt eine starke Frameworkunterstützung und bestitzt eine sehr große Sprachenunterstützung für Open Source Frameworks.Der Nachteil ist, dass eine Dokumentation der API nicht ohne weiteres exportiert werden kann.
Das Fazit ist, das Swagger trotz dieses Nachteiles den Anforderungen von GoHyper am ehesten entspricht. Letztlich hängt die Wahl der geeigneten API von der individuellen Arbeitsweise ab.
[Spi][APIb][APIa]

2.2 JavaScript Framework

Ein Update kann nicht ohne erheblichen Aufwand von Angular 1.x auf Angular 2.x eingespielt werden. Diese beiden Versionen sind inkompatibel zu einander, weshalb die Gelegenheit genutzt und überprüft wird, welche Frameworks derzeit sonst verfügbar sind. Zur Unterstützung der JavaScript (JS) Implementierung werden im Rahmen dieser Arbeit die Frameworks Angular 5, React und Vue näher betrachtet.
Alle drei Frameworks arbeiten nach dem Einwegdatenflussprinzip und sind

daher geeignet, um mit Redux zu arbeiten. Was Redux genau ist wird am Ende des Kapitels genauer erläutert. Desweiteren können alle drei Frameworks auch mit Komponenten umgehen. Komponenten sind eigens erstellte Hypertext Markup Language (HTML) Konstrukte und können wie Standard-HTML-Konstrukte verwendet werden. Somit wird eine deutlich höhere Wiederverwertbarkeit des Codes erreicht.
AngularJS ist am umfangreichsten von den drei Frameworks. Neben Komponenten hat AngularJS auch Templates, Filter (oder Pipes genannt) und einige weitere Möglichkeiten, mit welchen umfangreiche Projekte visualisiert werden können. AngularJS wurde von Google entwickelt und ist besonders für den Fall geeignet, dass TypeScript verwendet werden soll. Darüber hinaus wird eine genaue Struktur vorgegeben, wie der Quellcode aufgebaut sein muss und verwendet beim Rendern zwei verschiedene Threads: den Service-Worker-Thread und den normalen Browser-Thread, welcher für den UI-Bereich verwendet wird. Daher ist der Renderprozess bei Angular marginal schneller als bei den anderen beiden Kandidaten. [Angb]
Vue kann ebenfalls mit Templates arbeiten, welche aber sowohl in reinem HTML- als auch im JSX-Format [JSX] vorliegen können. Der Programmierer kann hier selbständig agieren und das Format verwenden, was ihm geeigneter erscheint. Desweiteren kann Vue eigene Attribute[1] in die Komponenten eintragen, welche sich ohne zusätzlichen programmiertechnischen Aufwand direkt verwenden lassen. Im Gegensatz zu AngularJS verwendet Vue nur den Browser Thread, weshalb es beim Rendern etwas langsamer als AngularJS ist.[Vue]
React ist im Vergleich deutlich schmaler und gehört eher in die Kategorie Bibliothek. Unabhängig davon eignet sich React dazu, mit dem Model View Controller (MVC) Prinzip, wie auch AngularJS unter der Version 1.x, zu arbeiten. Es arbeitet mit dem Einweg-Datenfluss und passt daher ebefalls gut zu Redux. Der Programmierer kann hier besser auf Besonderheiten im Programm eingehen. [Rea]
Letztlich haben alle drei Projekte ihre Vor- und Nachteile und auch hier hängt es vom Entwickler beziehungsweise vom zu visualisierenden Projekt ab, welches ausschlaggebend ist und letztlich verwendet werden sollte. AngularJS eignet sich für ein größeres Team, da der Stil des Quellcodes sehr strikt beibehalten werden muss. Vue ist sinnvoll, falls eine Demo oder ähnliches implementiert werden soll. Dazu gehören kleine Projekte, die bestenfalls nicht viel Zeit in der Umsetzung benötigen (sollen). React ist für Projekte mit speziellen Problemen oder ohne aufwändige Logik geeignet.
Da GoHyper derzeit nicht viel Logik enthalten, die Entwicklung mit React aber schon etwas geüber ist, wurde sich letztlich für React entschieden. [Pet] [Neu]

[1] https://vuejs.org/v2/guide/syntax.html#Attributes

2.3 Design

Das Layoutdesign wurde von GoHyper 1.0 soweit wie möglich übernommen. Da jedoch der Stil nicht ohne weiteres übernommen werden konnte, wird nun Material-UI[2] verwendet, welches sehr gut für Implementierungen mit React verwendet werden kann. Desweiteren ist die Dokumentation der einzelnen Elemente ebenso wie die möglichen Funktionen sehr detailliert. Die Funktionen von GoHyper 2.0 umfassen das Setzten, Auslesen, Editieren und Löschen von Annotationen, Highlights und Kommentaren. Dabei sollen die Annotationen mittels der API im BvN hinterlegt werden, die Highlights und Kommentare hingegen lokal.
Da ursprünglich die Quotes Hyperlinks, Tags, Highlights und Kommentare beinhalteten, musste das User Interface (UI) für diesen Zweck überarbeitet werden. Daraus folgte eine seperate Ansicht für jede Funktion. Weitere Unterschiede werden in Kapitel 3 beschrieben.

2.4 Der Workflow mit Redux

Um den Aufbau zu verstehen, ist es wichtig nachvollziehen zu können, wie Redux[Red] funktioniert.
Redux ist ein Predictable-State-Container. Also ein Store, welcher verschiedene States einer bestimmten Application speichern kann. Dieser wird mit Default-Werten initialisiert, jedoch kann er auch als ein konsistenter Speicher verwendet werden. Dies erleichtert den Umgang mit dem Speicher im Allgemeinen, da es nur einen Speicherzugriff benötigt und die Verwendung von zusätzlichen Datenbanken oder anderen Arten, Daten für einen Internetbrowser zu speichern, überflüssig macht.
Der Reduxspeicher wird durch Actions angesprochen. Diese enthalten nur den aktuellen Zustand der Daten, die geändert werden sollen. Die Actions durchlaufen dann verschiedene Reducer und werden vom Entwickler bestimmt. Der Speicher richtet sich nach den Reducern und wird so grob unterteilt. Die unterteilten Speicher können als konsistent markiert werden. Für GoHyper werden die Reducer CurrentPage, Highlights, Comments und Tags und TagsEdit verwendet. Als konsistent sind Highlights, Comments und TagsEdit markiert. Die Reducer aktualisieren den eigentlichen Speicher (State genannt), welche mittels einer Dispatcher Funktion nur die Teile des UI aktualisiert, welche von der jeweiligen Speicheränderung betroffen sind.
Die REST API wird zwischen den Aufruf der Actions und den Reducern eingesetzt. Die Actions sind alles JavaScript Object Notation (JSON) Objekte, welche alle den Key 'type' haben. Dieser 'type' bestimmt auch für welche Reducer die Actions wichtig sind.

[2]URL: `https://material-ui-next.com/`

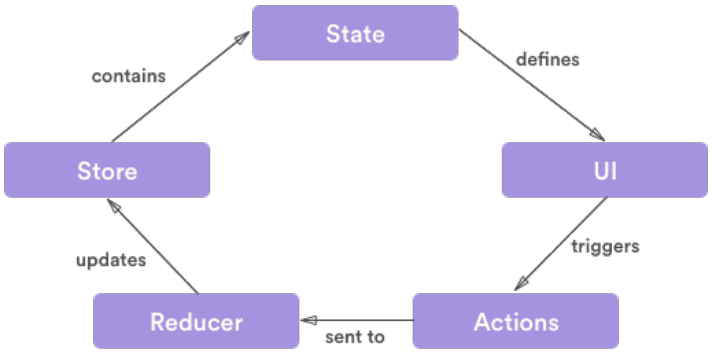

Abbildung 2.1: Current Page Reducer [San]

React in Verbindung mit Redux geht davon aus, dass durch eine Handlung des Nutzers genau eine Action aufgerufen wird. Dies wäre in Verbindung mit einer asynchronen API, wie die REST API, von Nachteil. Aus diesem Grund gibt es die Möglichkeit, die Typen noch mit zusätzlichen Eigenschaften zu verbinden, sodass die Typen nun auch aussagen, ob ein Request abgeschickt wurde, eine Response zurückgekommen ist oder ein Error erhalten wurde. Dies erleichtert auch die Handhabung mit einer asynchronen API.

3 Recherche

In der Version 1.0 konnte GoHyper spezielle Quotes setzen. Ein Quote zeichnete sich dadurch aus, dass der Benutzer zunächst eine Stelle in einem Fließtext markieren und diese Stelle dann mit beliebig vielen Hyperlinks und Tags versehen konnte. Desweiteren konnte der User an dieser Stelle auch Kommentare hinterlassen und war in der Lage, sich eine Übersicht aller Quotes anzeigen zu lassen. Die Quotes konnten nach einer bestimmten Reihenfolge sortiert oder gefiltert werden. Sollte der User ein bestimmtes Quote in der Übersicht auswählen, so wurde die Seite geladen, auf welcher der Quote gesetzt worden ist. Sobald ein Quote gesetzt war, unabhängig davon ob noch zusätzliche Hyperlinks, Tags oder einen Kommentar gesetzt wurden, war diese Textstelle direkt hervorgehoben (gehighlightet). Der User konnte also auch einen leeren Quote setzen, sollte er nur eine bestimmte Textstelle highlighten wollen.

3.1 Unterschiede in den Extentions

In der neu entwickelten Version von GoHyper sind die Quotes entfernt. Der User hat nun nicht mehr die Möglichkeit beliebig viele Hyperlinks und Tags zu setzen. Stattdessen kann er eine Textstelle selektiert und an dieser Stelle eine Annotation setzten. Diese Annotation wird in Verbindung mit WikiData[1] vervollständigt. Über eine zweite API werden nach Informationen über den Begriff in WikiData gesucht. Existiert dort exakt ein Eintrag, so wird die Annotation nach einer Bestätigung durch des Users gesetzt. Existieren mehrere Einträge, so wird der User aufgefordert, einen bestimmten Eintrag aus einer Liste auszuwählen und erst dann wird die Annotation gesetzt.
Durch die Einführung der Annotation sind die Quotes vollständig verschwunden. Es gibt noch eine Übersicht sämtlicher Einträge. Diese Annotationen richten sich nach der Terminologie von semantic Annotationen [MBKB+15]. Ist eine Annotation erzeugt, wird in Zukunft durch GoHyper beim Aufruf der Seite, auf die eine Annotation referenziert, automatisch die Informationen von WikiData geladen. Eigene Referenzen mittels URL kann der User nicht setzten. Auch wird zwischen Annotationen und Highlights für den User getrennt. So wird zwar eine Annotation für alle anderen User angezeigt, jedoch bleiben die Kommentare und Highlights lokal und können nicht mit anderen synchronisiert werden.
Wird eine Annotation gesetzt, wird diese Annotation für alle anderen User, die GoHyper verwenden, ebenfalls angezeigt. Das Speichern der Annotations-Daten geschieht nicht mehr lokal, was die Zusammenarbeit zwischen mehreren

[1] URL: `https://www.wikidata.org/`

Personen deutlich erleichtert. Derzeit existiert jedoch noch keine Anmeldungen, sodass alle User im Backend von Neonion als ein default User agieren.
Desweiteren existieren keine serverseitigen Funktionen zum Editieren und Löschen der Annotationen, weshalb hierfür Stabs eingesetzt wurden. Erläuterungen dazu folgen im nächsten Kapitel. Die Speicherung der Daten hat sich verändert. Während unter GoHyper 1.0 eine Indexdatenbank als konsistener Speicher verwendet wurde, welches die Suche nach einzelnen Einträgen erschwerte, kann nun leichter in verschiedenen JavaScript-Objekten gesucht werden. Da die Speicherung der Highlights und Kommentare weiterhin lokal passiert, ist hier ein voller Funktionsumfang, einschließlich dem Löschen und Editieren, vorhanden.

3.2 Übereinstimmungen der zwei GoHyper Versionen

Übereinstimmend mit der Version 1.0 ist geblieben, dass die Kommentare und Highlights weiterhin existieren und lokal gespeichert werden. Dies könnte für den User zu Verwirrungen führen, da die Annotationen im BvN gespeichert werden. Die Usability ist weitestgehend identisch geblieben. So werden neue Annotationen, Kommentare sowie Highlights weiterhin über das Kontextmenü angesteuert. Der Aufbau von GoHyper hat sich nicht verändert. Somit ist auch die Usability für den Nutzer weitestgehend identisch geblieben. Der Nachteil dabei ist, dass der Nutzer im Falle von Annotationen nicht mehr differenzieren kann, welche Einträge er selbst getätigt hat und welche Einträge von einem anderen User eingetragen wurden. Ebenfalls ist es derzeit nicht möglich Einträge im Backend von Neonion zu ändern oder zu löschen. Dies betrifft nur die Annotationen.

3.3 Andere Extensions und Tools

Auch andere Entwickler arbeiten derzeit an Tools und Extensions. So entwicklete Windows in seinem Browser Edge gleich die Funktionen, Elemente zu Highlighten und zu Kommentieren, mit. Die Highlights und Kommentare können auch mit anderen geteilt werden. Jedoch existiert dort keinerlei Möglichkeit, Begriffe zu taggen oder zu annotieren.
Der Browser Semantic Web Browser unterstützt speziell die Semantic Tags. Dieser Browser erlaubt es, nach Tags zu suchen und neue zu setzten. Die Informationen sind mit MediaWiki[2] verknüpft. Jedoch erlaubt dieser Browser keine Highlights oder Kommentare.

[2]URL: `https://www.mediawiki.org/`

4 Entwicklung

In diesem Abschnitt wird die Neuentwicklung von GoHyper dokumentiert. Dazu werden die Unterschiede in der Version 1.0 zur Neuentwicklung genannt und welche Aspekte übernommen wurden. Auch wird die Aufgabe der API beschrieben, welche Aspekte bereits vorbereitet sind und wie mit Problemen umgegangen wurde. Da die Implementierung funktionsorientiert ablief, wird der Aufbau von GoHyper immer anhand eines Funktionsbeispieles erklären. Im ersten Abschnitt dieses Kapitels wird der Aufbau des Redux-Speichers anhand der Highlights erörtert. Im zweiten Abschnitt dieses Kapitels liegt der Fokus auf der Funktionsweise des Redux-Speichers im Zusammenspiel mit React. Dies wird exemplarisch anhand der Kommentare erklärt. Im nachfolgenden Abschnitt wird erörtert, wie die clientseitige API entwickelt wird und wie und an welcher Stelle sie eingebunden wird. Zum Schluss wird dann noch darauf eingegangen, wie die Daten von WikiData geladen werden.

4.1 Der Speicher

Da GoHyper in der Version 1.0 ebenfalls mit Highlights arbeitet, wurde sich daran orientiert. In der voherigen Version von GoHyper musste zunächst eine Quote erstellt werden. Ob diese Quote jedoch noch Tags, Hyperlinks oder aber einen Kommentar enthält, spielt keine Rolle. In jedem Fall wurde der zuvor selektierte Bereich gehighlightet.
Die folgenden Informationen wurden für ein Highlight erhoben:

- URL: Verweist auf die Webseite, auf welcher das Highlight hinterlegt werden soll.
- Startpointer: Bestimmt den Startpunkt im Document Object Model (DOM), wo das Highlight beginnt.
- Endpointer: Bestimmt den Endpunkt im DOM, wo das Highlight endet.
- Timestamp: Speichert, wann das Highlight gesetzt wurde.

Da GoHyper Version 1.0 alles in einer Indexdatenbank speicherte und in dieser Arbeit mit Redux gearbeitet wird, ist der Aufruf der jeweiligen Speicher unterschiedlich. Es werden jedoch dieselben Daten erhoben.
Um später alle Highlights auf einer bestimmten Seite zu finden, wird die URL als ID verwendet. Die Struktur sieht dann unter Redux wie folgt aus:
Der Index innerhalb der Struktur nummeriert die Highlights auf einer Seite.
Da Redux die Informationen als JSON-Format speichert, reicht es, wenn dem

```
export const highlight = (url, index, startPointer, endPointer, timeStamp) => {
  let setHighlight = {};
  setHighlight[url] = {
    index: {
      startPointer: startPointer,
      endPointer: endPointer,
      timeStamp: timeStamp
    }
  };
  return setHighlight;
};
```

Abbildung 4.1: Highlight Struktur in Redux

Reducer ein JS-Objekt übergeben wird. Auf diese Art ist keine Umrechnung und auch kein Adapter nötig, um Daten zu speichern oder aus dem State auszulesen.
Der Aufruf, um ein entsprechendes Highlight in der Version 2.0 zu setzten, ist identisch zu dem Aufruf, ein Quote zu erstellen, wie in Version 1.0. Dies geschieht jeweils über das Kontextmenü im Webbrowser.
Eine weitere Möglichkeit ist die Aktivierung des Extension-Buttons im Webbrowser, im Header. Wird dieser aktiviert während der User ein Wort selektiert hat, öffnet sich ein Pop-Up direkt unter dem Button, in welchem sich der Benutzer entscheiden kann, ob er ein Highlight, einen Kommentar oder eine Annotation setzten möchte.

4.2 Redux-React Workflow

Die Speicherstruktur für die Kommentare ähnelt jener der Highlights sehr stark. Es gibt jedoch noch einen zusätzlichen Wert für die Kommentare. Im Kapitel 2.4 wurde bereits erklärt, wie der grundsätzliche Workflow mit der React-Redux-Kombination ist. Es wird nun nochmals drauf eingegangen, was mit den Daten passiert, wenn sie in Redux gespeichert oder ausgelesen werden.
Beim ersten Start der Extension werden Defaultwerte geladen. Im späteren Verlauf, wenn bereits einige Kommentare und andere Daten existieren, lädt Redux diese Informationen automatisch. Der Workflow zum Erstellen, Löschen und Editieren von Kommentaren ist jeweils identisch. Exemplarisch wird dieser Vorgang anhand der Erstellung eines Kommentares dargestellt. Durch den User wird auf einer Internetseite im Fließtext eine bestimmte Stelle markiert. Über das Kontextmenü wählt er den Reiter 'Kommentar erstellen' aus oder geht dafür über den Button im Header von Google Chrome. Sobald dies getätigt ist, wird eine Action erstellt. Diese Action ist ein JS Objekt und enthält alle nötigen Daten.
Eine solche Action wird in der folgenden Abbildung erzeugt. Das Objekt muss, damit es als Action aktiv werden kann, einen Typen haben, welcher im Reducer benötigt wird.
Anschließend wird diese Action durch die verschiedenen Reducer geleitet. Dort

```
export const createComment = (url, indexSize, startPointer, endPointer, timeStamp, comment) => {
  let commentAction = {};
  commentAction[url] = {};
  commentAction[url][indexSize + 1] = {
    comment: comment,
    endPointer: endPointer,
    startPointer: startPointer,
    timeStamp: timeStamp
  };
  return commentAction;
};
```

Abbildung 4.2: Create an Comment Action

wo der 'type' der Action mit einem 'type' eines Reducers übereinstimmt wird dann die entsprechende Funkion ausgeführt.

```
import * as TYPE from './../type/currentPage';

const currentPage = (state = {tag: false, comment: false, highlight: false, overview: false}, action = {}) => {
    let resultState;

    switch (action.type) {
        case TYPE.SET_FRONTEND:
            resultState = {comment: true, tag: false, highlight: false, overview: false};
            break;
        case TYPE.SET_OVERVIEW:
            resultState = {overview: true, tag: false, comment: false, highlight: false};
            break;
        case TYPE.SET_HIGHLIGHT:
            resultState = {highlight: true, tag: false, comment: false, overview: false};
            break;
        case TYPE.COMMENT:
            resultState = {tag: true, comment: false, highlight: false, overview: false};
            break;
        default: {

        }
    }

    if (resultState === undefined) {
        return state;
    } else {
        return Object.assign(...state, resultState);
    }
};

export default currentPage;
```

Abbildung 4.3: Current Page Reducer

Der Reducer in der Abbildung ist an dieser Stelle sehr simpel gehalten und ist vergleichbar mit den Routen von AngularJS. Nachdem die Action durch alle Reducer gewandert ist, wird verglichen, welche Teile des Speichers aktualisiert worden sind und welche nicht. Anschließend werden nur jene JS Dateien aufgerufen, welche von den manipulierten Daten betroffen sind. Dadurch aktualisiert auch React nur jene UI Elemente, welche die manipulierten Daten verarbeiten müssen. Dadurch ergibt sich ein Geschwindigkeitsvorteil gegenüber vielen anderen Frameworks, welche den gesamten DOM neu rendern würden.

4.3 Einbindung der Neonion API

In diesem Abschnitt wird genauer darauf eingegangen, wie das BvN mit der Extension kommuniziert. In den Quotes von GoHyper Version 1.0 kann der User sowohl Tags, als auch Hyperlinks getrennt voneinander verwenden.

In der Version 2.0 von GoHyper werden Annotationen verwendet. Diese sind in drei Aspekte unterteilt.

- selektierter String: ein selektierter String aus dem Fließtext einer Webseite, welcher getaggt werden soll.
- Kategorie: kategorisiert den selektierten String. Die Kategorie dient zur Unterstützung von Maschinen, damit diese z. B. bei einer Suchanfrage einen besseren Kontext ermitteln können.
- URL / String: Verlinkung zu z. B. WikiData, um Informationen über den selektierten String abzurufen. Sollte keine URL von WikiData hinterlegt sein, wird nur der String angezeigt.

Die Kategorie der Einträge wird automatisch ermittelt. Sollte dies nicht möglich sein, hat der Benutzer die Möglichkeit, selber eine Kategorie auszuwählen. Dadurch lassen sich folgende benötigte Spezifikationen für die Annotationen ermitteln:

- URL: die Seite, auf welcher sich die Annotation befindet
- path: die Lokalisation der Annotation innerhalb dieser Seite
- Kategorie: die Kategorie der Annotation
- Context: der Hyperlink bzw. das Ziel der Annotation

Die aus Swagger generierte API arbeitet asynchron, was bedeutet, dass der Browser nicht aktiv auf eine Response wartet. Dadurch besteht die Möglichkeit, während eines Funktionsaufrufes durch die API z. B. eine Warteanimation anzeigen zu lassen.
Damit diese Aufrufe auf der einen Seite durch eine Action erfolgreich aufgerufen und auf der anderen Seite erfolgreich durch die Reducer geleitet werden können, werden APIs im Datenfluss nach der Erstellung von Actions und vor dem Eintritt in den ersten Reducer eingebaut.

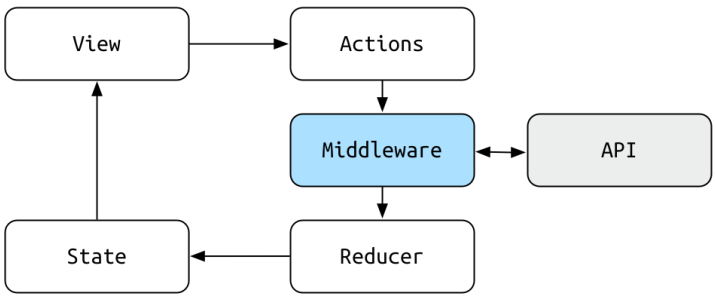

Abbildung 4.4: React Redux Workflow with an API [Cha]

4.4 Erstellen einer Annotation

Zum Erstellen einer Annotation sind zwei API Aufrufe nötig. Zuerst muss ein Target erstellt werden. Aufbauend auf einem Target wird die eigentliche Annotation erstellt.
Zum Erstellen eines Targets sind zwei Informationen nötig, welche mittels der API an das BvN übergeben werden.
Die PUT-Methode:

- URL: In die Uniform Resource Identifier (URI) zum Erstellen eines Targets wird eine Internationalized Resource Identifier (IRI) mit angehangen. Dabei darf die Methode nicht vergessen werden.
  Bsp.: url = http://{0}/targets/{1} –> http://127.0.0.1:8301/targets/target%3A1
- Methode: 'PUT'
- Body: Zudem wird eine Identifikator (ID) im Body des Funktionaufrufes im JSON-Format übergeben.

Somit ist der Request zum Erstellen eines Targets vollständig. Es gibt drei verschiedene Response-Möglichkeiten. Diese sind jeweils durch einen Code unterteilt.

- Code 201: Das Erstellen des Targets verlief erfolgreich.
- Code 409: Kommt zurück als Response im Falle eines Konfliktes. Dies ist der Fall, wenn z.B. bereits ein entsprechendes Target existiert.
- Code 400: Wird erhalten, falls der Request fehlerhaft ausgeführt wurde.

Sobald ein entsprechendes Target vorhanden ist, kann auch die Annotation erstellt werden. Durch das Erstellen von Targets wurde nun die URL zu den Webseiten gespeichert, auf welchen sich dann später die Annotationen befinden. Somit muss zum Erstellen einer Annotation der selektierte String, die Kategorie und die URL an BvN übergeben werden. An die Funktion zum Erstellen der Annotation werden folgende Variablen übergeben.

- URL: Innerhalb der URL werden zwei IRI übergeben werden. Zum einen die IRI des Targets und zum anderen die IRI der Annotation übergeben.
- Body: Der Body enthält noch weitere Variablen um alle Variablen der Annotationen zu vervollständigen.
  - context
  - ID
  - type
  - target

Auch für den Request gibt es drei unterschiedliche Response Möglichkeiten, wie schon bereits bei dem Request der Targets.

- Code 201: Das Erstellen der Annotation war erfolgreich.
- Code 409: Es gibt einen Konflikt.
- Code 404: Wenn das Target zu der Annotation nicht gefunden werden konnte.

4.5 Abrufen von Annotationen

Die Annotationen können sowohl einzeln, als auch komplett vom Server abgerufen werden. Dazu liegen unterschiedliche Methoden bereit.
Um sämtliche Annotationen zu erhalten, werden zunächst sämtliche Targets angefordert. Der Abruf mittels der API von BvN benötigt nur die URL des Servers auf dem sich BvN befindet. Als Response kommt eine JSON-Datei zurück, welche eine Liste von Target-Repräsentationen enthält. Diese Funktion hat als Antwort nur den Code 200, da die Funktion erfolgreich ausgeführt wurde. Sollte keinerlei Target bei BvN hinterlegt sein, kommt nur eine leere Liste zurück. Sollten Targets hinterlegt sein, kommt eine Response im JSON-Format zurück, welche eine Liste der Targets enthält.
Für ein bestimmtes Target muss hingegen nicht die komplette Targetliste abgerufen werden. Um die ID eines bestimmten Targets zu erhalten, ist es ausreichend, die IRI des Targets an die URL des Servers anzuhängen. In diesem Fall wird abermals ein Object im JSON Format zurückgegeben, welches jedoch nur die ID des einzelnen Targets enthält.
Sobald die ID eines Targets lokal hinterlegt ist, können zu dieser ID sämtliche Annotationen von BvN geladen werden. Der Aufruf gestaltet sich zu dem der Targets als äquivalent. Die URL, welche aus dem Abruf eines einzelnen Targets erstellt wurde, wird nun um ’./annotations’ erweitert und als Response kommt ebenfalls ein JSON-Format zurück, welches sämtliche Annotationen zu diesem einen Target enthält. Um eine bestimmte Annotation zu erhalten, werden an die API Funktion ebenfalls nur die IRIs des Targets und der Annotation übergeben, welche zu einer entsprechenden URL zusammengebaut werden. Als Response kommt ebenfalls ein JSON-Format zurück, welches dann nur die einzelne Annotation enthält.
Ebenso wie bei den Targets gibt es für sämtliche Annotationen nur einen Response-Code und für die Anfrage von bestimmten Annotationen den Code 200 (erfolgreich gefunden) und den Code 404 (Annotation nicht gefunden).

4.6 Editieren und Löschen von Annotationen

Das Editieren und Löschen einer Annotation ist derzeit nur lokal möglich. Die serverseitigen Funktionen existieren noch nicht. Um dennoch in gewissem Rah-

men die Funktion zu ermöglichen, wurde jeweils ein Stub verwendet.
Da derzeit weder eine Annotation, noch ein Target manipuliert oder gelöscht werden können, reicht es, wenn die jeweiligen Informationen lokal überschrieben werden. Es muss jedoch nicht darauf geachtet werden, ob die bereits bestehenden Targets und Annotationen auch serverseitig neuere Informationen haben, da dies nicht möglich ist.
Aus diesem Grund reicht es beim Laden einer Webseite zu überprüfen, ob neue Targets oder Annotationen existieren. Ein serverseitiger Abgleich und eine Überprüfung auf lokale Manipulationen ist jedoch nicht nötig. Um dennoch GoHyper für die spätere Erweiterung dieser Funktionen vorzubereiten und ein Anschluss an die entsprechenden API Funktionen möglichst leicht zu gestalten, werden die Manipulationen von Targets und Annotationen in einem seperaten Reducer verarbeitet. In dem Fall, dass lokale Manipulationen vorgenommen werden, ist ein Abgleich dennoch nötig. So muss überprüft werden, ob zu den serverseitig heruntergeladenen Annotationen auch lokale existieren. Sollten lokale Annotationen existieren, werden diese überschrieben. Daraus resultiert folgender Workflow:
Sobald der User eine Seite lädt, wird zunächst überprüft, ob diese Seite bereits als Target existiert. Sollte dies nicht der Fall sein, passiert auch nichts weiter. Im Fall, dass bereits ein Target existiert, werden die dazu gehörigen Annotationen, wie im vorherigen Abschnitt 4.5 beschrieben, geladen. Diese Annotationen werden in ein entsprechendes JS-Objekt verpackt und eine Action wie im Abschnitt 4.2 stößt den entsprechenden Workflow an. Nachdem die serverseitigen Variablen lokal hinterlegt sind, findet ein Abgleich zwischen den serverseitigen Variablen und den lokalen Manipulationen statt. Sollten serverseitige Annotationen lokale Manipulationen aufweisen, werden die lokalen Manipulationen mit einer höheren Priorität verwendet.

4.7 WikiData

Die Annotationen werden letztlich mit den Daten aus WikiData angereichert. Dazu muss zunächst eine entsprechende Anfrage an WikiData gestellt werden, ob über einen bestimmten String Daten vorhanden sind. Dazu existieren drei unterschiedliche Möglichkeiten: Es gibt keine Daten, es gibt genau einen Eintrag und es gibt mehrere Einträge zu einem bestimmten String. Sollte exakt ein Eintrag vorhanden sein, wird dieser dem User vorgeschlagen, welcher den entsprechenden Eintrag nur noch bestätigen muss. Sollten mehrere Einträge vorhanden sein, kann der User sich einen dieser Einträge aussuchen. In jedem Fall hat der User die Option 'Unknowed Ressource'. In diesem Fall werden keine Informationen von WikiData geladen. Der User erhält dann die Möglichkeit, eigene Informationen zu hinterlegen. Sobald eine Annotation geladen wurde, wird überprüft ob eine URL von WikiData oder ein normaler String hinterlegt ist. Je nachdem werden die Informationen erneut abgerufen oder es wird nur der String angezeigt.

5 Fazit

5.1 Aktueller Stand

Zum Ende lässt sich sagen, dass von der GoHyper Version 1.0 kaum etwas übrig geblieben ist. Durch die Einführung des Semantic Webs hat die Extension stark an Flexibilität eingebüßt. Diese Einbußen waren jedoch notwendig, um die Qualität der angezeigten Informationen deutlich zu steigern. Da die serverseitige API von Neonion derzeit noch nicht so weit ist war ein Einsatz der Stubs unabdingbar. Diese müssen vorerst Manipulationen der Annotationen lokal speichern. Da die Highlights und Kommentare derzeit lokale gespeichert werden, können diese im Vergleich zu den Annotationen schneller geladen werden. Auch sind diese Daten derzeit nicht von einem Server abhängig und können autark von einem Server angieren. Positiv ist zudem, dass der Workflow durch das Einsetzen von React und Redux deutlich vereinfacht werden konnte. Das dürfte einer Weiterentwicklung von GoHyper zugute kommen.

5.2 Weitere Möglichkeit

Trotz der unausgereiften Entwicklung von GoHyper lassen sich bereits jetzt viele Möglichkeiten erahnen. Angefangen das die Kommentare und Highlights in Zukunft ebenfalls im BvN hinterlegt werden und somit besser Informationen ausgetauscht werden können. Durch die Kommentare und Highlights gestaltet sich die Informationssuche in Verbindung mit einem eigenen Account auch als persönlicher. Im Gegensatz zu den Ansatz anderer Internetbrowser-Extensions, welche derzeit ihren Fokus nur auf Annotationen, statt auch auf Kommentare und Highlights legen. Solange noch kein Programm implementiert ist, welches die Informationen durch Annotationen nicht nur verknüpft, sondern die Informationssuche auch durch Kommentare und Highlights personalisiert, gestaltet sich die Datenanreicherung für den Einzelnen im Web auch weiterhin als äußerst schwer. Derzeit ist es außerdem nur möglich, Informationen von WikiData abzurufen. Jedoch könnte die weitere Entwicklung in eine Richtung gehen, in welcher Informationen nicht nur abgerufen, sondern neue Informationen auf WikiData geschrieben werden.

Literaturverzeichnis

[Anga] Guide to angularjs documentation. URL: https://docs.angularjs.org/guide. 2018-01-30.

[Angb] What is angular? URL: https://angular.io/docs. 2018-02-05.

[APIa] Swagger ui vs. api blueprint. URL: https://stackshare.io/stackups/api-blueprint-vs-swagger-ui. 2018-04-01.

[APIb] Swagger vs. raml - which is better for building apis? URL: https://blog.vsoftconsulting.com/blog/is-raml-or-swagger-better-for-building-apis. 2018-04-01.

[Bac] Gohyper version 2.0. URL: URL: https://github.com/Quenton90/gohyper.

[Ber] Human-Centered Computing @ FU Berlin. neonion - collaborative semantic annotations. http://fub-hcc.github.io/neonion/. 2018-01-23.

[Blu] Documentation. URL: https://apiblueprint.org/documentation/. 2018-03-16.

[Cha] Ignacio Chavez. redux-cycle. URL: https://ignaciochavez.com/how-redux-puts-middleware-together/. 2018-04-23.

[Cli] Client neonion api. URL: URL: https://github.com/Quenton90/Neonion_Client_Api.

[els] Elsevier. URL: https://www.elsevier.com/. 2018-02-10.

[Geb] Jennifer Gebske. Iterative und benutzerorientierte entwicklung der browser-erweiterung gohyper. https://www.inf.fu-berlin.de/inst/ag-se/theses/Gebske16-GoHyper.pdf. 2016-01-20.

[JSX] Jsx. URL: https://en.wikipedia.org/wiki/React_(JavaScript_library)#JSX. 2018-04-03.

[MBKB+15] Claudia Müller-Birn, Tina Klüwer, André Breitenfeld, Alexa Schlegel, and Lukas Benedix. Neonion: Combining human and machine intelligence. In *Proceedings of the 18th ACM Conference Companion on Computer Supported Cooperative Work &*

Social Computing, CSCW'15 Companion, pages 223–226, New York, NY, USA, 2015. ACM.

[Neu] Jens Neuhaus. Angular vs. react vs. vue: A 2017 comparison. URL: https://medium.com/unicorn-supplies/angular-vs-react-vs-vue-a-2017-comparison-c5c52d620176. 2018-02-03.

[Pet] Michael Petrosyan. Angular 5 vs. react vs. vue. URL: https://itnext.io/angular-5-vs-react-vs-vue-6b976a3f9172. 2018-02-02.

[RAM] Raml version 1.0: Restful api modeling language. URL: https://github.com/raml-org/raml-spec/blob/master/versions/raml-10/raml-10.md/. 2018-03-16.

[Rea] Hello world. URL: https://reactjs.org/docs/hello-world.html. 2018-02-20.

[Red] Read me. URL: https://redux.js.org/. 2018-03-03.

[RES] Representational state transfer. URL: https://en.wikipedia.org/wiki/Representational_state_transfer. 2018-02-16.

[San] Gerard Sans. Redux overview. https://angularjs.de/artikel/angular-redux-einfuehrung/. 2018-04-25.

[Spi] Kai Spichale. Entwicklung und dokumentation von rest-apis. URL: https://jaxenter.de/entwicklung-und-dokumentation-von-rest-apis-20099. 2018-03-25.

[Swa] Docs. URL: https://swagger.io/docs/. 2018-03-16.

[Vue] Guide. URL: https://vuejs.org/v2/guide/. 2018-01-30.

[YAM] Yaml. URL: https://en.wikipedia.org/wiki/YAML. 2018-03-19.

Anhang:

```
export const getState = () => {
  let state = {
    comments: {
      url: {
        index: {
          comment: String,
          endPointer: String,
          startPointer: String,
          timeStamp: String
        }
      }
    },
    currentPage: {
      overview: Boolean,
      comments: Boolean,
      highlight: Boolean,
      tag: Boolean
    },
    highlight: {
      url: {
        index: {
          endPointer: String,
          startPointer: String,
          timeStamp: String
        }
      }
    },
    overview: {
      visible: {
        comments: Boolean,
        highlight: Boolean,
        tag: Boolean
      }
    },
    tag: {
      url: {
        index: {
          endPointer: String,
          startPointer: String,
          timeStamp: String,
          categorie: String,
          wikiURL: String
        }
      }
    }
  };
};
```

Abbildung 5.1: Aufbau des Reduxspeichers

Glossar

IRI

Entspricht der URI mit einem erweiterten Zeichensatz

JSON

Ist ein kompacktes Datenformat, welches es erlaubt, auch in JS-Code direkt verwendet zu werden.

MVC

MVC beschreibt die strickte Trennung von Daten bzw. Quellcode in der Frontend-Entwicklung. Dabei gibt es drei verschiedene Aspekte. Das Model, den Viewer und den Controller. Das Model enthält sämtliche Daten, der Viewer passt sich speziell mit der UI und der Controller ist für die Logik und Verarbeitung der Daten zuständig. Die saubere APIs soll gewährleisten, dass ein möglichst reibungsloser Austausch einer dieser Komponenten möglich ist ohne das der komplette Code erneut implementiert werden soll.

BvN

Neonion ist eine Webseite einer Forschungsgruppe geleitet von Frau Prof. Müller-Birn zur Untersuchung des Semantic Web

REST

Beschreibt einen bestimmten Typ von API [RES].

URI

Mittels der Uniform Resource Identifier werden Ressourcen identifiziert.

URL

Unter Uniform Resource Locator oder kurz URL ist ein Einheitlicher Ressourcenzeiger zu verstehen. Dieser kann auf eine Internetseite oder eine externe Datenbank verweisen.

Annotation

Ein User kann eine Annotation setzten, um Informationen mit einem zuvor selektierten Wort zu verknüpfen.[Ber]

Extension

Ein Tool ist ein Werkzeug. Im Kontext dieser Arbeit ist damit ein Programm gemeint, welche die verschiedenen Web Browser als Erweiterung installieren können. Diese Erweiterungen werden z. B. bei Firefox ”AddOn” und bei Google Chrome ”Extension” genannt. Sie stellen den Web

Browsern zusätzliche Funktionen oder Grafiken zur Verfügung. Der Begriff Tool wird als Synonym für AddOn, Extension o. ä. verwendet, wenn der Browser nicht näher spezifiziert werden soll.

Extension

Extension ist ein Sammelbegriff für die Erweiterungen des Web Browsers Google Chrome. Diese Erweiterungen können separat voneinander installiert werden und sollen Google Chrome neue Funktionen oder Layoutanpassungen zur Verfügung stellen, welche in der Vanilla Version nicht existieren.

Highlight

Das Highlight ist eine farbliche Markierung, um wichtige Elemente z. B. in einem Fließtext hervorzuheben.

Request

Als Request wird die Anforderung eines Clients an einen Server betitelt. Der Server wird aufgefordert, eine bestimmte Arbeit zu verrichten.

Response

Als Response ist die Antwort eines Server auf Basis eines Request gemeint.

Semantic Web

Das Semantic Web erweitert das derzeitig herkömmliche World Wide Web um einige Daten. Ziel ist es, dass z. B. Namen leichter für Rechner interpretierbar sind, ohne den Kontext analysieren zu müssen. So könnte beispielsweise der Name ”Berlin” die zusätzlichen Information ”Hauptstadt”, ”Deutschland”, usw. liefern.

Stab

Ein Stab wird ersetztend für einen anderen Code verwendet. Ein recht berühmtes Beispiel sind dafür die Stabs in einer Java API. Der lokale Code bekommt nicht mit, das eventuell ein Serveraufruf getätigt wurde, da ein entsprechender Stab lokal die Ausführung dieses Codeabschnittes simuliert.

Stub

Stubs stehen für Programmcode anstelle eines anderen. Dabei werden sie häufig in Fall von APIs verwendet, um eine lokale Verarbeitung vorzutäuschen, auch wenn es sich dabei um einen Schnittstelle für eine Server-Client-Architektur handelt.

Vanilla Version

Sie beschreibt eine Software, welche ohne jegliche Erweiterungen betrieben wird.

Webseite

Beschreibt unter einer Domain die Auftritte von privaten, unternehmerischen oder staatlichen Anbietern. Sowohl Webseiten, als auch herunterladbare Dokumente sind unter den Begriffen Website, Webautritt, Webpräsenz oder Webangebot zusammengefasst.

Akronyme

API	Application programming interface
BvN	Backend von Neonion
DOM	Document Object Model
HTML	Hypertext Markup Language
ID	Identifikator
IRI	Internationalized Resource Identifier
JS	JavaScript
JSON	JavaScript Object Notation
MVC	Model View Controller
REST	Representational State Transfer
UI	User Interface
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
Web	World Wide Web