Organisation und Annotation von Bookmarks – benutzerfreundliche Verwaltung von digitalen Lesezeichen im World Wide Web

Anna A. Kriener

MASTERARBEIT

eingereicht am
Fachhochschul-Masterstudiengang

Interactive Media

in Hagenberg

im Juni 2016

Erklärung

Ich erkläre eidesstattlich, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen nicht benutzt und die den benutzten Quellen entnommenen Stellen als solche gekennzeichnet habe. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt.

Hagenberg, am 27. Juni 2016

Anna A. Kriener

Inhaltsverzeichnis

Kurzfassung

Das Wiederfinden von wichtigen Informationen im World Wide Web kann sich bei der enormen Menge an verfügbaren Inhalten als sehr schwierig herausstellen. Auch die Verwaltung von immer größer werdenden Bookmark-Archiven wird zu einer Herausforderung. Bestehende Tools greifen diese Schwierigkeiten auf und bieten bereits Funktionalitäten, die Benutzerinnen bei deren Recherche im Web unterstützen sollen. Jedoch gibt es kein Tool, welches das Konzept von multimedialen Annotationen und einer alternativen grafischen Visualisierung von Bookmarks umsetzt.

Im Zuge dieser Arbeit wurde deshalb ein Konzept für ein webbasiertes Tool entwickelt, das die Online-Recherche für Webanwenderinnen erleichtern soll, indem zusätzliche Funktionen geboten bzw. unterschiedliche Funktionen vereint werden und damit das Verwenden von mehreren verschiedenen Tools überflüssig machen soll. Mit einer alternativen Visualisierung von Bookmarks in Mindmaps und mit multimedialen Annotationen in Form von Text, Hyperlinks, Bildern oder Videos soll Nutzerinnen das Wiederfinden von wichtigen Informationen vereinfacht werden. Dadurch soll eine höhere Flexibilität beim Recherchieren im World Wide Web erzielt werden.

Auf Basis des entwickelten Konzeptes wurde ein Prototyp implementiert, der als Grundlage für eine Machbarkeitsstudie diente. Die Ergebnisse aus dieser Evaluierung ermöglichten es, die Erfüllung der gesetzten Ziele und Anforderungen zu überprüfen.

Abstract

The retrieval of important information from the World Wide Web can be very difficult due to the enormous amount of available content. The management of growing bookmark archives can also be very challenging. Already existing tools try to address these difficulties and offer functionality to support users in their research on the web. However, none of these tools support multimedia annotations, nor do they provide an alternative graphical representation of bookmarks.

In the course of this work a concept for a web-based tool has been developed, that should make it easier for web users to search on the web. A tool that offers additional new functions as well as a combination of different existing functions should make it unnecessary to use multiple different tools. The provision of an alternative visualization of bookmarks in the form of mind maps, and the ability to create multimedia annotations in the form of text, hyperlinks, images, and videos should help users retrieve important information. With this, a greater flexibility in research can then be achieved.

Based on this concept, a prototype was implemented, which provided the basis for a feasibility study. The results from this evaluation made it possible to check if the set goals could be achieved and if the requirements could be fulfilled.

Kapitel 1

Einleitung

Noch vor mehr als 25 Jahren hatte Tim Berners-Lee den Traum von einem weltweiten Netzwerk, in dem Informationen miteinander verlinkt werden können und Menschen von überall auf der Welt ihr Wissen miteinander teilen können [2]. Seitdem hat sich das World Wide Web in hoher Geschwindigkeit zu einem System entwickelt, in dem Inhalte nicht nur passiv angesehen werden können, sondern in dem Nutzerinnen auch aktiv teilnehmen können. Das Web ist fester Bestandteil im Alltag geworden und hat sich mit unzähligen digitalen Dokumenten als zentrale Informationsquelle etabliert.

1.1 Problemstellung und Zielsetzung

Aufgrund der unüberschaubaren Anzahl an Webseiten und der nicht fassbaren Menge an Informationen im Web sind Anwendungen, die das Verwalten von wichtigen Inhalten unterstützen, unabdingbar geworden. Der Gebrauch von Bookmarks, um Informationen im Web wiederzufinden, wird von vielen Menschen jedoch vermieden, da der Verwaltungsaufwand für zu groß gehalten wird. Ein weiteres Problem ist das Wiederfinden von relevanten Informationen unter einer großen Menge von Inhalten auf einer Webseite. Es existieren bereits einige Tools, die diese Schwierigkeiten aufgreifen und Einzellösungen für die genannten Probleme bieten. Jedoch ermöglichen die meisten Tools nur das Verwalten der Bookmarks in Ordnern, aneinandergereiht in langen Listen. Verfügbare Annotations-Tools erlauben meist nur textuelle Kommentare, die nur zu Text-Inhalten hinzugefügt werden können. Das Ziel dieser Arbeit ist es, ein Tool zu entwickeln, das die Schwächen der bestehenden Tools aufgreift, neue Lösungen dafür bietet und deren Einzellösungen kombiniert. Es sollen multimediale Annotationen für multimediale Inhalte auf beliebigen Webseiten möglich sein, um mehr Flexibilität zu erreichen. Bookmarks mit den dazugehörigen Annotationen sollen, organisiert in individuellen Mindmaps als alternative Visualisierung, von Benutzerinnen besser verwaltet werden können.

1.2 Inhaltlicher Aufbau

Die vorliegende Arbeit gliedert sich insgesamt in sechs Kapitel. In Kapitel 2 wird ein Überblick über den aktuellen Stand der Technik gegeben. Es wird auf die Geschichte von Bookmarks und Annotationen eingegangen und über vergleichbare relevante Recherche-Tools berichtet. Danach wird in Kapitel 3 der Entwurf für den eigenen Prototyp *tisoor* vorgestellt. Dazu werden zuerst die Ausgangssituation und die Anforderungen definiert und anschließend wird das Konzept für den eigenen Ansatz erläutert. Die Implementierung des Prototyps wird in Kapitel 4 beschrieben. Es werden die verwendeten Technologien vorgestellt und anschließend wird auf die technische Ausführung eingegangen. Kapitel 5 befasst sich mit der Evaluierung des Prototyps *tisoor*. Hier wird eine Anforderungsreflexion durchgeführt, um die Erfüllung der definierten Anforderungen zu ermitteln. Des Weiteren werden die Ergebnisse eines Benutzertests durch die Think Aloud Methode und eines User Experience Fragebogens dargelegt. Die Arbeit schließt mit einer Zusammenfassung in Kapitel 6 ab. Es wird ein Resümee der gesamten Arbeit gezogen und ein Ausblick auf mögliche zukünftige Entwicklungen gegeben.

1.3 Gendering

Aus Gründen der besseren Lesbarkeit wird in dieser Arbeit auf eine geschlechtergerechte Schreibweise verzichtet und ausschließlich die Sprachform des generischen Femininums angewendet. Es wird an dieser Stelle darauf hingewiesen, dass die Verwendung der weiblichen Form sowohl weibliche als auch männliche Personen anspricht.

Kapitel 2

Stand der Technik

Dieses Kapitel befasst sich zu Beginn mit der Geschichte der digitalen Lesezeichen und der digitalen Annotationen. Es wird über bereits existierende Arbeiten und Studien berichtet, die sich mit Benutzerverhalten im Web, Bookmarks und Annotationen auseinandersetzen. Danach folgt eine Analyse der vier aktuellen webbasierten Tools *Diigo*, *scrible*, *Coggle* und *Video-Ant*, die bei der Recherche im Web und bei der Organisation von Online-Informationen unterstützen.

2.1 Geschichte der digitalen Lesezeichen

Bereits im Jahr 1994 haben Catledge und Pitkow in [4] analysiert, wie Nutzerinnen mit ihrem Webbrowser interagieren, und dahingehend versucht, Strategien für das Navigieren im Web zu charakterisieren. Hauptsächlich wurde via Hyperlinks und Back-Button (92,5%) durch das Web navigiert. Bookmarks hingegen wurden nur selten verwendet.

Eine weitere Studie von Tauscher und Greenberg im Jahr 1995 [19] konzentrierte sich darauf, wie sich das erneute Besuchen von Webseiten auf den Browser-Verlauf auswirken könnte. Dabei stellten sie fest, dass Nutzer 58% aller aufgerufenen Webseiten bereits in der Vergangenheit besucht hatten und dies in Zukunft besser von Browsern unterstützt werden sollte.

Obwohl diese beiden Studien zeigen, dass Bookmarks wenig genutzt wurden, bestätigt eine andere Studie aus dem Jahr 1996 [1], dass Bookmarks mehr an Bedeutung gewannen (mehr als 92% der Nutzerinnen hatten ein Bookmark-Archiv), diese aber bei steigender Anzahl und durch schlechte Visualisierung schwierig zu verwalten waren. Auch der eingeschränkte Zugriff auf das Archiv von nur einem Gerät wurde als Problem beschrieben.

Cockburn und McKenzie aktualisierten im Jahr 2000 in [5] die zuvor genannten Studien und stellten fest, dass Nutzerinnen bereits 81% aller besuchten Webseiten zuvor schon einmal besucht hatten. In der Studie wurde ebenfalls aufgezeigt, dass Bookmarks von Nutzerinnen sehr unterschiedlich

genutzt wurden. Es wurden generell kontinuierlich Bookmarks ins Archiv hinzugefügt, aber kaum wieder gelöscht, was sehr große Archive zur Folge hatte. Ein weiteres Problem war, dass etwa 5% der Bookmarks Duplikate waren und etwa 25% der gespeicherten Webseiten bereits ungültig waren.

Im Jahr 2001 bestätigten Jones, Bruce und Dumais in [8], dass Nutzerinnen aufgrund der bereits zuvor genannten Probleme nur ungern Bookmarks verwendeten. Sie wiesen darauf hin, dass eine verbesserte Funktionalität und grafische Repräsentation das Verwenden von Bookmarks verbessern würde.

Erst 2007 wurde erneut eine Studie [14] durchgeführt, die neue Ergebnisse brachte. Nutzerinnen navigierten nicht nur mehr zum Großteil via Hyperlinks und Back-Button (57,8%), sondern vermehrt über das Absenden von Formularen (15,3%) und das Öffnen von Webseiten in neuen Tabs (10,5%). Außerdem waren nur mehr 45,6% der aufgerufenen Webseiten bereits zuvor besucht worden. Bookmarks wurden nach wie vor aufgrund der aufwendigen Organisation nur selten genutzt. Nutzerinnen bevorzugten das erneute Suchen in einer Suchmaschine, um Informationen wiederzufinden, obwohl sie oft Schwierigkeiten hatten, sich an Suchbegriffe korrekt zu erinnern, und die Suchmaschinen aufgrund der Schnelllebigkeit des Web nach kurzer Zeit andere Ergebnisse lieferten. Dahingehend wurde in dieser Studie betont, dass es wichtig sei, verbesserte Tools zu entwickeln, die das Finden von Informationen auf bereits besuchten Webseiten erleichtern.

Mathur und Karahalios arbeiteten 2009 in [12] an einem interessanten Tool, das die Bookmarks einer Nutzerin auf drei verschiedene Arten graphisch darstellen konnte. In der ersten Ansicht wurden Bookmarks als verschieden große Kreise dargestellt und in Halbkreisen angeordnet. Je öfter einzelne Bookmarks verwendet wurden, desto größer waren deren Kreise. Die zweite Darstellung war die Anordnung der Bookmarks als gestapelte Quadrate auf einer Zeitleiste. In der dritten Ansicht wurden die Bookmarks als hierarchischer Baum in Form von Quadraten (Ordner) und zugehörigen Kreisen (Bookmarks) dargestellt.

Auch im Jahr 2013 wurde von Shen und Prior in [17] an einer Lösung gearbeitet, um die Verwendung von Bookmarks für Nutzerinnen brauchbarer zu machen, da nach wie vor die bereits seit Jahren bekannten Probleme vorhanden waren. Sie entwickelten keine graphische Repräsentation, sondern ein strukturiertes Bookmark-Schema mit nur vier vorgegebenen Hauptkategorien und jeweils zwölf Unterkategorien.

Abgesehen von der Verbesserung von Bookmark-Tools gibt es aber auch andere Möglichkeiten, wie etwa Web-Annotationen, die das Wiederfinden von Informationen und die Recherche im Web vereinfachen können.

2.2 Geschichte der digitalen Annotationen

Kawase, Papadakis, Herder und Nejdl sahen in [9] großes Potential in digitalen *in-context*[1] Web-Annotationen. Sie entwickelten das Annotations-Tool *SpreadCrumb* und konnten damit Vorteile beim Wiederfinden von Informationen gegenüber von traditionellen Bookmark-Tools und Suchmaschinen feststellen.

Bereits Marshall betonte 1997 in [11], dass das Hinzufügen von Annotationen eine wichtige Aktivität ist, die ein bedeutender Teil vom Lesen, Schreiben und Lernen sein kann. Da Web-Annotationen als sehr vielversprechend gesehen wurden, wurden in den vergangenen Jahren viele weitere Tools entwickelt, wie etwa *MADCOW* [3] oder *LEMO* [7].

Aufgrund der vielen unterschiedlichen Annotations-Tools existierte und existiert nach wie vor eine große Anzahl an Architekturen, die nicht kompatibel miteinander sind. Auch die Definition von Annotationen variiert je nachdem, in welchem Bereich diese eingesetzt werden.

Haslhofer, Jochum, King, Sadilek und Schellner arbeiteten in [7] deswegen an einem einheitlichen Annotations-Modell für Multimedia-Inhalte und unterschiedliche Typen von Annotationen. Ihre Anforderungen an Annotationen waren, dass diese verschiedene Medientypen zulassen und auf Teilinhalte einer Webseite (Textpassagen, Bilder, Frames in Videos) angewendet werden können. Außerdem sollte es möglich sein, verschiedene Dokumente miteinander zu verbinden (Hyperlinks). Annotationen sollten robust gegenüber Veränderungen im Dokument sein und direkt im Dokument platziert werden können. Ebenfalls wichtig ist die Möglichkeit für öffentliche und private Annotationen und das Teilen von Annotationen mit anderen Nutzerinnen. Unter diesen Anforderungen und noch einigen weiteren wurden verschiedene Tools getestet, wobei keines der 19 Tools allen Anforderungen gerecht wurde.

2.3 Vergleichbare Recherche-Tools

In den letzten Jahren wurden viele verschiedene Annotations- und Bookmark-Tools entwickelt, die zum Großteil im Web leider nicht mehr verfügbar bzw. nicht mehr auffindbar sind. Die Tools, die es nach wie vor geschafft haben sich durchzusetzen, unterstützen vollkommen unterschiedliche Funktionen in unterschiedlicher Qualität. Viele der Tools ermöglichen entweder nur das Verwalten von Bookmarks oder ausschließlich Web-Annotationen, aber nur wenige ermöglichen beides.

Im Folgenden werden acht aktuelle Annotations-Tools in Anlehnung an die von [7] aufgezeigten Anforderungen erwähnt. Dies umfasst natürlich bei

[1] Mit *in-context* ist hier gemeint, dass Annotationen direkt innerhalb der originalen Ressource platziert werden können und dort sichtbar sind.

	A.nnotate	*Bounce*	*Diigo*	*FloatNotes*	*Hypothes.is*	*MyStickies*	*scrible*	*Usersnap*
verschiedene Medientypen	–	–	–	–	–	–	–	–
Assoziationen	+	–	–	–	–	–	–	–
robust	+	+	+	–	+	–	+	+
kollaborativ	+	+	+	–	+	–	+	+
öffentlich/persönlich	–	–	+	–	+	–	–	–
in-context Platzierung	–	–	+	+	–	+	+	–
Verwaltung	+	–	+	–	+	+	+	–

Tabelle 2.1: Acht verschiedene Annotations-Tools unterstützen jeweils unterschiedliche Funktionalitäten. Bei keinem der Tools können verschiedene Medientypen (Text, Hyperlink, Bild oder Video) annotiert oder als Notiz hinzugefügt werden.

Weitem nicht die gesamten verfügbaren Anwendungen im Web, gibt aber einen generellen Überblick über den Stand der Technik. In Tabelle 2.1 ist ersichtlich, welche Tools welche Anforderungen erfüllen.

Mit „verschiedene Medientypen“ ist gemeint, dass Annotationen zu verschiedenen Medientypen (Text, Hyperlink, Bild, Video) hinzugefügt werden können und gleichzeitig die Annotation selbst verschiedene Medientypen beinhalten kann. „Assoziationen“ steht für die Möglichkeit, verschiedene Annotationen miteinander zu verbinden und Annotationen sind „robust“, wenn sie mit dynamischen Webinhalten umgehen können und nicht verloren gehen. Annotationen sollen auch mit anderen Nutzerinnen geteilt („kollaborativ“) und als „öffentlich“ oder „persönlich“ verwaltet werden können. Optimalerweise werden Annotationen direkt bei der Ressource platziert („in-context Platzierung“) und können später auch in einer eigenen Umgebung verwaltet werden („Verwaltung“).

Es ist ersichtlich, dass *Diigo*[2] im Gegensatz zu den anderen genannten Tools die meisten Anforderungen erfüllt. *A.nnotate*[3], *Bounce*[4] und *Usersnap*[5] erstellen einen Screenshot der gewünschten Webseite, welcher dann annotiert werden kann. *FloatNotes*[6] und *MyStickies*[7] ermöglichen nur das Erstellen von digitalen Haftnotizzetteln mit textuellen Kommentaren. *Hy-*

[2] https://www.diigo.com/
[3] http://a.nnotate.com/
[4] http://bounceapp.com/
[5] https://usersnap.com/de
[6] http://www.floatnotes.org/
[7] https://www.mystickies.com/

	coggle	*Delicious*	*Diigo*	*Dragdis*	*Historious*	*Instapaper*	*Pinterest*	*Pocket*
Listendarstellung	–	+	+	–	+	+	–	+
alternative Visualisierung	+	–	–	+	–	–	+	+
Teilen von Bookmarks	+	+	+	+	–	+	+	+

Tabelle 2.2: Verschiedene Bookmark-Tools ermöglichen unterschiedliche Arten der Visualisierung von Bookmarks. Die meisten Tools repräsentieren Bookmarks in Listen oder als Kacheln.

pothes.is[8] unterstützt zwar das Hinzufügen von Bildern und Hyperlinks, die Annotationen befinden sich aber nicht im Kontext. Mit *scrible*[9] können Notizen direkt bei der Ressource platziert werden, diese erlauben jedoch ausschließlich Text als Inhalt. Somit erfüllt jedes Tool einige Anforderungen, jedoch erfüllt keines alle und keines der Tools erlaubt das Annotieren und Hinzufügen von verschiedenen Medientypen.

Da die meisten Webanwendungen entweder Annotationen oder Bookmarks unterstützen und viele Bookmark-Tools nur eine einfache Listendarstellung anbieten, sind in Tabelle 2.2 acht aktuelle Tools angeführt, die teilweise eine alternative Darstellung ermöglichen. Hier wurde auch berücksichtigt, ob Informationen mit anderen geteilt werden können. Diese Auflistung ist ebenfalls keinesfalls vollständig.

Dragdis[10], *Pinterest*[11] und *Pocket*[12] bieten eine alternative Darstellung der Bookmarks als Kacheln an. Diese Kacheln beinhalten meist auch ein entsprechendes repräsentatives Bild. Alle Anwendungen bis auf *Historious*[13] ermöglichen das Teilen von Bookmarks. Darüber hinaus bietet *Delicious*[14] als Social Bookmarking-Tool noch umfassendere Möglichkeiten dahingehend. *Coggle*[15] ist zwar kein Bookmark-Tool, wurde hier aber angeführt, da es als Mindmapping-Tool eine interessante alternative Darstellungsmöglichkeit repräsentiert. *Diigo* ist hier als eines der wenigen Tools angeführt, das Annotationen und Bookmarks verwalten kann. Im nachfolgenden Abschnitt wird diese Webanwendung daher näher beschrieben.

[8] https://hypothes.is/
[9] http://www.scrible.com/
[10] https://dragdis.com/
[11] https://www.pinterest.com/
[12] https://getpocket.com/
[13] https://historio.us/
[14] https://delicious.com/
[15] https://coggle.it/

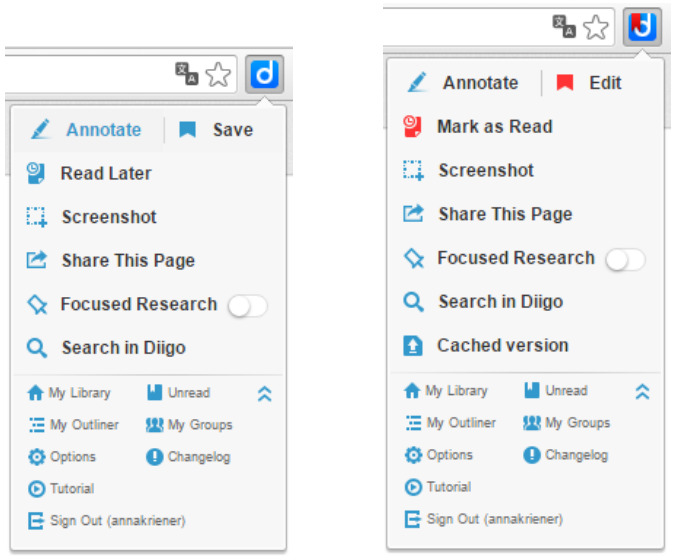
2.3.1 *Diigo*

Das Wort *Diigo* ist eine Abkürzung für „Digest of Internet Information, Groups and Other stuff“ und bedeutet so viel wie „Überblick über Internet-Informationen, Gruppen und andere Dinge“ [21]. *Diigo* ist eine Webanwendung, die es Nutzerinnen ermöglicht, ihre Informationen im Web mittels Bookmarks und Annotationen zu organisieren und im Überblick zu behalten. Das Team von *Diigo* arbeitet bereits seit dem Jahr 2005 an dieser Anwendung und hat festgestellt, dass ein Großteil der Informationen heutzutage online bezogen wird [21]. Sie haben ebenfalls festgestellt, dass dieser Vorgang der Informationsbeschaffung und die damit verbundene Organisation und Bearbeitung der Informationen oft sehr ineffizient ist. *Diigo* ermöglicht es, Informationen einfacher, übersichtlicher und produktiver zu organisieren und bietet darüber hinaus einige Funktionen, die andere Tools nicht haben. Diese Funktionalität ist unter anderem dafür verantwortlich, dass sich *Diigo* seit mehreren Jahren gegen konkurrierende Anwendungen durchsetzt [35]. Über die Jahre hinweg hat sich *Diigo* immer weiter entwickelt und mit jeder neuen Version wurden weitere Funktionen hinzugefügt. Anfänglich war es nur ein Online-Lesezeichenmanager, danach wurden Annotationen in Form von Text-Markierungen und Haftnotizzettel hinzugefügt. Mittlerweile ist die Anwendung zum gruppenbasierten kollaborativen Recherche-Tool geworden, mit dem Bookmarks gesammelt werden können und auch Annotationen möglich sind [21]. Die Anwendung ist bis jetzt nur in englischer Sprache verfügbar, somit beinhalten auch alle folgenden Screenshots der Anwendung englischen Text.

Installation

Um *Diigo* zu verwenden, sind nur wenige einfache Schritte notwendig. Auf der Website der Anwendung[16] muss ein neues Benutzerkonto angelegt werden. Dort kann auch die persönliche digitale Bibliothek mit allen gespeicherten Bookmarks eingesehen und verwaltet werden. Bei der Registrierung kann zwischen fünf verschiedenen Tarifen (Free, Basic, Standard, Professional und Teacher) gewählt werden, die jeweils unterschiedlichen Umfang an Funktionalität und Kapazität bieten [36]. Für diese Arbeit wurde der Tarif Free gewählt, mit dem alle nachfolgenden Betrachtungen durchgeführt wurden. Bezahlte Tarife bieten aber umfassendere Funktionen, wie etwa die Annotation von PDF-Dateien, Volltextsuche oder unlimitierte Kapazitäten. Um Bookmarks und Annotationen erstellen zu können, wird ein Benutzer-Tool installiert. *Diigo* bietet dazu Erweiterungen für die Browser *Firefox*, *Google Chrome*, *Internet Explorer* und *Safari*. Darüber hinaus gibt es das vom Browser unabhängige Tool *Diigolet* und Anwendungen für *iPhone*, *iPad* und *Android*. Hier wird die Erweiterung für *Google Chrome* verwendet. Nach

[16] https://www.diigo.com/



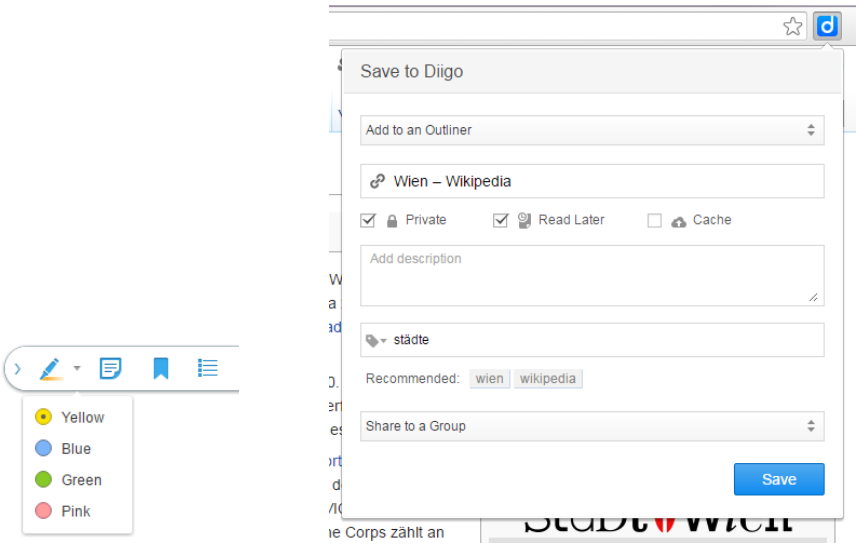
(a) (b)

Abbildung 2.1: Screenshot vom Menü der Browser-Erweiterung von *Diigo* (a) und von demselben Menü, nachdem eine Webseite als Bookmark gespeichert und diese als noch nicht gelesen markiert wurde (b).

der Installation erscheint in der Symbolleiste rechts oben im Browser ein neues Icon, mit dem ein Menü für alle weiteren Funktionen von *Diigo* geöffnet werden kann (siehe Abb. 2.1). Das bedeutet, dass alle Webseiten, die eine Manipulation durch Browser-Erweiterungen zulassen, mit *Diigo* auch bearbeitet werden können.

Funktionen

Über die Browser-Erweiterung von *Diigo* können einige Funktionen auf die aktuell besuchte Webseite angewendet werden (siehe Abb. 2.1). Die beiden wichtigsten sind „Annotate“ und „Save“. Mittels Klick auf den Button Annotate wird das Menü mit allen weiteren Funktionalitäten zum Annotieren der aktuellen Seite ein- und ausgeblendet (siehe Abb. 2.2). Wird der Button Save geklickt, erscheint ein Menü mit weiteren Einstellungen zum Speichern der aktuellen Seite als Bookmark (siehe Abb. 2.2). Dabei wird ein Link zur Webseite in der persönlichen digitalen Bibliothek abgelegt. Über die Funktion „Read Later“ wird automatisch ein Bookmark der aktuellen Webseite erstellt und diese wird dann automatisch geschlossen. Es gibt auch die Möglichkeit, einen Screenshot der aktuellen Seite zu erstellen. Hier kann der Bereich, der als Bild gespeichert werden soll, individuell angepasst und bearbeitet werden (siehe Abb. 2.3). Die aktuelle Webseite kann mittels der Option „Share This Page“ mit anderen Personen über *Twitter*, *Facebook*,



(a) (b)

Abbildung 2.2: Screenshots vom Menü zum Erstellen von Annotationen (a) und vom Menü zum Speichern von Bookmarks (b) in *Diigo*.

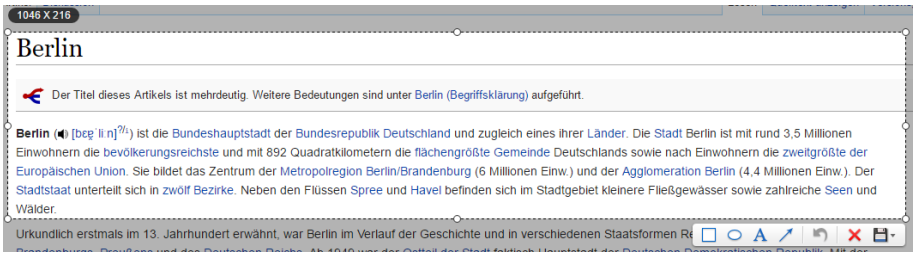


Abbildung 2.3: Screenshot von der Funktion zum Erstellen eines Screenshots einer Webseite von *Diigo*.

Google+ oder per E-Mail geteilt werden. Durch die Auswahl der Option „Annotated Link“ ist es möglich, die Webseite mitsamt allen Annotationen und Markierungen zu teilen. Die Funktion „Focused Research“ ist eine Premium-Funktion, die im Tarif Free nicht enthalten ist und hier nicht getestet wird. Mithilfe der Funktion „Search in Diigo“ können je nach Tarif Metadaten (Titel, URL, Annotationen und Tags) nach Stichworten durchsucht oder eine Volltextsuche (zusätzlich auch in allen als Kopie gespeicherten Webseiten) durchgeführt werden.

Bookmarks: Wie bei allen anderen Bookmark-Tools gibt es auch bei *Diigo* die grundlegende Funktion, Webseiten als Bookmarks abzuspeichern. Bereits beim Speichern von Bookmarks können zusätzliche Informationen

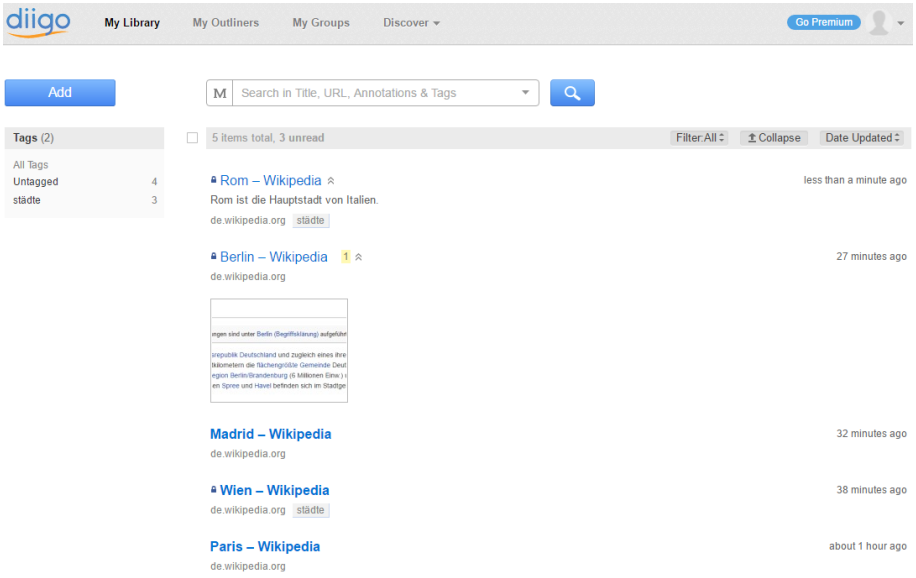


Abbildung 2.4: Screenshot von der digitalen Bibliothek von *Diigo*.

angegeben werden, wie etwa der Titel, eine textuelle Beschreibung oder Tags.[17] Es können auch sogleich Einstellungen vorgenommen werden, wie z. B. das Markieren der gespeicherten Webseite als privat oder als ungelesen. Mittels der Funktion „Cache“ kann die gesamte Webseite als Kopie gespeichert werden. Besonders für dynamische Webseiten, deren Inhalt sich ständig ändert, ist das von Vorteil, da Informationen bewahrt werden können. Ein Bookmark kann hier auch sogleich zu einem Outliner[18] hinzugefügt werden. In der digitalen Bibliothek werden alle gespeicherten Bookmarks als Liste dargestellt (siehe Abb. 2.4). Hier werden auch die Annotationen zu den jeweiligen Webseiten aufgelistet. Die digitale Bibliothek dient auch zur Verwaltung der persönlichen Bookmarks (löschen, bearbeiten und teilen).

Annotationen: Als besondere Eigenschaft von *Diigo* gelten Annotationen, die auf einer aktuell besuchten Webseite direkt bei der zu annotierenden Stelle hinzugefügt werden können. Dabei ist es möglich, selektierte Textpassagen in vier verschiedenen Farben zu markieren und Notizen in Form von Haftnotizzetteln direkt zum markierten Text oder für die gesamte Webseite hinzuzufügen (siehe Abb. 2.5). Globale Notizen können beliebig auf der Webseite platziert werden. Notizen zu Textpassagen werden dargestellt, indem am Ende einer Markierung ein repräsentatives Icon eingefügt wird. Farben

[17] Tags sind Stichworte oder Phrasen, die einem oder mehreren Bookmarks hinzugefügt werden können, um Bookmarks zu verwandten Themen zu gruppieren.

[18] Outliner sind bei *Diigo* gegliederte Listen, in denen Informationen von Benutzerinnen individuell strukturiert werden können.

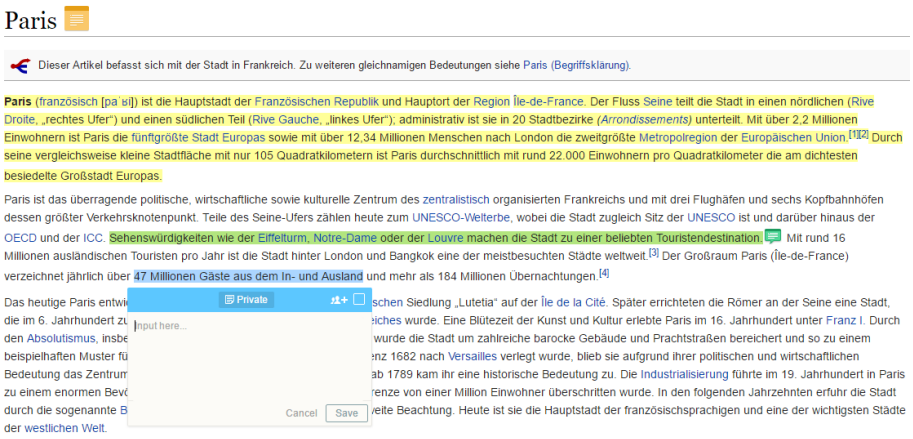


Abbildung 2.5: Screenshot von einem durch *Diigo* mit Annotationen versehenem Wikipedia-Eintrag über Paris [38].

können im Nachhinein geändert werden und auch Notizen können nachträglich zu markierten Texten hinzugefügt werden. Es gibt die Möglichkeit, Bilder zur persönlichen digitalen Bibliothek hinzuzufügen, jedoch können zu Bildern keine Notizen hinzugefügt werden. Videos können nicht gespeichert oder annotiert werden. Notizen können außerdem nur Text und keine Hyperlinks, Bilder oder Videos beinhalten. Als Übersicht über alle Annotationen auf einer Webseite kann eine Liste angezeigt werden. Diese Liste ist auch in der digitalen Bibliothek direkt bei dem entsprechenden Bookmark einsehbar (siehe Abb. 2.6). Noch nicht als Bookmark gespeicherte Webseiten werden beim Erstellen einer Annotation automatisch abgespeichert.

2.3.2 *scrible*

scrible ist ebenfalls wie *Diigo* eine Webanwendung, die es ermöglicht, Online-Annotationen in Form von Text-Hervorhebungen und Notizen zu erstellen und Webseiten als Bookmarks in einer digitalen Bibliothek zu speichern und zu organisieren. Die Anwendung *scrible* wurde etwa sechs Jahre nach *Diigo* im Jahr 2011 veröffentlicht [40]. Auch das Team um *scrible* hat festgestellt, dass, egal zu welchen Themengebieten, zum Großteil nur mehr online recherchiert wird [39]. Die Art, auf die die gewonnenen Informationen organisiert und bearbeitet werden (Ausdrucke auf Papier, ...), ist jedoch vielfach sehr veraltet und umständlich. *scrible* ermöglicht es, dass Nutzerinnen Informationen aus Online-Recherchen auch webbasiert verwalten können. Die Anwendung ist ebenfalls nur in englischer Sprache verfügbar, somit beinhalten auch alle folgenden Screenshots der Anwendung englischen Text.

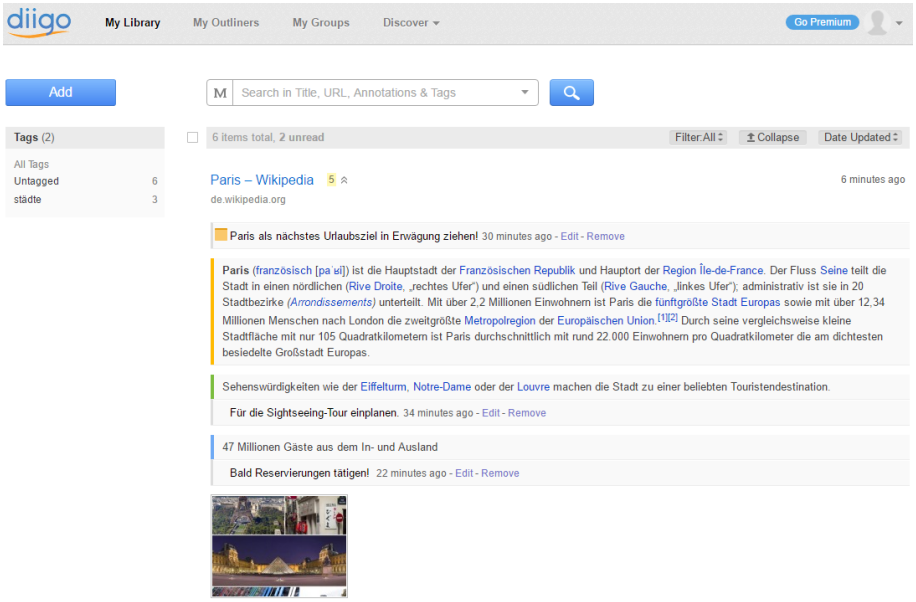


Abbildung 2.6: Screenshot von der digitalen Bibliothek von *Diigo* mit Bookmark und Annotationen.

Installation

Wie bei *Diigo* muss auch bei *scrible* auf der Website der Anwendung[19] ein Benutzerkonto angelegt werden. Hier wird auch die persönliche digitale Bibliothek mit allen Bookmarks und Annotationen verwaltet. Für *scrible* gibt es im Grunde keine unterschiedlichen Tarife [41]. Die Anwendung ist vollkommen frei verfügbar und nicht kostenpflichtig. Es gibt die Option einer Studenten-Version, die mehr Speicherplatz bietet, aber ebenfalls kostenlos ist. Nach der Registrierung wird ein Benutzer-Tool installiert. Dafür gibt es bei *scrible* zwei unterschiedliche Bookmarklets für *scrible* Toolbar (zum Erstellen von Annotationen) oder für *scrible* Bookmark (zum Erstellen von Bookmarks). Das jeweilige Bookmarklet wird einfach in der Browser-Lesezeichenliste abgelegt. Darüber hinaus gibt es auch ein Bookmarklet für *iPad* und eine Browser-Erweiterung für *Google Chrome*, welche für diese Arbeit verwendet wird. Mit der *scrible* Toolbar kann ein Menü eingeblendet werden, mit welchem auf aktuell besuchten Webseiten Annotationen hinzugefügt werden können (siehe Abb. 2.7).

[19] http://www.scrible.com/

Abbildung 2.7: Screenshot der *scrible Toolbar* zum Erstellen von Annotationen.

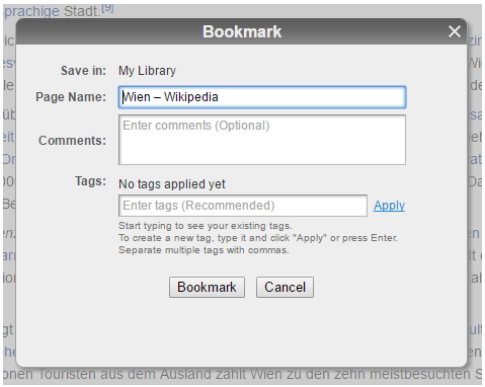


Abbildung 2.8: Screenshot von den Einstellungsmöglichkeiten beim Speichern von Bookmarks bei *scrible*.

Funktionen

scrible unterstützt zwar weniger Funktionen als *Diigo* (keine Screenshot-Funktion, Bookmark kann nicht als ungelesen markiert werden, ...), bietet aber umfassendere Möglichkeiten bei der Annotation von Webseiten. Die beiden wichtigsten Funktionen zum Hinzufügen von Annotationen und zum Speichern von Bookmarks sind aber auch bei *scrible* vorhanden. Alle Funktionalitäten zum Annotieren einer Webseite befinden sich bereits im Hauptmenü (siehe Abb. 2.7). Eine Webseite kann über das Bookmark-Symbol als Link in der persönlichen digitalen Bibliothek abgespeichert werden. Dafür öffnet sich ein neues Fenster mit weiteren Einstellungsmöglichkeiten (siehe Abb. 2.8). Die aktuelle Webseite kann über *Twitter*, *Facebook* und per E-Mail mit anderen Personen geteilt werden. Außerdem gibt es auch bei *scrible* die Möglichkeit, die Webseite mit allen hinzugefügten Annotationen über einen speziellen Link zu teilen. Wie bei *Diigo* kann auch bei *scrible* die gesamte Webseite als Kopie abgespeichert werden. Somit können auch hier Informationen von sich regelmäßig ändernden Webseiten bewahrt werden. Bei *scrible* ist nur ein Klick auf das Browser-Icon notwendig, um das Menü mit allen weiteren Funktionen einzublenden. Um das Menü zum Annotieren von Webseiten einzublenden, sind bei *Diigo* zwei Klicks nötig.

Bookmarks: Beim Abspeichern von Webseiten als Bookmark können bei *scrible* nur Seitenname (Titel), Kommentare (Beschreibung) und Tags an-

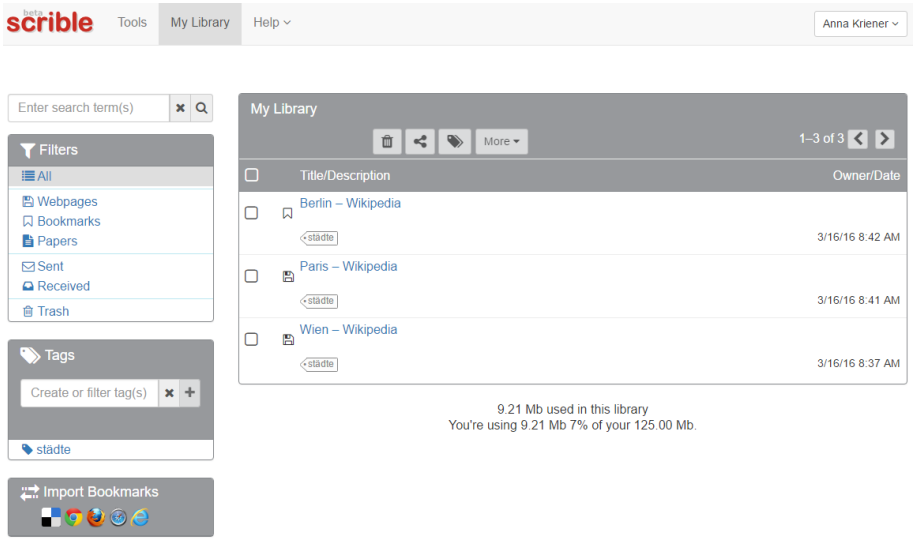


Abbildung 2.9: Screenshot von der digitalen Bibliothek von *scrible*.

gegeben werden (siehe Abb. 2.8). In der digitalen Bibliothek werden Bookmarks als Links in einer Liste dargestellt (siehe Abb. 2.9). Hier können Bookmarks auch einfach verwaltet werden (bearbeiten, löschen, ...). Volltextsuche (in als Kopie gespeicherten Webseiten) ist bei *scrible* auch in der kostenlosen Version verfügbar. Bookmarks können auch per *Drag and Drop* zur digitalen Bibliothek hinzugefügt werden, indem eine URL oder ein Link einfach in die Bibliothek gezogen wird.

Annotationen: Auch bei *scrible* können Annotationen direkt im Kontext platziert werden. Selektierter Text kann in 32 verschiedenen Farben hervorgehoben werden. Darüber hinaus kann auch die Schriftfarbe des selektierten Textes geändert werden (32 Farben), oder der Text kann unterstrichen, fett, kursiv oder durchgestrichen formatiert werden (siehe Abb. 2.10). Zu jeder Text-Formatierung können auch im Nachhinein Notizen in Form von Haftnotizzetteln hinzugefügt werden. Allgemeine Notizen für die gesamte Webseite sind nicht möglich. Wie bei *Diigo* können Notizen auch verschoben und überall auf der Webseite platziert werden. Hier ist es aber auch möglich, die Höhe und Breite der Notiz selbst anzupassen. Alle Farben können im Nachhinein geändert werden und bei *scrible* ist es auch möglich, Annotationen rückgängig zu machen und wiederherzustellen. Annotationen können auch ein- und ausgeblendet werden und Notizzettel können minimiert und maximiert werden. Es ist auch möglich, eine personalisierte Legende zu den jeweiligen Annotationen zu erstellen (gelbe Hervorhebung steht z. B. für „wichtige Informationen“). Bilder und Videos können von *scrible* nicht annotiert werden und es ist auch nicht möglich, Hyperlinks, Bilder oder Videos zu Notizen

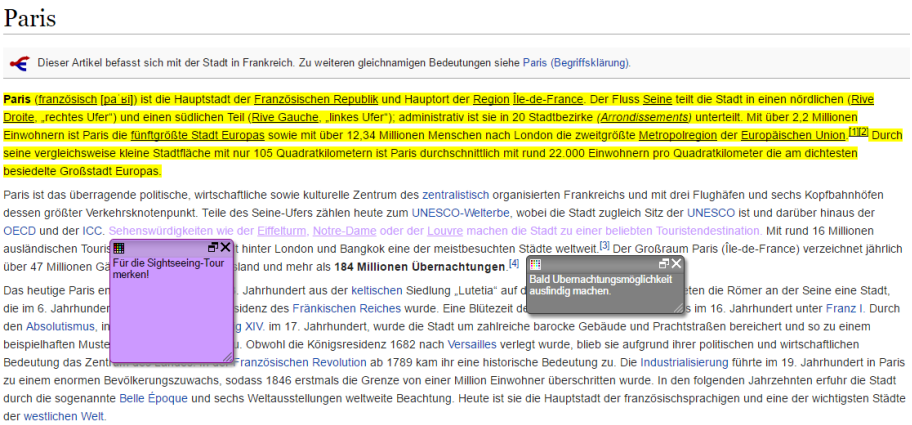


Abbildung 2.10: Screenshot von einem mit Annotationen versehenen Wikipedia-Eintrag [38] von *scrible*.

hinzuzufügen. In der digitalen Bibliothek wird zwar angezeigt, dass Annotationen bei einem Bookmark vorhanden sind, diese können dort aber nicht eingesehen werden.

2.3.3 *Coggle*

Coggle ist eine frei verfügbare Webanwendung, die es ermöglicht, im Browser individuelle Mindmaps[20] zu erstellen. Da *Coggle* ein Mindmapping-Tool ist, können im Gegensatz zu herkömmlichen linearen Textdokumenten hierarchische Strukturen erstellt werden. Darüber hinaus ist es möglich, gemeinsam in Teams gleichzeitig an einer Mindmap zu arbeiten und diese auch mit anderen zu teilen. Nach relativ kurzer Einarbeitungszeit können mit *Coggle* sehr schnell und einfach auch komplexe Mindmaps erstellt werden. *Coggle* ist ebenfalls nur in englischer Sprache verfügbar, weswegen auch alle folgenden Screenshots der Anwendung englischen Text beinhalten.

Erste Schritte

Um *Coggle* zu verwenden, ist keine Installation notwendig. Es wird lediglich ein *Google*-Konto benötigt, das zur Anmeldung auf der Webseite der Anwendung[21] verwendet wird. Danach kann zwischen drei verschiedenen Tarifen (Free, Awesome und Organization) entschieden werden, wobei für diese Arbeit der kostenlose Tarif Free gewählt wurde [33]. Nach der Anmeldung bzw.

[20] Das Wort Mindmap setzt sich zusammen aus den englischen Worten mind (Gedanken) und map ([Land]Karte) und beschreibt eine Möglichkeit, Gedanken, Ideen, Themen usw. visuell darzustellen.

[21] https://coggle.it/

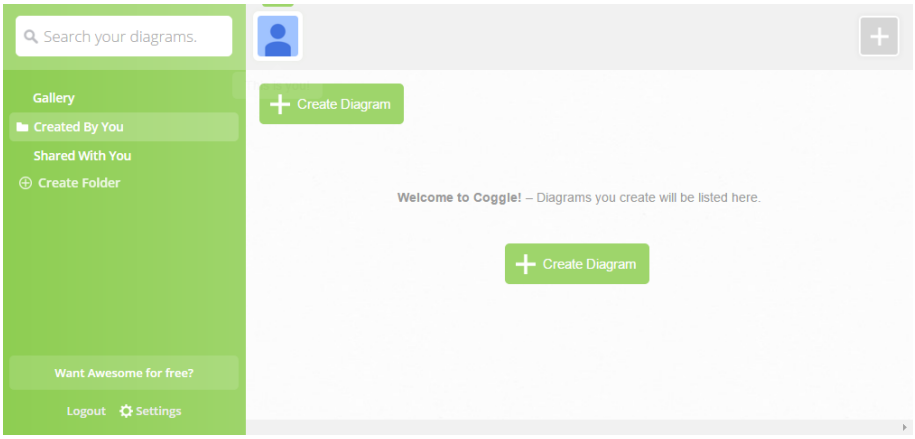


Abbildung 2.11: Screenshot der Webanwendung *Coggle* nach der Anmeldung.

Registrierung kann sofort mit dem Erstellen neuer Mindmaps begonnen werden, indem die Funktion „Create Diagram“ gewählt wird (siehe Abb. 2.11).

Funktionen

Standardmäßig wird bei *Coggle* automatisch ein erster Knoten als Ursprung erstellt, von dem aus sich alle weiteren Knoten hierarchisch verzweigen. Ausgehend vom Ursprung können nach oben, unten, rechts und links über Klick auf den Plus-Button neue Kind-Knoten hinzugefügt werden. Zu jedem Kind-Knoten können wiederum beliebig viele weitere Kind-Knoten hinzugefügt werden. Alle Knoten können verschoben und individuell angeordnet werden. Die Beschriftung jedes Knotens kann mittels Klick darauf einfach geändert werden. Bei *Coggle* ist es außerdem möglich, neben Text auch Hyperlinks oder Bilder zu einem Knoten hinzuzufügen, indem *Markdown* [34] verwendet wird. Darüber hinaus können mittels *Markdown* auch einfache Textformatierungen (fett, kursiv usw.) vorgenommen werden. Die Farben der einzelnen Zweige können entweder über direkten Klick auf eine Linie oder über ein Kontextmenü geändert werden (siehe Abb. 2.12). In diesem Menü sind auch andere Funktionen verfügbar, wie etwa das Hinzufügen, Löschen, Verschieben oder Kopieren von Knoten, das Hinzufügen von Kommentaren zum aktuellen Knoten oder das Erstellen von Links zwischen verschiedenen Knoten. Diese Links zwischen nicht verwandten Knoten werden durch strichlierte Linien dargestellt und ebenfalls mittels *Markdown* ausgezeichnet. Neben Kontextmenü und Mausbedienung sind zusätzlich auch einige Tastenkürzel und -kombinationen verfügbar, wie etwa die Taste Strg in Kombination mit dem Mausrad, um die Mindmap zu vergrößern oder zu verkleinern. In Abbildung 2.13 ist eine mit *Coggle* erstellte Mindmap zu sehen. Fertige Mindmaps kön-

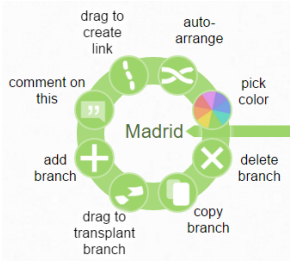


Abbildung 2.12: Screenshot des Kontextmenüs in *Coggle*.

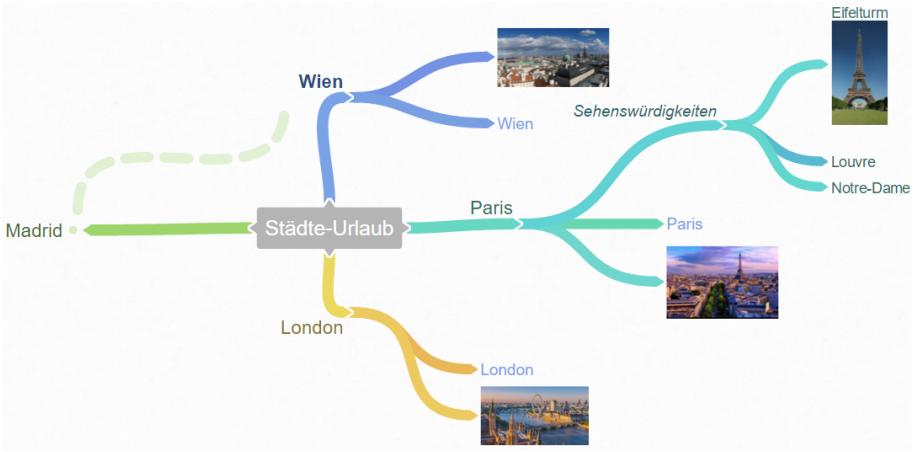


Abbildung 2.13: Screenshot einer Mindmap in *Coggle*.

nen als PDF, PNG, reines Textdokument oder als XML-basierte mm-Datei heruntergeladen werden.

2.3.4 *VideoAnt*

Die webbasierte Anwendung *VideoAnt* macht es möglich, zu beliebigen Zeitpunkten eines Videos Kommentare hinzuzufügen. Videos werden in die persönliche Online-Video-Bibliothek hinzugefügt und können dort verwaltet und annotiert werden. *VideoAnt* bezeichnet annotierte Videos als *Ants* und die Online-Bibliothek als *Ant Farm*. Es werden alle öffentlich verfügbaren Video-Dateien und *YouTube*-Videos unterstützt. *VideoAnt* wurde im Jahr 2015 veröffentlicht und ist kostenlos verfügbar [45].

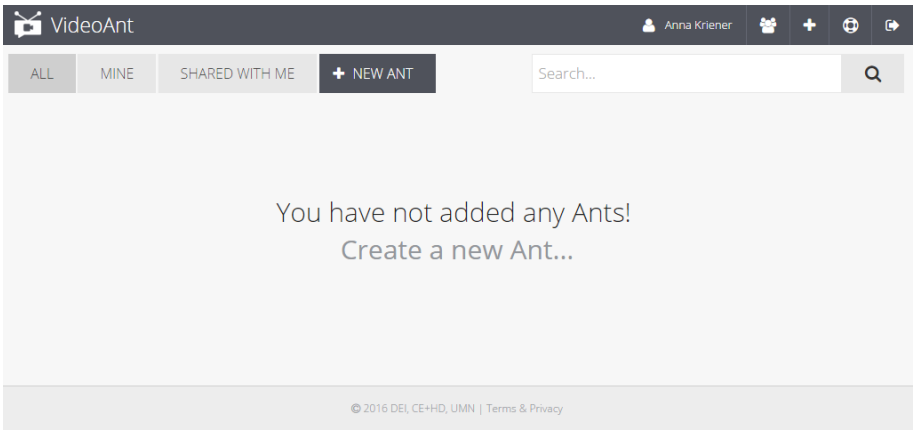


Abbildung 2.14: Screenshot der leeren Online-Video-Bibliothek nach der Anmeldung in *VideoAnt*.

Erste Schritte

Um *VideoAnt* zu nutzen, ist keine Installation notwendig. Zur Anmeldung auf der Webseite der Anwendung[22] wird entweder ein *Google*- oder ein *Facebook*-Konto verwendet. Nach der Anmeldung können in der persönlichen *Ant Farm* mit Klick auf den Button „New Ant“ sogleich neue Videos hinzugefügt werden (siehe Abb. 2.14).

Funktionen

Bei *VideoAnt* können zu jeder beliebigen Stelle im Video Annotationen bzw. Kommentare hinzugefügt werden. Mit Klick auf das Symbol „Add Annotation“ kann an der aktuellen Stelle ein neuer Kommentar erstellt werden. Dabei pausiert das Video und es erscheint rechts daneben ein Eingabefeld für persönliche Notizen (siehe Abb. 2.15). Gleichzeitig wird an der entsprechenden Stelle der Zeitleiste eine Markierung gesetzt. Alle gespeicherten Annotationen werden rechts neben dem Video aufgelistet und können einzeln wieder gelöscht und im Nachhinein auch bearbeitet werden (siehe Abb. 2.16). Markierungen auf der Zeitleiste können per *Drag and Drop* einfach verschoben werden. Mit Klick auf das Symbol „Open Settings“ können Einstellungen zum Video vorgenommen werden, wie etwa das Ändern des Titels oder das Hinzufügen einer Beschreibung. Hier können die Annotationen auch als Text, RSS, JSON oder XML exportiert werden. Ebenfalls hier ersichtlich ist der Code zum Einbetten des Videos mitsamt allen Annotationen. Außerdem ist es möglich, das Video mit anderen zu teilen oder andere Benutzerinnen oder Benutzergruppen zum Video hinzuzufügen, damit diese ebenfalls Annotationen hinzufügen können.

[22] https://ant.umn.edu/

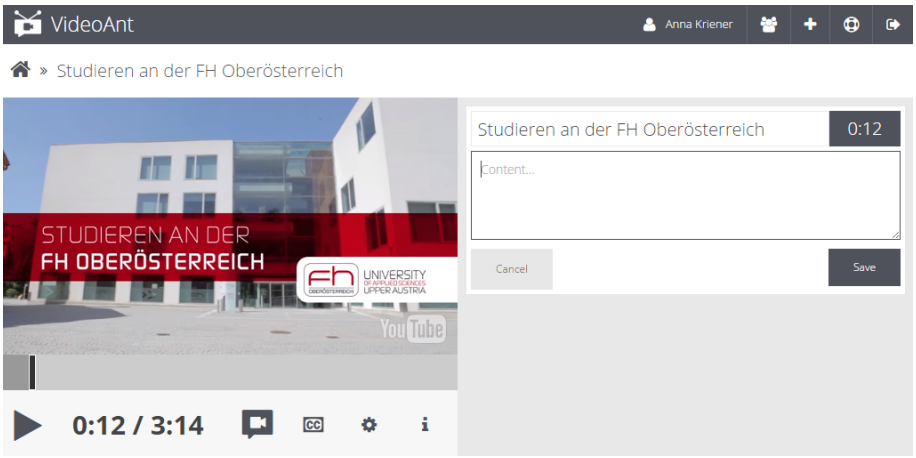


Abbildung 2.15: Screenshot vom Erstellen einer Annotation bei einem Video in *VideoAnt*.

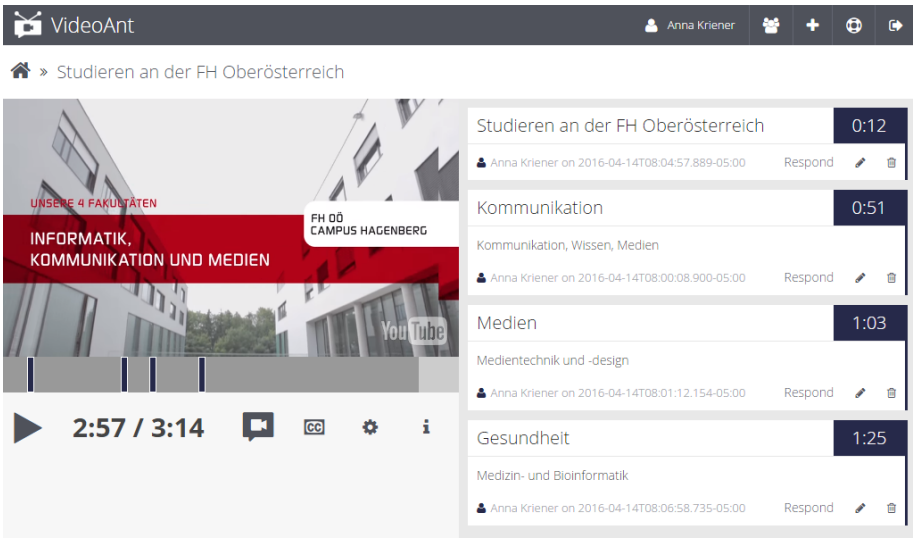


Abbildung 2.16: Screenshot von einem bereits mit mehreren Annotationen versehenen Video in *VideoAnt*.

2.4 Zusammenfassung

Aus allen bisherigen Studien bezüglich Benutzerverhalten im Web und Bookmarks kann resultiert werden, dass wichtige Webseiten oft wiederholt besucht werden, Bookmarks aufgrund des hohen Verwaltungsaufwandes nur wenig genutzt und eher Suchmaschinen zum Wiederfinden von Informationen verwendet werden, obwohl dies oft ineffizient ist. Online-Annotationen werden als potentielle Möglichkeit gesehen, Recherchen im Web und das

Wiederfinden von Informationen zu vereinfachen. Besonders Annotationen, die wie auf Papier auch im Web direkt bei der zu annotierenden Stelle platziert werden können, sind vielversprechend.

Aufgrund der allgemeinen Meinung, dass Online-Recherche einfacher und produktiver gestaltet werden sollte, wurden in den letzten Jahren viele Bookmark- und Annotations-Tools entwickelt, wobei hier auf die Webanwendungen *Diigo* und *scrible* näher eingegangen wurde. Da weder *Diigo* noch *scrible* eine Alternative zur Listendarstellung der Bookmarks anbieten, wurde das Mindmapping-Tool *Coggle* vorgestellt, das in Form von Mindmaps eine andere Möglichkeit zur Visualisierung von Bookmarks darstellt. Die Webanwendung *VideoAnt* wurde vorgestellt, da sie das Annotieren von Videos ermöglicht, was ebenfalls weder bei *Diigo* noch bei *scrible* möglich ist.

Kapitel 3

Eigener Ansatz und technische Gestaltung

Im folgenden Kapitel wird der Entwurf des Prototyps *tisoor* (tool in support of online research), welcher Unterstützung bei der Online-Recherche bieten soll, vorgestellt. Dazu werden die aus dem vorherigen Kapitel gewonnenen Erkenntnisse herangezogen und als Grundlage für die Entwicklung von *tisoor* verwendet. Zuerst wird die Ausgangssituation definiert und die Anforderungen an den Prototyp werden verdeutlicht. Danach folgt die Konzeptbeschreibung zur Erfüllung der Anforderungen und abschließend wird die geplante technische Gestaltung und Architektur des Prototyps beschrieben.

3.1 Ausgangssituation

Diese Arbeit beschäftigt sich mit der Recherche im Web, den damit verbundenen Problemen und den möglichen Lösungen dafür. Das bedeutet, dass auch der eigene Prototyp, wie die in Abschnitt 2.3 beschriebenen Tools, eine webbasierte Anwendung sein wird. Wie bereits in Kapitel 3 erwähnt, treten bei der Recherche im Web ohne Zuhilfenahme von z. B. Lesezeichenmanagern oder entsprechenden anderen Tools oft Schwierigkeiten auf, wie etwa das Wiederfinden von einmal gefundenen Informationen zu einem späteren Zeitpunkt. Um Probleme dieser Art zu lösen, gibt es bereits einige Webanwendungen, die auch zusätzliche Unterstützung bei der Online-Recherche bieten. Jedoch sind diese Tools oft in ihrer Funktionalität beschränkt, bzw. widmen sich diese oft nur dem Lösen bestimmter Teilprobleme, wie bereits in Abschnitt 2.3 aufgezeigt wurde.

Um die Situation zu verdeutlichen, soll hier ein Anwendungsfall als mögliches Szenario eingeführt werden. Beispielsweise möchte die Akteurin Susanne ihren Städteurlaub durch Europa planen. Dafür möchte sie

- Informationen zu Unterkünften, Sehenswürdigkeiten, Transfermöglichkeiten usw. sammeln,
- interessante Webseiten abspeichern,
- bestimmte Informationen zu einzelnen Städten gruppieren,
- wichtige Textpassagen und aussagekräftige Bilder oder Videos markieren und kommentieren,
- Verbindungen zwischen Informationen herstellen, um z. B. eine Route zu planen oder um ähnliche Informationen zu vergleichen,
- gesammelte Informationen einfach und nach persönlichen Vorstellungen entsprechend organisieren.

Unter den aktuellen Umständen und mit den momentan verfügbaren Möglichkeiten müsste Susanne mehrere verschiedene Tools verwenden und deswegen auch Informationen doppelt abspeichern und warten, um alle zuvor genannten gewünschten Funktionen nutzen zu können.

Mit der Entwicklung des Prototyps *tisoor* soll ein Tool geschaffen werden, das all diese verschiedenen Funktionalitäten in einer Anwendung vereint. Organisation und individuelle Darstellung von Bookmarks, Annotation von multimedialen Inhalten und Herstellen von Verbindungen zwischen gespeicherten Informationen soll mit Hilfe von nur einem Tool möglich sein.

3.2 Anforderungen

Das generelle Ziel der Umsetzung des Tools *tisoor* ist, dass die Online-Recherche für Webanwenderinnen erleichtert wird, indem zusätzliche Funktionen geboten bzw. unterschiedliche Funktionen vereint werden und damit das Verwenden von mehreren verschiedenen Tools überflüssig wird. Die Definition der folgenden Anforderungen orientiert sich an den gewonnenen Erkenntnissen aus dem vorhergehenden Kapitel.

3.2.1 Individuelle Visualisierung und Verwaltung von Bookmarks

Herkömmliche Lesezeichenmanager wie etwa von *Firefox* oder *Google Chrome* speichern Bookmarks in Form von Listen. Diese Listen werden bei größerer Anzahl von gespeicherten Webseiten jedoch schnell sehr unübersichtlich und sind mit dauerhaftem großen Verwaltungsaufwand verbunden. Es besteht zwar meist die Möglichkeit, Bookmarks in Ordnern zu gruppieren, eine visuelle Darstellung der Bookmarks, die das Wiederfinden einer gespeicherten Webseite unter einer großen Anzahl davon erleichtern könnte, gibt es allerdings nicht.

Neben der Listendarstellung gibt es auch Anwendungen wie etwa *Pinterest* oder *Pocket*, die Bookmarks in Kacheln angeordnet mit einem optiona-

len Bild darstellen. Diese Anordnung kann jedoch von der Benutzerin nicht individuell angepasst werden.

Ein Lesezeichenmanager sollte Bookmarks nicht nur in Listen anzeigen, sondern auch eine alternative visuelle Darstellungsmöglichkeit bieten, die von Benutzerinnen auch individuell angepasst werden kann. Eine grafische Visualisierung von Bookmarks kann auch Gruppierungen, Abhängigkeiten und Verbindungen zwischen einzelnen Bookmarks darstellen. Das Verwalten von Lesezeichen sollte somit intuitiver möglich sein, da Benutzerinnen selbst darüber bestimmen können, wo und wie sie ihre Bookmarks anordnen.

3.2.2 Multimediale Annotationen

Auch zum Erstellen von digitalen Annotationen existieren bereits Anwendungen, wie etwa *Diigo* oder *scrible*. Diese Tools bieten zwar umfassende Funktionalität, wie in Abschnitt 2.3.1 von *Diigo* und in Abschnitt 2.3.2 von *scrible* beschrieben wird, berücksichtigen jedoch weitgehend nur Textinhalte. Für Benutzerinnen besteht daher eine gewisse Einschränkung, wenn es um das Hinzufügen oder Annotieren von multimedialen Inhalten geht.

Die hier angeführten Anforderungen an digitale Annotationen richten sich großteils nach dem in Abschnitt 2.2 erwähnten Annotations-Modell für multimediale Inhalte aus [7], aber auch nach den gewonnenen Erkenntnissen aus Kapitel 2.

Unabhängigkeit von Webseiten

Digitale Annotationen sollen für alle im Web verfügbaren Seiten funktionieren. Benutzerinnen können demnach zu Inhalten jeder beliebigen aufgerufenen Webseite, die Manipulationen durch eine Browser-Erweiterung zulässt, digitale Anmerkungen hinzufügen.

Verschiedene Medientypen

Um das analoge Hinzufügen von Notizen zu Inhalten in z. B. einem Buch auch in der digitalen Welt nachzuempfinden, sollte es auf Webseiten ebenfalls machbar sein, zu verschiedenen Medientypen persönliche Anmerkungen hinzuzufügen. Webnutzerinnen sollen Textpassagen, Hyperlinks, Bilder und Videos gleichermaßen mit individuellen Kommentaren versehen können. Das bedeutet, dass einzelne Teilinhalte einer Webseite annotiert werden können. Darüber hinaus sollte die Möglichkeit bestehen, nicht nur verschiedene Medientypen zu annotieren, sondern auch multimediale Inhalte als Kommentar hinzuzufügen. Somit können Benutzerinnen beliebige Inhalte einer Webseite mit eigenen Texten, Hyperlinks, Bildern oder Videos ergänzen.

Platzierung im Kontext

Da, wie zuvor erwähnt, digitale Anmerkungen zu Teilinhalten einer Webseite hinzugefügt werden können sollen, liegt es nahe, diese Annotationen auch direkt im Kontext zu platzieren. Auch bei herkömmlichen Notizen in beispielsweise einem Schulbuch werden schriftliche Anmerkungen direkt bei der entsprechenden Stelle platziert oder Textpassagen direkt mit einem Leuchtstift markiert. Digitale Annotationen sollen daher ebenfalls direkt bei dem Inhalt, der bearbeitet wird, platziert werden.

Robustheit

Annotationen sollen, auch wenn sich Inhalte auf einer Webseite ändern oder Inhalte verloren gehen, bestehen bleiben. Benutzerinnen können ihre Anmerkungen nach wie vor einsehen, auch wenn Webinhalte, zu denen diese Annotationen hinzugefügt wurden, nicht mehr vorhanden sind. Dies setzt natürlich voraus, dass einmal erstellte Annotationen gespeichert werden. Eine wichtige Anforderung für dynamische Webseiten ist, dass Annotation auch bei verändertem oder gelöschtem Inhalt wiederhergestellt werden können. Diese Anforderung wird in dieser Arbeit jedoch nicht berücksichtigt.

3.2.3 Geräteunabhängigkeit

Ein beschriebenes Problem aus Abschnitt 2.1 ist der Zugriff auf gespeicherte Bookmarks von verschiedenen Geräten. Da Lesezeichen oft direkt im Browser verwaltet werden, können diese auch nur über das Gerät, auf dem dieser Browser installiert ist, abgerufen werden. Für Benutzerinnen ist es daher nicht möglich, von anderen Geräten oder mit anderen Browsern auf das persönliche Bookmark-Archiv zuzugreifen.

Darüber hinaus gibt es Anwendungen, wie etwa *OneNote*[1] von *Microsoft*, *Mendeley*[2] oder *Evernote*[3], die lokal auf verschiedenen Geräten installiert werden können und gespeicherte Recherche-Ergebnisse und Informationen automatisch synchronisieren, um von allen verwendeten Geräten darauf zugreifen zu können.

Ein Bookmark-Archiv basierend auf einer Webanwendung bedarf im Gegensatz zu lokalen Anwendungen keiner Installation und steht trotzdem allen verwendeten Geräten zur Verfügung. Es ist meist nur ein Benutzerkonto notwendig, um auf das persönliche Bookmark-Archiv von überall aus zugreifen zu können. Eine Browser-Erweiterung zum Erstellen von Bookmarks muss zwar bei einem bestimmten Browser installiert werden, kann aber trotzdem von verschiedenen Geräten und Betriebssystemen verwendet werden.

[1] http://www.onenote.com/
[2] https://www.mendeley.com/
[3] https://evernote.com/intl/de/

3.2.4 Benutzerfreundlichkeit

Usability steht nicht nur für die Benutzerfreundlichkeit einer Anwendung, sondern definiert auch deren Qualität. Eine Benutzerin sollte sofort erkennen können, welchen Nutzen eine Anwendung hat und wie sie funktioniert [18, S. 303–307]. Von Nielsen wird in [13, S. 26] der Begriff Usability als eine Kombination von mehreren Komponenten definiert und nicht als eine einzelne Eigenschaft eines User-Interfaces. Eine Anwendung sollte demnach einfach zu erlernen und effizient im Gebrauch sein, sodass eine Benutzerin produktiv damit umgehen kann. Eine Benutzerin sollte sich an die Bedienung einer Anwendung einfach erinnern können, damit diese nicht immer wieder neu erlernt werden muss. Des Weiteren sollte ein System eine geringe Fehlerrate aufweisen und für Benutzerinnen angenehm zu gebrauchen sein.

3.3 Konzept

Im Folgenden wird die geplante Herangehensweise zur Umsetzung des Prototyps *tisoor* vorgestellt. Wie bereits in Abschnitt 3.1 festgestellt, soll der Prototyp als Webanwendung umgesetzt werden, da dies die naheliegendste und zweckdienlichste Lösung ist.

3.3.1 Geräteunabhängigkeit

Wie zuvor bereits erwähnt, soll für den eigenen Prototyp eine Webanwendung realisiert werden, die als vorteilhafte Eigenschaft bereits Geräteunabhängigkeit mitbringt. Das Bookmark-Archiv soll als Website mit *Symfony* umgesetzt werden und sollte deshalb von unterschiedlichen Browsern und Geräten aufgerufen werden können. Bookmarks und Annotationen sollen mittels einer *Google Chrome* Browser-Erweiterung erzeugt werden können. Im Folgenden wird die geplante Architektur der Anwendung *tisoor* beschrieben.

3.3.2 Architektur

Die Webanwendung *tisoor* soll grundsätzlich aus zwei unterschiedlichen Teilen bestehen, dem Client und dem Server, die miteinander in Verbindung stehen und Informationen austauschen. Damit Webnutzerinnen beliebige Webseiten manipulieren können, also Annotationen hinzufügen können, ist eine Browser-Erweiterung notwendig. Browser-Erweiterungen fügen mittels JavaScript zusätzliche Funktionalität zum Browser hinzu. Dies ermöglicht es, einerseits Inhalte von besuchten Webseiten auszulesen und andererseits Inhalte und Funktionalität in besuchte Webseiten einzufügen. Diese Browser-Erweiterung ist, wie in Abbildung 3.1 zu sehen, der HTTP-Client. HTTP steht in diesem Fall dafür, dass der Client HTTP benutzt, um mit dem Server zu kommunizieren und Daten auszutauschen.

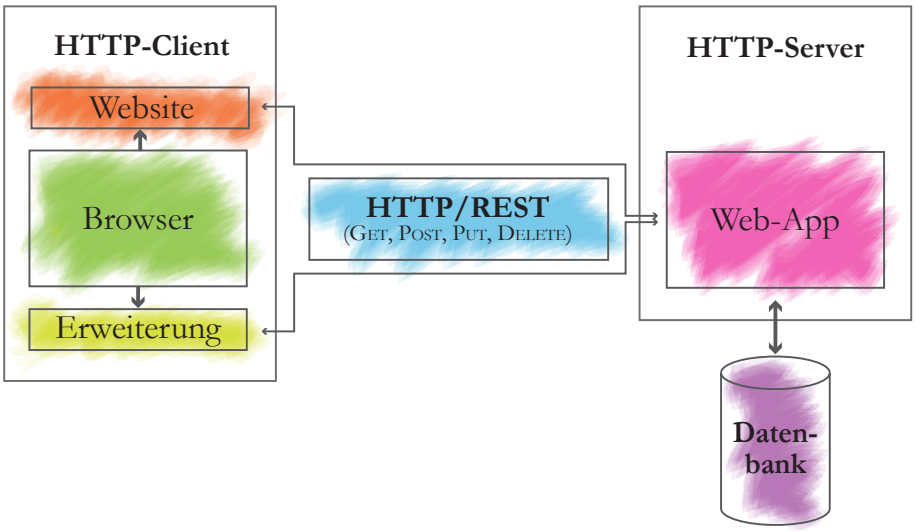


Abbildung 3.1: Architektur des Prototyps *tisoor*.

Der zweite Teil der Anwendung besteht aus dem HTTP-Server bzw. der Web-Applikation. Die Web-App steht in Verbindung mit der Datenbank und speichert dort neue Daten ab oder ruft benötigte Daten auf. Diese Daten (z. B. bereits erstellte Annotationen von einer Webseite) stellt der Server dem Client zur Verfügung und übermittelt diese via HTTP. Andererseits sendet der Client neue Daten (z. B. neue Annotationen) an den Server und die Web-App speichert diese dann in die Datenbank.

Die Kommunikation zwischen Client und Server findet, wie schon gesagt, über HTTP statt und basiert auf der REST Architektur [6]. REST ist ein leichtgewichtiger Architekturstil für Kommunikation zwischen Netzwerkanwendungen und nutzt meist einfache HTTP-Aufrufe, um Daten zu erzeugen (POST oder PUT), zu lesen (GET) oder zu löschen (DELETE).

Darüber hinaus sollen Benutzerinnen die auf dem Webserver abgelegte Website im Browser aufrufen können, um sich dort zu registrieren oder anzumelden und um auf dieser Website das persönliche Bookmark-Archiv einzusehen und zu verwalten. Der Webserver liefert auch dieser Website die benötigten Daten aus der Datenbank. Eine Benutzerin soll sich dementsprechend mit E-Mail-Adresse, Benutzername und Passwort anmelden, ihre Lesezeichen nachträglich bearbeiten und verwalten und sich wieder abmelden können.

Datenbankmodell

Abbildung 3.2 repräsentiert das Datenbankmodell für den Prototyp *tisoor*. Das Datenbankmodell ist ausgelegt für die Umsetzung einer relationalen tabellenbasierten Datenbank. Eine zentrale Tabelle ist die *Bookmark*-Tabelle

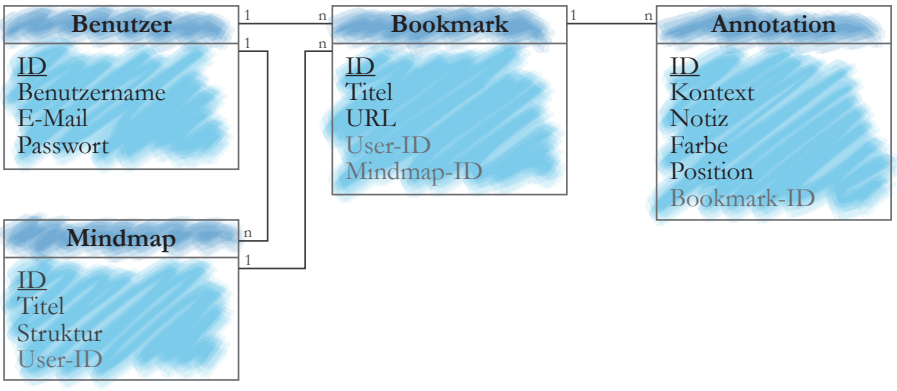


Abbildung 3.2: Datenbankmodell für die relationale Datenbank des Prototyps *tisoor*.

zum Abspeichern der Lesezeichen. Hier sollen Titel und URL eines Lesezeichens abgelegt und die Verbindung zur entsprechenden Benutzerin gelegt werden. Die *Benutzer*-Tabelle wird benötigt, um Lesezeichen, Annotationen und Mindmaps bestimmten Nutzerinnen zuordnen zu können und um die Anwendung für mehrere verschiedene Benutzerinnen zugänglich machen zu können, ohne dass diese sich in die Quere kommen. In dieser Tabelle sollen die wichtigsten Informationen zur Benutzerin, wie Benutzername, E-Mail-Adresse und Passwort, abgespeichert werden. Für das Abspeichern von Annotationen ist ebenfalls eine Tabelle notwendig, die *Annotation*-Tabelle. In dieser sollen der Kontext, also der annotierte Inhalt (Text, Bild-URL usw.), die hinzugefügte Notiz (falls vorhanden) und die Farbe der Annotation gespeichert werden. Außerdem soll bei Annotationen, die zu Textinhalten hinzugefügt wurden, die Position dieses Textes auf der Webseite gespeichert werden. Da Annotationen zu einer bestimmten Webseite gehören, soll hier auch eine Verbindung zum entsprechenden Lesezeichen gespeichert werden. In der *Mindmap*-Tabelle soll der Titel der Mindmap und die gesamte Struktur der Mindmap als JSON-Array gespeichert werden. Wiederum soll hier die Verbindung zur entsprechenden Benutzerin gesetzt werden. Zu erwähnen ist auch, dass in der *Bookmark*-Tabelle eine Beziehung zur *Mindmap*-Tabelle gesetzt wird, da Bookmarks zu einer bestimmten Mindmap hinzugefügt werden können. Die relationale Datenbank soll auf Basis dieses Datenbankmodells realisiert werden und die Möglichkeit bieten, Benutzerdaten (Benutzername, E-Mail, Passwort), Bookmark-Daten (Titel, URL), Annotations-Daten (Kontext, Inhalt, Position usw.), Mindmap-Daten und deren Beziehungen zueinander zu verwalten.

3.3.3 Individuelle Visualisierung und Verwaltung von Bookmarks

Lesezeichen sollen nicht nur in einer standardmäßigen Liste angezeigt, sondern auch in einer Mindmap visuell angeordnet werden können. Dadurch soll die Möglichkeit bestehen, Bookmarks individuell und einprägsam zu gruppieren und später einfacher abzurufen. Mit *Coggle* können, wie in Abschnitt 2.3.3 beschrieben, individuelle Mindmaps erstellt werden. Zusammengehörige Informationen und deren Beziehungen zueinander sind somit auf einen Blick sichtbar. Benutzerinnen sollen dabei, wie bei *Coggle*, Knoten einfach hinzufügen und wieder löschen können. Knoten sollen Text, Hyperlinks (Bookmarks), Bilder und Videos enthalten können und im Nachhinein noch bearbeitet werden können. Es soll möglich sein, dass ein Knoten auch eine Kombination aus verschiedenen Medientypen (Text mit Bild) beinhaltet. Zusätzlich wäre es bei größeren Mindmaps mit vielen Knoten von Vorteil, wenn einzelne Knoten, aber auch einzelne Teile der Mindmap verschoben werden können oder die gesamte Mindmap bewegt werden kann. Eine entworfene Mindmap kann abgespeichert und zu einem späteren Zeitpunkt wiederhergestellt werden, um diese später einsehen, aber auch, um im Nachhinein Änderungen oder Erweiterungen vornehmen zu können. Natürlich soll die Möglichkeit bestehen, zu unterschiedlichen Themengebieten auch mehrere unterschiedliche Mindmaps zu erstellen.

3.3.4 Multimediale Annotationen

Mit *tisoor* soll es möglich sein, multimediale Annotationen direkt im Kontext zu multimedialen Inhalten einer beliebigen Webseite hinzuzufügen. Dies soll mithilfe einer *Google Chrome* Browser-Erweiterung realisiert werden, die die Benutzerin einfach beim *Google Chrome* Browser installieren kann. Annotationen sollen über ein einheitliches Menü für Textpassagen, Hyperlinks, Bilder und Videos gemacht werden können. Es soll möglich sein, Text farbig zu markieren, in Anlehnung an das Markieren von Text auf Papier mit einem Leuchtstift. Dazu wird der zu markierende Text selektiert und anschließend soll zwischen vier verschiedenen Farben gewählt werden können. Bilder und Videos sollen ebenfalls markiert werden können – diese sollen farbig umrahmt werden. Somit werden Markierungen direkt im Kontext angezeigt. Alle markierten Inhalte sollen gespeichert und auch im persönlichen Archiv von der Benutzerin eingesehen werden können. Neben Markierungen sollen auch multimediale Kommentare mithilfe von *Markdown* für ausgewählte Inhalte möglich sein. Bei Text-Annotationen soll die Notiz als kleines Symbol im Kontext repräsentiert werden. Dieses Symbol soll später auch zum Bearbeiten der Notiz dienen. Bei Videos sollen Notizen zu einzelnen Bildern an beliebigen Stellen im Video hinzugefügt werden können. Alle Annotationen, die einmal erstellt wurden, sollen in der Datenbank abgespeichert werden

und sollen, wenn sie nicht von der Benutzerin selbst gelöscht werden, dauerhaft verfügbar sein. Das heißt, dass Inhalte, zu denen Anmerkungen gemacht wurden, auch nachdem sie in der originalen Quelle gelöscht wurden, in der Datenbank vorhanden und von der Benutzerin abrufbar sein sollen. Die Position von Annotationen bei Bildern und Videos soll über deren eindeutige URL festgestellt werden. Bei Text soll eine Suchfunktion verwendet werden, um Anmerkungen bei der richtigen Textstelle wiederherzustellen. Bei verändertem oder gelöschtem Inhalt auf einer Webseite können Annotationen auf diese Weise nicht wiederhergestellt werden. Eine Lösung dafür wäre, wie bei *Diigo*, Webseiten komplett abzuspeichern und bei festgestellten Unterschieden zwischen Kopie und Original, Annotationen in der Kopie wiederherzustellen. Dieser Lösungsansatz wird in dieser Arbeit jedoch nicht umgesetzt.

3.3.5 Benutzerfreundlichkeit

Da die in Abschnitt 2.3 analysierten Tools *Diigo*, *scrible*, *Coggle* und *VideoAnt* bereits getestete und etablierte Anwendungen sind, werden diese auch in puncto Benutzerfreundlichkeit und Interface als Orientierung für den eigenen Prototyp *tisoor* herangezogen. Um gewünschte Ergebnisse zu erzielen, sollen nur wenige einfache Schritte notwendig sein. Das bedeutet, dass mögliche Funktionen klar ersichtlich und deren Ergebnisse im Vorhinein abschätzbar sein sollen, und man das Ziel nur mit wenigen Mausklicks erreicht. Es soll auch für computertechnisch nicht versierte Benutzerinnen keine Einarbeitungszeit, Dokumentation oder Hilfestellung Dritter notwendig sein, um die Anwendung zu bedienen. Dafür sollen die Verwendung von vertrauten Konventionen, intuitiven Bedienmöglichkeiten und das grafisch reduzierte, simpel gestaltete Interface sorgen.

Annotationen

Ähnlich wie bei *Diigo* soll das Interface zum Erstellen von Annotationen lediglich aus zwei Buttons bestehen, die entweder zum Hinzufügen von Notizen oder zum Hervorheben eines selektierten Inhaltes dienen. Dieses Interface soll für Text, Bilder und Videos gleich sein. Bei bereits markierten Inhalten soll sich das Interface auf drei Buttons ändern, von denen einer zum Ändern der Farbe, der zweite zum Hinzufügen von Notizen und der dritte zum Löschen der Markierung verwendet werden soll. Wurde bereits eine Notiz erstellt, sollen im Interface wiederum nur zwei Buttons erscheinen, die das Ändern der Farbe und das Löschen der gesamten Annotation möglich machen. Ein grafischer Entwurf der einzelnen Buttons ist in Abbildung 3.3 zu sehen. Zum Erstellen von Notizen soll ein einfaches Textfeld erscheinen, in dem normale Textinhalte oder Hyperlinks, Bilder und Videos in Form von *Markdown*, wie bei *Coggle*, hinzugefügt werden können. Notizen sollen nach

(a) (b)

Abbildung 3.3: Grafischer Entwurf (a) von den Buttons, die das Interface zum Erstellen von Annotationen bilden. Der linke Button soll zum Hinzufügen von Notizen dienen, der Button rechts daneben soll zum Ändern der Farbe verwendet werden. Mit dem nächsten Button sollen Annotationen wieder gelöscht werden könne und der äußerst rechte Button soll zum Erstellen von Markierungen dienen. Der rechte Entwurf (b) zeigt die Liste, in der alle bereits erstellten Annotationen gesammelt angezeigt werden können. Textquelle [38] und Bildquelle [20].

dem Speichern als kleine Symbole direkt beim Inhalt dargestellt werden und sollen via Klick auf dieses Symbol bearbeitet oder gelöscht werden können. Dies ist auch in Abbildung 3.4 zu sehen, die einen Entwurf einer Webseite zeigt, die mit *tisoor* bearbeitet wird.

Bei Videos sollen, wie bei *VideoAnt*, einzelne Bilder bzw. bestimmte Stellen im Video kommentiert werden können. Dazu soll die Notiz in Kombination mit dem aktuellen Zeitpunkt im Video gespeichert werden. Gleichzeitig soll eine Markierung an der entsprechenden Stelle auf der Zeitleiste gesetzt werden. Mithilfe dieser Markierungen sollen erstellte Notizen nochmals eingesehen, bearbeitet oder gelöscht werden können.

Alle Annotationen auf einer Webseite sollen über ein zentrales Menü, dargestellt in einer Liste, gesammelt angezeigt werden können. Annotationen sollen automatisch gespeichert werden und sollen nicht manuell von der Benutzerin abgespeichert werden müssen. Bei einem erneuten Aufruf der Webseite sollen vorhandene Annotationen automatisch abgerufen und wiederhergestellt werden. Wurde die aktuelle Webseite noch nicht als Bookmark gespeichert, soll dies automatisch beim Erstellen der ersten Annotation passieren.

Mindmap

Mindmaps sollen einfach mittels eines Mausklicks erstellt werden können. Danach soll automatisch ein erster Knoten erzeugt werden, von dem aus al-

Abbildung 3.4: Beispielhafter Entwurf einer Webseite, die mit *tisoor* bearbeitet wird. Textquelle [38] und Bildquelle [20].

le weiteren Knoten hinzugefügt werden können. Bei jedem Knoten sollen mit Hilfe eines Plus-Buttons beliebig viele Kind-Knoten erzeugt werden können. Alle Knoten, bis auf den ursprünglichen Knoten, sollen über einen Minus-Button wieder gelöscht werden können. Die Benutzerin soll einen Knoten individuell platzieren können, indem sie diesen einfach mit der Maus an eine beliebige Position verschiebt. Mittels Klick auf einen Knoten soll dessen Inhalt angepasst werden können. Hier soll wiederum *Markdown* verwendet werden, um Hyperlinks, Bilder, Videos oder eine Kombination davon einzufügen. Gespeicherte Bookmarks sollen in einer Liste angezeigt werden und von dort einfach per *Drag and Drop* auf einen Knoten in der Mindmap gezogen werden können, um dort als Kind-Knoten hinzugefügt zu werden. Mögliche vorhandene Annotationen sollen automatisch beim entsprechenden Bookmark-Knoten als kleine farbige Kreise dargestellt werden. Zwischen einzelnen Knoten, die nicht direkt miteinander verwandt sind, sollen Assoziationen erstellt werden können. Diese sollen in Form von strichlierten Linien, wie bei *Coggle*, dargestellt werden. Mindmaps sollen von der Benutzerin manuell mittels Button-Klick gespeichert und zu einem späteren Zeitpunkt wieder aufgerufen werden können, um Bookmarks oder Informationen zu finden oder um weiter daran zu arbeiten. In Abbildung 3.5 ist ein Entwurf einer Mindmap, so wie sie in *tisoor* erstellt wird, zu sehen.

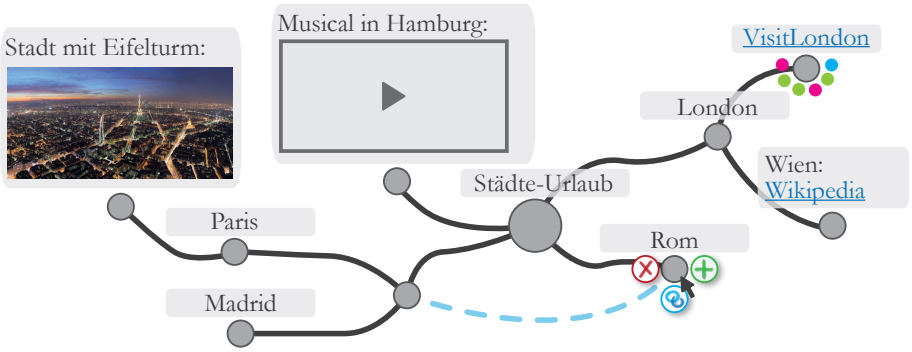


Abbildung 3.5: Grafischer Entwurf einer Mindmap, die mit *tisoor* erstellt wird. Bildquelle [20].

Kapitel 4

Implementierung

In diesem Kapitel wird die Umsetzung des Prototyps *tisoor* auf Basis des im vorherigen Kapitel erarbeiteten Konzepts beschrieben. Zuvor werden die verwendeten Web-Technologien genannt und insbesondere wird danach auch auf die *Google Chrome* Browser-Erweiterung[1] und das PHP-Framework *Symfony*[2] eingegangen. Anschließend folgt die Darlegung der Implementierung, wobei natürlich nicht auf jedes Detail eingegangen wird, sondern nur die wichtigsten Funktionalitäten und programmiertechnischen Lösungen beschrieben werden.

4.1 Verwendete Technologien

Zur Umsetzung des Prototyps *tisoor* wurden mehrere verschiedene Web-Technologien verwendet, wie etwa die JavaScript-Bibliothek *jQuery*[3], die darauf aufbauende User-Interface-Bibliothek *jQuery UI*[4] und das HTML-, CSS- und JavaScript-Framework *Bootstrap*[5]. Mithilfe der JavaScript-Bibliothek *Rangy*[6] wurde das Markieren von Textselektionen umgesetzt. *D3*[7] ist ebenfalls eine JavaScript-Bibliothek, die es möglich macht, Daten mittels HTML, CSS und SVG zu visualisieren. Diese Bibliothek wurde zur Realisierung der Mindmap genutzt.

Wie in Abschnitt 3.3.2 bereits beschrieben, besteht der Prototyp *tisoor* aus einer Browser-Erweiterung (Client) und einer Web-Applikation (Server). Die Browser-Erweiterung wurde als *Google Chrome* Erweiterung umgesetzt und für die Implementierung der Web-App wurde das PHP-Framework *Sym-*

[1] https://developer.chrome.com/extensions
[2] https://symfony.com/
[3] https://jquery.com/
[4] https://jqueryui.com/
[5] http://getbootstrap.com/
[6] https://github.com/timdown/rangy
[7] https://d3js.org/

fony verwendet. Diese beiden Web-Technologien werden im Folgenden näher beschrieben.

4.1.1 Google Chrome Erweiterung

Damit Annotationen für beliebige Webseiten funktionieren, ist es notwendig, dass zusätzliche Funktionalität zu einer Seite hinzugefügt und diese auch manipuliert werden kann. Eine Browser-Erweiterung bietet als Software-Anwendung, die beim Browser installiert wird, diese Möglichkeiten. Zur Implementierung dieser Anwendung werden Standard-Webtechnologien, wie Java-Script, HTML und CSS, verwendet.

Für den Prototyp *tisoor* wird eine Erweiterung für den *Google Chrome* Browser umgesetzt. Das bedeutet zwar, dass diese Erweiterung mit anderen Browsern nicht kompatibel ist, der am meisten genutzte Browser ist jedoch mittlerweile *Google Chrome* [42]. Darüber hinaus unterstützt *Google Chrome* einen Großteil der Funktionalität in Bezug auf CSS (77%) [23], HTML5 (85%) [24], SVG (83%) [27], JS API (88%) [25] und andere Funktionen (88%) [26], wie etwa Keyboard-Events. Ein weiteres Argument für eine *Google Chrome* Erweiterung ist die umfassende Dokumentation, die zur Verfügung steht.

Aufbau

In Abbildung 4.1 ist der grundlegende Aufbau einer *Google Chrome* Erweiterung zu sehen. Der Kern einer Erweiterung ist die Datei manifest.json [30]. Hier werden alle Metadaten und Informationen zur Erweiterung (Name, Beschreibung, Version, ...), die wichtigsten Ressourcen (JavaScript-Dateien, Bilder, ...) und Einstellungen (Berechtigungen, ...) angegeben. Beispielsweise wird in dieser Datei auch das Favicon angeführt, welches dann im Browser-Menü angezeigt wird. Darüber hinaus gibt es eine *Background Page* [28] in Form einer HTML- oder JavaScript-Datei, die das Verhalten der Erweiterung regelt. Diese Datei wird ebenfalls in der Manifest-Datei als background.html oder background.js angeführt. Hier kann z. B. auf einen Klick auf das Browser-Icon reagiert werden, die URL des aktuellen Browser-Tabs kann ausgelesen werden u. v. m.. Neben den beiden bereits genannten notwendigen Dateien kann eine *Google Chrome* Browser-Erweiterung auch weitere HTML-Dateien enthalten, die *UI Pages* [32], die zusätzliche Anpassungen zur Erweiterung enthalten können. Layout und Funktionalität kann hier mittels CSS und JavaScript ebenfalls hinzugefügt werden. Ein Popup bei Klick auf das Browser-Icon wird beispielsweise durch die *UI Page* popup.html realisiert. Die wichtigste Datei für den Prototyp *tisoor* ist die Datei content.js [29]. Sie erlaubt eine Manipulation der aktuellen Webseite, da sie im Kontext dieser Seite ausgeführt wird und somit ein Teil der Webseite ist. Hier kann die DOM-Struktur der Seite gelesen und manipuliert werden, nicht aber die

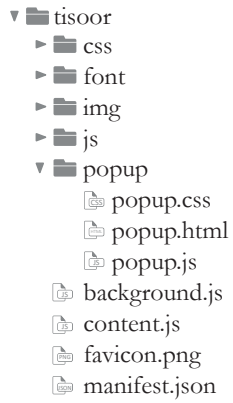


Abbildung 4.1: Die grundlegende Ordnerstruktur und die wichtigsten Dateien einer *Google Chrome* Erweiterung.

der *Background Page*. Da *Background Page* und *Content Script* voneinander abgeschnitten sind, können diese nur über *Messages* [31] miteinander kommunizieren.

4.1.2 Symfony 3.0

Zur Umsetzung der Webanwendung wurde *Symfony* in der momentan aktuellsten Version 3.0 gewählt. *Symfony* bietet hohe Skalierbarkeit und Flexibilität, eine große Menge an wiederverwendbaren, eigenständigen PHP-Komponenten und folgt dem Model-View-Controller-Pattern [10]. Obwohl eine sehr umfangreiche Dokumentation zur Verfügung steht, ist die Einarbeitungszeit in dieses Framework relativ hoch. Für diesen Prototyp konnte jedoch bereits aus vorhergehenden Projekten Erfahrung mit *Symfony* gesammelt werden. *Symfony* ist sehr stabil und zählt nach wie vor zu den am meist genutzten PHP-Frameworks [43].

Architektur

Da *Symfony* das MVC-Pattern unterstützt, ist auch die Architektur und Ordnerstruktur dementsprechend danach ausgerichtet. Steuerung, Präsentation und Modell werden in einem Projekt mittels Controller, Views und Entities getrennt organisiert. Controller sind PHP-Dateien, die Informationen von HTTP-Anfragen auslesen, allfällige Logik enthalten, um Daten zu verarbeiten, und HTTP-Antworten erzeugen und zurücksenden [16, S. 43]. Muss ein Controller Inhalte generieren (HTML, CSS, ...), wird diese Aufgabe an eine Template-Engine weitergeleitet, die Platzhalter in Templates mit Inhalten füllt. *Symfony* verwendet *Twig*[8] als Template-Engine [16, S. 71–73],

[8] http://twig.sensiolabs.org/

was kompakte, einfach lesbare und mit Funktionen erweiterbare Templates möglich macht. Entities sind einfache PHP-Klassen, die Daten enthalten, wie z. B. die Eigenschaften Titel und URL für die Klasse Bookmark. Zusätzlich werden hier Metadaten angegeben, die Regeln dazu enthalten, wie eine Klasse und deren Eigenschaften mittels *Doctrine*[9] auf eine bestimmte Tabelle in der Datenbank abgebildet werden. Diese Metadaten können in Form von YAML oder XML in eigenen Dateien oder direkt innerhalb der Klasse in Form von Annotationen angeführt werden [16, S. 100–102].

Die Ordnerstruktur eines *Symfony*-Projektes entspricht der zuvor beschriebenen Struktur und umfasst standardmäßig sieben Hauptordner (siehe Abb. 4.2). Der Ordner app enthält alle Einstellungen zur Anwendung, wie z. B. Routing-, Datenbank- oder Sicherheits-Einstellungen, sowie die Datei AppKernel.php, in der alle verwendeten Bundles registriert werden. Im Ordner bin befinden sich ausführbare Dateien, wie z. B. die Datei console zum Starten des Servers. Der eigentliche Projekt-Code befindet sich im Ordner src in Form eines oder mehrerer Bundles. Im Ordner tests können z. B. Unit-Tests angelegt werden und der Ordner var enthält automatisch generierte Dateien, wie etwa Cache- oder Log-Dateien. Alle Dateien, die zu fremden Bibliotheken oder Bundles gehören, wie z. B. zu *Doctrine* oder zu *Twig*, und von denen die eigene Anwendung abhängig ist, werden im Ordner vendor gespeichert. Im Ordner web befinden sich öffentlich zugängliche, statische Dateien, wie etwa Bilder, CSS- oder JavaScript-Dateien. Ein Bundle, so wie es für den Prototyp *tisoor* erstellt wurde, enthält standardmäßig einen Ordner Controller für alle Controller des Bundles (z. B. BookmarksController.php) und einen Ordner Resources, in dem alle Templates, Bilder, CSS-, JavaScript- und Konfigurations-Dateien abgelegt werden. Darüber hinaus gibt es einen Ordner Tests in dem wiederum Unit-Tests erstellt werden können. Zusätzlich wurden für diesen Prototyp noch die Ordner Entity, Form und Repository erstellt. Der Ordner Entity enthält die bereits erwähnten Entity-Klassen, wie etwa Bookmark.php. Alle benötigten Formulare können als eigene PHP-Klassen definiert werden, wie z. B. für die Entity Bookmark als BookmarkType.php, und werden im Ordner Form abgelegt. Spezielle Datenbank-Abfragen können in Methoden in einer zur Entity gehörenden Repository-Klasse ausgelagert werden. Diese Repository-Klassen, wie etwa BookmarkRepository.php, werden im Ordner Repository gespeichert.

4.2 Technische Ausführung

Die Basis von *tisoor* ist die mit *Symfony* umgesetzte Web-Applikation. Sie bietet einerseits die für die *Google Chrome* Erweiterung notwendig REST-Schnittstelle und andererseits die für die Benutzerin zugängliche Website mit dem persönlichen Bookmark-Archiv und der Mindmap-Funktion. Zur

[9] http://www.doctrine-project.org/

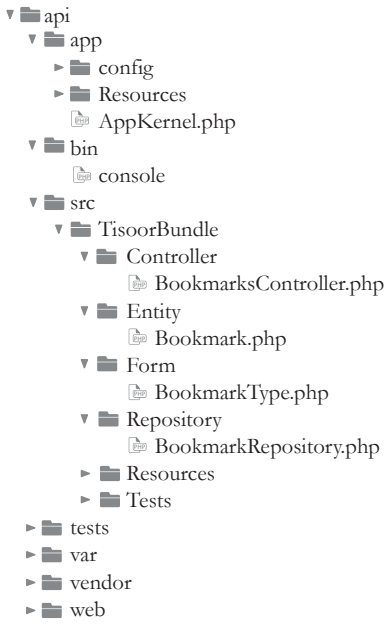


Abbildung 4.2: Die Abbildung zeigt die grundlegende Ordnerstruktur und die wichtigsten Dateien einer *Symfony*-Anwendung.

Realisierung der REST API wurde das bereits existierende *FOSRestBundle* für *Symfony* verwendet. Für eine simple Benutzerinnen-Verwaltung wurde zusätzlich das *FOSUserBundle* installiert. Dies vereinfacht die Implementierung einer grundlegenden Registrier-, Anmelde- und Abmelde-Funktion und das damit verbundene Speichern der Benutzerinnen in eine Datenbank bzw. auch das Abrufen der Benutzerinnen aus der Datenbank.

4.2.1 REST API

Damit Bookmarks und Annotationen für eine Benutzerin abgespeichert werden können, sind die in Abschnitt 3.3.2 erwähnten Tabellen notwendig. Dazu wurde eine neue Datenbank namens *api* angelegt. Anschließend wurden die Klassen *Bookmark*, *Annotation* und *User* mit den entsprechenden Eigenschaften erstellt. In Programm 4.1 ist ein Ausschnitt der Bookmark-Entity zu sehen. Mittels *Doctrine* wurden die Tabellen den Entities entsprechend automatisch in der Datenbank erstellt.

Um durch HTTP-Aufrufe Daten aus der Datenbank lesen und umgekehrt auch in die Datenbank speichern zu können, ist ein Controller nötig, welcher die unterschiedlichen HTTP-Aufrufe behandelt und die Daten entsprechend verarbeitet und versendet. Der Bookmarks-Contoller enthält dazu vier verschiedene Methoden, die jeweils auf GET-, POST-, PUT- und DELETE-Aufrufe unterschiedlich reagieren. In Programm 4.2 ist zu sehen, was pas-

```php
// TisoorBundle/Entity/Bookmark.php

/**
 * Bookmark
 *
 * @ORM\Table(name="bookmark")
 * @ORM\Entity(repositoryClass="TisoorBundle\Repository\BookmarkRepository")
 */
class Bookmark {
    /**
     * @var int
     * @ORM\Column(name="id", type="integer")
     * @ORM\Id
     * @ORM\GeneratedValue(strategy="AUTO")
     */
    private $id;

    /**
     * @var string
     * @ORM\Column(name="title", type="string", length=255)
     */
    private $title;

    /**
     * @var string
     * @ORM\Column(name="url", type="string", length=255)
     */
    private $url;
}
```

Programm 4.1: Ausschnitt aus der Bookmark-Klasse mit den Eigenschaften *id*, *title* und *url* und den Metadaten zur Abbildung in der Datenbank.

siert, wenn eine Benutzerin die URL http:localhost:8000/bookmarks/1 aufruft. Es wird in der Datenbank nach dem Lesezeichen mit der ID 1 gesucht und falls es existiert als JSON-Array (mit allen Eigenschaften) zurückgegeben. Bei POST-Anfragen ist zusätzlich ein Formular (BookmarkType.php) notwendig, das es ermöglicht, die mitgesendeten Daten auszulesen und in eine Bookmark-Instanz zu speichern. Dieses neue Lesezeichen wird anschließend in die Datenbank gespeichert. PUT-Aufrufe werden ähnlich behandelt wie POST-Aufrufe, nur wird zusätzlich die ID des Lesezeichens mitgesendet, um das entsprechende Lesezeichen in der Datenbank finden und aktualisieren zu können. Bei DELETE-Aufrufen wird ebenfalls die ID mitgesendet und das entsprechende Lesezeichen wird in der Datenbank gesucht und gelöscht. Die gleiche Struktur wurde auch für Annotationen umgesetzt.

Da es in bestimmten Fällen notwendig ist, Bookmarks und Annotationen nicht nur über deren ID ausfindig zu machen, sondern auch über eine Kom-

```
1 // TisoorBundle/Controller/BookmarksController.php
2
3 public function getAction($id) {
4     /* "get_bookmark"      [GET] /bookmarks/{id} */
5     $bookmark = $this->getDoctrine()->getRepository("TisoorBundle:
      Bookmark")->find($id);
6
7     if (!$bookmark instanceof Bookmark) {
8         throw new NotFoundHttpException('Bookmark not found');
9     }
10
11    return array('bookmark' => $bookmark);
12 }
```

Programm 4.2: Methode aus der BookmarksController-Klasse zur Behandlung von GET-Aufrufen.

```
1 # app/config/security.yml
2
3 access_control:
4     - { path: ^/login, roles: IS_AUTHENTICATED_ANONYMOUSLY }
5     - { path: ^/register, roles: IS_AUTHENTICATED_ANONYMOUSLY }
6     - { path: ^/archive, roles: ROLE_USER }
7     - { path: ^/mindmap, roles: ROLE_USER }
8     - { path: ^/mindmaps, roles: ROLE_USER }
9     - { path: ^/bookmarks, roles: ROLE_USER }
10    - { path: ^/annotations, roles: ROLE_USER }
11    - { path: ^/, roles: IS_AUTHENTICATED_ANONYMOUSLY }
```

Programm 4.3: Dieser Ausschnitt aus der Datei security.yml zeigt, wie die Zugriffsrechte der einzelnen Pfade vergeben werden.

bination von Eigenschaften, wurde für beide Enities eine eigene Repository-Klasse angelegt. Die Methoden dieser Repository-Klassen werden im Controller benutzt, um spezielle Datenbank-Aufrufe auszuführen, wie z. B., um Bookmarks nach deren URL und zugehöriger Benutzerin oder um Annotationen nach deren ID und zugehöriger Bookmark zu finden. Damit nur angemeldete Benutzerinnen Zugriff auf die REST-Schnittstelle haben, wurden in der Datei security.yml die entsprechenden Rechte gesetzt, wie in Programm 4.3 zu sehen. Benutzerinnen haben darüber hinaus nur Zugriff auf ihre eigenen Bookmarks, Annotationen und Mindmaps und können diese nur in angemeldetem Zustand einsehen.

4.2.2 Website

Die Website des Prototyps umfasst lediglich eine simple Startseite, jeweils eine Seite zum Registrieren und Anmelden für Benutzerinnen und eine simple Archiv-Seite, die alle Bookmarks und Mindmaps einer Benutzerin anzeigt. Der Default-Controller sorgt dafür, dass das Template für die Startseite gerendert wird. Er enthält außerdem eine Methode, um die aktuell angemeldete Benutzerin abzurufen. Ist eine Benutzerin angemeldet, kann sie ihr persönliches Archiv einsehen. Das Template dafür wird vom Archiv-Controller aufgerufen, welcher auch die Bookmarks und Mindmaps der aktuellen Benutzerin über deren ID abruft und an das Template weitergibt. Hier sind auch Methoden implementiert, die das Template zum Erstellen einer neuen Mindmap oder zum Anzeigen einer bestehenden Mindmap rendert. Die Benutzerinnen-Verwaltung wird hauptsächlich vom verwendeten *FOSUserBundle* geregelt. Dafür wurde eine User-Klasse angelegt, die von BaseUser aus dem *FOSUserBundle* abgeleitet wird. In der Datei config.yml wurde diese Klasse als User-Klasse angegeben und die Routing-Einstellungen wurden erweitert, um Registrieren und Anmelden möglich zu machen. Weitere Einstellungen oder Änderungen, bis auf das Überschreiben der Templates für Registrierung und Anmeldung, wurden nicht vorgenommen.

4.2.3 Mindmap

Wie bereits erwähnt, wurde zur Umsetzung der Mindmap die JavaScript-Bibliothek *D3* herangezogen. Mithilfe von *D3* können SVG-Elemente, wie Kreise, Rechtecke usw., einfach erzeugt werden und ähnlich wie bei *jQuery*, können DOM-Elemente selektiert und manipuliert werden. Darüber hinaus können aus gegebenen Daten SVG-Grafiken, wie z. B. einfache Balkendiagramme aber auch komplexe interaktive Info-Grafiken mit großen Datenmengen, erstellt werden. *D3* wurde für den Prototyp *tisoor* hauptsächlich verwendet, um grafische Elemente (Kreise, Linien) zu erzeugen und um diese per *Drag and Drop* verschieben zu können. Die Hauptfunktionalität der Mindmap befindet sich in der Datei mindmap.js. Standardmäßig wird beim Aufruf der Datei zuerst das `<svg>`-Element mit drei `<g>`-Elementen für die Gruppierung von Relationen (strichlierte Linien), Pfaden (normale Linien) und Knoten (Kreise) erzeugt. Dabei ist anzumerken, dass die Knoten zuletzt hinzugefügt wurden, da ansonsten die Linien über den Knoten gezeichnet werden. Falls eine bestehende Mindmap verfügbar ist, wird diese regeneriert, ansonsten wird ein erster ursprünglicher Knoten, der Haupt-Knoten, erstellt (siehe Abb. 4.3).

Neue Knoten erstellen

Neue Knoten werden erstellt, indem ein neues `<g>`-Element erzeugt wird, das x- und y-Koordinaten, Level, Typ, ID und CSS-Klassen enthält. Dieses

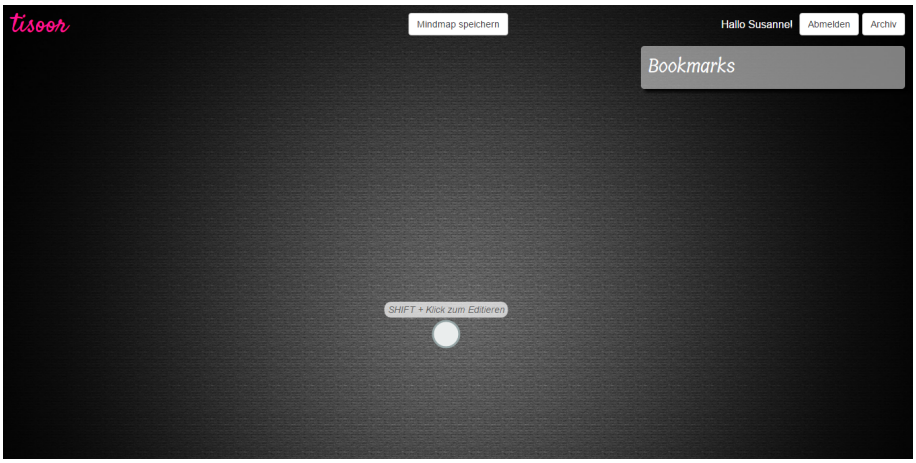


Abbildung 4.3: Screenshot einer neu erstellten Mindmap in *tisoor*.

Gruppen-Element wird zur Knoten-Gruppe hinzugefügt. Mittels der Koordinaten wird der neue Knoten über das `transform`-Attribut platziert. Level, Typ und eindeutige ID werden über `data`-Attribute beim Element gespeichert. Zu diesem Gruppen-Element werden als grafische Repräsentation des Knotens ein `<circle>`-Element und für die Inhalte ein `<foreignObject>`-Element hinzugefügt. Dieses Element wurde deswegen gewählt, da es auch HTML-Struktur (für Hyperlinks, Bilder, ...) enthalten kann. Der Level eines Knotens gibt Auskunft darüber, in welcher Hierarchie er sich befindet, also zu welchen Eltern-Knoten, Großeltern-Knoten, Urgroßeltern-Knoten usw. er gehört. Ein neuer Level wird erstellt, indem der Wert einer globalen Zähler-Variable zum Level des Eltern-Knotens mittels String-Konkatenation hinzugefügt wird. Der Level eines Knotens ist ausschlaggebend beim Verschieben von ganzen Teilen einer Mindmap, weil damit beim Bewegen eines Knotens alle Enkel, Urenkel usw. gefunden und ebenfalls bewegt werden können. Der Typ eines Knotens gibt an, ob es sich um den Haupt-Knoten, einen normalen Knoten oder einen Bookmark-Knoten handelt. Normale Knoten können Text, Hyperlinks, Bilder und Videos enthalten und deren Inhalt wird von der Benutzerin selbst gesetzt. Bookmark-Knoten enthalten automatisch die URL des Bookmarks als Hyperlink und werden von der Benutzerin erzeugt, indem ein Bookmark aus der Liste per *Drag and Drop* auf einen Mindmap-Knoten gezogen wird. Der Inhalt kann aber auch bei Bookmark-Knoten im Nachhinein noch bearbeitet werden. Bei einem Bookmark-Knoten wird außerdem die ID des Bookmarks als eindeutige ID beim Knoten gesetzt. Darüber hinaus werden vorhandene Annotationen, repräsentiert als kleine Kreise, als `<circle>`-Element in der entsprechenden Farbe der Annotation unterhalb des Knotens hinzugefügt. Der Kontext und die optionale Notiz der Annotation werden in einem separaten Fenster angezeigt, indem die Maus

über den Kreis bewegt wird. Zusätzlich werden zu jedem neuen Knoten Icons zum Löschen des Knotens und zum Erzeugen weiterer Kind-Knoten hinzugefügt. Außerdem wird auch ein Icon zum Erstellen von Beziehungen zu anderen nicht verwandten Knoten erzeugt. Eine Beziehung wird erstellt, indem dieses Icon per *Drag and Drop* auf einen beliebigen anderen Knoten gezogen wird. Verbindungen zwischen Eltern- und Kind-Knoten und Beziehungen werden als `<path>`-Elemente implementiert, die jeweils zur Pfad-Gruppe oder zur Relationen-Gruppe hinzugefügt werden. Pfade enthalten die Koordinaten (`d`-Attribut) und die Levels von Start- und End-Knoten (`data`-Attribute), damit sie richtig platziert werden und sich beim Verschieben von Knoten ebenfalls mitbewegen. Verbindungen zwischen Eltern- und Kind-Knoten werden als durchgehende Linien, Beziehungen als strichlierte Linien dargestellt.

Knoten-Inhalte bearbeiten

Bei neu erzeugten Knoten wird standardmäßig der Platzhalter-Text „SHIFT + Klick zum Editieren“ angezeigt. Wie dieser Text aussagt, können Inhalte mit gedrückter SHIFT-Taste und Klick auf den Inhalt jederzeit und immer wieder editiert werden. Bei Knoten ohne Inhalt erscheint ein simples leeres Textfeld, in das beliebiger Text eingegeben werden kann. Bei Knoten mit Inhalt erscheint dieser in Textform im Textfeld und kann von der Benutzerin editiert werden. Zusätzliche zum Textfeld werden außerdem drei Buttons angezeigt, die dazu verwendet werden können, *Markdown* für Hyperlinks (z. B. `[Paris-Hyperlink](http://de.parisinfo.com/)`), Bilder (z. B. `![Paris-Bild](http://www.imageurl.com/paris.jpg)`) oder *YouTube*-Videos (z. B. `?[Paris-Video](AQ6GmpMu5L8)`) in das Textfeld einzufügen. Für Videos gibt es keine *Markdown*-Syntax, daher wurde für diesen Prototyp das Fragezeichen als Zeichen für *YouTube*-Videos verwendet. Diese Notation wird in ein `<iframe>`-Element umgewandelt, um das Video einzubetten. Aufgrund von *Markdown* ist es möglich, zu einem Mindmap-Knoten eine Kombination von unterschiedlichen Medientypen hinzuzufügen. Da eine eigene Video-Notation hinzugefügt wurde, wurden auch zwei eigene Methoden implementiert, die einerseits Text aus dem Textfeld parsen und in HTML-Struktur umwandeln und andererseits HTML-Struktur parsen und daraus reinen Text mit *Markdown*-Syntax generieren. Zum Schließen des Textfeldes und gleichzeitigem Übernehmen des Inhaltes muss entweder Enter gedrückt oder der Schließen-Button geklickt werden. Danach erscheint der eingegebene Inhalt in HTML-Struktur beim Knoten. Hyperlinks können nun wie gewohnt geklickt werden, um die entsprechende Webseite in einem neuen Tab zu öffnen. Videos werden in Form von Thumbnails repräsentiert und bei Klick darauf in einem eigenen Player abgespielt.

```
1  // TisoorBundle/Resources/public/js/mindmap.js
2
3  // DELETE NODE (and all child nodes)
4  function deleteNode() {
5      var node = d3.select(this);
6      var currentLevel = $(this).attr("data-level");
7      var childNodes = $('g[data-level^=' + currentLevel + ']');
8
9      // also delete child nodes
10     childNodes.each(function (i) {
11         d3.select(this).remove();
12     });
13
14     // also delete paths
15     var pathsStart = $('path[data-level-start^=' + currentLevel + ']');
16     var pathsEnd = $('path[data-level-end^=' + currentLevel + ']');
17
18     pathsStart.each(function (i, e) {
19         d3.select(this).remove();
20     });
21
22     pathsEnd.each(function (i, e) {
23         d3.select(this).remove();
24     });
25
26     node.remove();
27 }
```

Programm 4.4: JavaScript-Funktion zum Löschen eines Mindmap-Knotens und allen damit verbundenen Elementen.

Knoten löschen

Wird ein Knoten gelöscht, werden auch dessen Kind-Knoten, Enkel-Knoten usw. gelöscht. Es werden außerdem alle mit diesem Knoten in Verbindung stehenden Pfade ebenfalls gelöscht. Um herauszufinden, welche Elemente zu löschen sind, wird hier wiederum der Level verwendet. Alle Elemente, deren Level exakt mit der Zahlenkombination des Levels von dem zu löschendem Knoten beginnt, werden auch gelöscht. In Programm 4.4 ist die `deleteNode()`-Funktion ersichtlich.

Mindmap speichern und wiederherstellen

Die Funktionalität zum Speichern und Wiederherstellen einer Mindmap wurde in die zwei Dateien storeMindmap.js und regenerateMindmap.js ausgelagert. Eine Mindmap wird gespeichert, indem alle Beziehungen, Pfade und Knoten jeweils in ein JSON-Array gespeichert werden. Dazu werden die Kind-Elemente der einzelnen Gruppen durchlaufen und die notwendigen

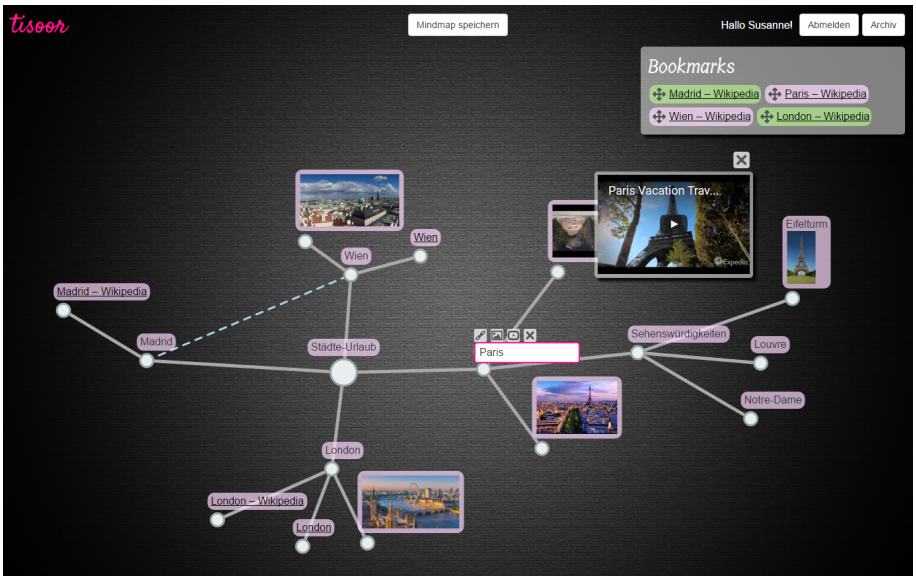


Abbildung 4.4: Screenshot einer in *tisoor* erstellten Mindmap.

Attribute als JSON-Objekt zum Array hinzugefügt. Diese drei Arrays, der Titel der Mindmap (Text des Haupt-Knotens) und die aktuelle Benutzerin werden dann via Ajax in die Datenbank gespeichert. Falls eine bereits bestehende Mindmap überarbeitet und erneut gespeichert wird, wird diese in der Datenbank aktualisiert. Hier wird auch für Mindmaps die REST-Schnittstelle verwendet.

Das Wiederherstellen einer Mindmap verläuft ähnlich wie das Speichern. Falls der Controller Mindmap-Daten an das Template übermittelt, werden diese im eingebetteten JavaScript-Code einer Variable zugewiesen. Nun ist auch ein Zugriff auf die Mindmap-Daten in der externen JavaScript-Datei möglich. Die einzelnen Arrays für Beziehungen, Pfade und Knoten werden dann durchlaufen und es werden HTML-Elemente erzeugt, bei denen die jeweiligen Attribute gesetzt werden, wie beim Erstellen neuer Knoten und Pfade. In Abbildung 4.4 ist eine mit *tisoor* erstellte Mindmap zu sehen.

4.2.4 Annotationen

Wie bereits mehrfach erwähnt, wurde zur Realisierung von Web-Annotationen eine *Google Chrome* Browser-Erweiterung implementiert. Die hauptsächliche Funktionalität befindet sich in der Datei content.js, die es als *Content Script* erlaubt, DOM-Manipulationen an beliebigen Webseiten durchzuführen. Da jedoch nur angemeldete Benutzerinnen Annotationen erstellen können, wird in der Datei background.js eine Anfrage zum Server geschickt, um die aktuelle Benutzerin zu ermitteln. Ist eine Benutzerin vorhanden, wird

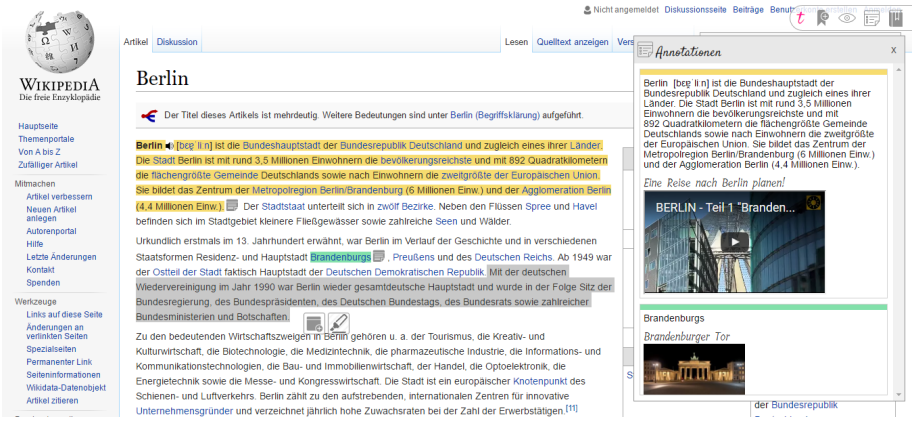


Abbildung 4.5: Screenshot der Wikipedia-Seite von Berlin [22], die gerade mit *tisoor* annotiert wird. Rechts sind alle bereits erstellten Annotationen in einer Liste ersichtlich. In der Mitte wird eine neue Annotation bei einem selektierten Text erstellt.

im Popup der Benutzername angezeigt, ist keine Benutzerin vorhanden, wird stattdessen ein Link zur Anmelde-Seite eingefügt. Die *Background Page* sendet außerdem eine Nachricht an das *Content Script*, wenn es eine Benutzerin gibt. Daraufhin wird im *Content Script* die Möglichkeit zum Erstellen von Annotationen freigegeben, indem das Hauptmenü angezeigt wird. Währenddessen werden weitere Anfragen an die REST-Schnittstelle gesendet, um festzustellen, ob die URL des aktuellen Browser-Tabs für diese Benutzerin bereits existiert, und wenn es bereits einen Bookmark-Eintrag gibt, wird angefragt, ob bereits Annotationen für diese Webseite zur Verfügung stehen. Alle bereits verfügbaren Daten (Bookmark und Annotationen) werden an das *Content Script* gesendet und dort in Variablen gespeichert. Das Hauptmenü der Browser-Erweiterung enthält lediglich vier Buttons. Einer dient zum Hinzufügen der aktuellen Webseite ins Bookmark-Archiv, mit einem weiteren kann die Funktion zum Annotieren der Seite aktiviert und deaktiviert werden, der nächste dient zum Ein- und Ausblenden der Annotations-Liste und der letzte Button verlinkt zum persönlichen Bookmark-Archiv. Existiert die aktuelle Seite bereits im Bookmark-Archiv, ist der Bookmark-Button deaktiviert und wird als grünes Bookmark-Symbol dargestellt. In der Annotations-Liste werden alle Annotationen der aktuellen Seite gesammelt in einer Liste angezeigt. Zusätzlich zum Aktivieren oder Deaktivieren der Annotations-Funktion werden auch alle bereits erstellten Annotationen auf der Webseite ein- und ausgeblendet. In Abbildung 4.5 ist eine Webseite zu sehen, die mit *tisoor* annotiert wird.

Annotationen erstellen

Neue Annotationen können erstellt werden, wenn die Funktion im Hauptmenü aktiviert wurde. Um Text zu Markieren wird dieser einfach mit der Maus selektiert. Anschließend erscheint ein Menü bestehend aus zwei Haupt-Buttons und jeweils vier Farb-Buttons. Hier kann entschieden werden, ob der selektierte Text nur farbig markiert werden soll oder ob zusätzlich sofort eine Notiz erstellt werden möchte. Wird der Button zum Markieren geklickt, werden mithilfe von *Rangy* alle reinen Text-Knoten der Selektion herausgefiltert und mit einem `<span>`-Element umschlossen. Dieses `<span>`-Element erhält ein `data`-Attribut, in dem die eindeutige ID dieser Annotation gespeichert wird, und die entsprechende CSS-Klasse, die die Hintergrundfarbe angibt. Da die reinen Text-Knoten manipuliert werden, ist auch eine Selektion von Hyperlinks, hervorgehobenen Textpassagen (fett, kursiv, ...), mehreren Paragraphen usw. möglich. Soll eine Markierung sogleich in einer anderen Farbe erstellt werden, muss einer der gewünschten Farb-Buttons unterhalb des Haupt-Buttons geklickt werden. Wird der Button zum Erstellen von Notizen geklickt oder einer der Farb-Buttons darunter, wird der selektierte Text ebenfalls, wie zuvor beschrieben, markiert und zusätzlich erscheint ein Fenster, in dem Notizen gemacht werden können. Dieses Fenster enthält ein Textfeld, drei Buttons zum Einfügen von *Markdown* für Hyperlinks, Bilder und Videos (siehe Abschnitt 4.2.3) und Buttons zum Abbrechen der Aktion und zum Speichern der Notiz. Bei Notizen für Textinhalte wird beim Speichern ein Icon nach dem letzten selektierten Text-Knoten eingefügt, das es ermöglicht, die Notiz später anzusehen und zu bearbeiten.

Bilder und Videos können ebenfalls einfach annotiert werden. Wird die Maus über ein Bild oder Video bewegt, erscheinen in der linken oberen Ecke des Bildes (oder Videos) ebenfalls die beiden Buttons zum Markieren und zum Hinzufügen von Notizen. Bilder werden markiert, indem, wie bei Text-Knoten, ebenfalls ein `<span>`-Element mit ID und entsprechender CSS-Klasse um das `<img>`-Element gelegt wird. Die CSS-Klasse für Bilder erzeugt jedoch einen farbigen Rahmen um das Bild. Bei *YouTube*-Videos umschließt das `<span>`-Element nicht das `<iframe>`-Element, sondern es wird direkt davor eingefügt, da ansonsten das gesamte Video neu geladen werden müsste. Die CSS-Klasse bezieht sich somit auf das nachfolgende `<iframe>`-Element. Notizen können bei Bildern genauso wie bei Text hinzugefügt werden, es wird jedoch kein Icon zum Einsehen oder Bearbeiten der Notiz erzeugt. Für diese Funktion erscheint bei annotierten Bildern und Videos im Menü ein zusätzlicher Button.

Bei *YouTube*-Videos können Notizen zu einzelnen, verschiedenen Zeitpunkten im Video hinzugefügt werden. Dazu wurde unter Verwendung der *YouTube*-API ein eigener Player implementiert (siehe Abb. 4.6). Dies war notwendig, damit der aktuelle Zeitpunkt im Video abgefragt werden kann und damit das Video einfacher geladen und gesteuert werden kann. Somit

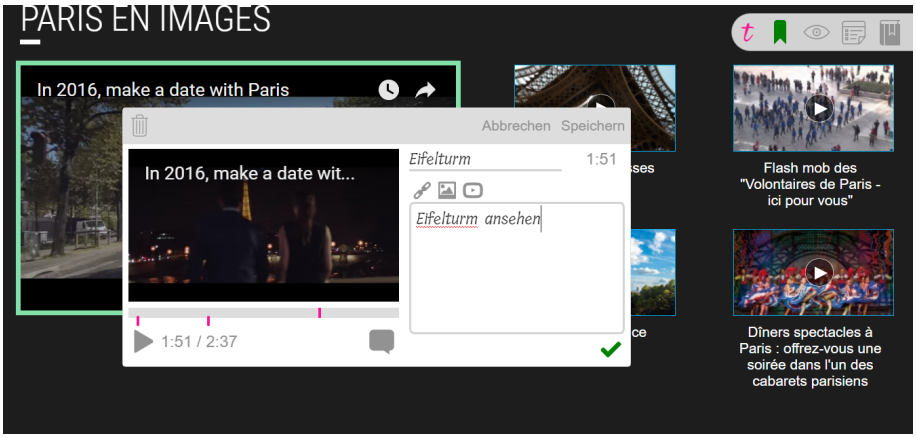


Abbildung 4.6: Screenshot der Website für das Fremdenverkehrsamt von Paris [37], auf der gerade ein eingebettetes *YouTube*-Video mit *tisoor* annotiert wird.

ist es möglich, zu jedem Zeitpunkt in einem Video eine Notiz hinzuzufügen. Mittels *Markdown* können auch hier Hyperlinks, Bilder oder Videos eingefügt werden. Dabei wird auf der Zeitleiste eine Markierung gesetzt, die es ermöglicht, wieder zu dieser Stelle und der entsprechenden Notiz zu wechseln. Jede Notiz wird als JSON-Objekt zu einem temporären Array hinzugefügt und dieses Array wird beim Speichern in der globalen Variable `annotations` abgespeichert.

Alle Annotationen auf einer Webseite werden in diesem globalen Array gespeichert. Bereits beim Erstellen von Markierungen wird ein Eintrag in diesem Array angelegt. Bei allen weiteren Änderungen der Annotation wird dieser Eintrag aktualisiert. In Programm 4.5 ist zu sehen, wie eine neue Annotation für Bilder im Array gespeichert wird. Die Felder `pos` und `matches` werden nur für Annotationen bei Textinhalten benötigt, um festzustellen an welcher Position auf einer Webseite sich der Text befindet (falls Textinhalte mehrfach auf einer Seite vorkommen). Auf Basis von diesem Array wird auch die Liste, in der alle erstellten Annotationen gesammelt angezeigt werden, erzeugt.

Annotationen bearbeiten

Wurden bereits Annotationen für Text, Bild oder Video erstellt, ändert sich das Menü, welches angezeigt wird, wenn die Maus darüber bewegt wird. Anstelle der Möglichkeit, eine Markierung zu erzeugen, kann nun die Farbe der bereits erzeugten Markierung geändert werden. Es kann nach wie vor eine Notiz hinzugefügt werden, wenn noch keine existiert. Dieser Button verschwindet ebenfalls, wenn bereits Notizen gemacht wurden. Anstelle

```
1 // tisoor/content.js
2
3 // store new annotation to annotations-array
4 function createNewImgAnnotation(annotationID) {
5     var highlightedElements = $("span[data-tisoor-annotation-id='" +
      annotationID + "']");
6
7     var item = {};
8     item.identification = annotationID;
9     item.type = "img";
10    item.pos = 0;
11    item.matches = 0;
12    item.color = highlightedElements.css("background-color");
13    item.context = img.attr("src");
14    item.note = "";
15    item.bookmark_id = bookmarkID;
16
17    annotations.push(item);
18 }
```

Programm 4.5: JavaScript-Funktion zum Speichern einer neuen Annotation für Bilder in das globale Array.

dessen erscheint ein neuer Button, der es ermöglicht, eine gesamte Annotation wieder zu löschen. Wird die Farbe einer Annotation geändert, wird die CSS-Klasse bei `<span>`-Elementen mit der gleichen ID ausgetauscht. Zusätzlich wird der Eintrag im globalen Array und die Annotations-Liste aktualisiert. Der Inhalt einer Notiz kann bearbeitet werden, indem der Button zum Anzeigen der Notiz gewählt wird. Hier wird die Notiz zum Ansehen in HTML-Struktur (mit allen Hyperlinks, Bildern und Videos) dargestellt. Soll diese Notiz editiert werden, wird der Button zum Bearbeiten geklickt und der Inhalt kann in reinem Textformat verändert werden. Das Konvertieren von *Markdown*-Syntax in HTML-Struktur und umgekehrt erfolgt wie bei der Mindmap (siehe Abschnitt 4.2.3).

Annotationen löschen

Beim Löschen von Annotationen werden bei Text-Annotationen die zuvor hinzugefügten `<span>`-Elemente wieder entfernt. Auch bei Bildern und Videos wird das `<span>`-Element wieder gelöscht. Hier ist darauf zu achten, dass nicht das gesamte Element gelöscht wird, da sonst auch die enthaltenen Inhalte der Webseite gelöscht werden. Zusätzlich wird der Eintrag im globalen Array gelöscht und die Annotations-Liste aktualisiert.

Annotationen speichern und wiederherstellen

Damit Annotationen nicht nur, wie zuvor erklärt, im globalen Array existieren, sondern auch in die Datenbank gespeichert werden, wird eine Annotation in Form eines JSON-Strings via Ajax an die REST-Schnittstelle gesendet. Damit eine Benutzerin ihre Annotationen nicht manuell speichern muss, wird beim Erstellen, Ändern und Löschen jeder Annotation ein Ajax-Aufruf durchgeführt. Existieren bereits Annotationen für eine Webseite, werden diese aus der Datenbank abgefragt und anschließend in das globale Array gespeichert. Dieses Array wird durchlaufen und je nach Typ der Annotation (Text, Bild, Video) wird eine eigene Funktion ausgeführt, die die Annotation wiederherstellt. Bei Bildern und Videos ist das relativ einfach, da die Webseite nach der eindeutigen URL durchsucht werden kann, bei Text gestaltet sich die Suche nach der richtigen Textpassage jedoch etwas schwieriger. Hier wird wieder *Rangy* zum Finden eines bestimmten Textes auf der Seite verwendet. Jedoch werden bei mehrfachem Vorkommen des Textes auf der Seite auch mehrere Treffer geliefert. Um dieses Problem zu lösen, wurde zuvor bereits die Position des Textes auf der Webseite gespeichert. Das bedeutet, dass, wenn ein bestimmtes Wort z. B. dreimal vorkommt und die Benutzerin das zweite Wort markiert, als Position die Zahl zwei gespeichert wird. Das Programm 4.6 zeigt zwei Funktionen, die das Wiederherstellen von Markierungen an der richtigen Stelle möglich machen.

In Zeile 4 und 5 wird mit *Rangy* ein Bereich im HTML-Dokument festgelegt (`body`), in dem nach dem Suchbegriff gesucht werden soll. Danach werden Einstellungen vorgenommen, wie z. B., dass bei der Suche auf Groß- und Kleinschreibung geachtet werden soll. Außerdem wird in den Suchoptionen auch der Suchbereich gesetzt. In Zeile 15 wird der Suchbegriff vorbereitet, indem spezielle Zeichen entfernt werden. Je nach abgespeicherter Farbe wird dann die Funktion `highlight` mit dem entsprechenden Parameter aufgerufen. In dieser Funktion findet das Hervorheben der richtigen Textpassagen statt. Die `while`-Schleife in den Zeilen 38 bis 45 wird sooft durchlaufen, sooft *Rangy* den Suchbegriff im Suchbereich findet. Die richtige Textpassage wird ermittelt, indem die zuvor eingeführte Zähler-Variable (Zeile 35) mit der abgespeicherten Position der Textpassage verglichen wird (Zeile 39). Stimmt die Position überein, wird in Zeile 40 jeder Textknoten des gefundenen Suchbegriffs mit einem `<span>`-Element umgeben. Dieses Element enthält die entsprechende CSS-Klasse (wird durch den Parameter `classAplier` übergeben), um die Markierung in der richtigen Farbe darzustellen.

```
1  // tisoor/js/restoreAnnotations.js
2
3  function highlightText(annotationID, color, text, position) {
4      var searchScopeRange = rangy.createRange();
5      searchScopeRange.selectNodeContents(document.body);
6
7      var options = {
8          caseSensitive: true,
9          wholeWordsOnly: false,
10         withinRange: searchScopeRange
11     };
12
13     range.selectNodeContents(document.body);
14     var st = text;     // search term
15     st = st.replace(/[\-\[\]\/\{\}\(\)\*\+\?\.\\\^\$\|]/g, "\\$&");
16
17     if (color == G) {
18         highlight(st, options, green, position);
19     } else if (color == Y) {
20         highlight(st, options, yellow, position);
21     } else if (color == B) {
22         highlight(st, options, blue, position);
23     } else if (color == P) {
24         highlight(st, options, pink, position);
25     }
26
27     var newHighlElem= $("span[data-tisoor-annotation-id='tmp']");
28     newHighlElem.attr("data-tisoor-annotation-id", annotationID);
29     newHighlElem.addClass("tisoor-highlight");
30 }
31
32 function highlight(st, options, classApplier, position) {
33     if (st !== "") {
34         st = new RegExp(st, "g");
35         var counter = 1;
36
37         // Iterate over matches
38         while (range.findText(st, options)) {
39             if(counter == parseInt(position)){
40                 classApplier.applyToRange(range);
41             }
42
43             range.collapse(false);
44             ++counter;
45         }
46     }
47 }
```

Programm 4.6: JavaScript-Funktionen zum Wiederherstellen von Text-Markierungen.

Kapitel 5

Evaluierung

Diese Kapitel befasst sich mit der Evaluierung des Prototyps *tisoor*. Dabei soll mittels einer Anforderungsreflexion festgestellt werden, ob die definierten Anforderungen an den Prototyp erfüllt wurden. Des Weiteren wird ein Benutzertest durchgeführt, um die Benutzerfreundlichkeit des Prototyps zu evaluieren und um mögliche Probleme bei der Bedienung der Anwendung zu erfassen. Als Usability-Methode dient die Think Aloud Methode. Abschließend werden die Ergebnisse des Fragebogens für die User Experience dargelegt.

5.1 Anforderungsreflexion

Im Folgenden wird die Erfüllung der in Abschnitt 3.2 angegebenen Anforderungen an den erstellten Prototyp *tisoor* evaluiert. Darüber hinaus wird der eigene Ansatz mit den bereits bestehenden Tools, die in Abschnitt 2.3 beschrieben wurden, verglichen. Ein Überblick wird in Tabelle 5.1 gegeben.

5.1.1 Individuelle Visualisierung und Verwaltung von Bookmarks

Für den Prototyp *tisoor* wurde im persönlichen Bookmark-Archiv die Funktion implementiert, individuelle Mindmaps zu erstellen. Dadurch können Benutzerinnen nicht nur ihre Bookmarks visuell organisieren, sondern auch zusätzliche Informationen wie Text, Hyperlinks, Bilder oder Videos in einer graphischen Darstellung verwalten. Benutzerinnen können Bookmarks und Informationen kategorisieren, gruppieren und ihren Vorstellungen entsprechend anordnen. Es können Abhängigkeiten und Verbindungen zwischen einzelnen Knoten hergestellt werden, die in Form von Linien visualisiert werden. Gespeicherte Bookmarks können an beliebiger Stelle einfach per *Drag and Drop* zur Mindmap hinzugefügt werden. Verbindungen zwischen einzelnen Mindmap-Knoten können auch via *Drag and Drop* erstellt werden. We-

	tisoor	*Diigo*	*scrible*	*Coggle*	*VideoAnt*
Bookmark-Visualisierung	+	–	–	+	
Multimediale Annotationen	+	–	–		○
Platzierung im Kontext	+	+	+		–
Robustheit	○	+	+		
Geräteunabhängigkeit	○	+	+	○	+

Tabelle 5.1: Die analysierten Tools erfüllen verschiedene Anforderungen zum Großteil (+), teilweise (○) oder nicht (–). Für die Tools *Coggle* und *VideoAnt* können manche Anforderungen nicht gewertet werden, da sie auf eine andere Funktionalität spezialisiert sind.

der *Diigo* noch *scrible* bieten eine alternative Darstellung von Bookmarks. Diese beiden Tools zeigen Bookmarks nur in Form einer Liste an, deswegen wird hier das Mindmap-Tool *Coggle* zum Vergleich herangezogen.

Da *Coggle* ein Tool ist, das rein auf das Erstellen von Mindmaps spezialisiert ist, ist die grafische Darstellung ausgereifter und die Funktionalität umfangreicher als beim eigenen Prototyp *tisoor*. Linien werden bei *Coggle* z. B. in verschiedenen Farben und als Kurven dargestellt, wohingegen bei *tisoor* nur graue, gerade Linien möglich sind. Zum Ändern des Textes bei einem Knoten ist bei *Coggle* nur ein Klick auf den Inhalt notwendig. Hier können Hyperlinks über einen Tooltip angeklickt werden. Beim eigenen Prototyp muss die Steuerungs-Taste zusätzlich gedrückt werden, um Inhalte zu ändern, dafür kann ein Hyperlink direkt angeklickt werden, um zur entsprechenden Seite zu wechseln. Das kann bei Mindmaps mit vielen Bookmarks praktischer für die Benutzerin sein. *Coggle* verwendet *Markdown*, um Hyperlinks und Bilder als Inhalt hinzuzufügen. Bei *tisoor* funktioniert das Hinzufügen von Links und Bildern ebenso, wobei zusätzlich auch die Möglichkeit besteht, Videos hinzuzufügen. Das Löschen von Knoten und das Herstellen von Verbindungen funktioniert bei *tisoor* direkt über einen Button beim Knoten. Wohingegen bei *Coggle* zum Löschen zusätzlich die Steuerungs-Taste gedrückt werden muss. Das Herstellen von Verbindungen erfolgt über ein eigenes Kontextmenü. Obwohl die Funktionalität von *Coggle* natürlich umfassender ist als die Mindmap-Funktion der eigenen Implementierung, erfüllt *tisoor* die Anforderung der individuellen Visualisierung und Verwaltung von Bookmarks, so wie sie in Abschnitt 3.2.1 beschrieben wird.

5.1.2 Multimediale Annotationen

Wie in Abschnitt 3.2.2 erwähnt, soll es möglich sein, multimediale Inhalte einer Webseite mit multimedialen Notizen zu ergänzen. Diese Annotatio-

nen sollen für beliebige Webseiten möglich sein, direkt im Kontext platziert werden und robust gegenüber Änderungen sein.

Mit der *Google Chrome* Browser Erweiterung von *tisoor* können einzelne Wörter oder Textpassagen einer Webseite in vier verschiedenen Farben markiert werden, indem der Text selektiert wird und das gewünschte Farbfeld geklickt wird. Die Markierung wird direkt beim Text als farbiger Hintergrund angezeigt. Auf ähnliche Weise können auch mit *Diigo* und *scrible* Texte markiert werden. Bei beiden Tools und auch beim eigenen Prototyp *tisoor* ist es möglich, die Farbe der Markierung im Nachhinein noch zu ändern. Sowohl bei *Diigo* als auch bei *scrible* kann zu einem selektierten oder bereits markierten Text eine Notiz in Form eines persönlichen Textes hinzugefügt werden. Mit *tisoor* ist es allerdings möglich, mithilfe von *Markdown* auch Hyperlinks, Bilder, Videos oder eine Kombination von alledem als Notiz einzufügen. Die Notiz wird wie bei *Diigo* und *scrible* als kleines Icon direkt beim annotierten Text angezeigt. Darüber hinaus können mit *tisoor* nicht nur multimediale Notizen erstellt werden, sondern es können auch multimediale Inhalte auf einer Webseite annotiert werden. Bilder und Videos können mit dem eigenen Prototyp im Gegensatz zu *Diigo* und *scrible* ebenfalls markiert werden, indem die Maus über das Bild oder Video bewegt und wie beim Text die gewünschte Farbe gewählt wird. Zusätzlich zur Markierung, die als farbiger Rahmen visualisiert wird, können auch multimediale Notizen hinzugefügt werden. Bei einem *YouTube*-Video können mit *tisoor* über einen eigenen Player sogar beliebige Stellen im Video mit Notizen ergänzt werden.

Bei *Diigo* ist es lediglich möglich, Bilder im persönlichen Archiv abzuspeichern und *scrible* unterstützt multimediale Medientypen in keiner Hinsicht. Daher wird für das Annotieren von Videos hier das Tool *VideoAnt* zum Vergleich herangezogen, wobei *VideoAnt* eine eigenständige Webanwendung ist und nicht mit Videos, die auf fremden Webseiten eingebettet sind, arbeitet. Das Erstellen von Notizen funktioniert bei *tisoor* ähnlich wie bei *VideoAnt*. *VideoAnt* bietet allerdings vor dem Erstellen der ersten Notiz eine kurze Anleitung dazu und zeigt noch kein Textfeld für den persönlichen Kommentar an. Es wird Zeitpunkt, Titel und Kommentar gespeichert und zusätzlich wird eine Markierung auf der Zeitleiste gesetzt. Bei *VideoAnt* werden im Gegensatz zu *tisoor* alle bereits erstellten Notizen eines Videos rechts daneben angezeigt. Dadurch kann jede einzelne Notiz jederzeit bearbeitet und gelöscht werden. Bei *tisoor* muss die Notiz auf der Zeitleiste ausgewählt werden, um Änderungen vorzunehmen. Markierungen können bei *VideoAnt* auf der Zeitleiste auch verschoben werden, was bei *tisoor* nicht möglich ist. *tisoor* unterstützt allerdings im Gegensatz zu *VideoAnt* multimediale Notizen. Das bedeutet, dass zu jeder beliebigen Stelle in einem *YouTube*-Video eine Kombination von Text, Hyperlinks, Bildern und Videos hinzugefügt werden kann.

Da viele Webseiten dynamisch erzeugt werden und deren Inhalte sich ständig ändern, ist es bei Annotationen wichtig, dass diese trotzdem nicht

verloren gehen. Bei *Diigo*, *scrible* und *tisoor* werden sowohl der annotierte Inhalt, als auch die erstellte Notiz im Archiv gespeichert und können somit von der Benutzerin auch dann noch eingesehen werden, wenn sich die Webseite verändert hat oder diese gelöscht wurde. Beim Wiederherstellen von Annotationen bei geänderten Inhalten reagiert *tisoor* genauso wie *Diigo* und *scrible*. Es wird nach den annotierten Textpassagen via Textsuche auf der gesamten Webseite gesucht. Gibt es Such-Ergebnisse, wird die Annotation der Position entsprechend bei der richtigen Textstelle wiederhergestellt, gibt es keine Ergebnisse, kann die Annotation weder bei *Diigo* noch bei *scrible* wiederhergestellt werden. Tauchen Textpassagen mehrmals auf einer Webseite auf, wird eine davon annotiert und eine andere gleiche Textpassage anschließend gelöscht oder verschoben, kann es passieren, dass die Annotation beim richtigen Text aber an der falschen Stelle wiederhergestellt wird. Dieses Problem tritt sowohl bei den beiden Tools *Diigo* und *scrible*, als auch beim eigenen Prototyp *tisoor* auf. *Diigo* und *scrible* bieten daher die Möglichkeit, gesamte Webseiten vollständig abzuspeichern. Diese können über das Archiv des jeweiligen Tools bearbeitet werden.

Die Anforderung der Robustheit konnte im Bezug auf das Wiederherstellen von Annotationen nicht vollkommen erfüllt werden. Mit *tisoor* ist es jedoch möglich, multimediale Inhalte auf beliebigen Webseiten zu markieren und mit multimedialen Notizen zu versehen. Annotationen werden dabei direkt im Kontext der Webseite platziert. Somit wurden die Anforderungen für multimediale Annotationen erfüllt, womit sich *tisoor* auch von bereits bestehenden Tools abhebt.

5.1.3 Geräteunabhängigkeit

Wie bereits in Abschnitt 3.2.3 erwähnt, soll das persönliche Bookmark-Archiv von verschiedenen Geräten und Browsern aufgerufen werden können. Wie bei *Diigo* und *scrible* wurde auch beim eigenen Prototyp *tisoor* das Bookmark-Archiv als Website umgesetzt. Somit ist es Benutzerinnen möglich, unterschiedliche Geräte und Browser zu nutzen und unabhängig davon die gespeicherten Lesezeichen einzusehen. Es ist keine Installation notwendig, die Benutzerin muss lediglich ein Benutzerkonto anlegen. Bezüglich mobiler Geräte wurde das Bookmark-Archiv von *tisoor* allerdings nicht getestet, wobei das Layout der Seite mit *Bootstrap* umgesetzt worden ist und sich somit an unterschiedliche Bildschirmgrößen anpasst. Das Bookmark-Archiv von *scrible* ist ebenfalls responsive, das von *Diigo* ist nicht für mobile Geräte optimiert.

Um Bookmarks zu erstellen, wurde für *tisoor* eine *Google Chrome* Browser Erweiterung umgesetzt. Das hat den Nachteil, dass eine Benutzerin nur mit dem *Google Chrome* Browser Lesezeichen in das *tisoor* Bookmark-Archiv speichern kann. Da eine Erweiterung beim Browser installiert werden muss, ist diese auch an den Browser gebunden. Das bedeutet, dass bei mehreren

verwendeten Geräten auch jeweils eine Installation notwendig ist. Der Vorteil ist jedoch, dass der *Google Chrome* Browser auf nahezu allen Geräten funktioniert und somit die *tisoor* Erweiterung zwar nicht plattformunabhängig aber geräteunabhängig ist (sofern Erweiterungen auf mobilen Geräten unterstützt werden). Dahingegen bieten *Diigo* und *scrible* Bookmarklets, die nicht installiert werden müssen und für alle gängigen Browser funktionieren. Darüber hinaus gibt es bei *Diigo* auch mobile Apps für *iOS* und *Android*. Die Anforderung der Geräteunabhängigkeit ist daher bei der *tisoor* Erweiterung insbesondere für mobile Geräte nur teilweise erfüllt. Das genannte Problem aus Abschnitt 2.1, dass Bookmarks nicht von allen verwendeten Geräten einsehbar sind, wurde aber gelöst.

5.1.4 Benutzerfreundlichkeit

Das User-Interface einer Anwendung trägt wesentlich zu deren benutzerfreundlicher Handhabung bei, wie bereits in Abschnitt 3.2.4 beschrieben. Mögliche anwendbare Funktionen sollen klar ersichtlich, einfach und in wenigen Schritten ausführbar sein. Auch Benutzerinnen mit geringem computertechnischen Hintergrundwissen sollen die Anwendung intuitiv bedienen können. Die Benutzerfreundlichkeit wird typischerweise gemessen, indem eine Reihe von Probandinnen vordefinierte Aufgaben mit einem System durchführt [13, S. 27]. Ein Benutzertest mit realen Benutzerinnen ist eine fundamentale Usability Methode, da er direkte Informationen darüber gibt, wie Benutzerinnen ein System bedienen und welche Probleme dabei auftreten [13, S. 165]. Eine Anforderungsreflexion würde hier dementsprechend wenig Aufschluss über die tatsächliche Benutzerfreundlichkeit des Prototyps *tisoor* geben, weswegen ein Benutzertest durchgeführt wurde, der im Folgenden beschrieben wird.

5.2 Think Aloud Benutzertest

Die Think Aloud Methode ist eine von mehreren verschiedenen Methoden, um die Usability einer Anwendung zu testen. Andere Methoden wären z. B. Beobachtung, Fragebögen oder Interviews, wobei die Think Aloud Methode die am meisten verbreitete und wertvollste Testmethode ist [13, S. 195], [18, S. 322]. Die Probandinnen sollen bei der Think Aloud Methode vordefinierte Aufgaben mit der zu testenden Anwendung lösen und dabei ihre Gedanken und Handlungen verbalisieren. Das ermöglicht es, herauszufinden, wie Benutzerinnen eine Anwendung wahrnehmen und einzelne Interface-Elemente interpretieren. Dabei können Erkenntnisse über problematische bzw. missverständliche Teile der Anwendung gewonnen werden [13, S. 195–198].

5.2.1 Probandinnen

An dem Benutzertest nahmen zwei Frauen und drei Männer im Alter zwischen 17 und 54 Jahren teil. Bei der Auswahl der Probandinnen wurde darauf geachtet, dass diese möglichst repräsentativ für die Endbenutzerinnen der Anwendung *tisoor* sind, wie Nielsen in [13, S. 175] betont. Daher war es wichtig, keine Probandinnen mit Fachkenntnissen auf den Gebieten Usability oder Webentwicklung auszuwählen. Andererseits sollten Personen teilnehmen, die zumindest Erfahrung mit der Bedienung eines Browsers und der Suche im WWW mitbringen. Die ausgewählten Personen verbringen zwischen einer und acht Stunden pro Tag im Internet, haben beruflich nichts mit Usability oder Webentwicklung zu tun und erfüllen somit die Kriterien als mögliche Endbenutzerinnen für die Think Aloud Methode.

5.2.2 Vorbereitung

Für die Durchführung der Think Aloud Methode wurden sieben Aufgaben formuliert, die die Probandinnen mit dem Prototyp *tisoor* lösen sollten. Bei der Ausarbeitung der Aufgabenstellungen wurde darauf geachtet, dass diese den möglichen Gebrauch der Anwendung repräsentieren und die wichtigsten Teile der Anwendung abdecken. Die erste Aufgabe wurde so gestaltet, dass sie leicht zu lösen ist, um der Benutzerin ein anfängliches Erfolgserlebnis zu garantieren. Alle weiteren Aufgaben hatten einen höheren Schwierigkeitsgrad [13, S. 185–187]. Die Aufgabenstellungen behandeln das Registrieren einer neuen Benutzerin, das Abspeichern von Bookmarks, das Hinzufügen von multimedialen Annotationen und das Erstellen einer Mindmap. In Abschnitt A.1 können die Aufgaben nachgelesen werden. Neben den Aufgaben für die Think Aloud Methode wurde auch ein kurzer Fragebogen vorbereitet, den die Probandinnen im Anschluss an den Benutzertest ausfüllen sollten. Dieser Fragebogen basiert auf dem *User Experience Questionnaire* aus [44] und ist in Abschnitt A.2 ersichtlich. Auf diesen Fragebogen und dessen Ergebnisse wird in Abschnitt 5.3 näher eingegangen. Zur Durchführung des Benutzertests wurde ein Laptop mit Maus für die Probandinnen zur Verfügung gestellt, auf dem bereits der *Google Chrome* Browser mit *tisoor*-Erweiterung installiert war. Vor jedem Test wurde im *Google Chrome* Browser der Verlauf gelöscht und die Startseite der *tisoor*-Website geöffnet. Außerdem wurde der Browser auf Standardeinstellungen zurückgesetzt und es wurden alle Lesezeichen des Browsers gelöscht. Der Prototyp *tisoor* benötigte für die definierten Aufgaben keine spezielle Vorbereitung.

5.2.3 Durchführung

Die Benutzertests wurden für jede Probandin einzeln in deren gewohnter Umgebung und zu einem festgelegten Termin durchgeführt. Vor dem Beginn des Tests wurde die Probandin über den Prototyp *tisoor* und den Grund des

Benutzertests aufgeklärt. Es wurde dabei betont, dass die Anwendung getestet wird und nicht die Benutzerin. Der Benutzerin wurde die Think Aloud Methode erklärt und sie wurde darauf hingewiesen, alle ihre Gedanken und Handlungen während des Tests laut auszusprechen. Der Benutzertest wurde von einer Expertin überwacht, die das Verhalten der Probandin protokollierte, und mit einer Video-Kamera aufgezeichnet. Diese Aufnahmen dienten der Expertin bei lückenhaften Protokollen zur nachträglichen Durchsicht der Tests.

Bei allen Probandinnen konnte beobachtet werden, dass sie Schwierigkeiten hatten, ihre Gedanken während des gesamten Tests fortlaufend laut auszusprechen. Sie mussten mehrmals daran erinnert werden ihre Handlungen zu verbalisieren. Bemerkenswert ist, das die benötigte Zeit zum Abschließen der Aufgaben für die fünf Probandinnen sehr unterschiedlich ausgefallen ist. Eine Benutzerin konnte alle Aufgaben in nur 20 Minuten lösen, während eine andere Probandin dafür 90 Minuten benötigte. Nach der Think Aloud Methode wurden die Probandinnen gebeten, den bereits zuvor erwähnten Fragebogen auszufüllen. Danach folgte ein kurzes Gespräch zwischen Expertin und Probandin, bei dem die Benutzerin nach Kommentaren zur Anwendung und Vorschlägen zur Verbesserung gefragt wurde.

5.2.4 Ergebnisse

Bei der Evaluierung des Prototyps *tisoor* mit der Think Aloud Methode tauchten einige Probleme auf, die im Folgenden zusätzlich mit möglichen Lösungsvorschlägen beschrieben werden.

Lange Ladezeiten

Bei der Evaluierung sorgte die Kommunikation zwischen der *tisoor* Browser-Erweiterung und der REST API mehrfach für sehr lange Ladezeiten. So dauerte es beispielsweise mehrere Sekunden, bis auch bei der Erweiterung die angemeldete Benutzerin angezeigt wurde. Da das Menü zum Erstellen von Bookmarks und Annotationen nur angezeigt wird, wenn eine Benutzerin angemeldet ist, dauerte es auch einige Sekunden, bis das Menü eingeblendet wurde. Diese langen Ladezeiten verursachten mehrfach Verwirrung bei den Probandinnen. Sie versuchten sich erneut anzumelden oder suchten in den Browser-Einstellungen oder im Kontextmenü nach Funktionen, um Bookmarks abzuspeichern.

Auch das Wiederherstellen von Annotationen erzeugte durch die sehr hohe Ladezeit einige Schwierigkeiten. Einige Probandinnen klickten mehrere Male den Button zum Wiederherstellen von Annotationen, da nicht angezeigt wird, dass die Seite gerade lädt. Dadurch kam es mehrmals zum Absturz des Browsers.

Ein Teil der Lösung wäre, alle Ladezeiten mittels grafischer Animation

zu repräsentieren, damit die Benutzerin zumindest erkennt, dass die Anwendung arbeitet. Der zweite Teil der Lösung wäre, die Kommunikation zwischen Erweiterung und REST API und den Algorithmus zum Wiederherstellen von Annotationen zu optimieren bzw. eine alternative Implementierung dafür zu finden.

Annotationen aktivieren

Markierungen und Notizen können auf einer Webseite erst erstellt werden, wenn im Menü Annotationen aktiviert wurden. Das führte bei allen Probandinnen zu Schwierigkeiten, da sie versuchten, Text zu markieren, obwohl Annotationen noch nicht aktiviert waren. Alle Benutzerinnen selektierten zuerst Textpassagen und suchten im Kontextmenü, in den Browser-Einstellungen oder im *tisoor*-Menü nach Funktionen zum Erstellen einer Markierung. Einigen war der Begriff Annotation nicht klar, wodurch sie nicht wussten, was der Button zum Aktivieren von Annotation bewirkt. Dass die Funktion zum Hinzufügen von Annotationen zuerst aktiviert werden muss, war für alle Probandinnen verwirrend und unpraktisch.

Eine einfache Lösung für dieses Problem wäre, Annotationen sogleich zu aktivieren. Damit würde die Bedienung um einiges erleichtert werden, da die Benutzerinnen mit dem richtigen Ansatz begonnen hatten (Selektion des Textes), die erwartete Funktionalität aber nicht gefunden haben.

Markdown

Um Hyperlinks, Bilder und Videos als Notiz zu einem Webinhalt hinzuzufügen oder als Inhalt bei einem Mindmap-Knoten einzufügen, wird *Markdown* verwendet. Die Probandinnen nutzten zwar die Buttons zum Einfügen des *Markdowns* für Bilder und Videos, wussten aber nicht, was sie damit anfangen sollten und löschten das komplette *Markdown* wieder. Einige nahmen an, dass ihr zuvor kopiertes Bild bereits mit Klick auf den Button zum Einfügen von *Markdown* eingefügt wurde. Nur zwei der Testpersonen durchschauten die Syntax und ersetzten den Link mit der eigenen Bildadresse, wobei ihnen die Funktionsweise trotzdem nicht klar war. Das Resultat der Evaluierung ist, dass *Markdown* ohne Hilfestellung für Endbenutzerinnen nicht zumutbar ist. In den anschließenden Gesprächen bestätigten alle Probandinnen, dass sie ohne Erklärung diese Syntax nicht verstanden hätten. Eine Teilnehmerin meinte sogar, dass das eingefügte *Markdown* ein Fehler der Anwendung war.

Eine mögliche Lösung wäre, eine Hilfestellung für *Markdown* anzubieten, in der Schritt für Schritt beschrieben wird, wie die Benutzerin vorgehen muss, um ein Bild oder Video einzufügen. Um die Anwendung jedoch ohne Hilfestellung möglich zu machen, müsste auf *Markdown* verzichtet werden und eine andere Lösung implementiert werden. Beispielsweise könnten Web- und Bildadressen direkt in das Textfeld eingegeben werden. Der Text könn-

te geparsed werden und gefundene Adressen könnten je nach Adresstyp in HTML-Struktur umgewandelt werden.

Notiz-Icon

Das Icon, welches beim Text eingefügt wird, um eine Notiz zu repräsentieren, wurde von einigen Benutzerinnen nicht wahrgenommen. Die Probandinnen suchten nach Möglichkeiten, die erstellte Notiz nochmals anzusehen, übersahen dabei aber das Icon. Zwei Benutzerinnen wiesen sogar darauf hin, dass das Icon unauffällig ist.

Das Icon könnte grafisch auffälliger gestaltet werden, z. B. größer oder farbig anstelle von grau, damit es Benutzerinnen besser finden können.

Video-Player

Damit bei *YouTube*-Videos zu verschiedenen Zeitpunkten Notizen hinzugefügt werden können, wurde ein eigener Player implementiert. Die Benutzerinnen versuchten direkt beim eingebetteten Original-Video Notizen hinzuzufügen und wussten nicht, wie sie dabei vorgehen sollten. Alle Probandinnen spielten zuerst das Original-Video ab und erwarteten eine Funktion, um direkt bei einem Zeitpunkt eine Notiz einfügen zu können. Der Gebrauch des Video-Players war für einige Teilnehmerinnen nicht klar. Es wurde von allen Probandinnen sofort in das inaktive Textfeld geklickt, um eine Notiz zu erstellen, dabei war zuerst nicht klar, dass zuvor im Player der Button zum Einfügen einer Notiz geklickt werden muss. Der Großteil der Benutzerinnen suchte nach dem Speichern der Notizen im Original-Video nach einem Hinweis darauf.

Eventuell könnte eine Zeitleiste mit Markierungen für die erstellten Notizen direkt beim Original-Video angezeigt werden, damit Benutzerinnen ihre Notizen einfacher kontrollieren können. Der teilweise missverständliche Video-Player könnte ersetzt werden, indem direkt auf dieser Zeitleiste, z. B. mit rechter Maustaste, neue Notizen erstellt werden können. Dadurch würde sich auch das Problem des inaktiven Textfeldes erübrigen.

Mindmap Knoten-Inhalt speichern

Wenn ein neuer Inhalt bei einem Mindmap-Knoten vergeben wurde, kann entweder Enter gedrückt oder der X-Button geklickt werden, um zu speichern und zu schließen. Alle Probandinnen versuchten zuerst auf den Hintergrund zu klicken, um den Inhalt zu übernehmen und das Textfeld zu schließen. Ein Großteil der Benutzerinnen gab an, das der X-Button hier missverständlich ist und eher zum Löschen des Inhaltes passen würde. Außerdem war es für zwei der Teilnehmerinnen ungewohnt, die Shift-Taste zu drücken und gleichzeitig mit der linken Maustaste zu klicken, um Inhalte zu editieren.

Um das genannte Problem zu lösen, könnten Inhalte einfach bei Klick auf den Hintergrund übernommen werden. Der X-Button könnte stattdessen entfernt werden oder zum Löschen des Inhaltes dienen. Eventuell könnte auch das Icon für den Button auf beispielsweise eine Speicherdiskette geändert werden, um die Funktion besser zu repräsentieren. Zum Editieren eines Inhaltes könnte, wie bei *Diigo*, ein einfacher Klick mit der linken Maustaste genügen.

5.3 User Experience Fragebogen

Wie bereits zuvor erwähnt, wurde von den Probandinnen nach der Think Aloud Methode ein kurzer Fragebogen ausgefüllt. Dieser Fragebogen beinhaltete neben Fragen zu demografischen Daten, wie z. B. Fragen zu Alter, Geschlecht oder Beruf, auch Gegensatzpaare von Eigenschaften, die eine Anwendung haben kann. Dieser Teil des Fragebogens zur Einschätzung der Anwendung basiert auf dem *User Experience Questionnaire* (UEQ) aus [44] und ist in Abschnitt A.2 ersichtlich. Der Fragebogen bietet eine einfache und schnelle Möglichkeit, die User Experience für eine Anwendung zu messen. Die User Experience gibt Auskunft darüber, wie eine Nutzerin die Bedienung einer Anwendung erlebt, bzw. wie sie sich dabei fühlt. Idealerweise bereitet die Auseinandersetzung mit einer Anwendung Freude oder zumindest Zufriedenheit [18, S. 289].

Da nur fünf Personen den Fragebogen ausfüllen konnten, können keine verlässlichen Ergebnisse garantiert werden. Obwohl alle Probandinnen an der Befragung mit vollkommener Zustimmung teilgenommen haben, würde eine größere Anzahl von Teilnehmerinnen laut [15] für stabilere Ergebnisse sorgen. Trotzdem kann aus diesen wenigen Antworten eine Tendenz erkannt werden.

5.3.1 Interpretation der Ergebnisse

Bei der Auswertung des UEQ werden die 26 Gegensatzpaare von Eigenschaften sechs verschiedenen Skalen zugeordnet. Die Werte einer Skala liegen im Bereich von -3 (extrem schlecht) bis $+3$ (extrem gut), wobei es durch die Berechnung der Mittelwerte sehr unwahrscheinlich ist, Werte über $+2$ oder unter -2 zu erreichen. Positive Evaluierungen liegen bei Werten über 0,8, neutrale Beurteilungen liegen im Bereich von $-0{,}8$ bis 0,8 und als negative Evaluierung gelten Werte unter $-0{,}8$ [15, S. 5].

Die Skalen geben Auskunft über Attraktivität, Durchschaubarkeit, Effizienz, Steuerbarkeit, Stimulation und Originalität der getesteten Anwendung. Die Attraktivität spiegelt den allgemeinen Eindruck wieder. Durchschaubarkeit gibt an, wie einfach eine Anwendung zu erlernen ist. Ob Aufgaben mit einer Anwendung ohne unnötige Anstrengungen gelöst werden können, wird durch die Effizienz angegeben. Die Steuerbarkeit deutet an, ob

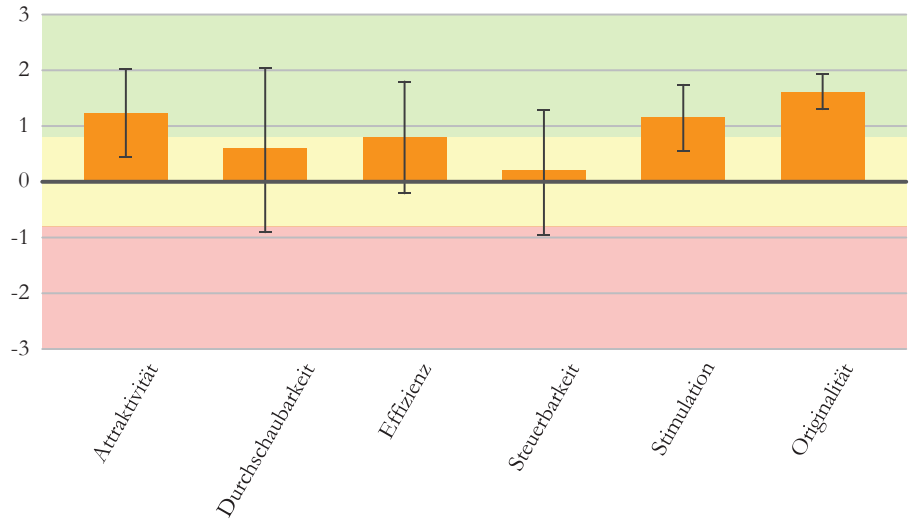


Abbildung 5.1: Ergebnis der User Experience für den Prototyp *tisoor*.

eine Probandin das Gefühl hat, die Interaktion mit der Anwendung unter Kontrolle zu haben. Ob die Bedienung einer Anwendung aufregend und motivierend ist, wird durch die Stimulation angezeigt. Die Originalität gibt an, ob eine Anwendung innovativ, kreativ und interessant für die Benutzerin ist [15, S. 2].

5.3.2 Ergebnisse

Das Ergebnis der User Experience für den Prototyp *tisoor* sagt aus, dass Attraktivität, Stimulation und Originalität positiv und Durchschaubarkeit, Effizienz und Steuerbarkeit neutral beurteilt wurden. Mit dem Wert 1,6 wurde die Originalität am besten beurteilt, die Steuerbarkeit war mit einem Wert von 0,2 die am schlechtesten bewertete Skala, wobei dieser Wert immer noch im neutralen Bereich liegt. In der Abbildung 5.1 sind die Ergebnisse grafisch repräsentiert.

Um einen besseren Eindruck der Qualität einer Anwendung zu bekommen, bietet UEQ die Möglichkeit, die gemessene User Experience der eigenen Anwendung mit Ergebnissen von bestehenden Produkten zu vergleichen. UEQ bietet dazu einen Benchmark, der Daten von 163 Produkt-Evaluierungen mit insgesamt 4818 Teilnehmerinnen enthält. Der Benchmark wird bei UEQ in fünf Kategorien (Exzellent, Gut, Überdurchschnittlich, Unterdurchschnittlich, Schlecht) eingeteilt. Erreicht eine Skala einen exzellenten Wert, bedeutet das, dass die Anwendung bezüglich dieser Eigenschaft zu den 10% der besten Ergebnisse gehört. Gut bedeutet, dass 10% der Ergebnisse aus dem Benchmark besser und 75% schlechter sind. Sind 25% der Ergebnisse besser und 50% schlechter, liegt die Anwendung mit dieser

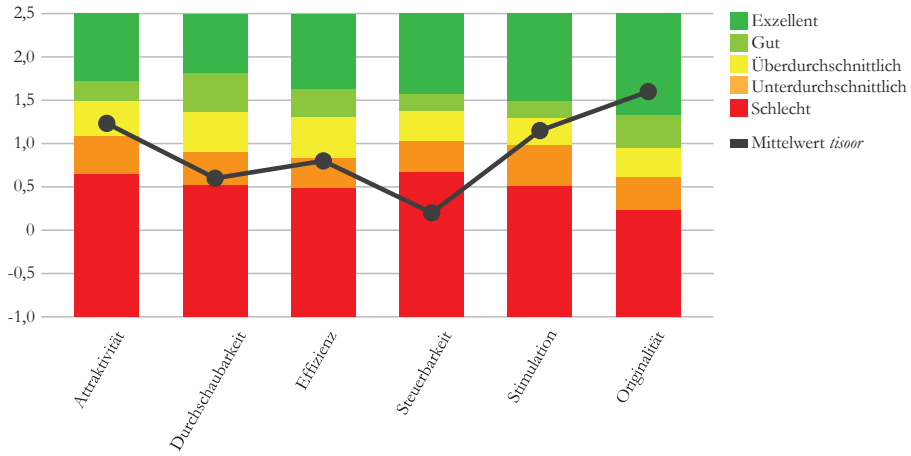


Abbildung 5.2: Ergebnis der User Experience im Vergleich zum Benchmark für den Prototyp *tisoor*.

Eigenschaft über dem Durchschnitt, bei umgekehrter Verteilung der Prozente liegt die Anwendung unter dem Durchschnitt. Erreicht eine Skala einen schlechten Wert, gehört sie zu den 25% der schlechtesten Ergebnisse. Die Graphik in der Abbildung 5.2 zeigt, wo sich die Ergebnisse der User Experience des eigenen Prototyps *tisoor* im Vergleich zum Benchmark befinden. Die Steuerbarkeit des Prototyps ist im Vergleich schlecht, Durchschaubarkeit und Effizienz liegen unter dem Durchschnitt, Attraktivität und Stimulation sind überdurchschnittlich und die Originalität des Prototyps erzielt einen exzellenten Wert.

5.4 Fazit

Bei der Evaluierung des Prototyps *tisoor* konnte festgestellt werden, dass ein Großteil der definierten Anforderungen erfüllt werden konnte. Im Bezug auf Robustheit von Annotationen gegenüber Veränderungen konnte jedoch keine allumfassende Lösung vorgelegt werden. Trotzdem wurden jene Anforderungen, mit deren Lösung sich *tisoor* von bestehenden Tools abgrenzt, erfolgreich erfüllt.

Der Benutzertest durch die Think Aloud Methode zeigte, dass Benutzerinnen den Prototyp großteils ohne Hilfestellung bedienen können. Neben kleineren Schwierigkeiten bei der Bedienung des Prototyps lag das größte Problem bei dem Verständnis von *Markdown*. Eine Verwendung ohne Hilfestellung wäre hier nicht möglich. Die multimedialen Annotationen und die Visualisierung von Bookmarks in Mindmaps wurden von den Benutzerinnen als durchaus positiv und gebrauchstauglich betrachtet.

Die Ergebnisse der User Experience aus den Fragebögen bestätigen die

Beobachtungen aus den Benutzertests. Die Durchschaubarkeit, Effizienz und Steuerbarkeit des Prototyps wurde weniger gut beurteilt. Attraktivität, Stimulation und Originalität wurden von den Benutzerinnen hingegen positiv evaluiert.

Kapitel 6

Zusammenfassung

Ziel dieser Arbeit war es, eine Möglichkeit zu finden, die Recherche im Web für Benutzerinnen zu vereinfachen, indem Funktionen bestehender Tools mit speziellen neuen Funktionalitäten in einem eigenen Prototyp kombiniert werden sollten. Es sollte möglich sein, Bookmarks nicht nur in Listen anzuordnen, sondern diese auch in Mindmaps visuell zu gruppieren und organisieren. Zu multimedialen Inhalten auf beliebigen Webseiten sollten multimediale Annotationen hinzugefügt werden können, um relevante Informationen einfacher wiederzufinden und um diese mit persönlichen Notizen versehen zu können. Bei der Analyse verschiedener Recherche-Tools konnte festgestellt werden, dass die meisten Tools nur auf gewisse Funktionalität spezialisiert sind. Eine Benutzerin müsste demnach mehrere verschiedene Tools verwenden, um beispielsweise Annotationen erstellen zu können und Bookmarks in einer Mindmap verwalten zu können. Multimediale Annotationen wurden von keinem der beschriebenen Tools unterstützt. Dies bestätigte die Relevanz des eigenen Ansatzes und half bei der Definition der Anforderungen und der Entwicklung des Konzeptes für den Prototyp. Im Konzept wurde die theoretische Lösung zur Erfüllung der Anforderungen erarbeitet. Neben der Architektur und dem Datenmodell für die Anwendung wurde hier auch die Funktionsweise der Mindmap und der Annotationen sowie die Gestaltung des User-Interfaces festgelegt. Die Implementierung des Prototyps *tisoor* erfolgte anschließend auf Basis des Konzeptes. Zur Realisierung von Annotationen wurde eine *Google Chrome* Erweiterung umgesetzt. *Symfony* diente zur Implementierung einer REST API und einer Webanwendung für das Bookmark-Archiv und die Mindmap-Funktion. Um herauszufinden, ob die definierten Anforderungen auch erfüllt werden konnten, wurde der entwickelte Prototyp als Grundlage für eine Machbarkeitsstudie verwendet. Fünf Probandinnen evaluierten die Anwendung durch die Think Aloud Methode und lieferten dabei Erkenntnisse über problematische oder missverständliche Teile des Prototyps. Die Probandinnen wurden außerdem gebeten einen Fragebogen zur User Experience auszufüllen. Das Ergebnis daraus lässt er-

kennen, dass die Anwendung einen durchaus positiven Eindruck hinterlässt, die Steuerbarkeit aber als weniger gut empfunden wird. Zusätzlich wurde eine Anforderungsreflexion durchgeführt, die Aufschluss über die Erfüllung der Anforderungen geben sollte.

6.1 Ausblick

Da das World Wide Web mittlerweile für viele Menschen zur Hauptinformationsquelle geworden ist und sich die Online-Recherche als durchaus mühselig und schwierig herausstellen kann, ist es für Benutzerinnen eine Unterstützung Recherche-Tools zu verwenden. Die Probandinnen aus den Benutzertests gaben auch an, dass sie den Prototyp *tisoor* im Alltag benutzen würden. Dazu müssten in einem weiteren Schritt Lösungen für die aus der Evaluierung erkannten Probleme gefunden und umgesetzt werden. Mit einer weiteren Evaluierung des überarbeiteten Prototyps könnte kontrolliert werden, ob eine bessere Bedienbarkeit erzielt werden konnte.

Zukünftig könnte das Konzept auch um eine Lösung für das Verhalten von Annotationen bei dynamischen Webseiten erweitert werden, da das Thema Robustheit gegenüber Veränderungen im Bezug auf das Wiederherstellen von Annotationen in dieser Arbeit nicht berücksichtigt werden konnte. Auch die Konzipierung weiterer Funktionen wäre denkbar, wie etwa das Hinzufügen von Skizzen direkt auf einer beliebigen Webseite.

Abschließend kann festgehalten werden, dass das Konzept von den Probandinnen als originell und positiv empfunden wurde. Für eine problemlose Nutzung im Alltag sind aber jedenfalls noch Verbesserungen durchzuführen. Durch zusätzliche Erweiterungen könnte *tisoor* eine umfassende Funktionalität bieten und somit die Verwendung von mehreren verschiedenen Recherche-Tools überflüssig machen.

Anhang A

Evaluierung des Prototyps

A.1 Aufgaben für die Think Aloud Methode

Folgende Aufgabenstellungen sollten die Probandinnen bei der Think Aloud Methode mit dem Prototyp *tisoor* lösen:

1. Du möchtest einen Städte-Urlaub durch Europa planen und recherchierst dazu im Internet. Als Unterstützung bei deiner Online-Recherche möchtest du eine neue Webanwendung verwenden: *tisoor*. Um die Anwendung nutzen zu können, registriere dich auf der Website:
   - E-Mail-Adresse: benutzer@test.at,
   - Benutzername: Benutzer,
   - Passwort: 0000.
2. Drei Städte, die du unbedingt besuchen möchtest, sind Wien, Paris und London. Benutze den Browser, um den Wikipedia-Eintrag von Wien zu finden. Speichere diese Wikipedia-Seite mit *tisoor* als Bookmark ab. Speichere auch die Wikipedia-Seiten von Paris und London als Bookmark ab.
3. Du möchtest nun wichtige Textpassagen im Wikipedia-Eintrag von Wien hervorheben. Wechsle dazu wieder zur Wikipedia-Seite von Wien. Markiere eine beliebige Textpassage in der Farbe Gelb. Zu einer anderen beliebigen Textpassage fügst du den Text „Wien ist eine wunderbare Stadt!“ als eigene Notiz hinzu.
4. Du möchtest nun auch ein Bild der Stadt Wien und ein YouTube-Video vom Wiener Riesenrad zum Wikipedia-Eintrag von Wien hinzufügen. Erstelle eine neue Notiz bei der Überschrift „Wien“. Füge ein beliebiges Bild und ein beliebiges Video ein. Suche dazu im Internet nach einem Bild der Stadt Wien und auf YouTube (https://www.youtube.com/) nach einem Video vom Wiener Riesenrad.
   TIPP: Bildadressen kannst du mit rechtem Mausklick auf ein Bild ko-

pieren. Den YouTube-Video-Code erhältst du, indem du die letzten 11 Zeichen (z. B. 4OIpdxnzwx4) der Web-Adresse kopierst, die im Browser in der Adressleiste zu sehen ist.

5. Du hast auf der Webseite http://de.parisinfo.com/ ein nettes Video über Paris entdeckt und möchtest eigene Notizen dazu geben. Wechsle im Browser zu dieser Website. Scrolle auf der Seite nach unten, um zu „Videos von Paris“ zu kommen. Füge an zwei beliebigen unterschiedlichen Stellen im Video jeweils eine eigene Notiz hinzu und speichere deine Anmerkungen.
6. Du hast bereits Informationen zu deinem Städteurlaub gesammelt und Bookmarks gespeichert. Nun möchtest du diese Informationen organisieren. Erstelle dazu in deinem persönlichen Archiv eine neue Mindmap. Benenne den Hauptknoten um in „Städteurlaub“. Erzeuge drei weitere Knoten mit den Titeln „Wien“, „Paris“ und „London“.
7. Du hast bereits eine Mindmap für deinen Städteurlaub erstellt. Füge nun deine Bookmarks als neue Knoten zu den jeweiligen passenden Städte-Knoten hinzu. Erstelle beim Knoten „London“ einen weiteren Kind-Knoten und füge als Inhalt ein Bild der Stadt London ein. Suche dazu im Internet ein beliebiges Bild aus.

A.2 User Experience Fragebogen

Die Abbildungen A.1 und A.2 zeigen den Fragebogen, den die Probandinnen im Anschluss an den Think Aloud Test ausfüllen sollten. Dieser Fragebogen basiert auf dem *User Experience Questionnaire* aus [44].

Fragebogen zur Web-Anwendung *tisoor*
Bitte gib deine Beurteilung ab.

Um die Web-Anwendung *tisoor* zu bewerten, fülle bitte den nachfolgenden Fragebogen aus. Er besteht aus Gegensatzpaaren von Eigenschaften, die die Web-Anwendung haben kann. Abstufungen zwischen den Gegensätzen sind durch Kreise dargestellt. Durch Ankreuzen einer dieser Kreise kannst du deine Zustimmung zu einem Begriff äußern.

Beispiel:

attraktiv	○	⊗	○	○	○	○	○	unattraktiv

Mit dieser Beurteilung sagst du aus, dass du die Web-Anwendung eher attraktiv als unattraktiv einschätzt.

Entscheide möglichst spontan. Es ist wichtig, dass du nicht lange über die Begriffe nachdenkst, damit deine unmittelbare Einschätzung zum Tragen kommt.

Bitte kreuze immer eine Antwort an, auch wenn du bei der Einschätzung zu einem Begriffspaar unsicher bist oder findest, dass es nicht so gut zur Web-Anwendung passt.

Es gibt keine „richtige“ oder „falsche“ Antwort. Deine persönliche Meinung zählt!

1. Allgemein

Alter: ____ *Geschlecht:* männlich O weiblich O

Beruf: _____

2. Internetnutzung

Wie viel Zeit verbringst du durchschnittlich jeden Tag im Internet?

_____

Wozu nutzt du das Internet (mehrere Antworten möglich)?

_____

Abbildung A.1

3. Einschätzung zur Web-Anwendung *tisoor*
Kreuze bitte nur einen Kreis pro Zeile an.

	1	2	3	4	5	6	7		
unerfreulich	○	○	○	○	○	○	○	erfreulich	1
unverständlich	○	○	○	○	○	○	○	verständlich	2
kreativ	○	○	○	○	○	○	○	phantasielos	3
leicht zu lernen	○	○	○	○	○	○	○	schwer zu lernen	4
erfrischend	○	○	○	○	○	○	○	einschläfernd	5
langweilig	○	○	○	○	○	○	○	spannend	6
uninteressant	○	○	○	○	○	○	○	interessant	7
unberechenbar	○	○	○	○	○	○	○	voraussagbar	8
schnell	○	○	○	○	○	○	○	langsam	9
neu	○	○	○	○	○	○	○	alt	10
unbedienbar	○	○	○	○	○	○	○	bedienbar	11
gut	○	○	○	○	○	○	○	schlecht	12
kompliziert	○	○	○	○	○	○	○	einfach	13
abstoßend	○	○	○	○	○	○	○	anziehend	14
veraltet	○	○	○	○	○	○	○	modern	15
unangenehm	○	○	○	○	○	○	○	angenehm	16
vorhersagbar	○	○	○	○	○	○	○	unvorhersagbar	17
abwechslungsreich	○	○	○	○	○	○	○	eintönig	18
zuverlässig	○	○	○	○	○	○	○	unzuverlässig	19
ineffizient	○	○	○	○	○	○	○	effizient	20
übersichtlich	○	○	○	○	○	○	○	verwirrend	21
stockend	○	○	○	○	○	○	○	flüssig	22
aufgeräumt	○	○	○	○	○	○	○	überladen	23
schön	○	○	○	○	○	○	○	hässlich	24
sympathisch	○	○	○	○	○	○	○	unsympathisch	25
unauffällig	○	○	○	○	○	○	○	auffällig	26

Vielen Dank!

Abbildung A.2

Anhang B

Inhalt der DVD

B.1 Masterarbeit

Pfad: /

Kriener_Anna_2016.pdf Masterarbeit (Gesamtdokument)

B.2 Abbildungen

Pfad: /Abbildungen

*.pdf . . . . . Vektorgrafiken

*.png . . . . . Screenshots und Rastergrafiken

B.3 Literatur

Pfad: /Literatur

*.pdf . . . . . Kopien der verwendeten Literatur.

B.4 Online-Quellen

Pfad: /Online-Quellen

*.pdf . . . . . Kopien der verwendeten Online-Quellen.

B.5 Projekt

Pfad: /Projekt

/Chrome Erweiterung . Enthält Quell-Code, sowie alle benötigten Dateien für die *Google Chrome* Erweiterung.

/Symfony Backend . . .	Enthält Quell-Code, sowie alle benötigten Dateien für das *Symfony* Backend.

Quellenverzeichnis

Literatur

[1] David Abrams, Ron Baecker und Mark Chignell. „Information Archiving with Bookmarks: Personal Web Space Construction and Organization“. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. Los Angeles, California, USA: ACM Press/Addison-Wesley Publishing Co., Apr. 1998, S. 41–48 (siehe S. 3).

[2] Tim Berners-Lee u. a. „World-Wide Web: The Information Universe“. *Electronic Networking: Research, Applications and Policy* 2.1 (1992), S. 52–58 (siehe S. 1).

[3] Paolo Bottoni u. a. „MADCOW: A Multimedia Digital Annotation System“. In: *Proceedings of the Working Conference on Advanced Visual Interfaces*. Hrsg. von Maria Francesca Costabile. Gallipoli (LE), Italy: ACM Press, Mai 2004, S. 55–62 (siehe S. 5).

[4] Lara D. Catledge und James E. Pitkow. „Characterizing browsing strategies in the World-Wide Web“. *Computer Networks and ISDN Systems* 27.6 (1995), S. 1065–1073 (siehe S. 3).

[5] Andy Cockburn und Bruce McKenzie. „What do web users do? An empirical analysis of web use“. *International Journal of Human-Computer Studies* 54.6 (2001), S. 903–922 (siehe S. 3).

[6] Roy T. Fielding. „Architectural Styles and the Design of Network-based Software Architectures“. Diss. Irvine: University of California, 2000 (siehe S. 27).

[7] Bernhard Haslhofer u. a. „The LEMO Annotation Framework: Weaving Multimedia Annotations with the Web.“ *International Journal on Digital Libraries* 10.1 (2009), S. 15–32 (siehe S. 5, 24).

[8] William P. Jones, Harry Bruce und Susan T. Dumais. „Keeping Found Things Found on the Web“. In: *Proceedings of the Tenth International Conference on Information and Knowledge Management*. Atlanta, Georgia, USA: ACM, Nov. 2001, S. 119–126 (siehe S. 4).

[9] Ricardo Kawase u. a. „The Impact of Bookmarks and Annotations on Refinding Information“. In: *Proceedings of the 21st ACM Conference on Hypertext and Hypermedia*. Toronto, Ontario, Canada: ACM, Juni 2010, S. 29–33 (siehe S. 5).

[10] Glenn E. Krasner und Stephen T. Pope. „A Cookbook for Using the Model-View Controller User Interface Paradigm in Smalltalk-80“. *J. Object Oriented Program.* 1.3 (Aug. 1988), S. 26–49 (siehe S. 36).

[11] Catherine C. Marshall. „Annotation: From Paper Books to the Digital Library“. In: *Proceedings of the Second ACM International Conference on Digital Libraries*. Philadelphia PA, USA: ACM Press, 1997, S. 131–140 (siehe S. 5).

[12] Pooja Mathur und Karrie Karahalios. „Using Bookmark Visualizations for Self-reflection and Navigation“. In: *Extended Abstracts on Human Factors in Computing Systems*. Hrsg. von Dan R. Olsen Jr. u. a. Boston, Massachusetts, USA: ACM, Apr. 2009, S. 4657–4662 (siehe S. 4).

[13] Jakob Nielsen. *Usability Engineering*. San Francisco: Morgan Kaufmann, 1993 (siehe S. 26, 56, 57).

[14] Hartmut Obendorf u. a. „Web Page Revisitation Revisited: Implications of a Long-term Click-stream Study of Browser Usage“. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. San Jose, California, USA: ACM, 2007, S. 597–606 (siehe S. 4).

[15] Martin Schrepp. *User Experience Questionnaire Handbook. All you need to know to apply the UEQ successfully in your projects*. Sep. 2015. URL: http://www.ueq-online.org/ (siehe S. 61, 62).

[16] SensioLabs. *Symfony – The Book*. Version 3.0. Mai 2016. URL: https://symfony.com/pdf/Symfony_book_3.0.pdf (siehe S. 36, 37).

[17] Siu-Tsen Shen und Stephen D. Prior. „My Favorites (Bookmarks) Schema: One Solution to Online Information Storage and Retrieval“. In: *Proceedings of the 2013 International Conference on Information Systems and Design of Communication*. Hrsg. von Carlos J. Costa und Manuela Aparicio. Lisboa, Portugal: ACM, Juli 2013, S. 33–40 (siehe S. 4).

[18] Torsten Stapelkamp. *Interaction- und Interfacedesign. Web-, Game-, Produkt- und Servicedesign, Usability und Interface als Corporate Identity*. Heidelberg: Springer-Verlag, 2010 (siehe S. 26, 56, 61).

[19] Linda Tauscher und Saul Greenberg. „How People Revisit Web Pages: Empirical Findings and Implications for the Design of History Systems“. *International Journal of Human Computer Studies* 47.1 (1997), S. 97–137 (siehe S. 3).

Online-Quellen

[20] URL: https://upload.wikimedia.org/wikipedia/commons/e/e6/Paris_Night.jpg (besucht am 18. 05. 2016) (siehe S. 31–33).

[21] *About Diigo*. URL: https://www.diigo.com/about (besucht am 14. 03. 2016) (siehe S. 8).

[22] *Berlin Wikipedia*. URL: https://de.wikipedia.org/wiki/Berlin (besucht am 10. 06. 2016) (siehe S. 46).

[23] *Can I use – Support tables for CSS*. URL: http://caniuse.com/#cats=CSS (besucht am 12. 05. 2016) (siehe S. 35).

[24] *Can I use – Support tables for HTML5*. URL: http://caniuse.com/#cats=HTML5 (besucht am 12. 05. 2016) (siehe S. 35).

[25] *Can I use – Support tables for JS API*. URL: http://caniuse.com/#cats=JS%20API (besucht am 12. 05. 2016) (siehe S. 35).

[26] *Can I use – Support tables for other functionality*. URL: http://caniuse.com/#cats=Other (besucht am 12. 05. 2016) (siehe S. 35).

[27] *Can I use – Support tables for SVG*. URL: http://caniuse.com/#cats=SVG (besucht am 12. 05. 2016) (siehe S. 35).

[28] *Chrome Extension – Background Pages*. URL: https://developer.chrome.com/extensions/background_pages (besucht am 14. 05. 2016) (siehe S. 35).

[29] *Chrome Extension – Content Scripts*. URL: https://developer.chrome.com/extensions/content_scripts (besucht am 14. 05. 2016) (siehe S. 35).

[30] *Chrome Extension – Manifest File Format*. URL: https://developer.chrome.com/extensions/manifest (besucht am 14. 05. 2016) (siehe S. 35).

[31] *Chrome Extension – Message Passing*. URL: https://developer.chrome.com/extensions/messaging (besucht am 03. 06. 2016) (siehe S. 36).

[32] *Chrome Extension – Overview*. URL: https://developer.chrome.com/extensions/overview (besucht am 14. 05. 2016) (siehe S. 35).

[33] *Coggle Plans*. URL: https://coggle.it/plans (besucht am 14. 04. 2016) (siehe S. 16).

[34] *Daring Fireball: Markdown*. URL: https://daringfireball.net/projects/markdown/ (besucht am 03. 06. 2016) (siehe S. 17).

[35] *Diigo, A Tool For Highlighting And Adding Sticky Notes To The Web, Gets A Facelift*. URL: http://techcrunch.com/2013/08/19/diigo-a-tool-for-highlighting-and-adding-sticky-notes-to-the-web-gets-a-facelift/ (besucht am 14. 03. 2016) (siehe S. 8).

[36] *Diigo Plans and Pricing*. URL: https://www.diigo.com/premium/pricing_table_details (besucht am 14. 03. 2016) (siehe S. 8).

[37] *Fremdenverkehrsamt Paris*. URL: http://www.parisinfo.com/ (besucht am 10. 06. 2016) (siehe S. 48).

[38] *Paris Wikipedia*. URL: https://de.wikipedia.org/wiki/Paris (besucht am 18. 05. 2016) (siehe S. 12, 16, 31, 32).

[39] *scrible*. URL: https://www.scrible.com/ (besucht am 14. 03. 2016) (siehe S. 12).

[40] *Scrible Launches Rich Web Annotation App To The Public*. URL: http://techcrunch.com/2011/05/04/scrible-launches-rich-web-annotation-app-to-the-public/ (besucht am 15. 03. 2016) (siehe S. 12).

[41] *scrible Plans*. URL: https://www.scrible.com/plans (besucht am 15. 03. 2016) (siehe S. 13).

[42] *StatCounter Global Stats – Top 9 Desktop, Mobile & Tablet Browsers from April 2015 to April 2016*. URL: http://gs.statcounter.com/#desktop+mobile+tablet-browser-ww-monthly-201504-201604 (besucht am 12. 05. 2016) (siehe S. 35).

[43] *The Best PHP Framework for 2015: SitePoint Survey Results*. URL: http://www.sitepoint.com/best-php-framework-2015-sitepoint-survey-results/ (besucht am 14. 05. 2016) (siehe S. 36).

[44] *UEQ – User Experience Questionnaire*. URL: http://www.ueq-online.org/ (besucht am 04. 06. 2016) (siehe S. 57, 61, 68).

[45] *VideoAnt Documentation*. URL: https://ant.umn.edu/documentation (besucht am 14. 04. 2016) (siehe S. 18).