Bachelorarbeit am Institut für Informatik der Freien Universität Berlin,

Arbeitsgruppe Software Engineering

# Iterative und benutzerorientierte Entwicklung der Browser-Erweiterung GoHyper

Jennifer Gebske
Matrikelnummer: 4518614
jennifer@gebske.de

Betreuer: Prof. Dr. Lutz Prechelt
Zweitgutachterin: Prof. Dr. Claudia Müller-Birn

Berlin, 20. Januar 2016

**Zusammenfassung**

Diese Arbeit dokumentiert den iterativen und benutzerorientierten Entwicklungsprozess der Browser-Erweiterung GoHyper. Die entwickelte Erweiterung ist ein persönliches Recherchewerkzeug, das die lokale Speicherung von Textstellen und deren Verlinkung im Browser ermöglicht. Die Textauswahl wird auf der zugehörigen Webseite farblich markiert und kann mit zusätzlichen Informationen wie Mehrfachverlinkung und Schlagworten versehen sowie kommentiert werden.

Unter Zuhilfenahme von Methoden, beispielsweise Anforderungsgesprächen, Prototyping, Workflow-Analyse und Usability-Tests, wurde die Anwendung stetig verbessert. Ein separates Modul zur De-/Serialisierung der Position im DOM-Baum wurde mit der Methode des Behavior Driven Development entwickelt und in GoHyper integriert.

Ergebnis des Projekts ist eine getestete und im Chrome Web Store veröffentlichte Browser-Erweiterung.

## Eidesstattliche Erklärung

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

20. Januar 2016

Jennifer Gebske

## Danksagung

Mein Dank gilt Prof. Dr. Lutz Prechelt für die Betreuung dieser Arbeit.
Hervorheben möchte ich die folgenden fleißigen Testpersonen, Gesprächspartner und Gesprächspartnerinnen, die GoHyper ausprobiert und mir darüber ergiebige Rückmeldungen gegeben haben: Henning Möldner, Alexander Naydenov, Edith Pahlke-Gebske, Sonja Peschutter, Sabine Redlich, Elisabeth Sassi, Nico Schlömer, Marco Seelbinder, Daniel Seidel, Lyudmila Vaseva und Falko E. Vedder. Ihre Anregungen haben die Realisierung des Projekts vorangetrieben.
Hardyna Vedder hat mir durch inspirierende Gespräche sowie ihre Testberichte bei der Entwicklung geholfen.
Ich möchte André Gaul für sein überaus hilfreiches Feedback und seine aufmunternde Unterstützung danken.

# Inhaltsverzeichnis

**1 Einleitung** **1**
1.1 Motivation . . . 1
1.2 Ziel . . . 1
1.3 Aufbau . . . 2

**2 Methodik** **3**
2.1 Anforderungsanalyse . . . 3
2.2 User Interface Prototyping . . . 4
2.3 Implementierung . . . 4
2.3.1 Prototyping . . . 4
2.3.2 Testgetriebene Entwicklung . . . 5
2.4 Usability Tests . . . 5

**3 Recherche** **7**
3.1 Verwandte Anwendungen . . . 7
3.1.1 Hypothesis . . . 7
3.1.2 Microsoft Edge . . . 8
3.1.3 Browser-Erweiterungen . . . 8
3.2 Aktuelle Entwicklungen . . . 9

**4 Entwicklungsiterationen** **11**
4.1 Iteration I: Konkretisierung der Idee . . . 11
4.2 Iteration II: Kontext und Kernfunktionalität . . . 13
4.3 Iteration III: Workflow-Analyse . . . 15
4.4 Iteration IV: On-site Usability Tests . . . 17
4.5 Iteration V: Remote Usability Tests . . . 19

**5 Technische Entscheidungen** **23**
5.1 Softwarearchitektur . . . 23
5.2 Frontend . . . 24
5.3 Datenspeicher . . . 25
5.4 DOM-Position-Serializer . . . 26

**6 Fazit und Ausblick** **30**

**A Dokumentation** **31**

**Literatur** **35**

# 1 Einleitung

## 1.1 Motivation

Vorliegende Bachelorarbeit dokumentiert den iterativen Entwicklungsprozess der Browser-Erweiterung[1] GoHyper. Die Ideen zu dieser Anwendung stammen von Hardyna Vedder und wurden von Prof. Dr. Lutz Prechelt in [2] weiter ausformuliert. Die Erweiterung wurde von mir programmiert und während der Entwicklung mit Hilfe von ersten Nutzern iterativ verbessert.
GoHyper ist ein individuelles Recherchewerkzeug im Chrome Browser, mit dem Textstellen – sogenannte Quotes – markiert und mit Links zu anderen Quotes oder Webseiten versehen werden können. Die zu GoHyper hinzugefügten Quotes für eine bestimmte Internetseite, genauer gesagt URL, werden farblich hinterlegt und sind anklickbar. Sie können mit Schlagworten oder einem Kommentar informationssteigernd ergänzt werden. Die Quotes sind sowohl auf der zugehörigen Webseite als auch auf einer in einem eigenen Tab geöffneten Übersicht aller gesammelten Daten zugänglich.
Zielgruppe der Erweiterung sind alle Personen, die ihre Informationssuche im Browser besser strukturieren sowie ihre Ergebnisse speichern und verknüpfen möchten.
Wie in Kapitel 3 gezeigt wird, gibt es bereits zahlreiche Erweiterungen, die Annotationen und Markierungen unterstützen. Eine Anwendung, mit der ein persönliches Netz von Quotes und Links aufgebaut werden kann und durch die Mehrfachverlinkung unterstützt wird, existierte jedoch vor GoHyper noch nicht.
Der Programmcode von GoHyper ist auf GitHub [3] unter der GNU GPLv3 zu finden. Von den ersten Schritten bis zur jetzigen im Chrome Web Store veröffentlichten Version[2] ist die Entstehung des Codes dort zu verfolgen.

## 1.2 Ziel

Das Ziel dieser Arbeit ist die Entwicklung eines Minimum Viable Product (MVP)[3] unter Einbeziehung von Nutzern und deren Rückmeldungen. Dabei kommen Methoden aus dem Bereich des User Centered Designs bzw.

[1] Browser-Erweiterungen arbeiten wie Webseiten und viele fügen dem Browser über eine Benutzerschnittstelle zusätzliche Funktionalität hinzu [1].

[2] Die Erweiterung ist unter `https://chrome.google.com/webstore/detail/gohyper/bemkdkdpdcepkncpclmcphgaddaameff` zu erreichen.

[3] MVP ist ein Konzept für die Produktentwicklung, das versucht Funktionalitäten, Qualität, Ressourcen und Zeit zu balancieren. Laut William S. Junk hat man den Punkt des MVP dann erreicht, wenn eine weitere Reduktion von Funktionalität ein Produkt ergibt, das einen zu begrenzten Nutzen hat, sodass sein Erfolg auf dem Markt unwahrscheinlich sei [4]. Ziel muss es demnach sein, mindestens die für das MVP nötigen Funktionen zu entwickeln. Für Eric Ries ist ein MVP „that version of a new product which allows a team to collect the maximum amount of validated learning about customers with the least effort“ [5]. In dieser Arbeit werden die mitklingenden wirtschaftlichen Aspekte ausgeklammert.

Usability Engineerings zum Einsatz, siehe Kapitel 2. Währenddessen gilt es herauszufinden, was GoHyper ist und was es sein kann. Durch Prototyping und Nutzereinbeziehung wird die Anwendung inkrementell erweitert und verbessert.
Es wird untersucht, welche Erweiterungen es bereits gibt und wie sich GoHyper noch besser von den bestehenden abheben kann. Technische Herausforderung ist es, auszuloten, welche Darstellungen der Extension möglich sind. Es stellt sich außerdem die Frage, wie die Daten gespeichert werden können. Darüber hinaus muss ein Weg gefunden werden, wie die Position eines Quotes im DOM-Baum wiedergefunden werden kann, damit eine farbliche Markierung möglich ist.
Dabei werden die entstehenden Versionen und Ideen einer kleinen Nutzergruppe zum Testen vorgelegt. Die Rückmeldungen werden in neuen Iterationen berücksichtigt.
Der Fokus dieser Arbeit liegt auf der Dokumentation des Entwicklungsprozesses. Technische Entscheidungen werden in Kapitel 5 lediglich angerissen. Technische Details, wie z.B. Algorithmen, können in den Dokumentationen sowie in dem Code auf GitHub [3, 6] nachvollzogen werden.

## 1.3 Aufbau

Kapitel 2 beleuchtet die Methoden, die zur benutzerzentrierten und iterativen Herangehensweise innerhalb dieses Projektes Verwendung finden.
In Kapitel 3 beschäftige ich mich mit der Frage, welche anderen Werkzeuge bereits existieren und gebe darüber einen vergleichenden Überblick. Weiterhin thematisiere ich die aktuellen Veränderungen in der Entwicklung von Firefox-Erweiterungen und legitimiere dadurch meine Fokussierung auf Chrome-Erweiterungen.
Auf die insgesamt 5 Entwicklungsiterationen von GoHyper gehe ich in Kapitel 4 detailliert ein. Dort beschreibe ich die für jede Iteration formulierten Ziele, ihren Ablauf und gebe jeweils ein kurzes Fazit.
Ausgewählte technische Entscheidungen werden in Kapitel 5 dokumentiert. Fragen zur entworfenen Softwarearchitektur und zum Datenspeicher werden dort beantwortet. Einen ausführlicheren Einblick gebe ich in ein Modul zur De-/Serialisierung der Position des markierten Textbereichs innerhalb des DOM-Baums. Dieses Modul wurde aufgrund der allgemeinen Anwendbarkeit separat von GoHyper entwickelt und ist auf GitHub ebenfalls unter der GNU GPLv3 verfügbar [6].
Die Arbeit schließt in Kapitel 6 mit Ausblick und Fazit ab.
Die Funktionsweise der finalen Version ist im Anhang A visualisiert.

# 2 Methodik

In diesem Kapitel gehe ich auf die Entscheidungen zur Methodenwahl ein. Es wurden mehrere Methoden verwendet, um schnell eine funktionierende und für Benutzer brauchbare[4] Browser-Erweiterung zu entwickeln. Bezüglich des iterativen Entwickelns, der Nutzereinbeziehung und der schnellen Reaktion auf Anforderungsänderungen stehen Konzepte wie Agile Softwareentwicklung[5] und Lean Development[6] als Leitgedanken über dem Projekt.
Resultat einer Iteration, also eines vordefinierten Abschnitts des Gesamtprojekts, ist eine fertiggestellte und durch Nutzer getestete Version des Produkts. Eric Ries spricht in seinem Buch „The Lean Startup“ in diesem Kontext von einer „*Build-Measure-Learn* feedback loop“ [9, S. 22]. Demnach gilt es, zuerst das zu lösende Problem herauszufinden, daraufhin ein MVP zu entwickeln, dieses zu testen und aus eventuellen Fehlern oder falschen Annahmen etc. so schnell wie möglich zu lernen.
Zur Einbeziehung der Nutzer habe ich mir Methoden des User Centered Designs und Usability Engineerings ausgesucht, die ich in dieser Arbeit nur kurz beschreiben möchte. Laut Richter und Flückiger beschäftigt sich User Centered Design damit, wie Nutzer systematisch in die Produktentwicklung einbezogen werden können, um gute Benutzbarkeit zu erreichen [7, S. 16]. Usability Engineering legt nach der Definition weniger den Fokus auf Benutzereinbeziehung und betont die systematische Anwendung von Methoden und Techniken, um benutzbare Produkte zu entwickeln [7, S. 172]. Sie heben außerdem hervor, dass Usability Engineering die Aufgabe verfolgt, „die Funktionalität eines Produktes auf ein für den Benutzer ideales Minimum zu reduzieren“ [7, S. 7].
Die genannten Aspekte sind für das GoHyper-Projekt aufgrund der kurzen Entwicklungszeit von Interesse. Die in diesem Kapitel vorgestellten Methoden kommen in den Beschreibungen der Iterationen in Kapitel 4 zum Einsatz, werden aber nicht zwangsläufig linear hintereinander – wie es die hier vorgenommene Reihenfolge suggeriert – angewendet.

## 2.1 Anforderungsanalyse

Zuerst kommt es darauf an, mehr über die Benutzer, ihre Kontexte und Bedürfnisse sowie über die zu entwickelnde Software und ihren Einsatz herauszufinden. Im Rahmen von Anforderungserhebungen können Interviews geführt oder bestehende Systeme analysiert werden [7, S. 22, S. 27].

[4] Eine Anwendung ist in dieser Arbeit nach Richter und Flückiger brauchbar, „wenn sie von den vorgesehenen Benutzern einfach erlernt und effizient verwendet werden [kann] und diese damit ihre beabsichtigten Ziele und Aufgaben zufriedenstellend ausführen können“ [7, S. 4].

[5] Z.B. Agiles Manifest auf `http://agilemanifesto.org/` (Stand 14.01.2016).

[6] Z.B. Ebert/Abrahamsson/Oza [8] und Ries [9].

Ich werde dazu informelle Befragungen durchführen, in denen ich offene Fragen stelle, und persönliche Gespräche mit potenziellen Nutzern führen. Wenn eine erste Version, also ein Prototyp[7], entwickelt ist, werde ich diesen nach der Befragung zeigen bzw. testen lassen und nach Verbesserungsvorschlägen sowie aufgetretenen Problemen fragen. Zuerst soll jedoch immer das Gespräch stattfinden, damit die Nutzer nicht vom Prototypen in ihren Ideen voreingenommen sind.

## 2.2 User Interface Prototyping

Die durch erste Anforderungsanalysen gewonnenen Erkenntnisse können beispielsweise durch User Interface Prototyping visuell umgesetzt werden. Hierbei gilt es, die Anforderungen zu klären sowie eine Benutzerschnittstelle zu entwerfen und zu verbessern.
In dieser Phase werden Funktionsweise, Abläufe und Bedienelemente bestimmt. Es handelt sich um eine iterative Aufgabe, die auch mit Mock-ups oder Papier-Prototypen durchgeführt wird [7, S. 62]. Man kann diese Prototypen mit Benutzern gemeinsam erstellen oder sie ihnen zur Visualisierung und Evaluierung der Benutzerschnittstelle vorlegen und damit interagieren lassen.
Ein Nachteil dieser Methode ist, dass sich manche Testpersonen nicht auf diese spielerische Herangehensweise einlassen können oder wollen, da gerade Papier-Prototypen viel Imaginationskraft erfordern. Der geringe Ressourceneinsatz zur Umsetzung – hier vor allem Zeit, in größeren Projekten auch Geld – ist vorteilhaft.
Anstatt die Funktionsweise der Anwendung nur mit Worten zu beschreiben, können Benutzer ihre Ideen optisch darstellen. Damit können Missverständnisse vermieden werden und einem eventuell fehlenden Vokabular zur Beschreibung von Software wird Abhilfe geschafft.
Die in diesem Kapitel aufgeführten Methoden sind zum Teil stark miteinander verwoben. So werden die während des User Interface Prototyping erstellten Prototypen nicht isoliert verwendet, sondern in Usability Tests (siehe Abschnitt 2.4) einer Prüfung unterzogen.

## 2.3 Implementierung

### 2.3.1 Prototyping

Da am Anfang des Projekts noch keine Software zum Testen und Visualisieren der Idee zur Verfügung stand, konnte ich mit Hilfe von Prototyping (vergleiche Kapitel 5.2) dieses Manko beheben. Prototyping kann dabei nach

[7] Während ein Prototyp entworfen wird, um gestalterische und technische Fragen zum Produkt zu beantworten, gebraucht man ein MVP, um die grundlegende Geschäftsidee zu testen [9, S. 93f]. Hier steht das Testen der Idee ohne geschäftliche Absichten im Vordergrund.

[10, S. 60] wie folgt definiert werden: „A [...] software development technique in which a preliminary version of part or all of the [...] software is developed to permit user feedback, determine feasibility, or investigate timing or other issues in support of the development process.“
Prototyping beschleunigt das Erreichen von ersten Versionen, die getestet werden können. Damit wird ein iteratives und inkrementelles Verbessern der Software unter Nutzereinbeziehung bereits in einer frühen Entwicklungsphase begünstigt.
Bevor die Realisierung der Software beginnt – aber auch oft noch danach –, ist es sinnvoll, eine grobe Struktur des Programms zu entwerfen. Dazu können formale sowie informale Entwurfsmuster der Softwarearchitektur verwendet werden, siehe Kapitel 5.1.

### 2.3.2 Testgetriebene Entwicklung

Ein wichtiger Bestandteil von GoHyper ist das von mir separat entwickelte Modul DOM-Position-Serializer, das in Kapitel 5.4 näher beschrieben wird. Bei der Entwicklung des Moduls habe ich zuerst Testfälle formuliert und daraufhin den Programmcode geschrieben.
Die von mir verwendete Art der testgetriebenen Entwicklung wird als Behavior Driven Development (BDD) [11] bezeichnet. Bevor überhaupt Funktionen implementiert werden, schreibt man Tests, die das Verhalten des zu entwickelnden Programms definieren. Nach der Implementierung wird überprüft, ob die Tests erfolgreich sind. Wenn das der Fall ist, verhält sich die Software wie erwünscht.
Darüber hinaus hätte das Test-Framework Selenium verwendet werden können, um Benutzertests für die Anwendung im Browser zu simulieren. Da mir jedoch zum einen Testpersonen zur Verfügung standen und ich zum anderen mehr Wert auf deren Rückmeldungen zur Benutzbarkeit, Benutzerfreundlichkeit und Funktionalität lege, habe ich von diesem Gebrauch abgesehen.
Weiterhin werden der Continuous Integration Dienst Travis sowie Gemnasium zur Überprüfung der Abhängigkeiten im Projekt sowie zur Information über Aktualisierungen der im GoHyper-Projekt integrierten Programme verwendet.

## 2.4 Usability Tests

Der jeweils aktuelle Entwicklungsstand wird mit Benutzern getestet und optimiert. Nutzertests können zum Teil in extra dafür eingerichteten Laboren unter Beobachtung stattfinden. Ich werde die weniger formalen Usability Walkthroughs zum Testen und Optimieren der ersten Prototypen benutzen [7, S. 26]. Bei diesen moderierten Tests können Nutzern Aufgaben gegeben werden. Sie werden dabei beobachtet und hinterher zu Anregungen, Auffälligkeiten etc. befragt.

Spätere Versionen der Erweiterung werden als Remote Usability Tests auf den Computern der Benutzer und ohne meine Anwesenheit getestet. Die Testpersonen haben lediglich die Beschreibung und Bildschirmfotos von Go-Hyper im Chrome Web Store zur Verfügung. Ich werde sie ohne Vorgabe von Aufgaben bitten, die Anwendung zu testen und mir Rückmeldungen zur Funktionsweise und Bedienbarkeit oder allgemeine Anregungen zu geben.
Anmerkungen sowie Anfragen nach einer bestimmten Funktionalität, die jedoch nur von wenigen Testpersonen zurückgemeldet werden, können mit Hilfe eines Online-Fragebogens noch anderen Testpersonen gestellt werden. Nicht jeder Nachfrage nachzugehen, ist gerade in Bezug auf die Vermeidung von Bloatware[8] und im Hinblick auf das MVP wichtig.
Während in den anfänglichen Tests offene Fragen gestellt werden, wird im finalen Testdurchlauf ein Fragebogen an ausgewählte Testpersonen geschickt, in dem ein Großteil der Fragen nur mit Ja und Nein zu beantworten ist. Damit kann herausgefunden werden, ob andere Benutzer die von Einzelnen angefragte Funktionalität ebenfalls wünschen.
Resultat dieser hier beschriebenen Methode ist die Prüfung der Software auf „Usability“ [7, S. 87]. Man möchte damit Probleme aufspüren, die daraufhin priorisiert und in Lösungsmöglichkeiten transformiert werden müssen.

[8] Siehe `https://de.wikipedia.org/wiki/Bloatware` (Stand 11.01.2016).

# 3 Recherche

## 3.1 Verwandte Anwendungen

Das folgende Kapitel bildet meine Recherche zu verwandten Anwendungen ab. Ich möchte hiermit einen Überblick zu bestehenden Recherchewerkzeugen geben und diese mit den Ideen von GoHyper vergleichen. Kriterien, mit denen ich andere Anwendungen aus- und untersuche, sind – sofern Informationen dazu vorhanden sind – zum Beispiel Benutzerfreundlichkeit, Aktualität der Software, Nutzerzahl und angebotene Funktionen. Mein Fokus liegt in diesem Kapitel vor allem auf den Funktionen.

### 3.1.1 Hypothesis

Hypothesis ist ein 2011 gegründetes Open-Source Software Projekt, das unter anderem eine Chrome-Erweiterung zur Annotation von PDFs und Webseiten bereitstellt. Die Anwendung unterstützt das Kommentieren und gemeinschaftliche Diskutieren im Internet. Um diese nutzen zu können, ist eine Anmeldung mit Benutzername und Passwort notwendig.
Die annotierten Textstellen werden, wie in GoHyper vorgesehen, farblich gekennzeichnet und ihnen können Schlagworte hinzugefügt werden. Es gibt einen öffentlichen und einen privaten Modus, bei dem Annotationen entweder mit allen angemeldeten Benutzern geteilt und diskutiert oder nur für den persönlichen Gebrauch verwendet werden können.
Aufgrund der Ladezeit hat es sehr lange gedauert ein Benutzerkonto anzulegen, außerdem ist das Programm währenddessen abgestürzt. Nach dieser anfänglichen Hürde läuft die Anwendung aber problemlos und es ist spannend zu verfolgen, was andere Nutzer auf Internetseiten kommentieren und diskutieren.
Eine spezielle Unterstützung von Links zwischen den Kommentaren oder Hyperlinks wird nicht angeboten. In ein Textfeld kann jedoch ein Kommentar geschrieben und Hyperlinks können eingefügt werden. Hypothesis stellt außerdem einen Link der Annotation bereit, der zitiert werden kann und alle Informationen, wie Textstelle, Kommentar, Tags und Internetseite der Textstelle, enthält.
Hypothesis hat im Chrome Web Store 14.035 Nutzer (Stand 14.01.2016) und verfolgt das Ziel, das gesamte Internet gemeinschaftlich annotierbar zu machen. Diesen Anspruch hat GoHyper nicht. Es ist jedoch denkbar, dass in einer späteren Version die gesammelten Quotes mit anderen Nutzern geteilt werden. Hypothesis bietet die Funktion des Scrollings zum markierten Text an. Außerdem werden die Textstellen noch auf einer Scrollbar angezeigt. Die Anwendung sowie farbliche Markierungen können ein- und ausgeschaltet werden.
Im Unterschied zu Hypothesis soll bei GoHyper kein Benutzerkonto nötig sein und die Daten sollen im lokalen Speicher des Browsers abgelegt wer-

den. Außerdem stellt Hypothesis keine Unterstützung bei der Eingabe von Hyperlinks, wie eine Suche in bereits verwendeten URLs, bereit.

### 3.1.2 Microsoft Edge

Microsofts Betriebssystem Windows 10, das 2015 veröffentlicht wurde, bietet Microsoft Edge als neuen Standardbrowser an. Edge Annotation ermöglicht persönliche Markierungen und Notizen auf Webseiten anzulegen. Inhalte können kommentiert und Zeichnungen hinzugefügt werden. Die annotierten Seiten können mit anderen Edge Nutzern geteilt werden.
Hier liegt der Schwerpunkt vor allem auf der Möglichkeit der Textmarkierung und Kommentierung auf Webseiten für Microsoft Benutzer. Eine Unterstützung von Verlinkungen der markierten Textstellen oder das Hinzufügen von Hyperlinks wird nicht angeboten.

### 3.1.3 Browser-Erweiterungen

Es existieren zahlreiche Erweiterungen[9], die Notizzettel direkt am Text[10] oder die farbliche Textmarkierung in Webseiten[11] und deren Annotation bzw. all dies zusammen anbieten.
Wenn man etwa nach den Schlagworten „Hyperlinks“ und „Links“ sucht, werden Erweiterungen angezeigt, die das Kopieren und Sammeln von Links sowie das Öffnen von Links in einem neuen Tab anbieten. Fügt man außerdem den Suchbegriff „Tag“ hinzu, werden Erweiterungen angezeigt, die eine Notizzettelfunktion anbieten. Darüber hinaus gibt es Erweiterungen, wie qSnap, die Bildschirmfotos und Annotationen verbinden.
Webtags ist zum Beispiel eine Chrome-Erweiterung mit der man Schlagworte zu Internetseiten hinzufügen kann. Leider sieht man hier jedoch weder direkt, ob man bereits Schlagworte hinzugefügt hat – erst wenn man auf das Icon klickt –, noch werden die markierten Stellen farblich hervorgehoben. Gut an dem Tool ist, dass es zu einem Ankerpunkt scrollt, sodass der verschlagwortete Text im Blickfeld liegt.
Andere Erweiterungen bieten auch Notizen und Schlagworte im Zusammenhang mit Lesezeichen an, so zum Beispiel die Firefox-Erweiterung HyperAnchor oder die Chrome-Erweiterung Hyperbook. Die Daten werden nicht im lokalen Speicher des Browsers aufbewahrt. Für eine Übersicht aller gesammelten Lesezeichen wird man bei Hyperbook auf `getbook.co` weitergeleitet. Leider funktionieren manche Buttons, das angebotene Kontextmenü und die Suchfunktion nach Schlagworten nicht.

[9] Die im Folgenden genannten Erweiterungen sind unter ihrem jeweiligen Namen im Chrome Web Store oder auf der Seite der Firefox Add-ons zu finden.
[10] Z.B. Sticky Notes oder FloatNotes.
[11] Beispielsweise Textmarker. Leider sind hier die Markierungen jedoch nach einem Neuladen nicht mehr vorhanden.

Eine mit 4.229.552 Nutzern (Stand 14.01.2016) im Chrome Web Store sehr weit verbreitete Erweiterung ist der Evernote Web Clipper. Die Anwendung ist eine Erweiterung der Lesezeichen durch Notizen. Hier können ganze Internetseiten oder Schnappschüsse dieser – nicht nur die URL – nach einer vorherigen Benutzeranmeldung gespeichert werden. Zum Beispiel sind Zeichnungen oder die Verpixelung von Bildern möglich. Die Auswahl einzelner Textstellen samt Verschlagwortung und deren farbliche Markierung werden ebenfalls unterstützt. Wie bei Hyperbook ist eine Übersichtseite aller Daten auf eine separate Internetseite – hier `evernote.com` – ausgelagert.
Annotary existiert sowohl als Erweiterung für den Firefox als auch den Chrome Browser. Die Erweiterung erfordert eine Benutzeranmeldung, danach stehen einem Funktionen, wie Notizzettel, farbliche Markierung der Textstellen und Lesezeichen mit Kommentarfunktion, zur Verfügung. Die gespeicherten Daten können öffentlich geteilt oder privat genutzt werden. Eine meiner Meinung nach besonders gelungene Übersicht aller persönlich gesammelten Daten ist auf `annotary.com` aufrufbar. Hierbei ist zum Beispiel zu erwähnen, dass die gesammelten Daten nicht in Tabellenform, sondern als Kacheln mit Fotos zu den gespeicherten Webseiten angezeigt werden.
Nach meiner Recherche zu Chrome- und Firefox-Erweiterungen gibt es kein Angebot, das Textmarkierungen mit Links zu Quotes oder mit Hyperlinks unterstützt. Somit wurde festgestellt, dass GoHyper ein Desiderat im Bereich der Browser-Erweiterungen darstellt. Ein weiterer Vorteil von GoHyper, der mir erst bei meiner Recherche bewusst wurde, ist die lokale Speicherung der persönlichen Daten, auf die einige Benutzer besonderen Wert legen. Damit geht auch einher, dass man für die GoHyper-Erweiterung kein Benutzerkonto anlegen braucht.

## 3.2 Aktuelle Entwicklungen

Aufgrund aktueller Entscheidungen im Bereich der Browser-Erweiterungen habe ich einen Fokus auf Chrome- anstatt Firefox-Erweiterungen gewählt. Mir ist bewusst, dass der Firefox Browser in Deutschland von mehr Menschen genutzt wird als die Alternative Chrome.[12] Neue Entwicklungen bei Mozilla, die den Firefox Browser und seine Erweiterungs-API betreffen, haben mich jedoch dazu veranlasst, mit einer Erweiterung für Chrome zu beginnen.
Am 21.08.2015 kündigte Kev Needham auf dem Mozilla Add-ons Blog die neuen Entwicklungen für den Firefox Browser und seine Erweiterungen unter dem Titel „The Future of Developing Firefox Add-ons“ [14] an. Die neu

[12] Der Marktanteil des Firefox Browsers unter den in Deutschland verwendeten Desktop Browsern liegt bei fast 43 Prozent. Chrome kommt dagegen auf einen Anteil von ca. 29 Prozent [12]. Betrachtet man jedoch die weltweiten Marktanteile, dann liegt Chrome mit fast 54 Prozent an der Spitze [13]. Firefox hat einen weltweiten Marktanteil von ca. 18 Prozent.

entwickelte aber noch in der Implementierung befindliche Erweiterungs-API heißt WebExtensions. Sie soll mit den APIs von Chrome und Opera weitgehend kompatibel sein, sodass eine Entwicklung über Browsergrenzen hinweg einfacher wird. Das bedeutet auch, dass die XPCOM- und XUL-basierten Add-ons für Firefox irgendwann nicht mehr unterstützt werden, da nicht alle mit der neuen Multiprozessarchitektur von Firefox (Electrolysis oder auch e10s) [15] vereinbar sind. Die Tatsache, dass eine Veröffentlichung von WebExtensions auf `https://addons.mozilla.org` momentan (Stand September/Oktober 2015) noch nicht unterstützt wird [16], sprach für einen primären Fokus auf Chrome-Erweiterungen.

# 4 Entwicklungsiterationen

In diesem Kapitel ist die Durchführung der iterativen Entwicklung von GoHyper dokumentiert. Es wurden mehrere in Kapitel 2 beschriebene Methoden mit verschiedenen Testpersonen und potenziellen Nutzern angewendet, mit dem Ziel eine funktionierende und für Nutzer brauchbare Browser-Erweiterungen zu entwickeln.

## 4.1 Iteration I: Konkretisierung der Idee

Ziel dieser Iteration war die Annäherung an das GoHyper-Projekt. Grundlegende Anforderungen wurden erhoben und diskutiert. Außerdem galt es sich mit der Chrome API zur Entwicklung von Erweiterungen vertraut zu machen. Schließlich sollte ein Prototyp zur ersten Visualisierung der Ideen von GoHyper erstellt und anderen Personen vorgestellt werden.
Prof. Dr. Lutz Prechelt hat nach seinem Gespräch mit Hardyna Vedder ihre und die ihm entstandenen Ideen zu GoHyper in [2] festgehalten. Die dortigen Beschreibungen zum Workflow und zur Funktionalität von GoHyper konnte ich als vorläufigen Anhaltspunkt nehmen, um mich dem Projekt zu nähern. Mein Vorhaben habe ich in dieser Phase mit André Gaul, Nico Schlömer und Daniel Seidel – er möchte im Rahmen seiner Bachelorarbeit das GoHyper-Projekt fortsetzen – besprochen. Außerdem habe ich mir, wie in Kapitel 3 beschrieben, einen Überblick über verwandte Anwendungen und technische Möglichkeiten verschafft.
Mit Hilfe der Chrome Extension API und den zugehörigen Anleitungen[13], des CSS Frameworks Bootstrap und des JavaScript Frameworks AngularJS konnte ich schnell einen Prototypen entwickeln, der den vom Benutzer markierten Text, die URL der aktuell aufgerufenen Webseite sowie den Titel ausliest. Diese Daten wurden in einem Pop-up, das von der Erweiterungs-API unterstützt wird, angezeigt. Weitere Formularelemente (Buttons, Eingabefelder für Schlagworte und Kommentare) habe ich ergänzt. Ein Bildschirmfoto der ersten GoHyper-Version ist in Abbildung 1 zu sehen.
Die Öffnung des Pop-ups wurde durch den Klick auf das Icon – bisher nur ein Bild als Platzhalter – der Erweiterung in der Toolbar ausgelöst. Zu diesem Zeitpunkt gab es weder ein Kontextmenü, das durch einen Rechtsklick auf den markierten Text erscheint, noch eine farbliche Markierung des ausgewählten Texts. Die Speicherung von Daten (mehr dazu in Kapitel 5.3) war ebenfalls noch nicht möglich. Nun galt es diese Version technisch weiterzuentwickeln und mit Hilfe von potenziellen Nutzern inkrementell zu verbessern bis eine brauchbare Lösung gefunden ist.
Die aktuelle Version konnte ich bereits den gerade genannten Personen zeigen und mit deren Feedback wurde der Prototyp verbessert. Der momen-

[13] Ein Startpunkt findet sich auf `https://developer.chrome.com/extensions` (Stand 18.01.2016).

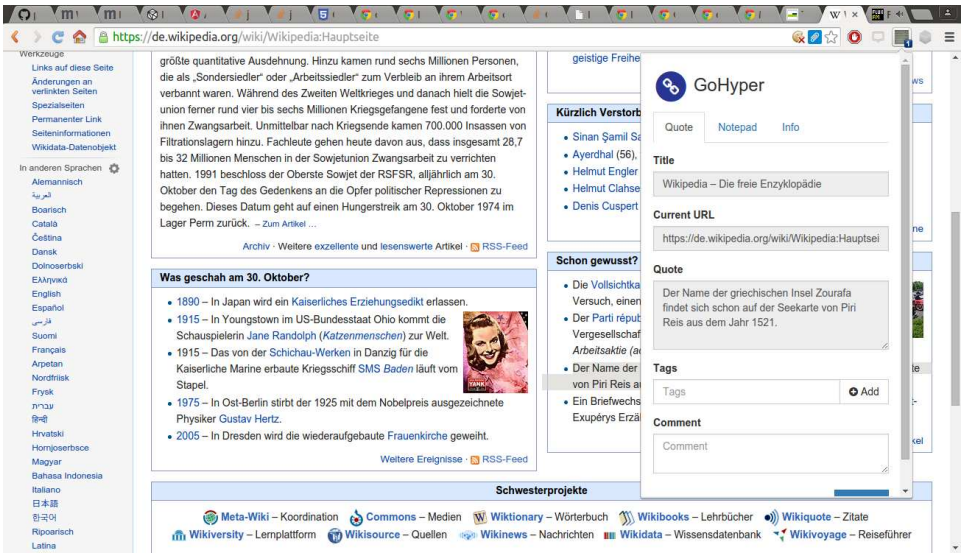



Abbildung 1: Erste Version

tane Arbeitsablauf war wie folgt: Der Nutzer markiert einen Textbereich und klickt dann auf das GoHyper Icon rechts oben im Browser-Fenster. Diese Vorgehensweise erschien sowohl mir als auch den in dieser Iteration befragten Personen nicht intuitiv. Verbesserungsvorschläge, wie ein Quote ausgewählt werden kann, reichten von einem Rechtsklick auf die Textstelle bis hin zu einem Button oder kleinen Pop-up direkt am ausgewählten Text, das sich durch die Selektion mit der Maus öffnet.
Ich entschied mich nach Rücksprache mit den Testpersonen vorläufig für folgenden Ablauf unter Verwendung des Rechtsklicks, da das automatische Öffnen eines Buttons oder kleinen Pop-ups störend wirken kann.

- Benutzer markieren den Text „abc“ mit der Maus und führen darauf einen Rechtsklick aus.
- Ein Kontextmenü mit dem Eintrag „Add ‚abc‘ to GoHyper“ wird angezeigt und ist auswählbar.
- Wird dieses Feld angeklickt, soll sich das Pop-up der Erweiterung mit dem ausgewählten Text „abc“, der URL und dem Titel der Seite als vorbelegte Felder öffnen.

Den gerade dargestellten Workflow musste ich leider aufgrund von eingeschränkten technischen Möglichkeiten überdenken und nach anderen Optionen suchen. Die Chrome API verbietet nämlich das Öffnen des Pop-ups aus der Anwendung heraus. Das Pop-up wird einzig und allein durch den Klick eines Benutzers auf das Erweiterungs-Icon ausgelöst [17, 18]. Daher musste zur Umgehung dieses Hindernisses an einer anderen technisch-gestalterischen Möglichkeit gearbeitet werden, mehr dazu in Abschnitt 4.3.

## 4.2 Iteration II: Kontext und Kernfunktionalität

Am 23. November 2015 fand ein Treffen mit Hardyna Vedder, der Ideengeberin des Projekts, und Daniel Seidel statt. Ich wollte gerne mehr über die Ursprungsidee erfahren und herausfinden, in welchem Umfeld diese entstanden ist. Durch dieses Treffen habe ich mir erhofft die GoHyper-Anwendung insgesamt zu verbessern und vor allem die Hauptfunktionen weiter zu konkretisieren.
Hardyna Vedder ist Autorin und forscht zu Bildungsthemen. Sie hat einige Projekte mit Schulen durchgeführt, in denen die von ihr entwickelte Papillon-Methode mit Schülerinnen und Schülern ausprobiert wurde. Die Kernelemente dieser Methode werden mit „lesen – surfen – vernetzen“ [19] zusammengefasst. In dem Gespräch hat sie uns unter anderem gesagt, dass sie eine Befürworterin des Lernens nach thematischen Schwerpunkten und nicht nach Fächern ist.
Mit Hilfe der zu diesem Zeitpunkt aktuellen GoHyper-Version konnten zu einer Textmarkierung mehrere Hyperlinks gespeichert werden. Damit kann der Benutzer zu einem für ihn relevanten Text weiterführende Informationen speichern und somit für sich persönlich das Angebot der jeweiligen Internetseite thematisch erweitern.
Nach unserem Gespräch über Hardyna Vedders Hintergrund und ihre Ideen habe ich ihr die aktuelle Version der Anwendung (siehe Darstellung 2) vorgestellt und wir haben über Verbesserungsmöglichkeiten geredet. Diese Version behandelt noch nicht die Positionierung der Quotes im HTML-Dokument. Es hat sich während des Gesprächs herausgestellt, dass ihr die Mehrfachverlinkung zu bestimmten Themen direkt am Text auf der entsprechenden Webseite wichtig ist. Beispiele zu dieser Idee finden sich auch auf ihrer Internetseite `http://www.einstein21.org/` und vor allem in ihrem dort online verfügbaren Buch „Goldstaub“, das Grundlage des Papillon-Lernkonzepts ist.
Das quantitative und qualitative Erweitern der Internetseite mit Hilfe von GoHyper ist momentan jedoch nur individuell möglich. Das Teilen dieser Ergebnisse mit anderen Nutzern, die auch die GoHyper-Erweiterung installiert haben, wäre eine lohnenswerte Möglichkeit der Weiterarbeit. In Hardyna Vedders Beispielen sind die zusätzlichen Links jedoch für andere Leser und nicht nur für die persönliche Recherche gedacht.
Eine wichtige Erkenntnis des Gesprächs war, dass die bereits verwirklichten Ideen im Prototypen und ihre Vorstellungen nicht voneinander abweichen. Sie hat die Möglichkeit der Mehrfachverlinkung von Textstellen betont und legt weniger Wert auf das Verlinken von Quotes untereinander.
Frau Vedder schlug vor, Textstellen nicht nur mit Schlüsselwörtern zu versehen, sondern noch Kategorien zur hierarchischen Einordnung anzubieten. Zum Beispiel könnte der Begriff „Gehirn“ zur Medizin und dieser wiederum in eine Kategorie „Bildung und Wissen“ eingeordnet werden. Hardyna

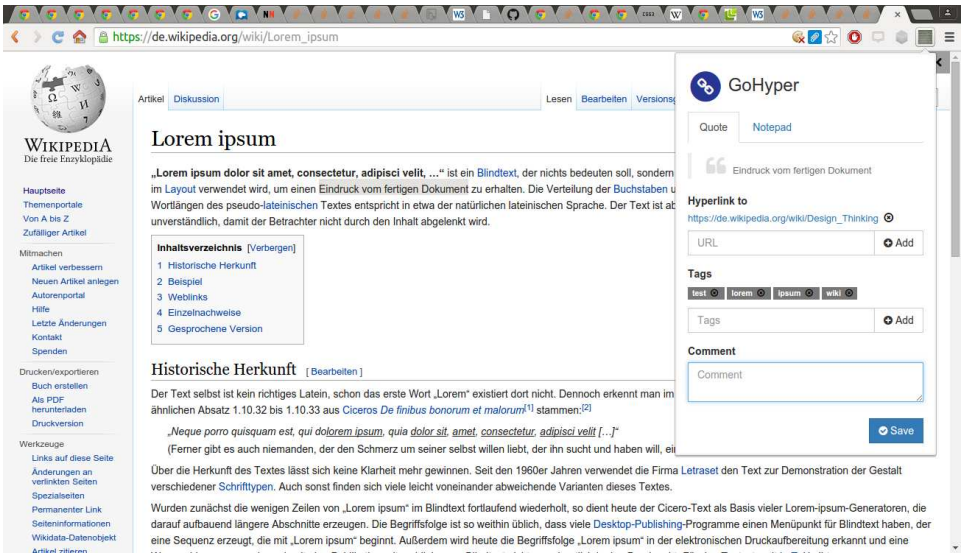



Abbildung 2: Zweite Version

Vedder informierte uns über das Projekt ThingLink, das das Hinzufügen von Links auf Fotos und Videos anbietet. Angeheftete Links, die sich als Pop-up öffnen, können wiederum zu Videos, Musikstücken, Fotos etc. führen.
Als weitere Anregung für GoHyper hat sie uns die Verwendung von Farben gegeben. Mit diesen können Zitate oder Schlagworte nach einer zusätzlichen Dimension strukturiert werden. Diese Möglichkeit bietet auch die Online-Projektmanagementsoftware Trello an. Bisher sind diese Anfragen nach mehr Funktionalität allerdings noch nicht umgesetzt. Nach dem Treffen habe ich mich auf das Verbessern der Anwendung bezüglich der Mehrfachverlinkung sowie das Herausarbeiten der Verlinkung zwischen Quotes konzentriert.
Ein Treffen mit der Übersetzerin Sabine Redlich war in dieser Iteration sehr aufschlussreich, da wir feststellen mussten, dass sie für ihre Arbeit eine andere Anwendung bräuchte. Sie muss sich für manche Projekte in neue Themenfelder einarbeiten und sich das dort verwendete Vokabular aneignen. Diese Arbeit findet im Browser statt, aber sie verwendet dafür keine Erweiterung, sondern merkt sich die Wortpaare aus deutscher und fremdsprachlicher Vokabel zum Beispiel in einem Text- oder Tabellenprogramm bzw. in einer speziellen Software zum Übersetzen. Bei ihren Recherchen sind Quellen in Form von URLs nicht so wichtig wie etwa bei wissenschaftlicher Informationssuche. Eine Verwendung der GoHyper-Software ist nicht ausgeschlossen und sie könnte diese eventuell auch für sich fruchtbar verwenden, aber ihre Bedürfnisse weichen stark von den Vorstellungen anderer potenzieller Nutzer ab. Andere in Kapitel 3.1.3 aufgeführte Erweiterungen mit einer Notizzettelfunktion könnten ihre Anforderungen wahrscheinlich besser erfüllen.

## 4.3 Iteration III: Workflow-Analyse

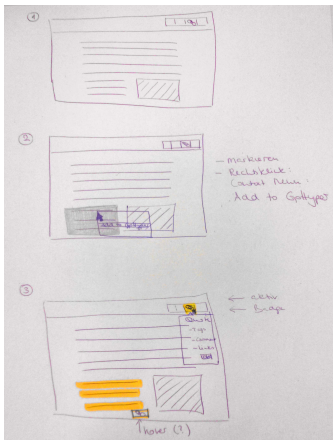
Wie ich in Abschnitt 4.1 kurz erläutert habe, musste anstatt der Benutzung des Pop-ups eine andere Lösung gefunden werden. Um Möglichkeiten des Workflows innerhalb der Anwendung durchzuspielen sowie Alternativen zum Pop-up zu besprechen, habe ich mich mit der Interaction Designerin Elisabeth Sassi getroffen. Sie gab mir die Idee mit Papier-Prototypen zu arbeiten, mit Hilfe derer man die Anwendung und ihre möglichen Abläufe testen kann.
Während des Treffens haben wir den existierenden Prototypen gebraucht und an der Darstellungsoptimierung gearbeitet. Wir überlegten, dass man anstelle des Pop-ups einen Schieber, der sich auf der aktuellen Webseite befindet, verwenden kann. Dieser wird dann nach Bedarf geöffnet – überdeckt in dem Fall einen Teil der darunterliegenden Seite – und nach Erledigung einer Aufgabe wieder geschlossen.
Mir war zu diesem Zeitpunkt zwar bewusst, dass eine teilweise Verdeckung der Seite Benutzer eventuell irritiert, wenn der markierte Text genau darunter liegt. Aber da der ausgewählte Quote nochmal im Schieber wiederholt wird, dachte ich, dass dies kein Problem darstellt. In der in Abschnitt 4.5 beschriebenen Iteration wird der Schieber nochmal thematisiert.
Wir haben in dieser Iteration folgenden Ablauf entwickelt:

- Der Text wird mit der Maus markiert.
- Der Benutzer klickt mit der rechten Maustaste auf die Markierung und führt dort über das angebotene Kontextmenü das Hinzufügen aus.
- Danach geht der Schieber auf und dort wird der markierte Text nochmals angezeigt.
- Weiterhin können zusätzliche Informationen ergänzt werden.
- Nach erfolgreicher Speicherung der Daten schließt sich der Schieber und der Zähler der Erweiterung – von Chrome „Badge“[14] genannt – wird erhöht.
- Wenn die Nutzer auf eine bereits gespeicherte Textmarkierung klicken, öffnet sich der Schieber ebenfalls und Angaben können bearbeitet oder gelöscht werden.

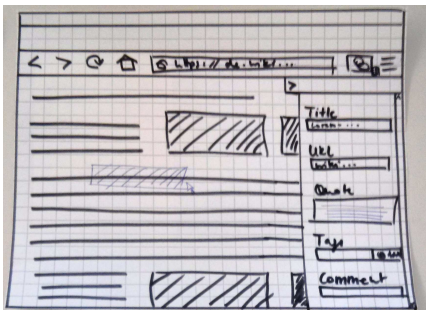
Während unserer Workflow-Analyse haben wir auch über die Möglichkeit einer Gesamtdarstellung aller gespeicherten Daten (zu allen Webseiten) gesprochen. Als Idee haben wir zu diesem Zeitpunkt festgehalten, dass dieser

[14] Die Chrome API unterstützt das Anzeigen weniger Zeichen am Icon der Erweiterung. Mehr dazu findet man unter `https://developer.chrome.com/extensions/browserAction` (Stand 31.12.2015).
Die GoHyper Badge gibt an, wie viele Quotes sich auf der aktuellen Webseite befinden.

Überblick in einem neuen Tab durch einen Klick auf das GoHyper Toolbar Icon ausgelöst werden kann. Diese Möglichkeit habe ich programmiert, aber in einer späteren Iteration verworfen, da die Benutzer der Browser-Erweiterung eine Interaktion mit diesem Icon gewohnt sind und dahinter die Auslösung der Hauptfunktionalität erwarten. So kündigt es auch Chrome bei der Installation der Erweiterung an.
Eine Gesamtübersicht, wie beispielsweise bei der Lesezeichen-Anwendung zu sehen, war uns aber wichtig. Eine weitere Anregung, die Elisabeth Sassi gab, war die Darstellung der Übersicht nicht in Tabellen-, sondern in Kachelform, wie sie etwa bei Pinterest oder Instagram zu finden ist. Anstatt der dort gezeigten Bilder oder Fotos, schlug sie vor, das Favicon der Webseite, zu der der Quote gehört, anzuzeigen. Dadurch kann auch das Auffinden von relevanten Informationen beschleunigt werden.
Weiterhin sind während des Treffens mit Elisabeth Sassi sehr gute Ideen entstanden, die in der aktuellen Version von GoHyper existieren oder in der Zukunft umgesetzt werden können. So schlug sie beispielsweise eine Volltextsuche vor, um relevante Quotes zu finden und präferierte diese Möglichkeit vor anderen Filterungen.
Ich kann feststellen, dass Paper Prototyping eine Möglichkeit darstellt, um Nutzervorlieben und Arbeitsschritte zu erkennen. Außerdem kann ich diese Methode in einem zukünftigen Projekt noch vor dem Software Prototyping einsetzen. Diese Methode diente hier hauptsächlich dazu, die Umstellung vom Pop-up auf den Schieber zu visualisieren und den neuen Arbeitsablauf zu verstehen. Erste Zeichnungen der Abläufe und ein Beispiel eines Papier-Prototypen sind in Abbildung 3 visualisiert.



(a) Erste Visualisierungen des Ablaufs




(b) Papier-Prototyp

Abbildung 3: Beispiele der Workflow-Analyse

Nach der Besprechung der Papier-Prototypen und Rückmeldung der Test-

personen aus Iteration I habe ich ein Mock-up[15] entworfen, in dem die Idee des Schiebers technisch und gestalterisch umsetzt wird. Dieses Mock-up ist in Abbildung 4 dargestellt.

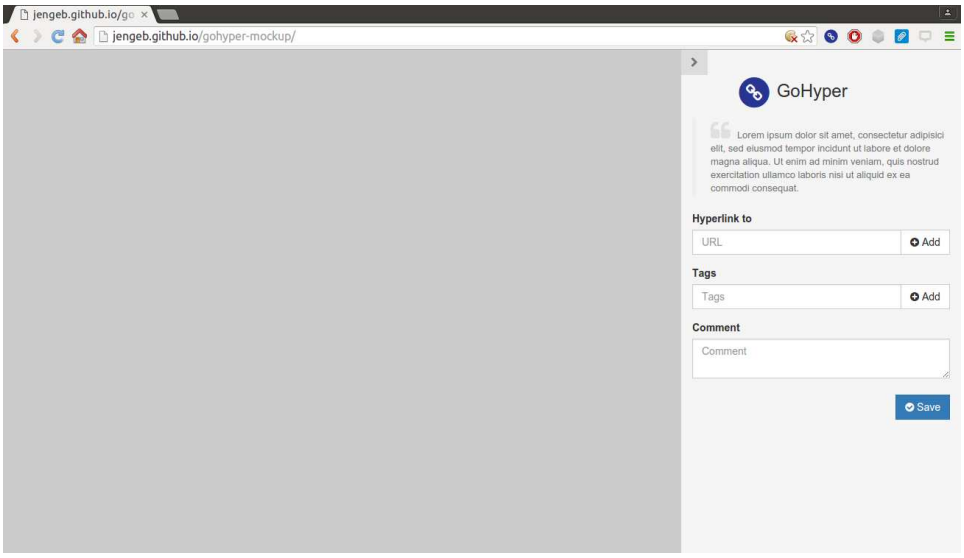



Abbildung 4: Mock-up

Außerdem wurden nun die während dieser Iteration gewonnenen Erkenntnisse, vor allem bezüglich des verbesserten Arbeitsablaufs, in der GoHyper Extension technisch umgesetzt. Für die Einsicht in technische Details verweise ich auf Kapitel 5.2.

## 4.4 Iteration IV: On-site Usability Tests

In dieser Iteration liegt das Augenmerk hauptsächlich darauf, dass Personen die Anwendung selbstständig mit einer kurzen Einführung unter meiner Beobachtung testen. Zentrale Frage dieser Iteration ist die der Bedienbarkeit und Benutzerfreundlichkeit von GoHyper. Etwaige Probleme sollen durch das Testen der Software und die anschließende Befragung aufgedeckt werden.
Dazu habe ich die während dieser Iteration aktuelle Software-Version mit 2 Nutzern getestet, die die Erweiterung an meinem Notebook unter Beobachtung ausprobiert haben. Dabei habe ich aber kleine Hilfestellungen bezüglich des Starts – wie man die Erweiterung bedient: Textstelle markieren und dann Rechtsklick – gegeben. Ich konnte während der Nutzerbeobachtung feststellen, dass zwar verstanden wurde, dass in das Eingabefeld „Hyperlink to“ URLs eingegeben werden können. Diese wurden aber – wie Nutzer es bei der Eingabe in das Browser-Suchfeld gewöhnt sind – ohne „http://“ bzw.

[15] Siehe `http://jengeb.github.io/gohyper-mockup/` (Stand 18.01.2016).

„https://“ eingetragen und stellten daher keine vollwertigen Hyperlinks dar. Beim Anklicken dieser unvollständigen Links erreicht man dann nicht die gewünschte Seite, sondern erhält einen Fehler. Hier sollten die Benutzereingaben validiert und gegebenenfalls mit dem Schema samt spezifischem Teil [20] ergänzt werden.
Weiterhin konnte ich beobachten, wie die Nutzer Schlüsselwörter eingeben. Die eine Testperson schrieb alle Schlüsselbegriffe nur durch ein Leerzeichen getrennt in das Eingabefeld und klickte dann auf „Hinzufügen“, im Gegensatz dazu fügte die andere Testperson die Begriffe einzeln hinzu. Dabei kann man festhalten, dass alle Testpersonen wussten, dass und wie sie dieses Feld benutzen können. Hilfetexte, die sich zum Beispiel hinter kleinen Icons verbergen, habe ich an dieser Stelle nicht als notwendig erachtet, da Nutzer diese Funktion verwenden konnten. In der im nächsten Unterkapitel beschriebenen Iteration wurde u.a. dazu ein Fragebogen verfasst.
Darüber hinaus habe ich beobachtet, dass Nutzer durch eine Hervorhebung bzw. Schattierung der Buttons besser darauf hingewiesen werden können, dass ihre Eingaben noch abgespeichert werden müssen.
Während eines weiteren Treffens mit der Interaction Designerin Elisabeth Sassi schlug sie vor, dass die Reihenfolge der Quotes in der lokalen Übersicht auch an die Reihenfolge der Stellen im Text angepasst werden könnte. Eine ähnliche Umsetzung gibt es bei Hypothesis, die sogar eine entsprechende Markierung der Textstelle am Scrollbalken anzeigen. Da andere Testpersonen dazu keine Vorliebe äußerten, habe ich zunächst eine zeitliche Sortierung gewählt, sodass neu hinzugefügte Einträge immer ganz oben in der Liste erscheinen.
Weiterhin haben während des Projekts 2 Treffen mit Daniel Seidel stattgefunden, in denen wir gemeinsam – wie zum Teil in Abschnitt 4.1 schon erwähnt – an technischen und gestalterischen Verbesserungen der Anwendung gearbeitet haben. Einige der dort diskutierten Funktionalitäten, wie zum Beispiel das Optimieren der Suchfunktionen, will er im Rahmen seiner Bachelorarbeit bearbeiten. In der globalen Anzeige aller Quotes werden bisher erst dann Einträge nach Tags durchsucht, wenn nach der Eingabe die Enter-Taste gedrückt wird. Hier könnte zum Beispiel noch eine Suche, die nach jedem Tastendruck ausgelöst wird, implementiert werden.
Des Weiteren ist uns aufgefallen, dass die Größe des Kommentarfeldes abhängig von der eingegebenen Textlänge dynamisch angepasst werden kann. Wir hatten auch einige Überlegungen zur farblichen Gestaltung von Tags, wie sie etwa bei Trello und Github angeboten wird. Doch ich habe mich mit Fokus auf das MVP gegen diese Zusatzfunktionen entschieden.
Andere Tags, die schon für die aktuelle Internetseite in anderen Quotes verwendet wurden, könnten in dem Eingabefeld mit Hilfe eines Dropdown-Menüs vorgeschlagen werden. Daniel Seidel und ich stimmten überein, dass das automatische Scrollen zu einem bestimmten Quote eine wichtige und umzusetzende Funktion ist. Von ihm stammt weiterhin der Vorschlag, dass

man für Quotes, die nicht im HTML-Dokument wiedergefunden und farblich markiert wurden, die Farbe der Badge am GoHyper-Symbol ändert und damit die Nutzer darauf hinweist. Ein anderer Nutzer meldete mir in diesem Zusammenhang auch zurück, dass diese Quotes in der Übersicht speziell gekennzeichnet werden könnten.
Nach diesen Gesprächen und Tests sowie der Integration einiger der dort erzielten Erkenntnisse in die Anwendung, sollte die Erweiterung mit Anwendern frühzeitig unter realen Bedingungen – ohne Hilfestellung und Beobachtung – getestet werden.

## 4.5 Iteration V: Remote Usability Tests

Um meine Version des MVP der Erweiterung noch mit Benutzern in ihrer realen Umgebung zu testen, habe ich GoHyper am 25. Dezember 2015 im Chrome Web Store veröffentlicht. Eine aktuelle Bildschirmaufnahme (Stand 17.01.2016) ist in Abbildung 5 zu sehen. Nach ein paar Tagen der Tests sowie der Verbesserungen und Aktualisierungen habe ich André Gaul, Edith Pahlke-Gebske, Hardyna Vedder und Daniel Seidel gebeten, die Erweiterung auf ihren Computern auszuprobieren und mir Rückmeldung über ihre Erfahrungen zu geben. Die aus dieser Iteration gewonnenen Erkenntnisse werde ich im Folgenden beschreiben.

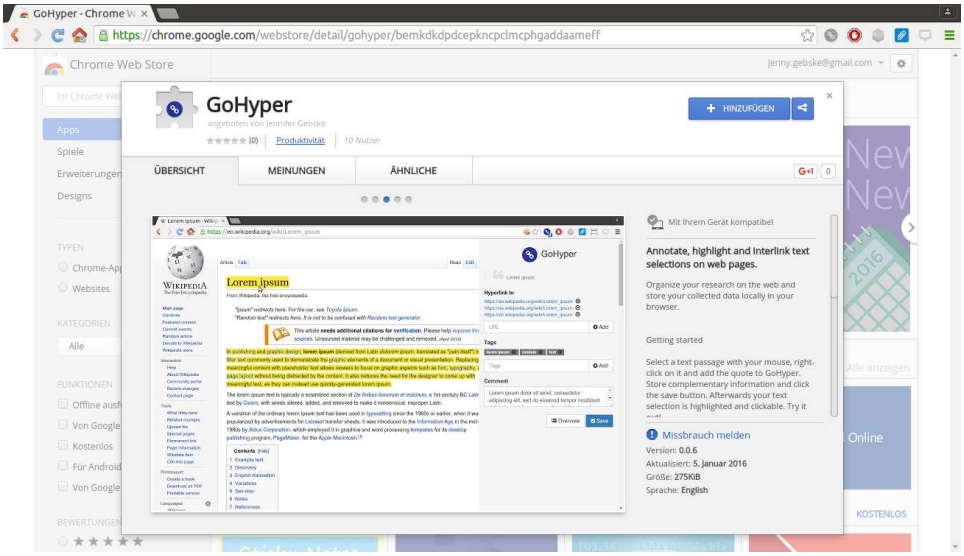
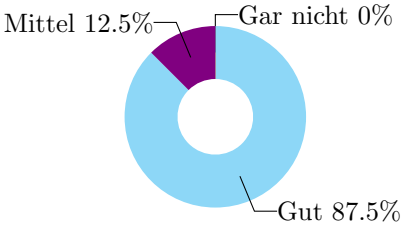



Abbildung 5: GoHyper im Chrome Web Store

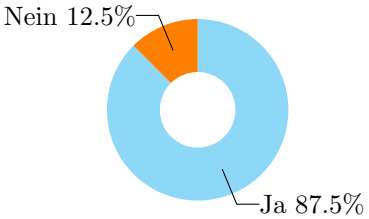
In dieser Phase habe ich den Testnutzern keine Vorgaben oder Fragen vorab gestellt und mich zuerst auf ihre Rückmeldungen konzentriert. Hardyna Vedder hat weitere Testpersonen für mich rekrutiert und mir ihren sowie einen weiteren Testbericht per E-Mail geschickt.

Nach dem Testdurchlauf und dem Nutzer-Feedback habe ich innerhalb eines Gesprächs noch Fragen zur Benutzbarkeit und zur Funktionalität gestellt sowie nach Verbesserungsmöglichkeiten gefragt. Während dieser Iteration wollte ich herausfinden, ob die Nutzer die Anwendung benutzen und für sich nutzbar machen können. Der Programmcode sollte verbessert und allgemeine Ideen sollten für zukünftige Versionen aufgenommen werden.
Eine Testperson berichtete mir, dass sie nicht wusste, was sie in dem Eingabefeld zur URL eintragen soll. Daraufhin habe ich mich entschieden den Platzhalter des Eingabefeldes anzupassen, weil dieser eventuell zu technisch war. Jetzt erscheint dort anstatt „URL“ der Text „Search for quote links or paste a link“. Eine weitere Rückmeldung betrifft ebenfalls dieses Feld. Der Testperson war nicht bewusst, dass Links zu anderen Quotes oder Webseiten hinzugefügt werden können. Es wurde nämlich vorgeschlagen, dass hier die URL der aktuell geöffneten Webseite bereits vorgegeben wird. Durch den eindeutiger formulierten Platzhalter habe ich versucht dieses Missverständnis zu beseitigen. Weiterhin kann dazu die Beschreibung der Extension im Chrome Web Store verbessert werden, indem die Möglichkeiten von GoHyper besser formuliert und hervorgehoben werden. Wobei hier nicht zu erwarten ist, dass alle Anwender diese Beschreibung auch lesen. Falls weitere Nutzer ähnliche Erfahrungen berichten, sollte diese Rückmeldung weiter erforscht werden.
Zusätzlich haben noch sechs weitere Testpersonen die Anwendung zwischen dem 6. und 14. Januar 2016 eigenständig ausprobiert. Um noch ein besseres Meinungsbild aller Testpersonen zu bekommen, habe ich aus ausgewählten Rückmeldungen dieser und der vorherigen Iteration einen kurzen Fragebogen entwickelt. Ziel war es mit dieser kleinen Stichprobe herauszufinden, welche Funktionen höchste Priorität haben und essentiell für GoHyper sind. Die Ergebnisse der Umfrage sind in den Ringdiagrammen der Abbildung 6 veranschaulicht.
Auf einen Teil der Rückmeldungen möchte ich im Folgenden noch explizit eingehen. Einige Rückmeldungen habe ich erwartet, andere waren in dieser Testphase jedoch überraschend. Zu erwarten war beispielsweise, dass Verbesserungen in der Darstellung aller Quotes vorgeschlagen wurden, wie das Zusammenführen von Webseitentitel und URL, sodass dadurch ein Link entsteht. Diese Idee habe ich sofort umgesetzt und sie führte zu einer übersichtlicheren Darstellung.
Weiterhin wurde das Schließen des Iframes durch das Klicken auf die Internetseite als störend empfunden. Es stellte sich nämlich heraus, dass ein Nutzer in das Kommentarfeld Text aus der darunter liegenden Webseite kopieren wollte. Dies wurde im bisherigen Arbeitsablauf noch nicht bedacht. Dieses Feedback war zum Teil auch absehbar und wird in einer kommenden Version umgesetzt.
Unerwartet für mich kam der Wunsch, dass die GoHyper-Oberfläche auf jeder Webseite als globales Fenster zu sehen sein soll. Dies war deshalb un-

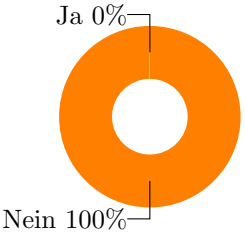
erwartet, weil diese Rückmeldung erst nach mehreren Iterationen aufkam. Diesem Vorschlag – bisher nur von einer Testperson geäußert – muss gegebenenfalls nochmal nachgegangen werden.
Einige Anregungen von den Testpersonen habe ich sofort umgesetzt, andere sind bei GitHub zu den Issues aufgenommen und mit Prioritäten gekennzeichnet. Mir war dabei wichtig, dass die Kernelemente von GoHyper weiter herausgearbeitet werden und dass die Anwendung im Sinnes des MVP benutzbar wie benutzerfreundlich ist. Weniger wichtig war mir zu dem Zeitpunkt das Einbauen von Zusatzfunktionen und optionalen Einstellungen, wie zum Beispiel die Konfiguration von Farben, oder der Vorschlag, die GoHyper-Daten als Excel-Dokument exportieren zu können. Unter den Issues des GoHyper-Projekts sind weiterhin Aufgaben, die zwar wichtig sind, die ich jedoch im Rahmen dieser Bachelorarbeit nicht weiter bearbeiten werde. Beispielsweise sind die Suchstrategien innerhalb der gesammelten Daten verbesser- und erweiterbar.



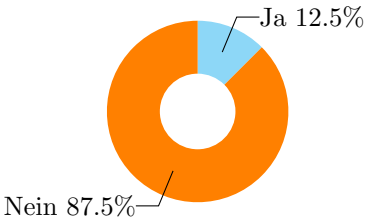

(a) Wie bist du mit der Browser Erweiterung GoHyper zurecht gekommen?



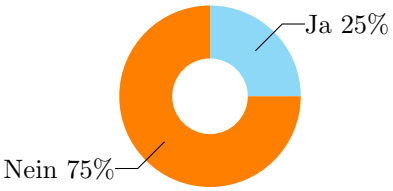

(b) War das Starten der Anwendung durch das Markieren von Text und den Rechtsklick verständlich?



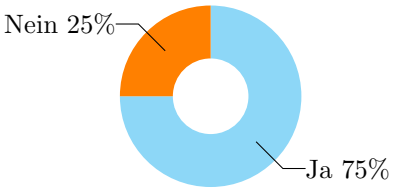

(c) Gab es Probleme bei der Eingabe von Tags?



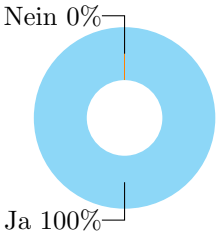

(d) Soll das Löschen eines Quotes nur in der jeweiligen Editieransicht möglich sein? (d.h. nicht in der Listenansicht)



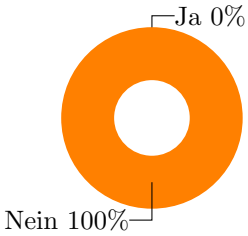

(e) Möchtest du die Farbe der Markierung (momentan nur Gelb) verändern können?




(f) Möchtest du die Anzeige der Markierung ein- und ausschalten können?




(g) Soll man von der Übersicht, die „Quotes on this page“ zeigt, zur Markierung eines Quotes auf der Internetseite springen können?




(h) Haben dir Erklär- oder Hilfetexte gefehlt?

Abbildung 6: Ergebnisse der Online-Umfrage mit 8 Testpersonen

# 5 Technische Entscheidungen

In den folgenden Unterkapiteln beschreibe ich relevante Implementierungsdetails der Chrome-Erweiterung GoHyper. Ich werde dabei auf technische Umsetzungen eingehen und jeweils kurz erläutern, weshalb ich bestimmte Programmiersprachen und Frameworks eingesetzt habe. Außerdem werden Probleme und die dazu gefundenen Lösungen thematisiert.
Vorab sei erwähnt, dass zur technischen Realisierung dieser Arbeit die Paketverwaltungsdienste Bower und npm, der Buildservice gulp sowie Versionsverwaltung mit Git auf GitHub und der Continuous Integration Dienst Travis verwendet werden.
Zur Programmierung der Erweiterung habe ich die JavaScript API[16] des Chrome-Browsers benutzt. Die APIs zur Entwicklung von Erweiterungen der Browser Chrome, Firefox und Opera sollen sich, wie in Kapitel 3.2 näher beschrieben wurde, in Zukunft angleichen.

## 5.1 Softwarearchitektur

In den nachfolgenden Ausführungen gebe ich einen Überblick zu den Komponenten der Erweiterung und gehe auf ihr Zusammenspiel ein. Die drei Programmkomponenten sind das Background Script, das Content Script und der Iframe. Abbildung 7 visualisiert deren Zusammenwirken.

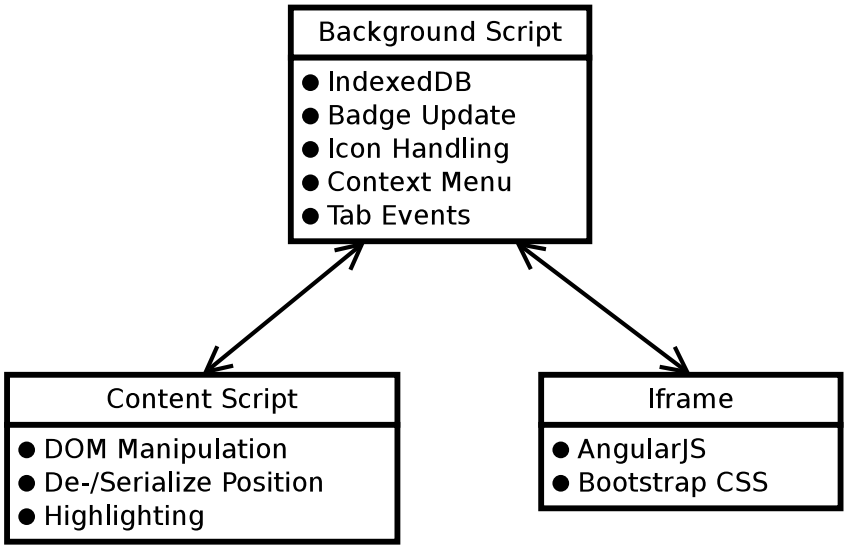



Abbildung 7: Softwarearchitektur

[16] Siehe `https://developer.chrome.com/extensions/api_index` (Stand 09.01.2016).

Die Kommunikation der Komponenten erfolgt über Nachrichtenaustausch[17] in Form von JSON-Objekten, wobei kein direkter Austausch zwischen Content Script und Iframe möglich ist. Kommuniziert wird in dem Fall stellvertretend über das Background Script, das die Daten weiterleitet.

**Background Script** Im Background Script, das in seiner Form als Event Page[18] nur bei Bedarf geladen wird, besteht eine Verbindung zum Datenspeicher, siehe Unterkapitel 5.3. Wird ein Quote hinzugefügt oder entfernt, dann erneuert das Script den Wert innerhalb der Badge, deren Farbe hier ebenfalls gesetzt wird. Es detektiert auch Tab-Ereignisse, zum Beispiel den Wechsel oder die Aktualisierung eines Tabs. Außerdem ist das Background Script für das Kontextmenü und Erweiterungs-Icon zuständig und erkennt, ob der Nutzer es angeklickt hat. Ist dies der Fall wird eine Nachricht an den AngularJS Code im Iframe gesendet. Verschickt das Background Script eine Mitteilung, können sowohl Content Script als auch der Iframe diese empfangen („Broadcast“).

**Content Script** Der JavaScript Code des Content Scripts läuft nicht im Kontext der Erweiterung, sondern in dem der Webseite [1]. Dieses Script ist für DOM Manipulationen verantwortlich. Es kann Code in eine vom Benutzer aufgerufene Webseite einfügen und diese damit verändern. Hier wird die Position des Quotes de-/serialisiert und der Quote-Text farblich markiert, wenn ein entsprechender Auftrag vom Background Script kommt. Diese Funktionalität wird von dem DOM-Position-Serializer bereitgestellt, den ich in Unterkapitel 5.4 beschreibe.

**Iframe** GoHyper Nutzer sehen von der hier dargestellten Struktur hauptsächlich den Iframe. Der dort laufende AngularJS Code sendet beispielsweise Nachrichten an das Background Script, um Daten zu speichern oder abzurufen. Den Iframe beleuchte ich in den nachfolgenden Ausführungen.

## 5.2 Frontend

Zur Programmierung des Frontends habe ich HTML5, CSS und JavaScript (ECMAScript 5) verwendet. Dabei habe ich auf einige Frameworks zurückgegriffen, die mir bei der Gestaltung der Extension geholfen haben. Das CSS Framework Bootstrap setze ich für Typographie, Buttons, Tabellen und andere Seiten- und Formularelemente ein. Bootstrap fördert die schnelle Entwicklung des Frontends und ist daher sehr hilfreich für das Prototyping.

[17] Beispiele zum Nachrichtenaustausch findet man auf `https://developer.chrome.com/extensions/messaging` (Stand 09.01.2016).

[18] Mehr dazu unter `https://developer.chrome.com/extensions/event_pages` (Stand 09.01.2016).

Zur Darstellung des GoHyper Icons sowie der kleinen Icons innerhalb der Buttons nutze ich Font Awesome.
Darüber hinaus habe ich mich für das Model-View-Controller Framework AngularJS entschieden. Es unterstützt die schnelle und strukturierte Softwareentwicklung sowie das Prinzip der Single-Page-Webanwendung. D.h. die Internetseite (hier im Rahmen der Browser-Erweiterung) besteht aus einem HTML-Dokument, dessen Inhalte dynamisch geladen werden können. Damit wird auf das komplette Neuladen der Seite verzichtet. Dieses Prinzip ist auch unter dem Namen AJAX bekannt und für eine eventuell später angedachte Client-Server-Kommunikation vorteilhaft.
Durch das in AngularJS integrierte String Escaping wird sichergestellt, dass kein schädlicher HTML- oder JavaScript-Code eingebettet werden kann. Solches Cross Site Scripting muss beim – eventuell in Zukunft implementierten – Austausch von Daten zwischen Benutzern vermieden werden.
Für Funktionen wie Typeahead und Tooltips habe ich UI Bootstrap in der Version 0.14.3 verwendet. UI Bootstrap ist eine Sammlung von AngularJS Directives speziell für Bootstrap. Directives erweitern HTML um neue Attribute oder Tags [21].
Die Oberfläche von GoHyper ist als Inlineframe – auch Iframe genannt – realisiert. Dieses Element wird benutzt, um ein anderes HTML-Dokument, zum Beispiel anderer Anbieter, in einem vorgegebenen Bereich innerhalb des aktuellen HTML-Dokuments einzubetten [22]. Die GoHyper-Oberfläche wird als Iframe in Internetseiten injiziert und wird somit Teil dieser. Die farblichen Markierungen (siehe Abschnitt 5.4) werden durch `<span>` Inline-Elemente in der im Browser aufgerufenen Internetseite realisiert.

## 5.3 Datenspeicher

Am Anfang stellte sich die Frage, wie und wo die Daten gespeichert werden. Eine clientseitige Browser-Datenbank wurde vorerst einer serverseitigen Datenbank vorgezogen, da zur Erreichung des MVP keine serverseitigen Funktionen, zum Beispiel das Teilen der Daten mit anderen Nutzern, notwendig sind. Durch Modularisierung des Progammcodes wurde jedoch die Möglichkeit des zukünftigen Verwendens anderer Datenbanken berücksichtigt. Die Verwendung einer clientseitigen Datenbank hat auch die Vorteile, dass der Datenzugriff offline möglich ist und durch Unabhängigkeit vom Netzwerk performant ist [23].
Auf der Clientseite bieten sich im Browser momentan zwei persistente Datenspeicher an: Web Storage und die Indexed Database API. Die vormals dritte Alternative Web SQL wird seit 2010 nicht mehr empfohlen [24].
Web Storage besteht aus den Schlüssel-Wert-Speicher Local und Session Storage, wobei der letztgenannte nicht persistent ist [25]. IndexedDB zeichnet sich durch die Speicherung von Objekten aus. Ich habe IndexedDB dem Local Storage gegenüber vorgezogen, hauptsächlich weil IndexedDB eine asyn-

chrone API hat sowie Transaktionen und Indizes unterstützt [26]. Der letzte Punkt ist, besonders für die Suche in größeren Datenmengen, entscheidend. IndexedDB hat jedoch auch den Nachteil, dass ihre komplexe API aufgrund des asynchronen Zugriffs zu einer großen Anzahl verschachtelter Callbacks führt [23]. Am Anfang des Projekts habe ich ein AngularJS Modul verwendet, das die IndexedDB API mit AngularJS nutzbar macht.[19] Da dort jedoch nicht alle Funktionen der nativen IndexedDB API implementiert sind, habe ich von der Verwendung dieses Moduls wieder abgesehen.
Ein Quote-Objekt kann mit folgenden Informationen in der IndexedDB gespeichert werden:

Beispiel 1: Quote-Objekt

```
{
  id: 680,
  url: "https://en.wikipedia.org/wiki/Autonomous_car",
  title: "Autonomous car - Wikipedia",
  quote: "Autonomous car",
  quoteLocation: {
    start: {path: [5, 7, 0], offset: 0},
    end: {path: [5, 9, 1], offset: 0}
  },
  hyperlinks: [
    "http://www.zeit.de/mobilitaet/2015-09/roboterauto-trend
        -oder-hype",
    "https://en.wikipedia.org/wiki/Google_self-driving_car",
    "http://www.heise.de/tp/artikel/46/46954/1.html",
    "http://www.heise.de/ct/artikel/Die-sieben-Huerden-zum-
        selbstfahrenden-Auto-2764145.html",
    "http://www.autonomos.inf.fu-berlin.de/"
  ],
  tags: [
    "Auto", "selbstfahrend", "autonom"
  ],
  comment: "Wie viel soll so ein Auto kosten?",
  createTimestamp: "2016-01-17T16:16:46.125Z",
  updateTimestamp: "2016-01-17T16:23:50.955Z"
}
```

Weitere Informationen werden momentan nicht erfasst. Auf einem Quote-Objekt können die CRUD-Operationen[20] ausgeführt werden.

## 5.4 DOM-Position-Serializer

Der DOM-Position-Serializer ist ein von mir geschriebenes und auf GitHub veröffentlichtes Modul zum Serialisieren und Deserialisieren von Positionen innerhalb eines HTML-Dokuments, genauer DOM-Baums. Unter der Serialisierung versteht man die Transformation einer Eigenschaft, z.B. die Anfangs-

[19] Siehe `https://github.com/bramski/angular-indexedDB` (Stand 30.12.2015).
[20] CRUD steht für Create, Read, Update und Delete.

bzw. Endposition einer Markierung, in eine serielle Datenstruktur, z.B. einen String, um diese abzuspeichern oder über ein Netzwerk zu transportieren. Die Deserialisierung ist die Rücktransformation zur originalen Eigenschaft.
Bei meiner Recherche nach bereits vorhandenen Lösungen bin ich auf die von Tim Down entwickelte JavaScript Range und Selection Bibliothek Rangy[21] gestoßen. Rangy unterstützt zwar – neben zahlreichen anderen Funktionen – das Serialisieren und Deserialisieren von Bereichen im DOM, aber die Veränderungen des DOM durch die Erweiterung führen mit Rangy zu einer falschen Positionsbestimmung, zum Beispiel wenn Quotes gelöscht werden.
Daher habe ich den DOM-Position-Serializer entworfen, der hinzugefügte DOM-Elemente einer vorgegebenen Klasse ignoriert. Dies ist für die farbliche Markierung der Quotes, die in der Funktion `highlight` im Content Script passiert, notwendig. Die vom Nutzer ausgewählte Textstelle wird mit dem `<span>` Element umschlossen, auf dem die CSS-Klasse „goHyper“ gesetzt ist. Diese Klasse sorgt für die farbliche Markierung des Quotes. Hierbei ist entscheidend, dass das hinzugefügte `<span>` Element die Struktur des DOM-Baums, wie in den Beispielen 2 und 3 zu sehen, verändert.[22]
Die Serialisierung einer Position wird ausgehend vom `<body>` Element als Pfad im DOM-Baum und als Offset im Blatt-Element (Textknoten) berechnet. Der Pfad ist ein Array von Null-basierten Indizes, die das jeweilige Element in der Liste der Kind-Elemente angeben. In Beispiel 2 wird die Anfangsposition von „Hallo“ mit `{path: [1, 0], offset: 0}` serialisiert. Die Endposition wird als `{path: [1, 0], offset: 5}` serialisiert. Wenn „Hallo“ nun durch die Extension markiert wird, erhält man Beispiel 3.
Wird nun auch noch „Welt“ ausgehen von dem DOM in Beispiel 3 serialisiert, erhält man dafür die Startposition `{path: [1, 1], offset: 0}` und die Endposition `{path: [1, 1], offset: 4}`. Löscht der Nutzer den Quote „Hallo“, ist die Deserialisierung von „Welt“ ausgehend von dem empfangenen Originaldokument in Beispiel 2 nicht mehr möglich. Die serialisierte Position gibt es nicht mehr und damit ist eine Markierung nicht möglich.
Daher berechnet das Modul alle Serialisierungen bezüglich des Original-DOM. Für beide Stellen von „Welt“ in Beispiel 2 und 3 wird jeweils `{path: [1, 0], offset: 6}` als Anfangs- und `{path: [1, 0], offset: 10}` als Endposition des Textbereichs bestimmt, da der eingefügte Elementknoten mit der goHyper-Klasse im DOM-Position-Serializer nicht mitgezählt wird.
Nachdem ich die Problematik der De-/Serialisierung der DOM-Position erforscht hatte, habe ich angefangen Testfälle zu formulieren, denen der zu entwickelnde DOM-Position-Serializer standhalten sollte. Somit habe ich bei der Entwicklung dieses Moduls die Methode des BDD (siehe Kapitel

[21] Siehe `https://github.com/timdown/rangy` (Stand 16.01.2016).

[22] Whitespace zwischen Elementen ist im DOM-Position-Serializer berücksichtigt, wird jedoch in den hier aufgeführten Beispielen zum besseren Verständnis ignoriert. Mehr zu diesem Thema findet man z.B. unter `https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Whitespace_in_the_DOM` (Stand 19.01.2016).

Beispiel 2: Original DOM

```
<body>
  <div></div>
  <h1>
    Hallo Welt
  </h1>
</body>
```

Beispiel 3: Manipuliertes DOM

```
<body>
  <div></div>
  <h1>
    <span class="goHyper">
      Hallo
    </span>
    Welt
  </h1>
</body>
```

2.3.2) verwendet und dazu das JavaScript Test Framework Mocha eingesetzt. Als das Verhalten der zu schreibenden Funktionen durch die Testfälle definiert war, habe ich die Funktionen implementiert und anschließend überprüft, ob die Tests erfolgreich sind. Die Ergebnisse der Tests sind unter `https://travis-ci.org/jengeb/dom-position-serializer` einsehbar.
Im Folgenden stelle ich die beiden Funktionen zur Serialisierung und Deserialisierung genauer vor.

```
serializePosition(node, offset, rootNode, className)
```

Die Funktion zum Serialisieren einer Position bekommt als Eingabewerte einen Knoten im DOM-Baum, den Offset innerhalb des Knotens, den Wurzelknoten des Dokuments sowie einen optionalen Klassennamen. Elementknoten, auf denen die hier angegebene Klasse angewendet wird, werden in der Positionsbestimmung herausgerechnet. Zur Serialisierung eines ausgewählten Textbereichs, der sich auch über mehrere Knoten strecken kann, muss die Funktion sowohl für Anfangs- als auch Endposition mit ihren zugehörigen Knoten aufgerufen werden.
Die Serialisierungsfunktion startet im angegebenen Knoten und traversiert den DOM-Baum bis zur definierten Wurzel. Sie berechnet dabei den Pfad vom Knoten unter Berücksichtigung des Offsets bis zur Wurzel und rechnet Elementknoten heraus, auf der die goHyper-Klasse gesetzt ist. Trifft die Funktion auf so einen Knoten, betrachtet sie auch die Kinderknoten und bezieht dieses in den Pfad und den Offset mit ein. Somit wird für Pfad und

Offset der ursprüngliche Zustand des DOM-Baums, ohne die vorgenommenen Markierungen, benutzt.
Ergebnis der Funktion ist ein Objekt, das den Pfad von der Wurzel bis zum ausgewählten Knoten innerhalb des DOM-Baums sowie den dortigen Offset zum angegebenen Zeichen angibt.

```
deserializePosition(position, rootNode, className)
```

Die Deserialisierung ist für das Wiederfinden einer Position in einem HTML-Dokument zuständig. Pfad und Offset einer Position – in der Funktion `position` genannt – werden in ein Objekt mit dem tatsächlichen Zielknoten samt Offset umgewandelt. Auch bei dieser Funktion kann ein Klassenname angegeben werden, mit dem der Originalzustand des DOM-Baums ohne Elementknoten mit dieser Klasse benutzt wird. Die Funktion startet im Wurzelknoten und traversiert den Baum bis zur angegebenen Position.
Diese deserialisierten Positionen werden der Funktion `highlight` übergeben, die den Zwischenbereich farblich markiert und anklickbar macht. Für diese Funktionalität greife ich auf das „Class Applier Module“[23] zurück, das das Hinzufügen und Entfernen von Klassen auf Knoten im DOM-Baum ermöglicht.
Ein noch offenes Problem sind Webseiten mit dynamischem Inhalt. Dort ausgewählte Quotes können zwar in den Übersichten von GoHyper angesehen werden, sie werden jedoch nicht entsprechend deserialisiert und farblich markiert, da sie entweder gar nicht mehr oder an anderer Stelle im DOM-Baum zu finden sind.

[23] Zu finden unter `https://github.com/timdown/rangy/wiki/Class-Applier-Module` (Stand 16.01.2016).

# 6 Fazit und Ausblick

Zusammenfassend lässt sich festhalten, dass in kurzer Zeit eine Chrome-Erweiterung unter Einbeziehung von potenziellen Nutzern sowie ein zusätzliches Modul zur De-/Serialisierung einer Position im DOM-Baum entwickelt und veröffentlicht wurden. Die in den Iterationen I und II herausgearbeiteten Kernfunktionen – Mehrfachverlinkung und Verknüpfung von Quotes – sind umgesetzt. Optimierungen des Workflows, Verbesserungen der Bedienbarkeit und weitere Benutzerrückmeldungen konnten gewinnbringend in das Projekt GoHyper einfließen.
Die verwendeten Methoden, mit denen die Ideen der Benutzer Eingang in die Anwendung gefunden haben, stellen sich als fruchtbar heraus. Obwohl man einerseits feststellen kann, dass durch den Methodeneinsatz weniger Zeit zur Softwareentwicklung zur Verfügung stand, hat sich dadurch andererseits im Sinne des MVP eine Fokussierung auf das Wesentliche ergeben. Damit wurde das Ziel, eine brauchbare Browser-Erweiterung zu entwickeln, schneller erreicht.
GoHyper bietet ein großes Potenzial zur Weiterentwicklung. Es gibt Nutzerrückmeldungen (siehe Issues [3]), die in der Anwendung bisher unberücksichtigt sind. Außerdem können Suchstrategien innerhalb der gesammelten Daten verbessert und Probleme im Umgang mit dynamischen Webseiten behandelt werden. Es besteht Bedarf sich mit überlappenden Annotationen zu beschäftigen.
Neben technischen Entwicklungen bleiben auch angrenzende soziale Themen offen. Beispielsweise wäre das Teilen gesammelter Daten mit anderen Nutzern möglich. Auch Fragen, wie Nutzer mit Hilfe von GoHyper gemeinschaftlich recherchieren können und wie man sie bei dieser Zusammenarbeit am besten unterstützt, sind noch zu beantworten.

# A Dokumentation

Im Folgenden umreiße ich die Funktionsweise der Chrome-Erweiterung GoHyper anhand von Bildschirmaufnahmen. In Abbildung 8 sieht man rechts oben neben dem Browser-Suchfeld das kleine GoHyper Icon. Führt man darauf einen Mausklick aus, öffnet sich die GoHyper-Oberfläche und zeigt an, dass auf der aktuellen Webseite noch keine Quotes hinzugefügt wurden.

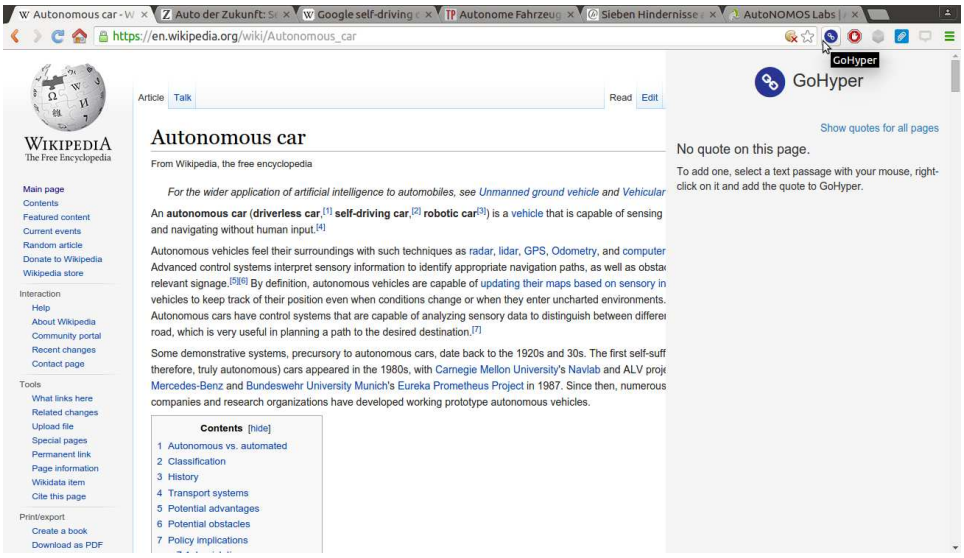



Abbildung 8: GoHyper-Oberfläche nach Mausklick auf das Erweiterungs-Icon (auf einer Webseite ohne Quotes)

Durch Rechtsklick auf einen mit der Maus markierten Text öffnet sich das Kontextmenü, siehe Abbildung 9. Wählt man dort „Add ‚xyz' to GoHyper" aus, wird die GoHyper-Oberfläche angezeigt, siehe Abbildung 10. Links, Tags und zusätzliche Informationen können eingetragen werden. Nach der Betätigung des Save Buttons, werden die Daten gespeichert, die Textstelle wird Gelb hinterlegt und die Badge des Erweiterungs-Icons zeigt eine 1 (Anzahl der Quotes auf der aktuellen Seite), siehe dazu Abbildung 11.
Die Aufnahme in Abbildung 12 zeigt das Ergebnis eines Klicks auf den Gelb markierten Text. Man sieht die GoHyper-Oberfläche mit allen zusätzlichen Daten zum Editieren und Löschen des Quotes.
Abbildung 13 visualisiert das Resultat eines Mausklicks auf das GoHyper Icon für eine Webseite, zu der bereits Quotes gespeichert sind. Man erhält eine Übersicht von allen Quotes zur aktuellen URL.
Möchte man auf alle gesammelten Daten zugreifen, klickt man dort auf „Show quotes for all pages" und bekommt in einem neuen Tab die Ansicht, die Bildschirmfoto 14 illustriert.

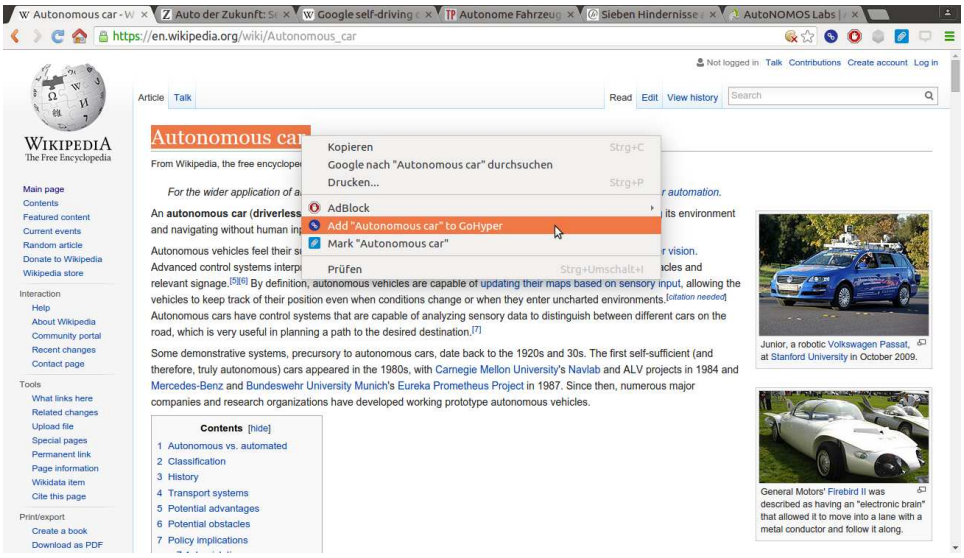



Abbildung 9: Start der Anwendung durch Textauswahl, Rechtsklick und Kontextmenü

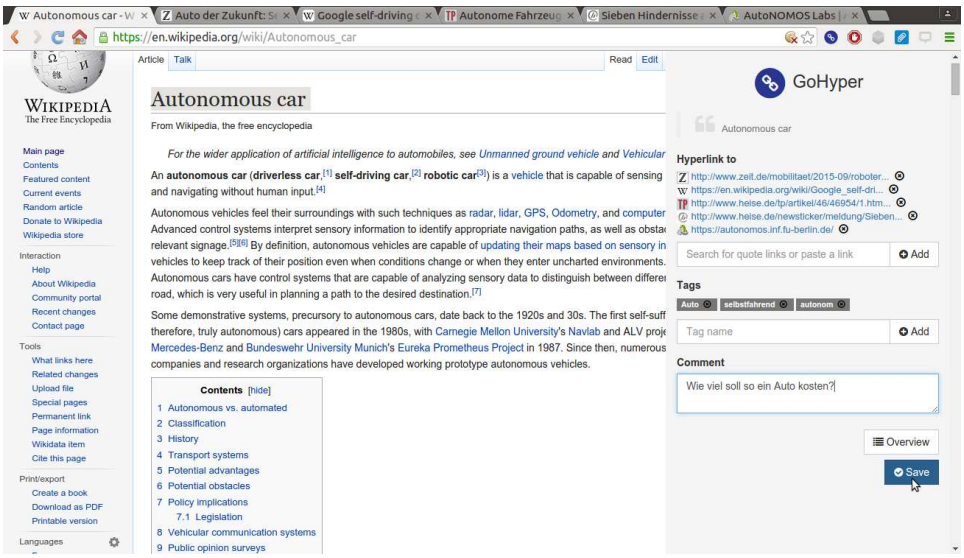



Abbildung 10: Speichern eines Quotes mit zusätzlichen Informationen

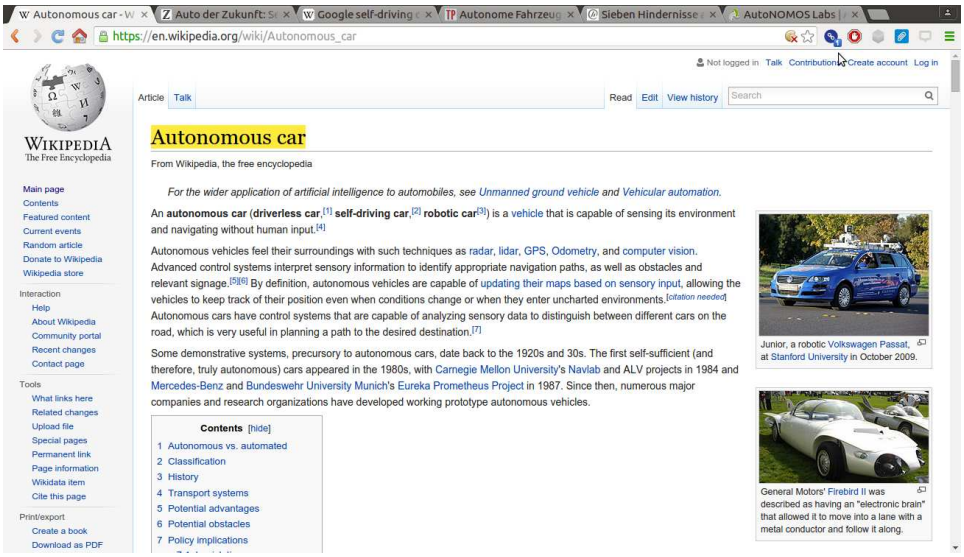



Abbildung 11: Nach der Speicherung ist Quote farblich markiert und Badge hochgezählt

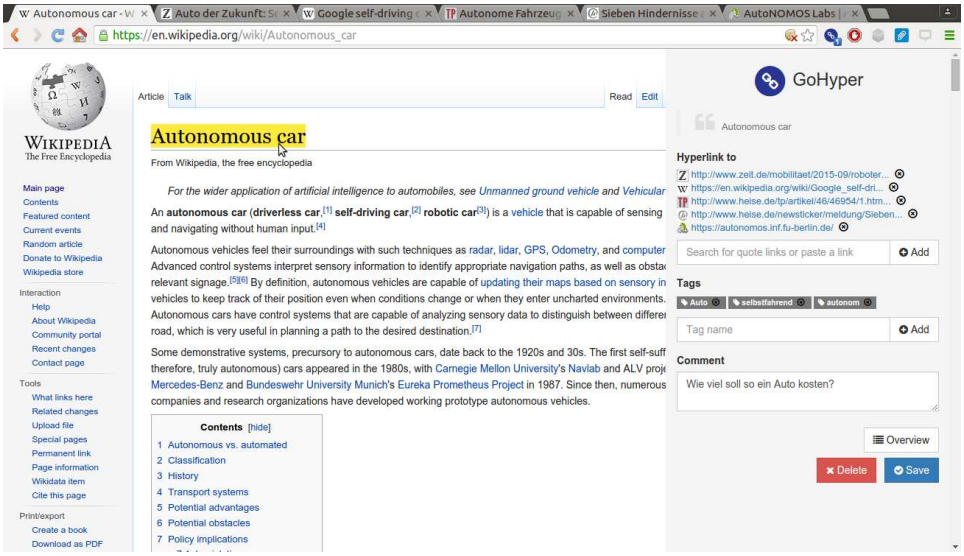



Abbildung 12: Wird ein Quote angeklickt, öffnet sich die Oberfläche von GoHyper mit den zugehörigen Daten

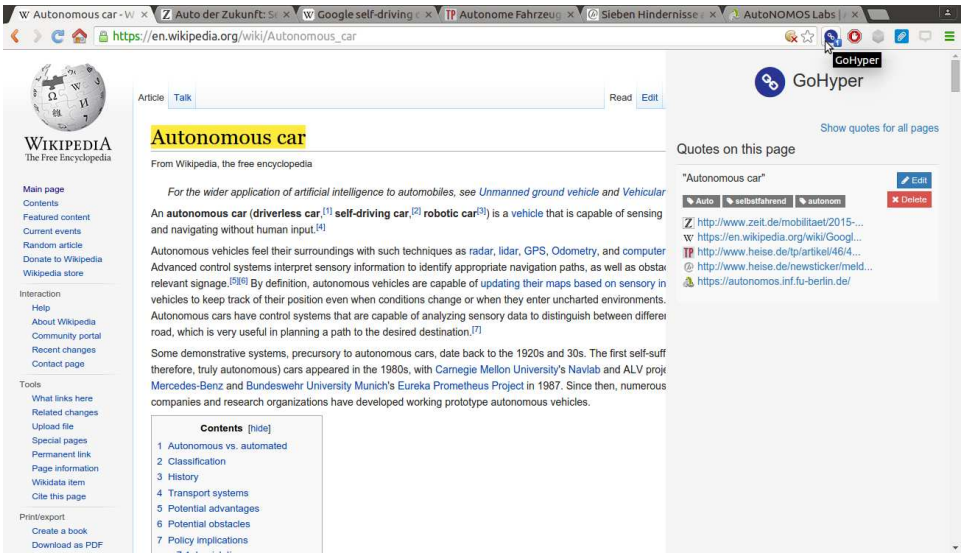



Abbildung 13: GoHyper-Oberfläche nach Mausklick auf Icon (auf einer Webseite mit einem Quote)

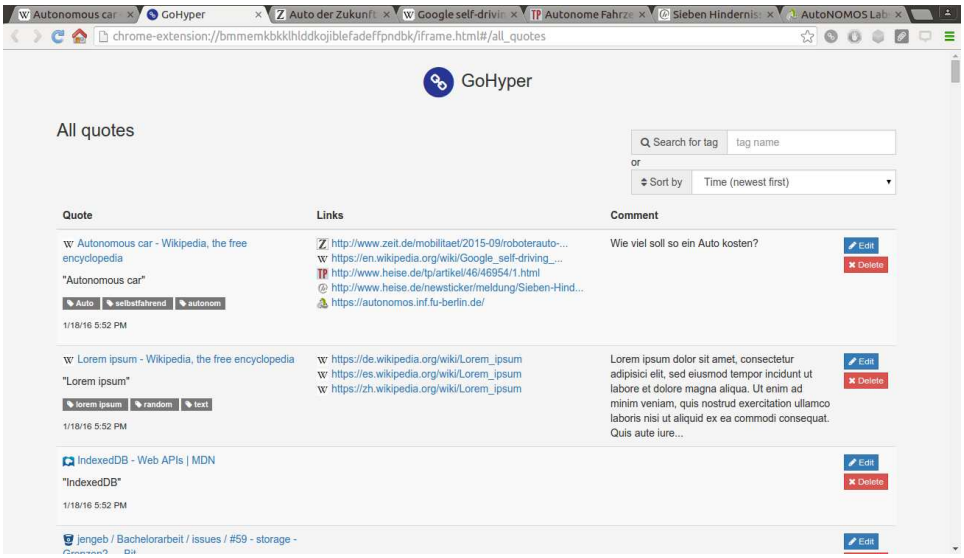



Abbildung 14: Zur Übersicht aller gespeicherten Daten gelangt man über den Link „Show quotes for all pages“

# Literatur


[1] *Chrome Extensions*. URL: https://developer.chrome.com/extensions/overview (Stand 13. 01. 2016).

[2] *GoHyper: Ein neuartiges Recherchewerkzeug (GoHyper: A new kind of web research tool) [Bachelorarbeit]*. URL: http://www.inf.fu-berlin.de/w/SE/ThesisGoHyper (Stand 02. 01. 2016).

[3] Jennifer Gebske. *GoHyper*. 2015–. URL: https://github.com/jengeb/gohyper.

[4] William S. Junk. *The Dynamic Balance Between Cost, Schedule, Features, and Quality in Software Development Projects*. April 2000. URL: http://www.cs.pomona.edu/classes/cs121/supp/DynamicBalance.pdf (Stand 18. 01. 2016).

[5] Eric Ries. *Minimum Viable Product: a guide*. 3. Aug. 2009. URL: http://www.startuplessonslearned.com/2009/08/minimum-viable-product-guide.html (Stand 18. 01. 2016).

[6] Jennifer Gebske. *DOM-Position-Serializer*. 2015–. URL: https://github.com/jengeb/dom-position-serializer.

[7] Michael Richter und Markus D. Flückiger. *Usability Engineering kompakt*. 3. Aufl. Springer, 2013.

[8] Christof Ebert, Pekka Abrahamsson und Nilay Oza. "Lean Software Development". In: *IEEE Software* 29.5 (2012), S. 22–25.

[9] Eric Ries. *The LEAN STARTUP: How Constant Innovation Creates Radically Successful Businesses*. London; New York: Portfolio Penguin, 2011.

[10] *IEEE Standard Glossary of Software Engineering Terminology*. Techn. Ber. 1990.

[11] *Behavior-driven development*. URL: https://en.wikipedia.org/wiki/Behavior-driven_development (Stand 10. 01. 2016).

[12] *StatCounter Desktop Browsers in Germany 2015*. URL: http://gs.statcounter.com/%5C#desktop-browser-DE-monthly-201412-201512-bar (Stand 31. 12. 2015).

[13] *StatCounter Desktop Browsers 2015*. URL: http://gs.statcounter.com/%5C#desktop-browser-ww-monthly-201412-201512-bar (Stand 31. 12. 2015).

[14] Kev Needham. *The Future of Developing Firefox Add-ons*. 21. Aug. 2015. URL: https://blog.mozilla.org/addons/2015/08/21/the-future-of-developing-firefox-add-ons/ (Stand 30. 12. 2015).

[15] *Multiprocess Firefox*. URL: https://developer.mozilla.org/en-US/Firefox/Multiprocess_Firefox (Stand 30. 12. 2015).

[16] *WebExtensions*. URL: https://wiki.mozilla.org/WebExtensions (Stand 30. 12. 2015).

[17] *Stackoverflow: How can I open my extension's pop-up with JavaScript?* URL: http://stackoverflow.com/questions/10479679/how-can-i-open-my-extensions-pop-up-with-javascript (Stand 07. 01. 2016).

[18] *Chromium Issue 30491: Feature request: open extension popup bubble programmatically*. URL: https://code.google.com/p/chromium/issues/detail?id=30491 (Stand 02. 01. 2016).

[19] *PapillonMethod - Rhizomatic Knowledge World*. URL: http://www.einstein21.org/Papillon.htm (Stand 13. 01. 2016).

[20] *Uniform Resource Locator*. URL: https://de.wikipedia.org/wiki/Uniform_Resource_Locator (Stand 05. 01. 2016).

[21] *Creating Custom Directives*. URL: https://docs.angularjs.org/guide/directive (Stand 07. 01. 2016).

[22] *MDN <iframe> Summary*. URL: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/iframe (Stand 08. 01. 2016).

[23] Michael Mahemoff. *Client-Side Storage*. 1. Okt. 2010. URL: http://www.html5rocks.com/en/tutorials/offline/storage/ (Stand 08. 01. 2016).

[24] *Web SQL Database*. URL: http://www.w3.org/TR/webdatabase/ (Stand 08. 01. 2016).

[25] *Web Storage*. URL: http://www.w3.org/TR/webstorage/ (Stand 08. 01. 2016).

[26] *Indexed Database API*. URL: http://www.w3.org/TR/IndexedDB/ (Stand 08. 01. 2016).