Mathias Reppe

Kollaborativer LaTeX Abstraktions Web Editor

Master's Thesis

Graz University of Technology

Institut für Informationssysteme und Computer Medien
Head: Univ.-Prof. Dipl.-Ing. Dr.techn. Frank Kappe

Supervisor: Ao.Univ.-Prof. Dipl.-Ing. Dr.techn. Nikolai Scerbakov

Graz, Juli 2015

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Graz, _____ _____

Date Signature

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Graz, am _____ _____

Datum Unterschrift

Kurzfassung

Das Schreiben von wissenschaftlichen Arbeiten ist eine Herausforderung. Der Aufbau eines strukturierten Texts, untermauert mit Argumenten und nachvollziehbaren Quellen erfordert einen enormen Aufwand. Steckt ein Verfasser so viel Energie in solch eine Arbeit, will er nicht, dass seine Leistung durch eine schlechte Formatierung des Dokuments geschmälert wird. Dadurch kann es passieren, dass bereits während dem Verfassen der wissenschaftlichen Arbeit ein Teil des Fokus des Verfassers nicht auf den Inhalt sondern auf die Formatierung der Arbeit gelenkt wird. Auch am Ende der Arbeit kann ein enormer Arbeitsaufwand entstehen mit dem verwendeten Textverarbeitungssystem für eine ordentliche Formatierung zu sorgen. Abhilfe schafft hier die Verwendung des Textsatzsystemes LaTeX. Während dem Schreiben der Arbeit fokussiert sich der Verfasser ausschließlich auf den Text. Formatierungen werden über Befehle definiert. Generiert der Benutzer die Ausgabe des Dokuments erledigt LaTeX die Formatierung des Dokuments automatisch. Die Verwendung von LaTeX bedarf einer gewissen Einarbeitungszeit. Erst danach lernt man die Vorzüge dieser Software kennen und schätzen. Aufgrund dieser Einstiegshürde wird oft lieber auf bereits erlernte Textverarbeitungssysteme zurückgegriffen.
Diese Arbeit liefert eine Web Oberfläche zum Verfassen von LaTeX Dokumenten. Die Oberfläche ist ein Mittelweg zwischen *„What you see is what you get“* und Texteditor zum Schreiben mit LaTeX Befehlen. Der Fokus des Verfassers bleibt am Inhalt, das Erlernen von Befehlen ist nicht notwendig. Zusätzlich ist die Oberfläche kollaborativ um es mehreren Personen gleichzeitig zu ermöglichen am Dokument zu schreiben. Abgerundet wird der kollaborative LaTeX Editor durch Audio-, Video- und Textkommunikationsmöglichkeiten unter den Benutzern. Technische Herausforderung dieser Arbeit ist das Entwickeln einer kollaborativen Web Oberfläche mit dem Implementieren der dazu notwendigen Algorithmen sowie das Schaffen einer benutzerfreundlichen Oberfläche um LaTeX Dokumente zu verfassen.

Abstract

Scientific writing is a challenge. Writing a structured text based on conclusive arguments with comprehensible references takes an enormous effort. If an author spends so much energy in scientific writing he does not want that a bad formatting minimizes his performance. While writing it is possible that the author focuses on formatting instead of the content of the text. Also at the end of the work there is still an enormous amount of work to format the document. A workaround is to use the typesetting system LaTeX. While writing the author focuses exclusively on the text. The text gets formatted with commands. If the user generates the output of the document LaTeX handles automatically the formatting. Using LaTeX needs certain training. After that training the user appreciates the advantages of LaTeX. Thus of the needed training users tend to use well known word processing systems.
This work provides a web interface to write LaTeX documents. The interface is a middle course of a *"what you see is what you get"* editor and a text editor interface to write the document with LaTeX commands. The focus of the author remains on the content, learning commands is not necessary. Additionally the interface is collaborative. Several users could write simultaneously on the same document. The editor also provides audio, video and text communication between the users.
The technical challenge of this work is the development of a collaborative web interface with the needed algorithms as well as the development of a user-friendly user interface to write LaTeX documents without knowing any command.

Inhaltsverzeichnis

1 Einführung

An der technischen Universität Graz wird es den Studenten nahegelegt Laborprotokolle, wissenschaftliche Arbeiten und Abschlussarbeiten in LaTeX zu verfassen. Dadurch müssen sich Studenten früher oder später mit dem Textsatzsystem LaTeX auseinandersetzen und dessen Syntax erlernen. Im speziellen für Studierende der Fakultät Informatik ist das zusätzliche Erlernen einer Syntax keine große Herausforderungen, da sie während des Studiums bereits mehrere Syntaxen kennengelernt haben. Diese Studenten haben sich bereits auf die Eigenheiten bei der Verwendung einer Syntax eingestellt. Sobald die Einarbeitungszeit in LaTeX überwunden ist, lernen die Benutzer die Vorteile zu schätzen. Die Konzentration auf den Inhalt des Textes ist hier der größte Vorteil, die Formatierung des Dokuments erledigt dabei der LaTeX Compiler. Weitere Vorteile sind die automatisierte Erstellung von Quellen-, Inhalts-, Bild- und Tabellenverzeichnissen. Das komfortable Referenzieren innerhalb des Textes sowie zu externen Quellen macht LaTeX zum idealen Textsatzsystem für wissenschaftliche Texte [35].

Technische Wissenschaftler sind unter allen Forschenden und Studierenden eine Minderheit. Im Studienjahr 2012/13 waren an österreichischen öffentlichen Universitäten 17,85% der ordentlichen Studierenden für technische Studien gemeldet [54]. Für die restlichen 81,15% ist es mit hoher Wahrscheinlichkeit eine Hürde auf die Vorteile von LaTeX zu setzen. Zu hoch ist die Bequemlichkeit auf bereits erlernte Systeme zurück zu greifen wie z.B. das kommerzielle Textverarbeitungsprogramm *Microsoft Word*.

Es gibt somit eine potentielle Zielgruppe die unnötigen Zeitverlust beim Verfassen von Texten hat auf Grund der Probleme die diverse Textverarbeitungsprogramme mit sich bringen bzw. Zeitverlust auf Grund von einlernen in LaTeX erleiden. Ziel dieser Arbeit ist es die Vorteile von LaTeX einer möglichst breiten Zielgruppe ohne hohe Einlernphasen zugänglich zu machen.

Ein Lösungsansatz ist es, die LaTeX Syntax in einem *WYSIWYG* Editor zu abstrahieren. *WYSIWYG* ist die Abkürzung für den Englischen Satz „*What you see is what you get*“ und bedeutet, dass der Benutzer direkt den formatierten Output des Dokument bearbeitet. Eine vollständige *WYSIWYG* Abstrahierung würde

jedoch den Vorteil, dass man sich ausschließlich auf den Inhalt konzentriert aushebeln. Ziel ist es den Spagat zu schaffen Syntax bedienerfreundlich zu verbergen und den Fokus des Benutzers während dem Verfassen des Textes nicht auf die Formatierung zu lenken. Dies bezeichnet man auch als *WYSIWYM*. Diese Abkürzung steht für „*What you see is what you mean*“.

Das durch diese Arbeit entstehende Produkt wird als Web Applikation realisiert. Vorteile hierbei sind, dass keine lokale Installation am Rechner des Benutzers notwendig ist. Das Produkt ist dadurch Betriebssystem unabhängig einzusetzen. Updates werden zentral am Server ausgerollt, der verantwortliche Administrator muss sich keine Gedanken über Updates auf den Client Rechnern machen. Ein weiterer Vorteil der sich durch diese Architektur Entscheidung ergibt ist, dass die verfassten Texte überall auf der Welt über das Internet zugänglich gemacht werden können. Um Datenschutzbedenken von betreibenden Organisationen gerecht zu werden, wird es jedem offen stehen ob er einen bereits eingerichteten Server im Internet verwendet oder sich seine eigene Instanz im internen Netzwerk verwaltet. Auf Grund der Architektur Entscheidung wird das Produkt auch mobilen Endgeräten zugänglich gemacht. Hier ist im speziellen auf die Optimierung für kleine Bildschirmgrößen, Touch Eingabe und geringe Bandbreite zu achten.

Eine häufige Problemstellung ist es, dass ein Text durch mehrere Personen erstellt wird. Verwendet man hierbei binäre Dateiformate ist das Zusammenführen von mehreren Textteilen manuell zu erledigen. Verwendet man textbasierte Dateiformate ist es möglich das Zusammenführen durch eine Versionierungssoftware zu realisieren. Der zweite Schwerpunkt dieser Arbeit neben dem LaTeX basiertem Editor liegt auf dem gemeinsamen Arbeiten an einem Dokument. Es wird möglich sein in Echtzeit an gleichen Textstellen zu arbeiten. Dies kann man sich so vorstellen, dass man in einem Absatz mehrere Cursor sieht und sobald ein anderer Benutzer eine Eingabe tätigt, wird diese auch bei den anderen Benutzer ersichtlich. Das wohl bekannteste Beispiel für Echtzeit Zusammenarbeit ist *Google Docs* [22]. Eine besondere Herausforderung wird es, wenn die Verfasser räumlich getrennt arbeiten und so Diskussionen virtuell ausgetragen werden müssen. Eine benutzerfreundliche Lösung um Teile des Textes zu diskutieren wird angestrebt.

1.1 LaTeX Beispiel

Um das Argument mit der schwer zu erlernenden Syntax aus der Einführung zu untermauern zeigt der folgende Quellcode 1.1 wie die Tabelle 1.1 auf Seite

6 in Form von LaTeX Code zu schreiben ist. Hier wird ganz klar deutlich, dass LaTeX eine hohe Einlernphase hat.

```
\begin{center}
\begin{tabular}{|l|| c | c | c | c | c | c | c |}
\hline
&
\rot{Web basierte Loesung} &
\rot{\textit{WYSIWYG}} &
\rot{Abstraktion von \{LaTeX\}} &
\rot{Abstraktion von {\BibTeX}} &
\rot{Echtzeit Zusammenarbeit} &
\rot{Geeignet fuer wissenschaftliche Texte} &
\rot{Moeglichkeit eigenen Server aufzusetzen} \\
\hline \hline
Overleaf & \OK & \OK & \OK & & \OK & \OK & \OK \\ \hline
ShareLaTeX & \OK & & & & \OK & \OK  & \OK \\ \hline
Authorea & \OK & \OK & & & \OK & \OK & \\ \hline
Papeeria & \OK & & & & \OK & \OK & \\ \hline
Verbosus & \OK & & & & & \OK & \\ \hline
BlueLaTeX & \OK & & & & & \OK & \OK \\ \hline
SageMathCloud & \OK & & & & & & \OK & \\ \hline
LaTeX-lab & \OK & & & & & \OK & \\ \hline
Google Docs & \OK & \OK & & & & \OK & & \\ \hline
LyX & & \OK & \OK & & & \OK & & \\ \hline
LyX in Verbindung mit Owncloud & & \OK & \OK & & & & \OK & \OK \\ \hline
Owncloud Documents & \OK & \OK & & & & & \OK \\ \hline
\end{tabular}
\end{center}
```

Quellcode 1.1: LaTeX Quellcode Beispiel

1.2 Marktanalyse

Bei der Marktanalyse wurden sowohl webbasierte Systeme sowie Client Systeme betrachtet. Unter den *WYSIWYG* Client Systemen ist *LyX* [58] das System, das die meisten LaTeX Features abstrahiert. *LyX* bietet jedoch nicht direkt die Möglichkeit mehrere Personen an einem Dokument arbeiten zu lassen. Eine Möglichkeit Kollaboration mit *LyX* zu realisieren wäre es, die verwendeten Source Dateien von einem Versionierungstool überwachen zu lassen. Setzt man Versionierungstools ein, die normalerweise zum Entwickeln von Software eingesetzt werden wie z.B. *Subversion* oder *Git* hat man das Problem, dass keine Echtzeit Zusammenarbeit möglich ist. Jeder Benutzer muss sich manuell die Updates der Anderen holen und ggf. einen Konflikt händisch mergen. Verwendet man anstatt eines Versionierungstool einen Cloud Synchronisationsdienst wie z.B. *Dropbox* oder *OwnCloud* ist es nicht möglich zeitgleich an der selben Datei zu arbeiten. Diese Synchronisationsdienste bieten keine Möglichkeit Dateien zu mergen. Dies muss händisch erfolgen und ist daher nicht benutzerfreundlich

genug für die Zielgruppe dieses Projekts. Eine weiterer Nachteil wenn man *LyX* in Verbindung mit Synchronisationssoftware für Dateien einsetzt ist, dass man keine Diskussion über Textteile direkt in der Software starten kann.

Unter den webbasierten Systemen muss man zwischen LaTeX basierten Systemen und webbasierten Textverarbeitungssystemen unterscheiden. Das wohl bekannteste webbasierte Textverarbeitungssystem ist *Google Docs* [22]. In Sachen Zusammenarbeit setzt dieses System die Standards. Es ist möglich in Echtzeit die Änderungen der anderen Benutzer zu verfolgen und auch Diskussionen über den Text durchzuführen. Ein Nachteil von *Google Docs* ist, dass man keinen eigenen Server betreiben kann. Was jedoch *Google Docs* als Alternative für die Zielgruppe dieses Projekts auf jeden Fall entfallen lässt, ist die Möglichkeit wissenschaftliche Texte zu verfassen. Es ist nicht möglich zwischen Textabschnitten zu Referenzieren. Es ist nicht möglich ein Abbildungs-, Tabellen-, und Quellenverzeichnis zu erstellen. Eine Alternative zu *Google Docs* ist *Owncloud Documents* [43]. Diese Software kann auch auf einem eigenen Server installiert werden. Für wissenschaftliche Texte ist *OwnCloud Documents* jedoch gänzlich ungeeignet, da es nur die Funktionalitäten eines Texteditors mit einfachen Formatierungsmöglichkeiten wie Fett, Kursiv und Ähnlichem bietet.

Die LaTeX basierten Web Systeme muss man unterscheiden zwischen *WYSIWYG* Systemen und Quelltext Systemen. Quelltextsysteme sind *Verbosus* [61], *BlueLaTeX* [56], *SageMathCloud* [63] und *LateX-lab* [57]. Quelltextsysteme die zusätzlich Echtzeit Zusammenarbeit ermöglichen sind *ShareLaTeX* [59] und *Papeeria* [53]. Auf Grund der fehlenden LaTeXAbstraktions Funktionalität sind diese Systeme jedoch für die Zielgruppe dieses Projekts nicht geeignet.

Zwei Systeme die für die Zielgruppe dieses Projekts realisiert wurden, sind *Authorea* [3] und *Overleaf* [37]. Bei *Authorea* ist es nicht möglich einen eigenen Server zu betreiben, bei Overleaf gibt es eine kostenpflichtige Variante einen eigenen Server zu betreiben. Wie unterscheidet sich nun dieses Projekt von *Auhorea* und *Overleaf* abgesehen von der Möglichkeit eine eigene Instanz zu betreiben? Dieses Projekt wird eine vollständige Abstraktion von BibTeX bieten. Es wird nicht nötig sein eine einzige Zeile Code zu schreiben um auf Quellen zu referenzieren. Weiters sind Tabellen Bestandteil von vielen wissenschaftlichen Arbeiten und wie im LaTeX Beispiel auf Seite 2 ersichtlich besonders kompliziert zu realisieren mit LaTeX. Eine benutzerfreundliche Möglichkeit Tabellen zu zeichnen wird dieses Projekt beinhalten. Dieses Projekt wird auch besonderen Wert darauf legen den Fokus der Funktionalität auf wissenschaftliche Texte und die dafür benötigten Textbausteine zu legen. Die Oberfläche dieses Projekts wird minimalistisch gestaltet sein, um den Benutzer auf den Textinhalt zu fokussieren und nicht auf das Design. Dies wird dadurch ermöglicht, dass

dieses Projekt kein *WYSIWYG* Editor sondern ein LaTeX Abstraktions Editor oder anders gesagt ein *WYSIWYM* Editor wird.

In der Tabelle 1.1 auf Seite 6 sind die in der Marktanalyse untersuchten Systeme und dessen Features zusammengefasst.

	Web basierte Lösung	*WYSIWYG*	Abstraktion von LaTeX	Abstraktion von BibTeX	Abstraktion von Tabellen	Echtzeit Zusammenarbeit	Geeignet für wissenschaftliche Texte	Möglichkeit eigenen Server aufzusetzen
Overleaf [37]	✓	✓	✓			✓	✓	✓
Authorea [3]	✓	✓	✓			✓	✓	
ShareLaTeX [59]	✓					✓	✓	✓
Papeeria [53]	✓					✓	✓	
Verbosus [61]	✓						✓	
BlueLaTeX [56]	✓						✓	✓
SageMathCloud [63]	✓						✓	
LateX-lab [57]	✓						✓	
Google Docs [22]	✓	✓			✓	✓		
Owncloud Documents [43]	✓	✓						✓
LyX [58]		✓	✓		✓	✓	✓	
LyX [58] in Verbindung mit Owncloud [43]		✓	✓		✓		✓	✓

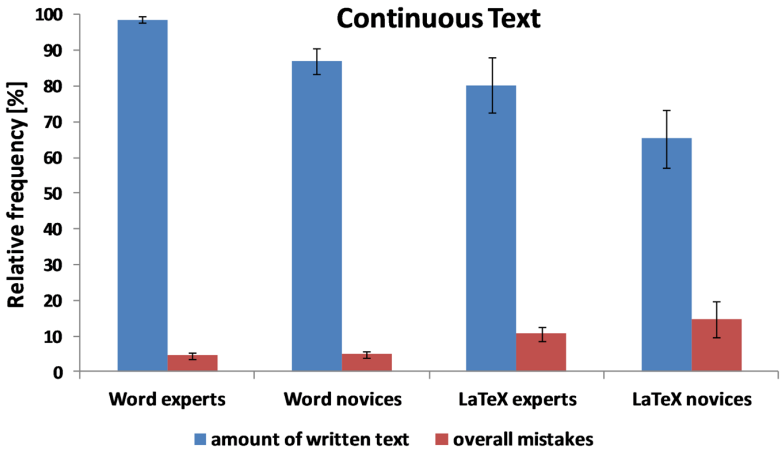
Tabelle 1.1: Marktanalyse

1.3 Motivation

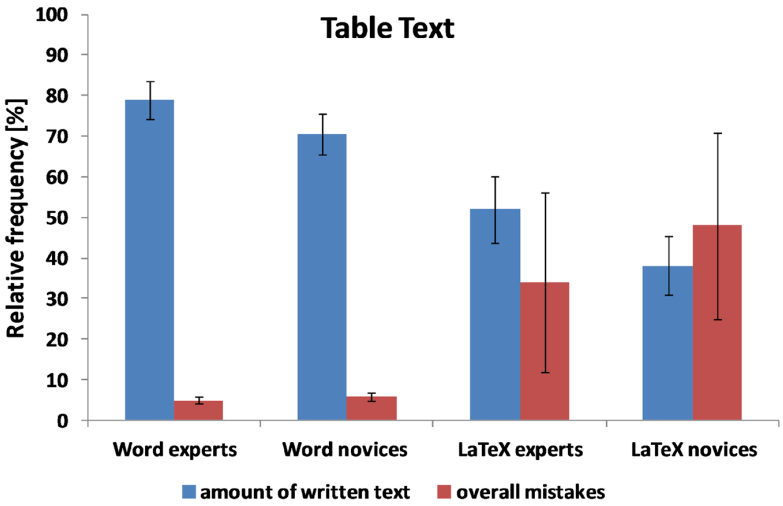
Die Studie *„An Efficiency Comparison of Document Preparation Systems Used in Academic Research and Development“* vergleicht *Microsoft Word* und LaTeX. Die Teilnehmer der Studie mussten drei verschiedene Texte innerhalb von 30 Minuten abtippen. Die Teilnehmer wurden in die Kategorien *„Word Anfänger“*, *„Word Experte“*, *„LaTeX Anfänger“* und *„LaTeX Experte“* eingeteilt. Der erste Text war ein reiner Fließtext mit Überschriften, der zweite Text war ein Fließtext mit einer komplexen Tabelle und der dritte Text war ein Fließtext mit komplexen mathematischen Formeln. [40]

Wie das Ergebnis der Studie in Abbildung 1.1 auf Seite 8 zeigt, ist es für reine Fließtexte und Fließtexte mit Tabellen effizienter *Microsoft Word* einzusetzen. Beinhaltet ein Text jedoch mehrere mathematische Formeln, ist es effizienter auf LaTeX zu setzen. Diese Studie motiviert für LaTeX ein User Interface zu schreiben, dass LaTeX auf gleiche Ebene mit *Microsoft Word* bringt, was das User Interface betrifft. Der Hauptvorteil von LaTeX, dass bei großen Arbeiten nicht auf die Formatierung geachtet werden muss und das das Layout des Endproduktes bezüglich setzen der Wörter bzw. Buchstaben, Wordtrennung und Formatierung dem von *Microsoft Word* überlegen ist [4, S.24], wird in dieser Studie nicht berücksichtigt. Warum LaTeX in dieser Hinsicht überlegen ist, wird näher im Kapitel LaTeX auf Seite 11 erläutert. Den Nachteil von LaTeX den diese Studie aufzeigt ist, dass es relativ komplex ist einfache Texte zu erstellen. Ein Nachteil der gänzlich durch dieses Projekt ausgemerzt werden wird.

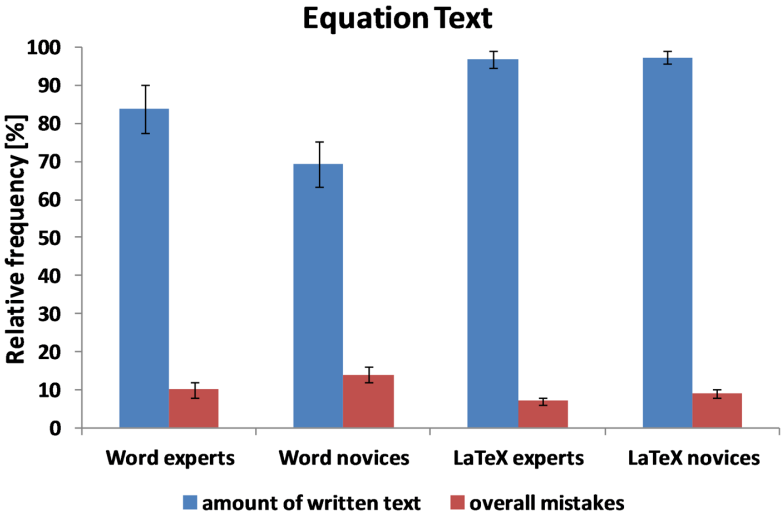
Die Aussagekraft der Studie *„An Efficiency Comparison of Document Preparation Systems Used in Academic Research and Development“* wurde jedoch durch den Artikel *„Word-processing war flares up on social media“* in Frage gestellt [64]. Gleich mehrere Wissenschaftler meldeten sich nach dem Veröffentlichen der Studie per Social Media zu Wort. Steven J. Gibbons schrieb z.B. auf Twitter: *„As a reviewer of scientific papers, I can confirm that ones prepared in LaTex are 500% better than those in Word for (cont).“* [64]. Des Weiteren schrieb Gibbons: *„Correct labelling of equations, figures, tables and including all references in the right order. LaTex wins hands down!“* [64]. Des Weiteren wird im Artikel *„Word-processing war flares up on social media“* [64] die wissenschaftliche Methodik zur Evaluierung der Systeme LaTeX und *Microsoft Word* angezweifelt. Der Autor rechtfertigt sich in einem Kommentar, dass es keine andere Möglichkeit gegeben habe die Systeme direkt zu vergleichen. Trotz der Relativierung der Aussagekraft der Studie durch Wissenschaftler ist die Aussagekraft der Studie für diese Arbeit von Bedeutung und die darin aufgezeigten Probleme werden in die Implementierung mit einfließen.



(a) Fließtext Ergebnis



(b) Fließtext mit einer komplexen Tabelle Ergebnis



(c) Fließtext mit komplexen mathematischen Formeln Ergebnis

Abbildung 1.1: LaTex - Microsoft Word Vergleich [40]

Im Artikel „*Word-processing war flares up on social media*“ [64] ist eine weitere wichtige Aussage für diese Arbeit zu lesen. Angie Pendergras sagt zur Verwendung von LaTex: „*The code required to write in LaTeX can be especially cumbersome when multiple authors are working on the same paper*“ [64]. Für Angie Pendergras, die am National Center for Atmospheric Research arbeitet ist es offensichtlich ein Problem eine Umgebung aufzusetzen, die es ermöglicht mit Hilfe einer Versionierungssoftware mehrere Personen an einer Quelldatei schreiben zu lassen. Als Informatiker muss man dies akzeptieren. Man muss akzeptieren, dass Wissenschaftler aus anderen Fachgebieten einen anderen Zugang zu Technologien haben, die für Informatiker einfach und selbstverständlich zu bedienen sind. Es sollte vielmehr ein Ziel sein, diese Technologien so zu abstrahieren, dass keine bzw. eine kurze Einlernphase nötig ist um von den Vorteilen profitieren zu können. Diese Aussage motiviert, dass User Interface kollaborativ zu gestalten. Jede Änderung eines Benutzers ist sofort für die anderen Benutzer sichtbar. Das Zusammenführen von verschiedenen Versionen entfällt.

1.4 Ziele

Aus der Einführung auf Seite 1 und der Marktanalyse auf Seite 3 ergeben sich für diese Arbeit folgende Ziele:

- LaTeX Abstraktions Web Editor
  - BibTeX Unterstützung
  - Tabellen Unterstützung
  - Export als PDF
- Echtzeit Zusammenarbeit von mehreren Benutzern
  - Möglichkeit Textteile zu diskutieren
- Firmen/Organisationen sollen die Möglichkeit haben, eine eigene Instanz des Systems auf eigener Hardware zu betreiben.

2 Stand der Technik

In diesem Kapitel wird das theoretische Hintergrundwissen für dieses Projekt aufgearbeitet. Es wird der *„Motor“* dieses Projekts, das Textsatzsystem TEX mit dem Makropaket LATEX näher betrachtet. Weiters werden die verwendeten Web Technologien zum Erstellen der Benutzeroberfläche wie z.B. *Ajax*, *WebSockets* und *WebRTC* aufgearbeitet. Der wichtigste und anspruchsvollste Punkt dieser Arbeit, dem Zusammenarbeiten von mehreren Benutzern an einem Textteil oder auch kollaboratives Arbeiten genannt und die dahinter liegenden Techniken werden ebenfalls behandelt.

2.1 LATEX

Übersetzt beschreibt sich das LATEX Projekt auf seiner Homepage selbst mit: *„LATEX ist ein hoch qualitatives Textsatzsystem. Es wurde entwickelt für die Erstellung von technischen und wissenschaftlichen Dokumentationen. LATEX ist ein de facto Standard für die Kommunikation und Publikation von wissenschaftlichen Dokumenten. LATEX ist als freie Software verfügbar.“* [35]

Bei LATEX handelt es sich um kein *„what you see is what you get“* System. Das Dokument wird in einer Quelltext Datei verfasst. Jegliche Formatierungen, Gliederung und andere Einstellungen die beim Erstellen eines Textes getroffen werden, werden mittels Befehlen in der Quelltext Datei durchgeführt [4, S.25]. Im Quellcode 2.1 auf Seite 12 wird der Inhalt einer einfachen LATEX Quelltext Datei gezeigt. Die Endung der LATEX Dateien ist *.tex*. Mit dem Programm *pdflatex* kann die Quelltext Datei in ein *PDF* umgewandelt werden. Die Abbildung 2.1 auf Seite 12 zeigt die Ausgabe die *pdflatex* in einer *PDF* Datei erzeugt.

An dieser Stelle wird darauf verzichtet näher auf die Verwendung von LATEX einzugehen. Es ist genügend Literatur zu diesem Thema vorhanden. Zu empfehlen ist das Buch von *Berndt* mit dem Titel LaTeX Der typographische Einstieg. [4]

1 Kapitel 1

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

Abbildung 2.1: Ausgabe des Quellcodes 2.1 in einem PDF.

```
\documentclass{article}
\begin{document}
\section{Kapitel 1}
Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod
    tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.
\end{document}
```

Quellcode 2.1: Einfaches LaTeX Dokument

2.1.1 Textsatzsystem vs. Textverarbeitungssystem

Um LaTeX von konkurrierender Software zur Erstellung von Texten abzugrenzen bzw. um die Vorteile hervorzuheben, müssen zuerst die Begriffe Textsatzsystem und Textverarbeitungssystem aufgearbeitet werden. Vorweg sei erwähnt, dass es sich bei LaTeX um ein Textsatzsystem und nicht um ein Textverarbeitungssystem handelt. Beim Textsatz handelt es sich um die Kunst Wörter eines Textes in eine drucktaugliche Form zu bringen. In der Praxis bedeutet dies z.B. einen Absatz so in Blocksatz zu formatieren, dass der gesamte Absatz korrekt auf den rechten Rand ausgeglichen wird. Zusätzlich sollen durch Silbentrennung keine großen Lücken zwischen den Buchstaben auf Grund des Blocksatzes entstehen. In der ursprünglichen Form des Buchdruckes war der Setzer ein eigener Beruf. Der Setzer bediente sich Bleibuchstaben um den zu druckenden Text zu erzeugen. Ein elektronisches Textsatzsystem ist eine Software Implementierung dieses Verfahrens. [4, S.21-27]

Im Gegensatz dazu ist laut Duden die Textverarbeitung: *„Verfahren zur Rationalisierung des Formulierens, Diktierens, Schreibens, Vervielfältigens o.Ä. von Texten“* [20]. Des Weiteren wird im Duden der Begriff Textverarbeitungssystem wie folgt definiert: *„Computer, mit dem in Verbindung mit geeigneter Software die Textverarbeitung elektronisch erfolgt“* [21]. Ein Textverarbeitungssystem ist also eine Software die einem dabei hilft Text zu verfassen und dies in einer möglichst benutzerfreundlichen Art und Weise. Zu Textverarbeitungssystemen bei denen

die Benutzerfreundlichkeit primäres Ziel ist sagt *Berndt* in LaTeX Der typographische Einstieg: *„Die Qualität der Ausgabe kann schon deshalb nicht im Mittelpunkt stehen, weil ein halbwegs gediegener Textsatz aufgrund zahlloser zu beachtender Aspekte einem solchen Bedienkomfort entgegensteht.“* [4, S.21-27].

Zusammenfassend ist also zu sagen, dass ein Textsatzsystem das Ziel hat sich um Layout und dem Setzen von Buchstaben bzw. Wörtern zu kümmern. Im Gegensatz dazu hat ein Textverarbeitungssystem das Ziel, dem Benutzer das Erstellen von Texten so einfach und intuitiv wie möglich zu ermöglichen. Auf Grund der verschiedenen Ziele gibt es einen Unterschied in der Ausgabequalität zwischen Textsatz- und Textverarbeitungssystemen. Dieser Unterschied wird im nächsten Kapitel Textsatz Beispiel *Microsoft Word* vs. LaTeX ersichtlich.

2.1.2 Textsatz Beispiel Microsoft Word vs. LaTeX

Personen die noch nie sonderlich auf den Textsatz geachtet haben können sich unter gutem und schlechtem Textsatz wenig vorstellen. Die Abbildung 2.2 auf Seite 14 visualisiert genau diesen Unterschied zwischen *Microsoft Word 2010* und LaTeX. Speziell in der Mitte des Absatzes sind die Unterschiede zu erkennen. Bei dieser Gegenüberstellung ist zu erwähnen, dass *Microsoft Word 2010* hier in der Standardkonfiguration verwendet wird.

Silbentrennung

Bei der Silbentrennung werden automatisch die Wörter nach Silben getrennt, um ein Ausrichten des Textes in Blocksatz ohne große Abstände zwischen den Wörtern zu ermöglichen. *Microsoft Word 2010* hat standardmäßig keine Silbentrennung aktiviert. Dies ist der Grund für die großen Abstände zwischen den Wörtern in der Abbildung 2.2 auf Seite 14. Bei einem anderem Versuch wurde *Microsoft Word 2008 (Mac)* mit Silbentrennung und LaTeX auf gleiche Art und Weise wie in Abbildung 2.2 verglichen. Die Auswertung hat ergeben, dass für einen Text mit 202 Wörtern LaTeX 4 Silbentrennungen und *Microsoft Word 2008* 9 Silbentrennungen durchführt. LaTeX ist also auch bei aktivierter Silbentrennung *Microsoft Word* beim Textsatz überlegen [46].

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Fischers Fritz fischt frische Fische mit dem Affen, der über den See fliegt. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Fischers Fritz fischt frische Fische mit dem Affen, der über den See fliegt. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Abbildung 2.2: Vergleich des Textsatzes von *Microsoft Word 2010* in Standardkonfiguration (linke Spalte) und LaTeX (rechte Spalte) [25]

Abbildung 2.3: Vergleich von Kerning mit *Microsoft Word 2008* in Standardkonfiguration (oben) und LaTeX (unten) [46]

Ligaturen

Ein weiterer Unterschied sind Ligaturen die *Microsoft Word* erst seit der Version 2010 beherrscht [25]. Dabei werden gewisse Buchstabenkombinationen näher aneinander gesetzt. Ligaturen sollen die Lesbarkeit eines Textes erhöhen [52, S.27]. *Microsoft Word 2010* hat sowie die Silbentrennung auch die Ligaturen nicht in der Standardkonfiguration aktiviert. Es folgen ein paar Beispiele zu Ligaturen. In der ersten Zeile werden Ligaturen verwendet und in der zweiten Zeile werden diese unterdrückt.

Mit Ligaturen: Wasserflasche, Sportfischen, abflauen, erfinden, höflich
Ohne Ligaturen: Wasserflasche, Sportfischen, abflauen, erfinden, höflich

Die Ligaturen im Beispiel wurden mit LaTeX erzeugt. Hier zeigt sich, dass LaTeX nicht perfekt arbeitet. Laut *Solbrig* in *„Zweisprachige Mikrotypografie“* werden im Deutschen Ligaturen nicht gesetzt wenn sie zwischen Wortstamm und Endung bzw. zwischen Wortfugen zweier miteinander verbundener Wörter stehen. Demnach sind die Buchstaben *fl* im Beispiel für das Wort *höflich* falsch gesetzt. Im Englischen werden Ligaturen immer gesetzt, egal an welcher Position sie sich im Wort befinden. Hier arbeitet LaTeX korrekt [52, S.27].

Kerning

Ein weiteres Feature, dass LaTeX unterstützt und bei *Microsoft Word* erst aktiviert werden muss ist Kerning. Dabei werden bestimmte Buchstabenkombinationen die es erlauben ineinander geschoben. In der Abbildung 2.3 auf Seite 15 wird Kerning anhand eines Vergleiches von *Microsoft Word 2008* in Standardkonfiguration und LaTeX gezeigt.

Zusammenfassend zum Kapitel Textsatz Beispiel *Microsoft Word* vs. LaTeX ist zu sagen, dass *Microsoft Word* sehr wohl auf die Grundsätze des Textsatzes eingeht.

Grundsätze wie Silbentrennung, Ligaturen und Kerning müssen jedoch erst aktiviert werden. Im Kapitel Textsatzsystem vs. Textverarbeitungssystem wird erwähnt, dass ein Textverarbeitungssystem wie *Microsoft Word* benutzerfreundlicher zu bedienen ist. *Microsoft Word* ist aber nur bezüglich der Verfassung des Textes benutzerfreundlicher zu bedienen, bezüglich der Einstellungen für Textsatz ist LaTeX klar überlegen. Bei LaTeX muss sich der Benutzer um keinerlei Einstellungen bezüglich des Textsatzes kümmern.

2.1.3 TeX, LaTeX, LaTeX Distributionen

In dieser gesamten Arbeit wird LaTeX als Textsatzssytem genannt. Genau genommen ist LaTeX jedoch nur ein Makropaket für TeX. TeX ist das eigentliche Textsatzsystem das zusätzlich Erweiterungen in Form vom Makros zulässt. Der Name TeX steht für die Textsatz Engine aber auch für die Makro Sprache die TeX definiert. Die durch LaTeX implementierten Makros für TeX ermöglichen eine einfache Verwendung von TeX. Die vorhandenen Makros sind auf die Bedürfnisse der Benutzer zugeschnitten wie z.B. das Erstellen von Überschriften, Inhaltsverzeichnis, Tabellenverzeichnis, Abbildungsverzeichnis, Titelblatt und vieles mehr. LaTeX bietet auch die Möglichkeit zusätzliche Pakete einzubinden. Dadurch ist es flexibel für Erweiterungen für Anwendungen aller Art. [4, S.28-34]

Es gibt eigene LaTeX Distributionen die eine schnelle Installation von TeX, LaTeX und häufig verwendeten Zusatzpaketen ermöglicht. Als Beispiel für das Betriebssystem Linux ist *TeX Live* [24] zu nennen. Eine beliebte Distribution für Windows ist *MiKTeX* [49].

2.1.4 Programme und Ausgabeformate

Wie im Kapitel LaTeX auf Seite 11 beschrieben, erzeugt man zuerst eine *.tex* Quelltext Datei die im Anschluss in das gewünschte Ausgabeformat kompiliert wird. Mit dem Programm *latex* kann die *.tex* in eine *DVI* Datei umgewandelt werden. [4, S.28-34] *DVI* steht für *Device independent file format* und soll wie der Name schon verrät eine Basis bilden um diese Datei für verschiedenste Geräte zu kompilieren. Ein Gerät kann sowohl ein Drucker sein aber auch eine Software wie z.B. ein PDF Viewer. [34, S.407] Als Gerätetreiber bezeichnet man jene Software die die *DVI* Datei in eine für das Gerät verständliche Datei umwandelt. Der Grund für diese Aufsplittung in *DVI* Datei und anschließender Umwandlung mit dem Gerätetreiber ist, dass es keinen Standard für Weitergabe von Daten an ein Ausgabegerät gibt. [5, S.318-320] Beispiele für Gerätetreiber

sind *dvipdf* [2] für die Umwandlung in eine *PDF* Datei oder *dvips* [47] für die Umwandlung in ein *PostScript* Datei. Sollte direkt eine PDF Datei benötigt werden, gibt es die Möglichkeit mit dem Programm *pdflatex* die *.tex* Quelltext Datei direkt in ein *PDF* umzuwandeln. [60]

2.2 Kollaboration

In diesem Kapitel werden Techniken behandelt, die es ermöglichen gleichzeitig an einem Text zu arbeiten. Der Text liegt dabei auf einem Server und mehrere Clients können den Text zeitgleich öffnen und editieren. Änderungen eines Clients werden in Echtzeit an die anderen Clients übermittelt. Techniken die keine Echtzeit Zusammenarbeit garantieren werden hier nicht behandelt. Ein Beispiel hierfür ist *Locking*. Beim *Locking* wird der Text oder Textteile gesperrt, sobald ein Benutzer diese editiert um Konflikte zu vermeiden. Techniken die nicht benutzerfreundlich sind werden ebenso nicht behandelt wie z.B. *Dependency-detection* [55]. Bei der *Dependency-detection* werden alle Änderungen von verschiedenen Benutzern mit einem Zeitstempel versehen. Solange die Änderungen ohne Konflikt in den Text aller Benutzer eingepflegt werden funktioniert das System. Sobald sich ein Konflikt ergibt, muss dieser manuell von den Benutzern behoben werden. Wenig Benutzerfreundlich ist auch die Methode *Reversible Execution* [48]. Dabei bekommt jede Änderung einen Zeitstempel oder eine fortlaufende Nummer zugewiesen von einer zentralen Stelle. Die Änderungen werden in einer Historie gespeichert. Ergibt sich ein Konflikt, können über die Historie die Konflikt auslösenden Änderungen rückgängig gemacht werden. Im Anschluss wird die einzuspielende Änderung durchgeführt und die zuvor rückgängig gemachten Änderungen erneut ausgeführt. Die erneut ausgeführten Änderungen werden basierend auf der einzuspielenden Änderung modifiziert, damit sich ein konsistenter Zustand ergibt. *Reversible Execution* ist nicht benutzerfreundlich, da das Rückgängig machen von Änderungen vor den Augen des Benutzers geschieht und für Verwirrung sorgt. [13]

2.2.1 Operational Transformation

Operational Transformation besteht aus zwei Komponenten. Einerseits die Operation und andererseits der Algorithmus zur Integration von Operationen. Operationen sind Funktionen die Änderungen an den Daten beschreiben. Im folgenden wird *Operational Transformation* angewendet auf Plaintext beschrieben. Für einen Plaintext ergeben sich die zwei Operationen *insert* und *delete*. *Insert* bekommt als Parameter eine Position und den Character welcher an dieser Position hinzuzufügen ist. *Delete* bekommt nur die Position des zu löschenden Character als Parameter. Dadurch ergeben sich folgende Funktionsdefinitionen und Menge von Operationen:

$$insert(i, c); i = index, c = character \tag{2.1}$$

$$delete(i); i = index \tag{2.2}$$

$$O = \{insert, delete\} \tag{2.3}$$

Generiert ein Benutzer eine Operation am Text, wird diese zuerst am eigenen Text ausgeführt und im Anschluss an den Server und weitere Clients gesendet. Dadurch ergibt sich folgendes Problem:

- Text Benutzer A:
  Opperational Transormation
- Operation Benutzer A:
  delete(1)
- Text Benutzer A nach Ausführung der eigenen Operation:
  Operational Transormation
- Text Benutzer A nach Ausführung der Operation von Benutzer B:
  Operational Transofrmation

- Text Benutzer B:
  Opperational Transormation
- Operation Benutzer B:
  insert(17, f)
- Text Benutzer B nach Ausführung der eigenen Operation:
  Opperational Transformation
- Text Benutzer B nach Ausführung der Operation von Benutzer B:
  Operational Transformation

Das Ergebnis von Benutzer A ist ungleich von Benutzer B (**Operational Transformation** $\neq$ **Operational Transformation**). Die Lösung für dieses Problem ist die Transformation von Operationen. Werden zwei Operationen auf die selbe Version des Textes ausgeführt, werden die Operationen so transformiert, dass egal in welcher Reihenfolge sie ausgeführt werden sich ein konsistenter Zustand des Textes ergibt. Für die Operationen *insert* und *delte* ergeben sich dadurch folgende Transformationen:

Definition von i_{o_j}: i = index für die Operation, o = beliebige Operation aus O, j = Ausführungsreihenfolge der Operationen.

if $i_{o_1} < i_{o_2}$ **then**
  if $o_1 == insert$ **then**
    $i_{o_2} = i_{o_2} + 1$
  end if
  if $o_1 == delete$ **then**

```
            i_{o_2} = i_{o_2} − 1
        end if
    else if i_{o_1} > i_{o_2} then
        if o_2 == insert then
            i_{o_1} = i_{o_1} + 1
        end if
        if o_2 == delete then
            i_{o_1} = i_{o_1} − 1
        end if
    else if i_{o_1} == i_{o_2} then
        if o_1 == delete && o_2 == delete then
            Keine Operation ausführen
        end if
        if o_1 == insert && o_2 == insert then
            i_{o_2} = i_{o_2} + 1
        end if
        if o_1 == insert && o_2 == delete then
            i_{o_2} = i_{o_2} + 1
        end if
    end if
```

Bekommt der Server also eine Operation übermittelt, transformiert er diese Operation gegen alle vorhandenen Operationen die aktueller sind. Danach kann die erhaltene Operation am Server ausgeführt werden und im Anschluss an alle Clients übermittelt werden. Der Client der die Operation initiieren hat, hat nun die Bestätigung, dass seine Operation ausgeführt wurde. Er befindet sich wieder in einem synchronisierten Zustand. Auch der Client muss Operationen transformieren. Erhält er eine Operation und hat gerade eine eigene Operation an den Server gesendet, muss er die erhaltene Operation gegen die gerade gesendete Operation transformieren. [13]

2.2.2 Differential Synchronization

Differential Synchronization ist ein von Neil Fraser entwickelter Algorithmus um Texte über Netzwerke zu synchronisieren die kein Quality of Service garantieren wie z.B. das Internet. Grundsätzlich werden nur Änderungen (Diffs) vom aktuellen Text zum vorherigen Text übertragen. Der Algorithmus ist symmetrisch, das bedeutet, dass Server und Client den gleichen Code ausführen. Wird der Algorithmus für mehrere Clients angewandt, unterscheidet sich der Server Code geringfügig vom Client Code.

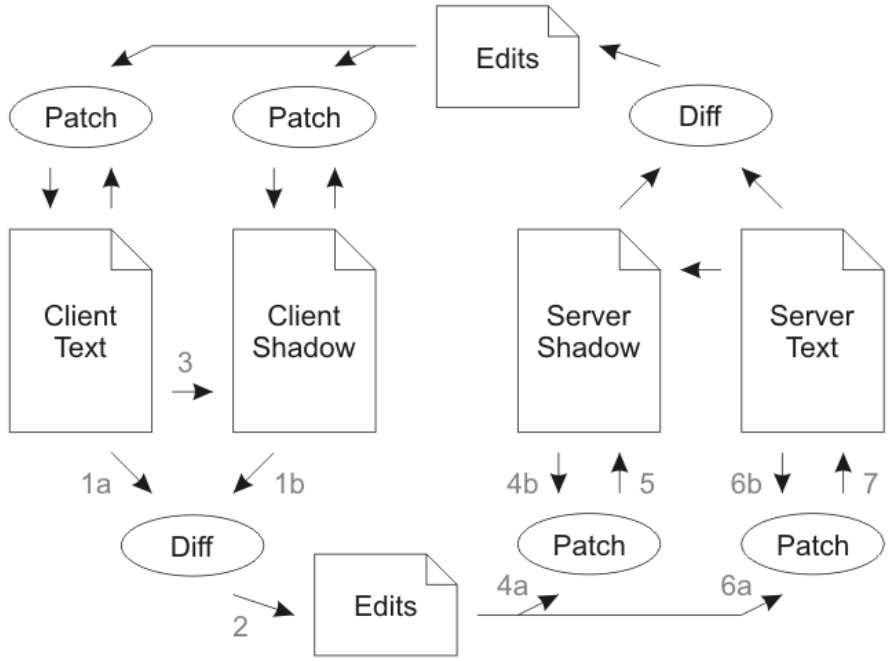


Abbildung 2.4: Daten-Flussdiagramm von Differential Synchronization [18]

Beim Differential Synchronization werden vom zu ändernden Server Text 3 Kopien erzeugt. Server Text, Server Shadow, Client Text und Client Shadow sind die Daten die nach der Initialisierung am Server bzw. am Client in identischem Zustand bereit stehen müssen. Änderungen des Benutzers werden am Client Text durchgeführt. Die folgende Erklärung des Algorithmus wird anhand der Abbildung 2.4 auf Seite 21 durchgeführt. Der Algorithmus wird nach einer beliebigen Änderung durch den Benutzer am Client Text asynchron angestoßen. Als erstes wird ein Diff zwischen Client Text und Client Shadow erzeugt (1a, 1b und 2). Nach dem Erzeugen des Diffs wird der Client Text auf die Client Shadow abgeglichen (3). Das erzeugte Diff wird im Anschluss an den Server gesendet. Das Diff wird am Server zuerst in die Server Shadow (4a, 4b und 5) und im Anschluss in den Server Text gepacht (6a, 6b und 7). Nun beginnt der gleiche Zyklus ausgehend vom Server. Es wird wieder zuerst ein Diff zwischen Server Text und Server Shadow erzeugt. Haben sich inzwischen Änderungen am Server durch einen anderen Client ergeben, sind diese in diesem Diff enthalten und werden nach dem gleichem Schema an den Client übergeben. Dieser Zyklus wird solange durchgeführt, bis sich keine Änderungen mehr für den Client bzw. für den Server ergeben. Der Algorithmus Differential Synchronization kann mit beliebigen Daten durchgeführt werden, die mittels eines Diff und Patch Algorithmus bearbeitet werden können. [18]

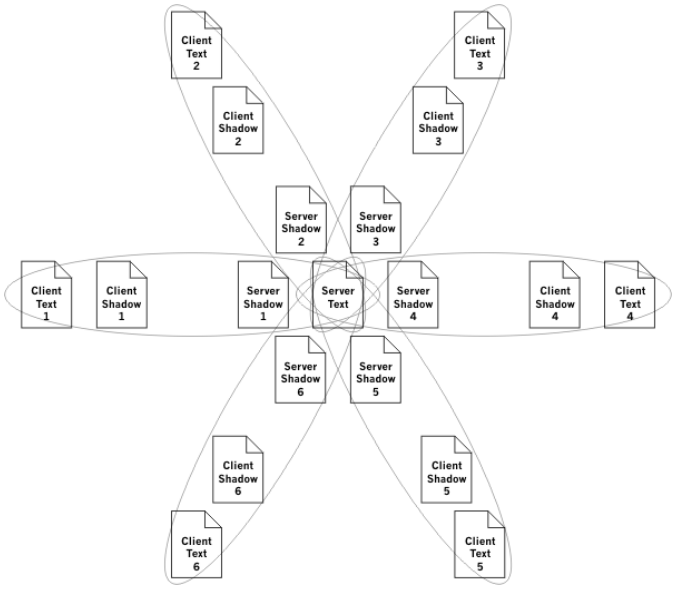


Abbildung 2.5: Konfiguration einer Differential Synchronization Umgebung mit mehreren Clients[18]

Die vorherige Erklärung war für einen Server mit einem Client. Verbinden sich mehrere Clients mit dem Server, benötigt jeder einzelne Client eine eigene Server Shadow am Server. Die Abbildung 2.5 auf Seite 22 visualisiert die Konfiguration mit mehreren Clients. Wird der Server Text durch einen Client geändert, bekommen die anderen Clients diese Änderung bei der nächsten Synchronisation mitgeteilt. [18]

Zusammenfassend zum Kapitel Kollaboration ist zu sagen, dass Differential Synchronization besser geeignet ist für kollaborative Web-Anwendungen. Der Differential Synchronization Algorithmus muss nur angestoßen werden, sobald sich eine beliebige Änderung in der Client Version ergibt. Im Anschluss wird das Diff zur vorherigen Version erzeugt. Wie diese Änderung zustande gekommen ist, ist egal. Es gibt somit einen zentralen Punkt an dem der Algorithmus ansetzt. Beim Operational Transformation müssen alle Benutzereingaben erfasst werden und in Operationen umgewandelt werden. Benutzereingaben können hierbei unter anderem sein: Kopieren und Einfügen, Autokorrektur, Drag and Drop oder Rückgängig. All diese Benutzereingaben abzufangen und in Operationen für *Operational Transformation* umzuwandeln stellt einen erheblichen Aufwand dar. Dieser erhebliche Aufwand ergibt sich im speziellen bei Web-Anwendungen. Hier muss zusätzlich die Kompatibilität mit allen gängigen Browsern berücksichtigt werden. [18]

2.3 Kommunikation

Arbeiten mehrere Personen am gleichen Standort an einem Dokument wird es eine Person geben, die die diskutierten Inhalte niederschreibt und andere Personen werden sich an der Diskussion über die zu verfassenden Inhalte beteiligen. Somit ergeben sich zwei Kommunikationsmöglichkeiten. Erstens kommuniziert die schreibende Person mit z.B. einem Textverarbeitungssystem und zweitens kommunizieren alle beteiligten Personen untereinander. Genau aus diesem Beispiel aus der Realität leiten sich die benötigten Kommunikationskanäle für eine Software ab, die dieses Beispiel virtualisieren soll. Erstens wird eine Kommunikation zwischen dem Dokument und den Verfassern benötigt. Auf Grund dessen, dass sich das Dokument zentral auf einem Server befindet wird dies im Kapitel Client ↔ Server Kommunikation auf Seite 23 behandelt. Zweitens wird eine Kommunikation zwischen den Verfassern benötigt, die eine Diskussion der Verfasser am gleichen Standort ersetzt. Hier ist an Sprach-, Video- und Textkommunikation zu denken und wird im Kapitel Client ↔ Client Kommunikation auf Seite 25 behandelt.

2.3.1 Client ↔ Server Kommunikation

Wie aus dem Kapitel Kollaboration ersichtlich, benötigen kollaborative Anwendungen eine bidirektionale Echtzeit-Verbindung zwischen dem Client und dem Server. In der Einführung wurde bereits begründet, dass es sich bei der Implementierung dieser Arbeit um eine Web Anwendung handeln wird. In der Spezifikation des HTTP/1.1 Protokolls Version wird festgelegt, dass der Client eine Ressource vom Webserver beantragt und der Webserver diese anschließend ausliefert. Es ist nicht möglich mit dem HTTP/1.1 Protokoll Änderungen am Server dem Client mitzuteilen. [16]
Auf Grund dessen werden in diesem Kapitel ausschließlich bidirektionale Client ↔ Server Echtzeit-Kommunikationsmöglichkeiten zwischen Webserver und Webclient behandelt.

Ajax Polling

Mit Hilfe der Einführung des *XMLHttpRequest* wurde die Basis geschaffen echtzeitfähige Webseiten zu implementieren. In der Spezifikation wird übersetzt geschrieben: *„Der Name des Objekts ist XMLHttpRequest (...) jede Komponente dieses Namens ist irreführend.“* [33]. Der erste Teil des Namens ist falsch, da nicht nur

XML sondern beliebige Nachrichten übertragen werden können. *Http* ist falsch, da es möglich ist über *http* und über *https* Daten zu übertragen. Request ist falsch, da es nicht nur den Request sondern den gesamten Protokollablauf des HTTP Protokolls spezifiziert. [33] Mit dem *XMLHttpRequest* ist es möglich per JavaScript einen Request an den Server abzusetzen und im Anschluss die erhaltenen Daten per JavaScript zu verarbeiten. Das Absetzen des Requests erfolgt asynchron. Dadurch ist der restliche Programmablauf der Webseite nicht gestört. [23, S.25-26]
Beim Polling wird nun der *XMLHttpRequest* genutzt um in regelmäßigen Abständen am Server nachzufragen, ob neue Daten für diesen Client vorhanden sind. Das bedeutet, dass hier nur Quasi-Echtzeitfähigkeit erreicht wird. Wie gut die Annäherung an Echtzeit ist, ist Abhängig vom Intervall der Nachfragen auf den Server. Das Intervall ist wiederum Abhängig von der Leistungsfähigkeit des Servers bzw. von der Anzahl der Benutzer. [23, S.27-18]
Eine Einsatzmöglichkeit für Ajax Polling ist, wenn die Aktualisierungsintervalle des Servers bekannt sind. Dies kann z.B. ein Sensor sein, der in einem bestimmten Intervall Daten aufnimmt. [44]

Ajax Long Polling

Ein HTTP Request sowie ein *XMLHttpRequest* öffnen eine TCP Verbindung um die Daten zu übertragen. Nach dem Übertragen der Daten wird diese Verbindung wieder geschlossen. Beim Ajax Long Polling wird die TCP Verbindung am Server solange offen gehalten, bis Daten für den Client vorliegen. Sollte das Timeout der TCP Verbindung erreicht sein, wird diese geschlossen. Erhält der Client Daten bzw. wird eine Verbindung auf Grund des Timeouts geschlossen, eröffnet der Client sofort wieder eine neue Verbindung, um neue Daten zu empfangen. Durch dieses Verfahren wird der Overhead durch andauernde Anfragen an den Server verringert und nahezu Echtzeit Fähigkeit erreicht.[23, S.28-29]

WebSockets

WebSockets ist die erste Technologie die spezifiziert wurde um eine bidirektionale Verbindung zwischen Client und Webserver aufzubauen. Es gibt zwei Standards, die von der Technologie *WebSockets* verwendet werden. Dies ist einerseits die Spezifikation des *WebSocket* Protokolls [15] und andererseits die Spezifikation der Websocket Client API [26] für die Verwendung von *WebSockets* mit *JavaScript*. Beim Ajax Polling und beim Ajax Long Polling wird für jede neue Anfrage

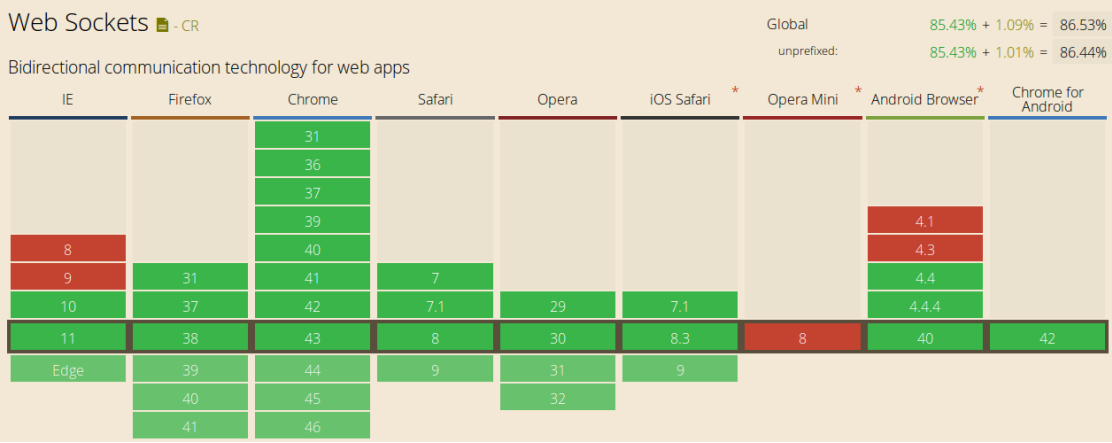


Abbildung 2.6: Browser Kompatibilität von *WebSockets* [9]

ob neue Daten am Server vorhanden sind eine neue TCP Verbindung geöffnet. Beim *WebSocket* Protokoll wird für ein Websocket eine TCP Verbindung geöffnet und diese offen gelassen. Des Weiteren hat jeder *„poll“* beim Ajax Polling und beim Ajax Long Polling den Overhead des HTTP Headers. Das *WebSocket* Protokoll bietet einen niedrigeren Overhead bei der Datenübertragung. [15] Eine Applikation, welche alle zwei Sekunden Daten am Client aktualisiert, hat mit der Realisierung von Websockets eine drei Mal bessere Antwortzeit als mit Ajax Long Polling. Der Protokolloverhead ist sogar 500mal kleiner. [14] Ein Test einer mit einer 4Hz Datenrate eines Sensors der seine Daten über das Internet an Clients verteilt auf der ganzen Welt sendet hat ergeben, dass Websockets im Schnitt bessere Antwortzeiten liefert als Ajax Polling und Ajax Long Polling. [44] *WebSockets* sind heutzutage mit allen aktuellen Browsern kompatibel wie in Abbildung 2.6 auf Seite 25 ersichtlich. [9]

Zusammenfassend zum Kapitel Client ↔ Server Kommunikation ist zu sagen, dass betreffend dem heutigen Stand der Technik für Echtzeit Applikationen *WebSockets* eingesetzt werden sollten. Dies kann natürlich nur realisiert werden, wenn man bezüglich der Kompatibilität zu alten Browsern Einschränkungen in Kauf nehmen kann.

2.3.2 Client ↔ Client Kommunikation

In diesem Kapitel wird behandelt, welche Technologie verwendet werden kann um eine Diskussion von zwei oder mehreren Personen am gleichen Standort zu virtualisiert. Würden alle Daten über den Server gesendet werden, wäre

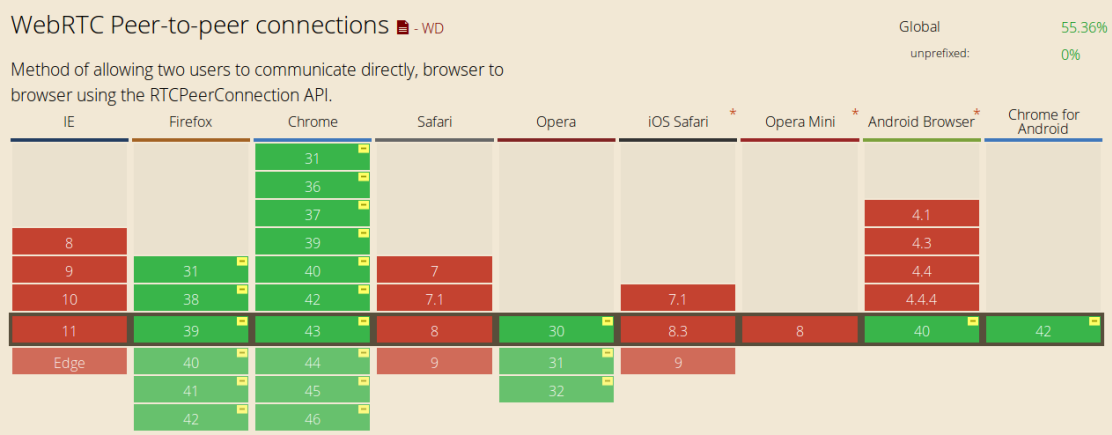


Abbildung 2.7: Browser Kompatibilität von *WebRTC* [10]

die benötigte Bandbreite am Server direkt proportional zur Benutzeranzahl. Ein besserer Ansatz ist, dass alle Clients direkt miteinander kommunizieren. Dies wiederspricht jedoch dem Grundkonzept von Web Applikationen in dem immer ein Client mit einem Server kommuniziert.

WebRTC

Die Abkürzung *WebRTC* steht für *Web Real-Time Communication* und ist eine *JavaScript API* und Standard der es erlaubt zwischen Browsern aber auch mobilen Geräten eine Peer-to-Peer Verbindung zu eröffnen. [28] *WebRTC* wird derzeit von den Browsern Chrome, Firefox und Opera unterstützt. Global sind somit über 50% der Geräte kompatibel mit diesem Standard. [10] Die Abbildung 2.7 auf Seite 26 zeigt die genaue Kompatibilität zu den aktuellen Browserversionen.

Über die Peer-to-Peer Verbindung können Text basierte Daten, Binärdaten, Video- bzw. Audiostreams oder der Bildschirm (Screensharing) übertragen werden. [39] Der automatische Verbindungsaufbau zwischen zwei Peers ist jedoch nicht ohne Server realisierbar. Beim sogenannten *Signaling* arbeitet ein Server als Vermittler um den Verbindungsaufbau zwischen den Peers zu ermöglichen. Um eine Verbindung über den Signaling Server aufzubauen müssen sich die Clients zuerst mit einer eindeutigen ID mit dem Signaling Server verbinden. Dies kann z.B. eine eindeutige URL am Server oder eine URL mit eindeutigem Parameter sein. Der Server kann über die ID die beiden Verbindungspartner zuordnen und nun z.B. über *WebSockets* die Signaling Nachrichten austauschen. Die Abbildung

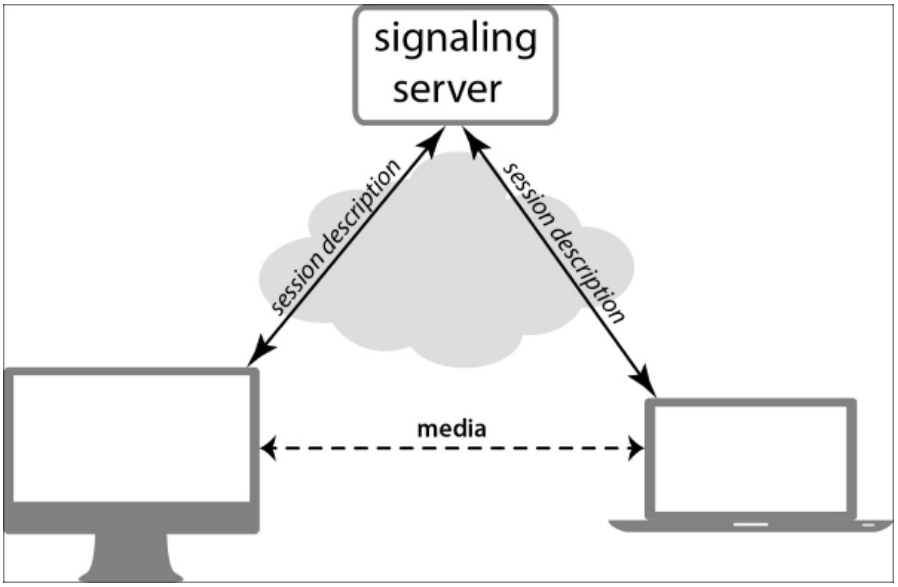


Abbildung 2.8: *WebRTC* Signaling [51]

2.8 auf Seite 27 stellt das Signaling Verfahren mit seinen Teilnehmer schematisch dar. Das Signaling ist nicht standardisiert. [39]

Steht der Kanal über den Signaling Server, muss jeder Peer bekannt geben wie er mit dem Internet verbunden ist. Wahrscheinlich ist es, dass jeder Peer sich in einem privaten Netzwerk befindet und über einen Router mit dem Internet verbunden ist. Damit die Geräte mit den privaten Adressen mit Geräten im Internet kommunizieren können, wird der Router *NAT* durchführen. Es kann sich zusätzlich auch noch eine Firewall zwischen dem Peer und dem Internet befinden. Um herauszufinden wie ein Peer mit dem Internet verbunden ist, kommt ein sogenannter *STUN* Server zum Einsatz. *STUN* steht für *„Session Traversal Utilities for NAT"*. Der Client verbindet sich mit dem STUN Server und der STUN Server liefert als Antwort genaue Informationen darüber wie der Client mit dem Internet verbunden ist. Diese Information wird *Session description* genannt. Die *Session description* wird im Anschluss über den Signaling Kanal an den jeweils anderen Peer gesendet. [39] Es gibt mehrere STUN Server die frei und öffentlich verfügbar sind. Muss z.B. die Antwortzeit eines STUN Servers garantiert werden, gibt es auch die Möglichkeit einen STUN Server selbst zu betreiben. Eine Server Implementierung des STUN Protokoll ist z.B. *STUNTMAN* [50].

Sollte es auf Grund der Netzwerkkonfiguration der einzelnen Peers nicht möglich sein die Peers direkt miteinander zu verbinden, kommt der *TURN*

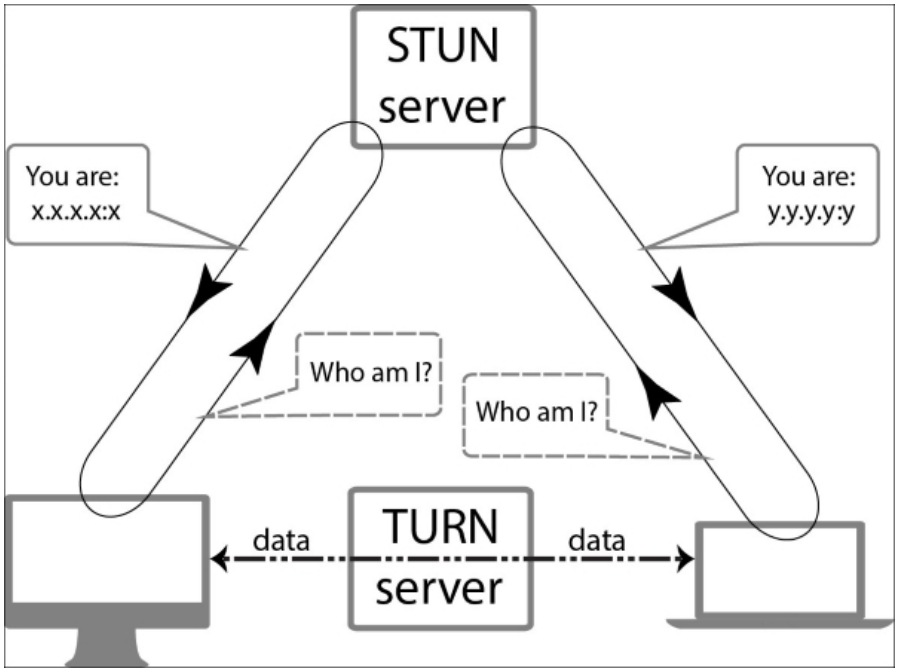


Abbildung 2.9: *WebRTC* STUN und TURN Server[51]

Server zum Einsatz. Dies ist ein Fallback Variante. Dabei werden die Daten nicht Peer-to-Peer sondern über den TURN Server versendet. In der Abbildung 2.9 auf Seite 28 wird das Zusammenspiel zwischen Peer und STUN Server bzw. falls benötigt das Zusammenspiel von Peers und TURN Server schematisch dargestellt. [39]

Über die *Media Stream API* kann per *JavaScript* auf Eingabegeräte des Rechners wie z.B. Webcam oder Mikrofon zugegriffen werden. Diese können dann über die *RTCPeerConnection* an andere Peers gesendet werden oder auch über das HTML5 Element <*video*> bzw. <*audio*> direkt ausgegeben werden. Die Codierung der Streams wird von der API durchgeführt. Für das Übermitteln von Text bzw. Binärdaten, steht die Klasse *RTCDataChannel* zur Verfügung. [39]

Es sind auch *JavaScript* Frameworks verfügbar, welche die *WebRTC* API abstrahieren. Ein Beispiel hierfür ist *SimpleWebRTC*. Die Verwendung eines solchen Frameworks ist von Vorteil, da hier oft benötigte Problemstellungen schnell und einfach realisierbar sind. Ein Beispiel wäre ein einfacher Video-Chat oder eine textbasierte Datenübertragung. [31]

Zusammenfassend zum Kapitel Client ↔ Client Kommunikation ist zu sagen, dass *WebRTC* noch nicht produktiv bei Produkten für die breite Masse eingesetzt werden kann, da nur ca. 50% aller installierten Browser mit dieser Technologie

kompatibel sind. Für die Entwicklung von neuen Anwendungen sollte diese Technologie jedoch in Betracht gezogen werden, da sie neue Möglichkeiten bezüglich der Skalierbarkeit von Applikationen bietet. Des Weiteren ist es möglich bei ganz speziellen Nieschenprodukten wo der Benutzer mehr oder weniger von dieser Anwendung abhängig ist auf *WebRTC* zu setzen. Die Benutzer müssen einfach informiert werden, dass sie auf die neueste Version von Chrome, Firefox oder Opera umsteigen. Die Einschränkung der Benutzerfreundlichkeit muss von Produkt zu Produkt neu beurteilt werden. Es bleibt die Frage offen, wann Microsoft mit dem Internet Explorer den Standard umsetzt. Dies wäre wohl der Durchbruch für *WebRTC*.

3 Problemstellung

Im folgenden Kapitel wird beschrieben, welche Ziele die einzelnen Module dieser Arbeit haben. Grundsätzlich gliedert sich die Arbeit in die drei Module Dokument Modul, Kommunikations Modul und Vorschau Modul. Das Projekt wurde in diese Module eingeteilt, da jedes Modul einzeln und eigenständig auch in anderen Software Projekten zum Einsatz kommen kann. Jedes Modul ist somit in sich abgeschlossen. Das Projekt wurde nach dem Modularitätsprinzip geteilt [62, S.145].

3.1 Dokument Modul

Das Dokument Modul ist der Kern dieser Arbeit und ermöglicht das kollaborative Zusammenarbeiten an einem Dokument. Jede Änderung im Dokument Modul wird bei allen anderen Clients, welche das selbe Dokument geöffnet haben in Echtzeit ersichtlich.

3.1.1 Editor

Im Editor können die einzelnen Dokument Komponenten und der Text erzeugt werden. Zu den einzelnen Dokument Komponenten zählen unter anderem:

- Kapitel
- Überschrift
- Unter Überschrift
- Unter unter Überschrift
- Formatierbarer Absatz
- Manueller Seitenumruch

Im formatierbaren Absatz können Wörter fett, kursiv und unterstrichen gestaltet werden. Es ist möglich nummerierte und unnummerierte Aufzählungen zu

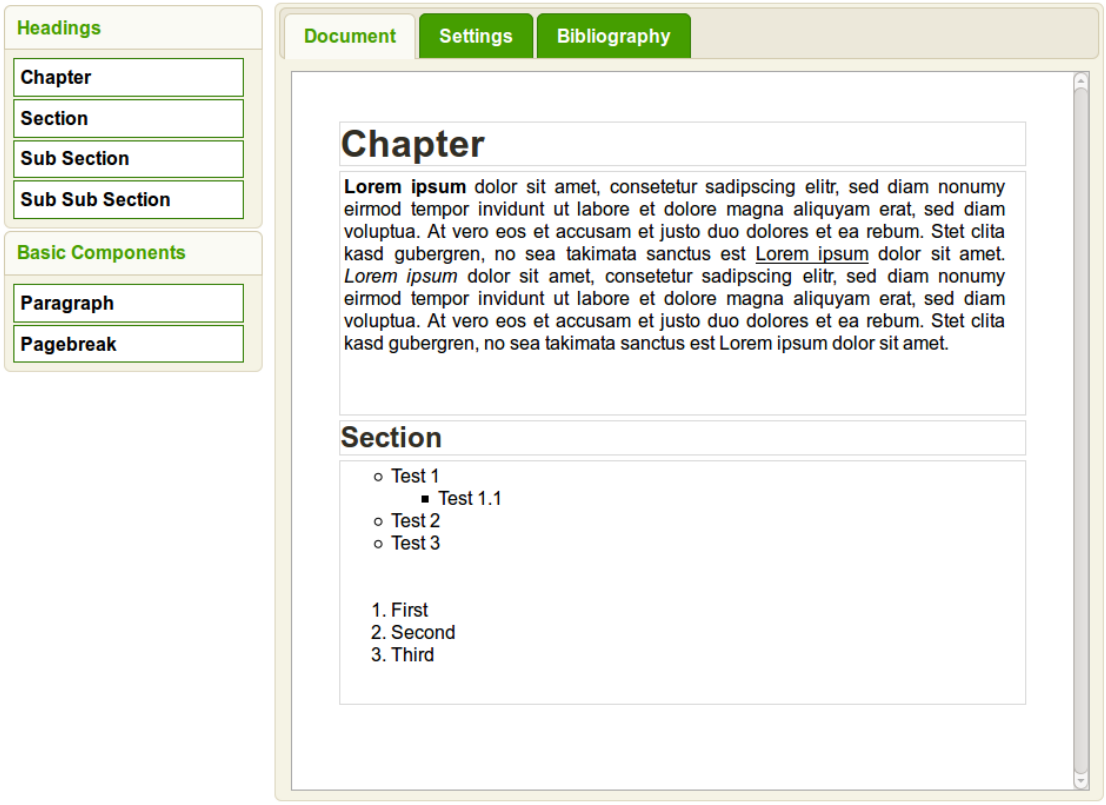


Abbildung 3.1: Benutzeroberfläche des Editors

erstellen. Der Editor wird unterteilt in eine Komponenten Box und in das Dokument selbst. Um dem Dokument neue Komponenten hinzuzufügen müssen diese per Drag and Drop von der Komponenten Box in das Dokument gezogen werden. Komponenten im Dokument können über einen Button direkt bei der Komponente gelöscht werden. Die Komponenten im Dokument besitzen eine Reihenfolge. Durch Drag and Drop kann die Reihenfolge der Komponenten im Dokument verändert werden. Die Abbildung 3.1 auf Seite 32 zeigt die Benutzeroberfläche des Editors mit der Komponenten Box auf der linken Seite und dem Dokument auf der rechten Seite.

3.1.2 Dokument Einstellungen

In den Einstellungen können die Metadaten des Dokuments bearbeitet werden. Zu den Metadaten zählen unter anderem:

- Dokumenttitel

- Autor
- Erstelldatum des Dokuments
- Anzeigen des Inhaltsverzeichnisses - Ja / Nein
- Anzeigen einer Titelseite - Ja / Nein

Die Abbildung 3.2 auf Seite 34 zeigt die Benutzeroberfläche der Dokument Einstellungen. Jede Änderung in den Einstellungen wird sofort bei allen anderen Clients, welche das selbe Dokument geöffnet haben ersichtlich.

3.1.3 Referenzen

Es ist möglich kollaborativ Referenzen zu erstellen, bearbeiten und zu löschen. Die Arten von Referenzen basieren auf den Möglichkeiten von BibLaTeX [36]. Nach dem Erstellen der Referenzen können diese im Editor verwendet werden.

3.1.4 Kommentare

Über Kommentare können Diskussionen zu einzelnen Komponenten gestartet werden. Kommentare können direkt über eine Komponente erzeugt, bearbeitet und gelöscht werden. Über die Kommentare kann z.B. eine Person ein Dokument verfassen und eine zweite Person gibt zu einzelnen Textteilen seine Meinung bekannt. Ein Anwendungsfall wäre das Korrekturlesen einer wissenschaftlichen Arbeit eines Studenten durch einen Professor.

3.2 Kommunikations Modul

Das Kommmunikations Modul soll so gut als möglich eine Diskussion mehrere Personen am runden Tisch ersetzen. Der Benutzer soll dabei selbst entscheiden welche Art von virtueller Kommunikation bzw. Diskussion er wählt. Es ist möglich mit einem einzelnen Benutzer zu Chatten oder über einen Gruppenchat eine Diskussion zu starten. Eine weitere Möglichkeit zur Diskussion bietet die Audio bzw. Videotelefonie direkt über den Browser. Besonders die Videotelefonie ist eine wichtige Kommunikationsmöglichkeit, da laut einer Studie des Institut für Demoskopie Allensbach und des Institut für Publizistik der Universität Mainz 59% der Wirkung einer Rede über Körpersprache und

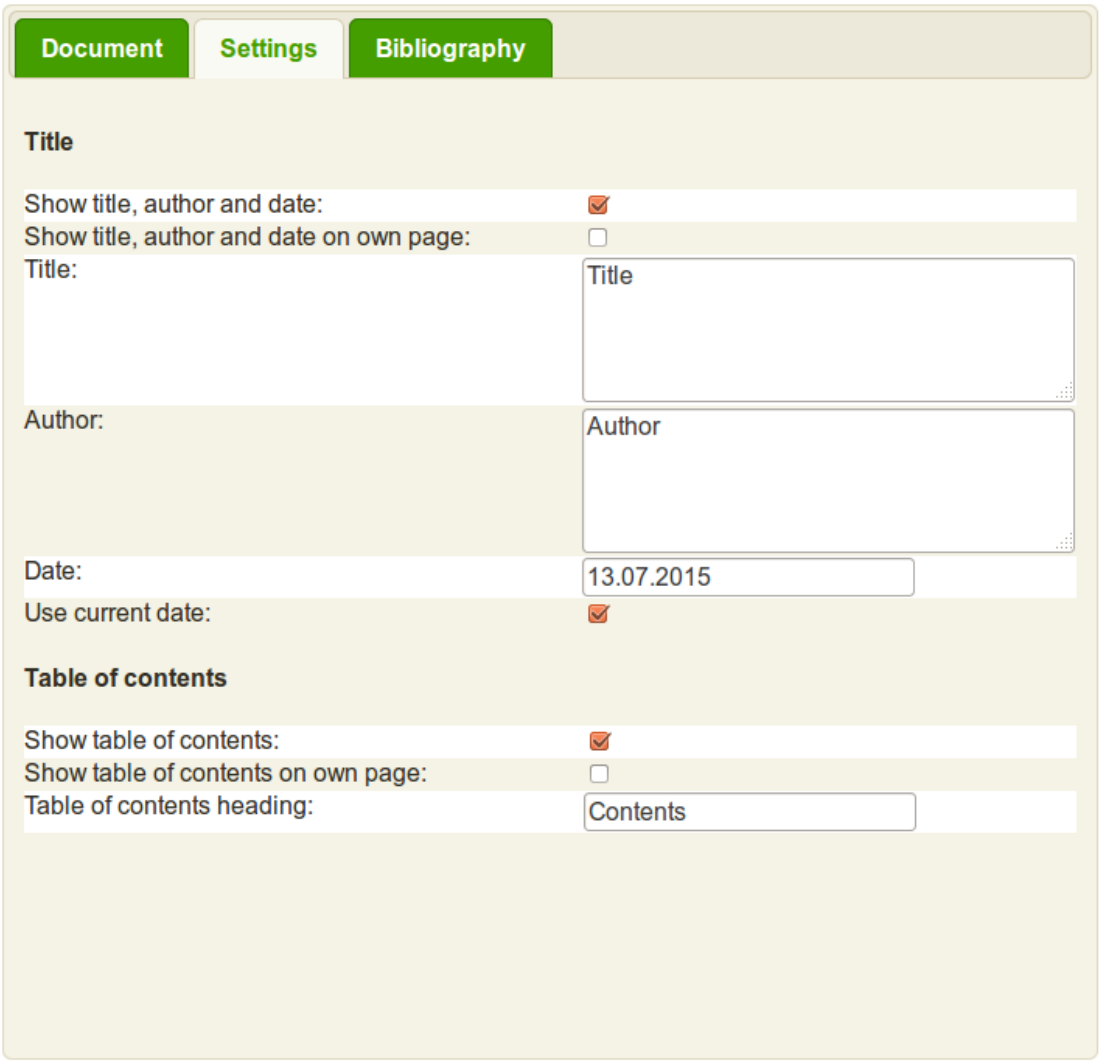


Abbildung 3.2: Benutzeroberfläche der Dokument Einstellungen

Erscheinungsbild übertragen werden [8]. Die Videotelefonie ermöglicht die so wichtige Nutzung des visuellen Kanals bei der Kommunikation.

3.3 Vorschau Modul

Das Vorschau Modul zeigt die aktuelle PDF Version des Dokuments. Wird eine Änderung am Dokument im Editor oder unter den Dokument Einstellungen vorgenommen wird die Vorschau automatisch aktualisiert. Das Aktualisieren der Vorschau kann etwas Zeit in Anspruch nehmen. Das Dokument muss am Server mit *pdflatex* kompiliert werden und im Anschluss als PDF an den Client ausgeliefert werden. Über das Vorschau Modul kann die aktuelle PDF Version des Dokuments heruntergeladen werden. Der im Browser integrierte PDF Viewer bietet auf die Möglichkeit im Dokument eine Suche zu starten. Die Abbildung 3.3 auf Seite 36 zeigt die Vorschau des in Abbildung 3.1 auf Seite 32 erstellten Dokuments.

Abbildung 3.3: PDF Vorschau des aktuellen Dokuments

4 Lösung

In diesem Kapitel wird beschrieben, auf Basis welcher Technologien diese Arbeit realisiert wurde. Es wird auf die Architektur der einzelnen Module aus dem Kapitel Problemstellung eingegangen sowie auf die wesentlichen technischen Details in den verschieden Modulen.

Die Applikation wurde als *JSF 2.2* Web Applikation realisiert. Der Server kommuniziert mit den Clients über ein *JSR 356 WebSocket*. Die Clients kommunizieren untereinander mit *WebRTC*.

4.1 Dokument Modul

Das Dokument Modul ermöglicht es Dokument Komponenten (z.B. Überschrift, Absatz, ...) hinzuzufügen, verschieben und zu löschen. Das Dokument Modul muss die Dokumentstruktur, kollaborative Inhalte und die Dokumenteinstellungen zwischen den Clients synchronisieren sowie eine aktuelle Version des Dokuments am Server halten. Das Synchronisieren der Dokumentstruktur wird mit Hilfe der *JSF JavaScript API* und *WebSockets* realisiert. Für das Synchronisieren von Dokumenteinstellungen wurden kollaborative JSF Komponenten implementiert. Das Synchronisieren von kollaborativen Inhalten wird mit Hilfe des Differential Synchronization Algorithmus ermöglicht.

4.1.1 Architektur

Das Dokument selbst sowie die einzelnen Dokument Komponenten wurden als *JSF Custom Components* realisiert. Die Kommunikation zwischen Server und kollaborativen Dokument Komponenten wird über ein *JSR 356 WebSockets* durchgeführt. Änderungen an der Dokumentstruktur werden mit Hilfe von *JSF Ajax* Requests an den Server übermittelt und im Anschluss vom Server über das *WebSocket* an die restlichen Clients weitergeleitet. Die Änderungen am DOM Baum der Clients werden durch die *JSF JavaScript API* durchgeführt.

Komponenten, JSF Renderer und Latex Renderer

Die folgende Beschreibung der Komponenten und deren Renderer basiert auf dem Klassendiagramm aus Abbildung 4.1 auf Seite 39. Ein LaTeX Dokument wird durch die Klasse *DocumentRoot* beschrieben. Die Klasse *DocumentRoot* hält in der Liste *children* alle Dokument Komponenten. Als Platzhalter für beliebige Dokument Komponenten wird in weiterer Folge die Klasse *DocumentComponentX* verwendet. Alle *DocumentComponentX* und die *DocumentRoot* leiten von der abstrakten Klasse *DocumentComponentBase* ab. Durch diese Klasse hält jede Dokument Komponente eine Referenz auf das Dokument (*DocumentRoot*). Diese Referenz wird im speziellen beim Rendern des PDF benötigt. Genauer beschrieben wird dies im Kapitel PDF Darstellung auf Seite 74.
Jeder Dokument Komponenten Klasse wird über die Annotation *RendererBinding* ein eigener LaTeX Render sowie ein eigener *JSF* Renderer zugewiesen. Die LaTeX Renderer leiten von der Klasse *LatexRenderer* und die *JSF* Renderer leiten von der Klasse *JsfRenderer* ab. Über den *JsfRenderer* ist es möglich ein *DocumentRoot* mit seinen Dokument Komponenten für die Auslieferung an den Client in Form von *HTML* und *JavaScript* zu rendern. Dies beinhaltet auch die Oberfläche für die Editierung der Dokument Struktur sowie der Dokument Komponenten am Client. Über den *LatexRenderer* ist es möglich ein *DocumentRoot* mit seinen Dokument Komponenten als LaTeX Quelltextdatei zu rendern um im Anschluss die Konvertierung in ein PDF durchzuführen.
Die Daten der einzelnen Dokument Komponenten werden direkt in den Dokument Komponenten Klassen gehalten (*DocumentComponentX*). Dies kann z.B. ein kollaborativer Text der Dokument Komponente Absatz sein oder der kollaborative Text einer Überschrift. Die Dokument Einstellungen aus Kapitel 3.1.2 auf Seite 32 werden direkt in der Klasse *DocumentRoot* gehalten. Die Renderer bekommen in ihren Rendering Methoden *encodeBegin* und *encodeEnd* die zugehörige Komponente übergeben und können so auf die Daten beim Rendern zugreifen.

4.1.2 Kollaborative Dokumentstruktur mit JSF

Wird ein Dokument von einem Benutzer geöffnet gibt es eine *DocumentRoot* und diese besitzt keine Dokument Komponenten. Der *DocumentRootJsfRenderer* liefert dem Benutzer also ein leeres Dokument auf die Web Oberfläche. Zusätzlich liefert der Renderer eine Oberfläche aus, um einerseits Komponenten hinzuzufügen, zu löschen oder in der Reihenfolge zu verschieben. Die Oberfläche steuert den Server mit den Befehlen *add*, *move* und *delete* an. Hat der erste

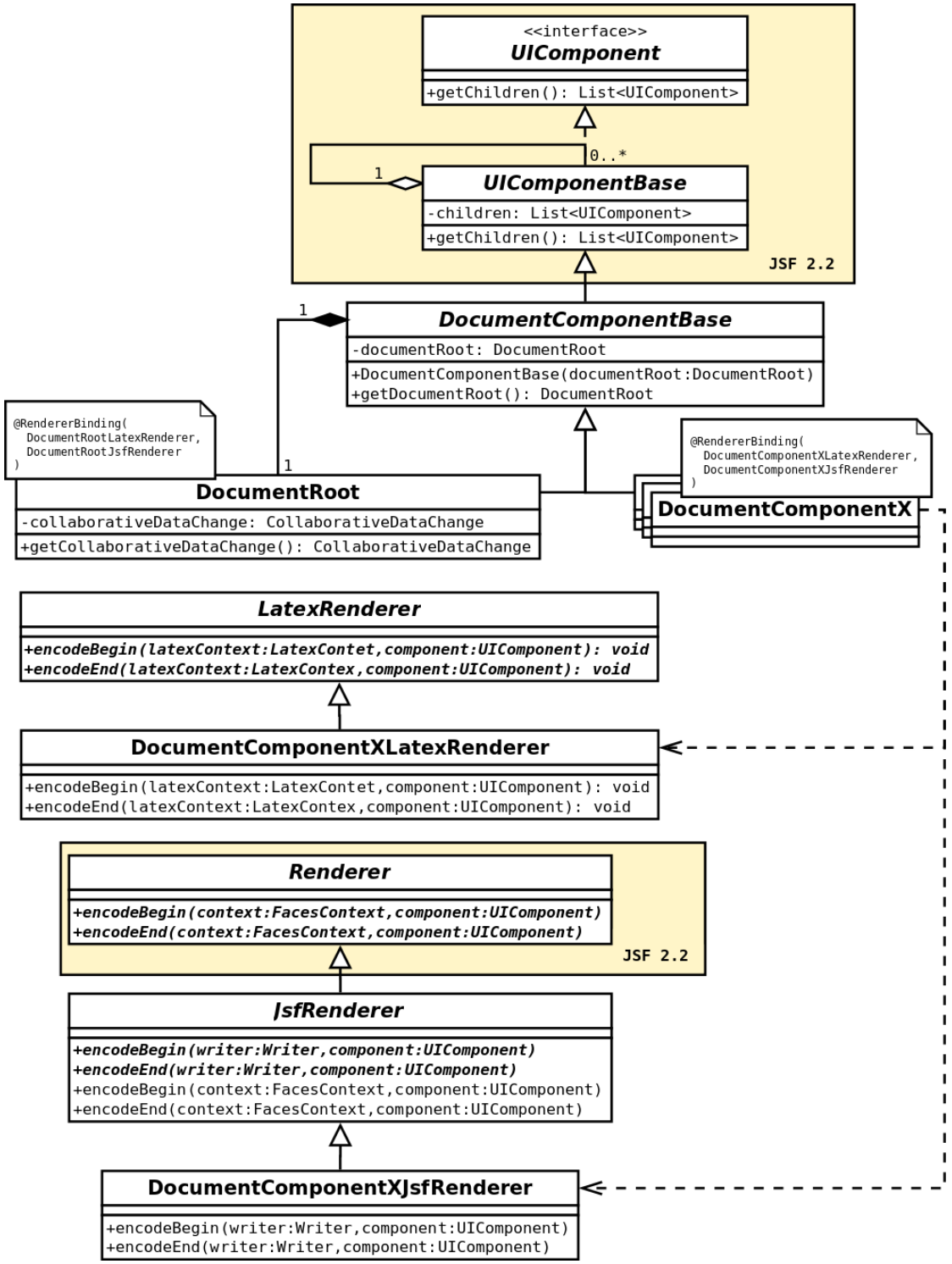


Abbildung 4.1: Klassendiagramm des Dokuments, der Dokument Komponenten und deren Renderer für *JSF* und LaTeX.

Benutzer nun bereits mehrere Komponenten hinzugefügt und ein zweiter Benutzer öffnet dasselbe Dokument, wird die *DocumentRoot* mit all seinen bereits vorhandenen Dokument Komponenten gerendert. Wird einer der Befehle *add*, *move* oder *delete* an den Server gesendet, wird zuerst der Befehl auf der Server Dokument Struktur ausgeführt. Es wird also z.B. beim Befehl *add* der Liste *children* von der Instanz der Klasse *DocumentRoot* ein neues Objekt vom Typ *DocumentComponentX* hinzugefügt. Nun muss die Änderung auch am DOM Baum aller Clients durchgeführt werden. *JSF* stellt hierfür bereits ein Verfahren zur Verfügung. Über die Methode *jsf.ajax.response* der *JSF JavaScript API* kann der DOM Baum eines Clients gezielt editiert werden [41]. Als Parameter wird ein JSF Response XML übergeben. Dieses XML beschreibt die Änderungen am DOM Baum. Der Quellcode 4.1 auf Seite 40 zeigt das JSF Response XML für den Befehl *add*. Der Quellcode 4.2 auf Seite 40 zeigt das JSF Response XML für den Befehl *delete*. Der Quellcode 4.3 auf Seite 40 zeigt das JSF Response XML für den Befehl *move*.

```
<?xml version='1.0' encoding='UTF-8'?>
<partial-response>
  <changes>
    <insert>
      <after id="previousId">
        <![CDATA[<!--insert HTML here-->]]>
      </after>
    </insert>
  </changes>
</partial-response>
```

Quellcode 4.1: JSF Response XML - Insert

```
<?xml version='1.0' encoding='UTF-8'?>
<partial-response>
  <changes>
    <delete id="id" />
  </changes>
</partial-response>
```

Quellcode 4.2: JSF Response XML - Delete

```
<?xml version='1.0' encoding='UTF-8'?>
<partial-response>
  <changes>
    <delete id="id" />
    <insert>
      <after id="newPreviousId">
        <![CDATA[<!--insert HTML here-->]]>
      </after>
    </insert>
  </changes>
</partial-response>
```

Quellcode 4.3: JSF Response XML - Move

Der Server wandelt also den erhaltenen Befehle in ein JSF Response XML um und sendet dieses im Anschluss per *WebSocket* an alle Clients. Über die *JSF JavaScript API* mit dem Befehl *jsf.ajax.response* und dem XML als Parameter werden dann am Client die Änderungen am DOM Baum durchgeführt. Der Ablauf beim Editieren der kollaborativen Dokument Struktur wird durch die folgende Aufzählung zusammengefasst:

1. Der Benutzer führ am Client über die Oberfläche eine Änderung durch.
2. Die Änderung wird erkannt und als Befehl an den Server gesendet.
    - Hier die möglichen Befehle (Parameter werden in Klammer gelistet):
        - add(type,previousId)
        - delete(id)
        - move(id,newPreviousId)
3. Der Server führt die Änderung an der aktuellen Dokument Struktur am Server durch.
4. Die Änderung wird durch ein JSF Response XML beschrieben.
5. Das JSF Response XML wird an alle Clients gesendet.
6. Die *JSF JavaScript API* ändert den DOM Baum aller Clients auf Basis des *JSF Response XML*.

Synchronisations Problem

Durch das genannte Verfahren zur Implementierung einer kollaborativen Dokumentstruktur ergibt sich ein Synchronisations Problem. Senden Client 1, und Client 2 den Befehl *add* zeitgleich und wollen an derselben Stelle eine neue Dokument Komponente hinzufügen, kann sich auf Grund von unterschiedlichen Übertragungsgeschwindigkeiten ein inkonsistenter Zustand ergeben. Die Reihenfolge der Dokument Komponenten auf Server, Client 1 und Client 2 könnte unterschiedlich sein. Dieses Problem kann dadurch gelöst werden, dass der Server die Befehle in einer Queue empfängt. Der nächste Befehl aus der Queue wird erst abgearbeitet, wenn der Server von jedem Client die Information erhalten hat, dass die letzte Änderung am Client erfolgreich ausgeführt wurde. Dadurch ist die Dokument Komponenten Reihenfolge von allen Clients und dem Server identisch. Dieses Verfahren verlangsamt natürlich das gemeinsame Ändern der Dokumentstruktur. Änderungen an der Dokumentstruktur werden jedoch relativ selten durchgeführt im Vergleich zu Änderungen am Inhalt. Deshalb wird dieses Bottleneck bewusst in Kauf genommen.

4.1.3 Differential Synchronization

Um die Daten der einzelnen Dokument Komponenten wurde der Algorithmus Differential Synchronization gewählt. Da in diesem Projekt der Server als JSF Web Applikation realisiert wurde muss die Serverseite des Algorithmus in *Java* und die Client Seite des Algorithmus in *JavaScript* realisiert werden. Neil Fraser stellt hierfür bereits fertige Implementierungen zur Verfügung [17]. Die Implementierung von Neil Fraser ist jedoch nur dazu geeignet um Plaintext zwischen Server und verschiedenen Clients zu synchronisieren. Für die Dokument Komponente *Kapitel* ist dies ausreichend. Der Text des Kapitels muss nicht formatiert werden. Die Formatierung für das Dokument ergibt sich bereits aus dem Typ der Dokument Komponente. Für die Dokument Komponente *Absatz* ist dies nicht ausreichend, da die Formatierungen des Textes mit synchronisiert werden muss.

HTML codieren

Verwendet man die Differential Synchronization Implementierung von Neil Fraser um HTML Quellcode zu synchronisieren, kann ein invalider Quellcode entstehen. Wird beispielsweise zeitgleich ein HTML Tag von zwei Benutzern editiert, kann es dazu führen, dass der Patch Algorithmus die zwei editierten Versionen mischt und dadurch ein ungültiger Tag entsteht. Eine primitive Lösung dieses Problems ist, vor dem Ausführen des Diff Algorithmus alle HTML Tags durch einzelne Unicode Zeichen zu ersetzen, welche nicht im Text vorkommen können. Vor dem Rendern der editierten Version am Client müssen die Unicode Zeichen wieder in HTML Tags umgewandelt werden können. [19] Mit dieser Methode können jedoch nur HTML Tags ohne Attribute ersetzt werden. Beispiele hierfür sind *<bold></bold>* oder **. Würde man HTML Tags mit Attributen ersetzen, müsste man für jede mögliche Attribut Kombination ein eigenes Unicode Zeichen definieren. Für das Setzen der Textfarbe mit dem Tag ** müsste also für jede mögliche Farbe ein eigenes Unicode Zeichen definiert werden. Dies ist nicht Praxistauglich.

HTML in JSON Transformation

Sieht man sich den Algorithmus im Kapitel Differential Synchronization auf Seite 20 genau an, basiert dieser lediglich auf einem Diff und einem Patch Algorithmus. Eine Möglichkeit um HTML mit Differential Synchronization zu

synchronisieren ist es, das HTML vor dem Erzeugen des Diff in JSON zu konvertieren. Im Anschluss wird vom JSON ein Diff zur vorherigen Version erzeugt. Das erzeugte Diff basiert auf RFC6902 [6]. RFC6902 ist ein Standard um JSON Patch Befehle zu beschreiben. Erhält ein Client oder der Server einen RFC6902 Patch muss er zuerst seine aktuelle HTML Version nach JSON konvertieren, dann den Patch ausführen und zum Schluss das JSON wieder in HTML für die Ausgabe konvertieren.
Die Differential Synchronization Implementierung von Neil Fraser für Plaintext ist Performance optimiert. Das Konvertieren nach JSON, Erzeugen des JSON Patches, Patchen des JSON und zurück Konvertieren in HTML kann mit dieser Performance nicht mithalten. Deshalb ist eine Mischung aus dem Diff-Patch über JSON und Diff-Patch für Plaintext die Lösung. Zuerst wird der RFC6902 Patch aus dem Diff der aktuellen und der Shadow Version des JSON erzeugt. Die Befehle des RFC6902 Patch zur Strukturveränderung bleiben unverändert. Diese sind *add*, *remove* und *move*. Der Befehl *replace* würde bei jeder Änderung am Client den gesamten geänderten Inhalt an alle anderen Clients bzw. an den Server senden. Dies ist das größte Performance Bottleneck, da der *replace* Befehl der am häufigsten auftretende Befehl sein wird im Vergleich zu Strukturveränderungen. Das Schreiben von 100 Zeichen und anschließende Fett setzen dieser Zeichen würde *100 replace* Befehl sowie 2 *add* und einen *move* Befehl auslösen. Dieses Beispiel zeigt, dass der *replace* Befehl der häufigste ist. Um die Performance des *replace* Befehls zu optimieren werden als Parameter des Befehls nicht der gesamte neue Text sondern nur ein Patch basierend auf der Plaintext Differential Synchronization Implementierung übermittelt.
Die folgende Aufzählung ist eine Zusammenfassung des Differential Synchronization für HTML mit der JSON Transformation. Die Zusammenfassung basiert auf der Abbildung 2.4 auf Seite 21.

1. Der Client Text ist HTML.
2. Die Client Shadow ist das HTML konvertiert in JSON.
3. Der Benutzer führt eine Änderung am Client Text durch bzw. formatiert diesen.
4. Der Client Text wird in JSON konvertiert.
5. Das JSON des Client Text wird mit dem JSON der Client Shadow verglichen. Daraus wird ein RFC6902 Patch erzeugt.
6. Die Befehle *add*, *remove* und *move* bleiben unverändert des RFC6902 Patches.
7. Der Befehl *replace* wird verändert.
    a) Vergleichen des neuen Textes mit dem alten Text aus dem Client Shadow JSON.

   b) Erzeugen eines Patches basieren auf der Plaintext Differential Synchronization Implementierung.
8. Übermitteln des Patches an den Server.
9. Die Server Shadow ist JSON.
10. Der Server Text ist JSON.
11. Patchen von Server Shadow und Server Text.

Erhält der Client einen Patch sind folgende Schritte am Client durchzuführen:

1. Client HTML in JSON konvertieren.
2. RFC6902 Patch am JSON durchführen unter Berücksichtigung des veränderten *replace* Befehls.
3. JSON zurück in HTML konvertieren.

Serverseitige Implementierung

Dieses Kapitel Zeigt, wie der Differential Synchronization Algorithmus serverseitig implementiert wurde. Hierzu zeigt Abbildung 4.2 das Klassendiagramm. Zentrales Element ist die nach dem Singleton Pattern implementierte Klasse *CollaborationBackend*. Die Methoden dieser Klasse werden über ein *WebSocket* vom Client aufgerufen. Die Klasse *CollaborationBackend* hält eine Map von *CollaborationSession*. Der Key der Map ist hierbei die ID der kollaborativ zu bearbeitenden Daten (*collaborationUuid*). Eine *CollaborationSession* beinhaltet alle Daten die benötigt werden um Differential Synchronization auf Daten anwenden zu können. Hierzu gehören die Server Daten und die Server Shadow Daten jedes Clients. Des Weiteren stellt die Klasse *CollaborationSession* Methoden bereit, um die aktuellen Daten zu patchen. Die verschiedenen Server Shadow Daten werden in einer Map von *CollaborativeConnection* gehalten. Der Key für diese Map ist die Session ID des *WebSockets* mit dem der Client zum Server verbunden ist. Daten die mit Hilfe von Differential Synchronization synchronisiert werden möchten, müssen von der Klasse *CollaborativeData* ableiten. Diese abstrakte Klasse beinhaltet die abstrakten Methoden *diff* und *patch*. Konkret implementiert wurden die Klassen zur Synchronisation von Plaintext (Klasse *CollaborativeText*) und JSON (*CollaborativeHtml*). Die Klasse *CollaborativeHtml* hält die HTML Daten in Form von JSON Objekten wie im Kapitel HTML in JSON Transformation auf Seite 42 beschrieben. Vorteil dieser Architektur ist, dass Differential Synchronization somit auf beliebige Daten anwendbar ist. Es muss nur möglich sein auf die Daten einen Diff und einen Patch Algorithmus anzuwenden und eine Klasse implementiert werden die von der Klasse *CollaborativeData* ableitet.

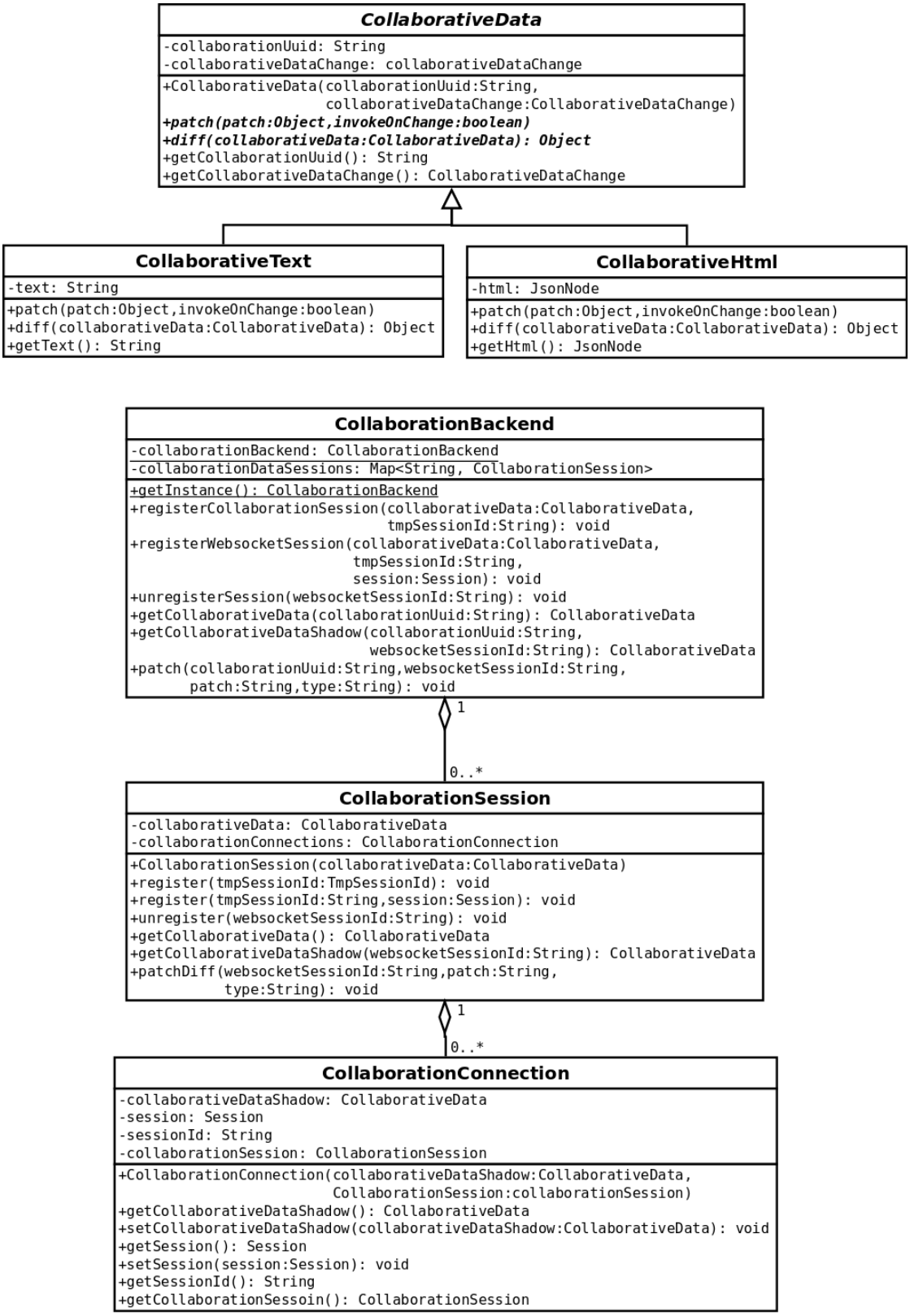


Abbildung 4.2: Klassendiagramm der serverseitigen Implementierung des Differential Synchronization Algorithmus.

Der Quellcode 4.4 auf Seite 46 zeigt den Quellcode der Klasse *CollaborationBackend*. Der Ablauf der Methodenaufrufe ist wie folgt. Kollaborative Komponenten wurde als JSF Custom Components realisiert. Beim Rendern einer solchen JSF Custom Component ist bekannt, welche *collaborationUuid* die kollaborative zu bearbeitenden Daten besitzen. Die *collaboratioUuid* ist also eine ID für einen kollaborative zu bearbeitenden Inhalt. Dies kann z.B. der Text eines HTML Input Textfeldes oder einer Textarea sein oder auch das *innerHTML* eines HTML DIV Tags mit Attribut *contenteditable* und zugehörigem Wert *true*.
Beim Rendern der JSF Komponente wird die Methode *registerCollaborationSession* aufgerufen. In dieser Methode wird überprüft, ob für diese *collaborationUuid* bereits eine *CollaborationSession* vorhanden ist. Ansonsten wird diese erzeugt. Im Anschluss wird das HTML der Komponente an den Client ausgeliefert. Nachdem der Client die Komponente geladen hat, wird eine *WebSocket* Verbindung zum Server aufgebaut. Im Anschluss wird der Befehl *register* per *WebSocket* an den Server übermittelt. Als Parameter wird die *collaborationUuid* sowie die *tmpSessionId* übergeben. Nach Erhalt dieses Befehls am Server, kann das *WebSocket* die Methode *registerWebsocket* im *CollaborationBackend* aufrufen. Dadurch wird eine Referenz auf die *WebSocket* Session in der Klasse *CollaborationConnection* gespeichert. Zugeordnet wird dies über *tmpSessionId*. Die Referenz auf die *WebSocket* Session ist dazu notwendig, um bei Änderungen an den Server Daten im Anschluss patches an alle Clients über das *WebSocket* senden zu können.
Wird am Client eine Änderung an den kollaborativen Daten durchgeführt, wird aus dem Unterschied von Client Daten und Client Shadow ein Diff erzeugt und im Anschluss als Patch über das *WebSocket* an den Server gesendet. Das *WebSocket* ruft im Anschluss die Methode *patch* des *CollaborationBackend* auf.
Schließt ein Client ein Browser Fenster, wird die Websocket Verbindung zum Server getrennt. Beim Trennen dieser Verbindung wird die Methode *unregisterSession* im *CollaborationBackend* aufgerufen. Dabei wird die zugehörige *CollaborationConnection* gelöscht. Beinhaltet die *CollaborationSession* keine *CollaborationConnections* mehr, wird auch die *CollaborationSession* gelöscht.

```
1  package at.reppeitsolutions.collaboration;
2
3  import at.reppeitsolutions.collaboration.model.CollaborativeData;
4  import java.util.Collections;
5  import java.util.HashMap;
6  import java.util.Iterator;
7  import java.util.Map;
8  import javax.websocket.Session;
9
10 /**
11  * Singleton which holds all CollaborationSessions
12  * @author Mathias Reppe <mathias.reppe@gmail.com>
13  */
14 public class CollaborationBackend {
15
```

```
16      private Map<String, CollaborationSession> collaborationDataSessions
17              = Collections.synchronizedMap(
18                      new HashMap<String, CollaborationSession>());
19
20      private CollaborationBackend() {
21      }
22
23      public static CollaborationBackend getInstance() {
24          return CollaborationBackendHolder.INSTANCE;
25      }
26
27      private static class CollaborationBackendHolder {
28
29          private static final CollaborationBackend INSTANCE
30                  = new CollaborationBackend();
31      }
32
33      /**
34       * Registers new collaborative data.
35       *
36       * @param collaborativeData Data for collaboration.
37       * @param tmpSessionId Temporary session id. Used until reference to
38       * WebSocket session is available.
39       */
40      public synchronized void registerCollaborationSession(
41              CollaborativeData collaborativeData,
42              TmpSessionId tmpSessionId) {
43          String collaborationUuid = collaborativeData.getCollaborationUuid();
44          CollaborationSession collaborationSession
45                  = collaborationDataSessions.get(collaborationUuid);
46          if (collaborationSession == null) {
47              collaborationSession = new CollaborationSession(collaborativeData);
48              collaborationDataSessions.put(
49                      collaborationUuid,
50                      collaborationSession);
51          }
52          collaborationSession.register(tmpSessionId);
53      }
54
55      /**
56       * Registers WebSocket for collaborative data.
57       * @param collaborationUuid ID of collaborative data
58       * @param tmpSessionId Used tmpSessionId when registered.
59       * @param session WebSocket session
60       */
61      public synchronized void registerWebsocket(String collaborationUuid,
62              String tmpSessionId,
63              Session session) {
64          CollaborationSession collaborationSession
65                  = collaborationDataSessions.get(collaborationUuid);
66          if (collaborationSession != null) {
67              collaborationSession.register(tmpSessionId, session);
68          } else {
69              throw new RuntimeException("Unable to set real session ID.");
70          }
71      }
72
73      /**
74       * Unregisters a WebSocket
75       * @param websocketSessionId ID of WebSocket Session
76       */
77      public synchronized void unregisterSession(String websocketSessionId) {
```

```
78          Iterator<CollaborationSession> iter
79                  = collaborationDataSessions.values().iterator();
80          while (iter.hasNext()) {
81              CollaborationSession collaborationSession = iter.next();
82              if (collaborationSession.unregister(websocketSessionId)) {
83                  iter.remove();
84              }
85          }
86      }
87
88      /**
89       * Returns collaborative data.
90       * @param collaborationUuid CollaborationUuid of collaborative data.
91       * @return Current server version of collaborative data.
92       */
93      public synchronized CollaborativeData getCollaborativeData(
94              String collaborationUuid) {
95          if (collaborationDataSessions.containsKey(collaborationUuid)) {
96              CollaborationSession collaborationSession
97                      = collaborationDataSessions.get(collaborationUuid);
98              return collaborationSession.getCollaborativeData();
99          }
100         return null;
101     }
102
103     /**
104      * Returns collaborative data shadow of one client
105      * @param collaborationUuid CollaborationUuid of collaborative data.
106      * @param websocketSessionId WebSocket session ID.
107      * @return Current server shadow of collaborative data.
108      */
109     public synchronized CollaborativeData getCollaborativeDataShadow(
110             String collaborationUuid,
111             String websocketSessionId) {
112         if (collaborationDataSessions.containsKey(collaborationUuid)) {
113             CollaborationSession collaborationSession
114                     = collaborationDataSessions.get(collaborationUuid);
115             return collaborationSession.
116                     getCollaborativeDataShadow(websocketSessionId);
117         }
118         return null;
119     }
120
121     /**
122      * Returns CollaborationSession by collaborationUuid.
123      * @param collaborationUuid CollaborationUuid of collaborative data.
124      * @return CollaborationSession
125      */
126     public CollaborationSession getCollaborativeSession(
127             String collaborationUuid) {
128         return collaborationDataSessions.get(collaborationUuid);
129     }
130
131     /**
132      * Sends command to all registered clients.
133      * @param collaborationUuid CollaborationUuid of collaborative data.
134      * @param collaborationCommand The command.
135      */
136     public void broadcast(String collaborationUuid,
137             CollaborationCommand collaborationCommand) {
138         CollaborationSession collaborationSession
139                 = getCollaborativeSession(collaborationUuid);
```

```
140         if (collaborationSession != null) {
141             collaborationSession.broadcast(collaborationCommand);
142         }
143     }
144
145     /**
146      * Patches collaborative data received from one client.
147      * @param collaborationUuid CollaborationUuid of collaborative data.
148      * @param websocketSessionId WebSocket Session ID.
149      * @param patch Patch string.
150      * @param type Type of patch (f.e. json, text)
151      */
152     public void patch(String collaborationUuid,
153             String websocketSessionId,
154             String patch,
155             String type) {
156         CollaborationSession collaborationSession
157                 = collaborationDataSessions.get(collaborationUuid);
158         if (collaborationSession != null) {
159             collaborationSession.patchDiff(patch,
160                     websocketSessionId,
161                     type);
162         }
163     }
164
165 }
```

Quellcode 4.4: Quellcode der Klasse CollaborationBackend.

Der Quellcode 4.5 auf Seite 50 zeigt den Quellcode der Klasse *CollaborationSession*. Eine *CollaborationSession* kapselt genau ein kollaborativ zu bearbeitendes Objekt. Die *CollaborationSession* beinhaltet die Server Version der zu synchronisierenden Daten und zusätzlich hält die Klasse eine Map von *CollaborationConnection*. Der Key der Map ist die *WebSocket Session ID*. In dieser Map werden alle Verbindungen zu den Clients gespeichert und zusätzlich die Server Shadow Daten.
Die Klasse stellt zwei *register* Methoden zur Verfügung. Einmal mit der *tmpSessionId* als Parameter. Das bedeutet die *CollaborationConnection* ist erzeugt, die Server Shadow wurde erzeugt, es ist jedoch noch keine Referenz zur *WebSocket* Session vorhanden, weil der Client noch keine *WebSocket* Verbindung zum Server aufgebaut hat und sich noch nicht mit dem Befehl *register* angemeldet hat. Die zweite *register* Methode mit der *tmpSessionId* und der *WebSocket Session* als Parameter dient dazu, um nach dem Erhalt des Befehls *register* vom Client, die *WebSocket Session* zu setzen.
Durch die Methode *patchDiff* wird ein vom Client empfangener Patch zuerst zuerst in die zugehörige Server Shadow des Clients gepatcht und im Anschluss in die Server Version gepatcht. Danach wird ein Diff zwischen allen Server Shadows und der Server Version erstellt. Ergibt das Diff einen Unterschied, wird aus diesem ein Patch erstellt und an den zugehörigen Client gesendet.

```
package at.reppeitsolutions.collaboration;

import at.reppeitsolutions.collaboration.model.CollaborativeData;
import at.reppeitsolutions.collaboration.model.CollaborativeHtml;
import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;
import com.github.fge.jsonpatch.FuzzyOperation;
import com.github.fge.jsonpatch.JsonPatch;
import com.github.fge.jsonpatch.PathValueOperation;
import java.io.IOException;
import java.util.Collections;
import java.util.HashMap;
import java.util.LinkedList;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.websocket.Session;
import name.fraser.neil.plaintext.diff_match_patch;

/**
 * One collaborative session holds the server version of the collaborative
 * data and all client shadow versions of the collaborative data.
 * @author Mathias Reppe <mathias.reppe@gmail.com>
 */
public class CollaborationSession {

    private CollaborativeData collaborativeData;
    private Map<String, CollaborationConnection> collaborationConnections
            = Collections.synchronizedMap(
                    new HashMap<String, CollaborationConnection>());
    private final ObjectMapper mapper = new ObjectMapper();

    public CollaborationSession(CollaborativeData collaborativeData) {
        this.collaborativeData = collaborativeData;
    }

    /**
     * Registers a new client. Creates the CollaborationConnection with the
     * server shadow of the collaborative data.
     * @param tmpSessionId
     */
    public synchronized void register(TmpSessionId tmpSessionId) {
        if (!collaborationConnections.containsKey(tmpSessionId.getId())) {
            collaborationConnections.put(
                    tmpSessionId.getId(),
                    new CollaborationConnection(collaborativeData.clone(),
                            this));
        }
    }

    /**
     * Adds the WebSocket session to a CollaborationConnection.
     * @param tmpSessionId
     * @param session
     */
    public synchronized void register(String tmpSessionId, Session session) {
        CollaborationConnection tmpCollaborationConnection
                = collaborationConnections.get(tmpSessionId);
        if (tmpCollaborationConnection != null) {
            tmpCollaborationConnection.setSession(session);
```

```
62              collaborationConnections.remove(tmpSessionId);
63              collaborationConnections.put(session.getId(),
64                      tmpCollaborationConnection);
65          }
66      }
67
68      /**
69       * Unregisters a client.
70       * @param websocketSessionId WebSocket session ID.
71       * @return if true there are no more active sessions
72       */
73      public synchronized boolean unregister(String websocketSessionId) {
74          collaborationConnections.remove(websocketSessionId);
75          return collaborationConnections.isEmpty();
76      }
77
78      /**
79       * Returns current server version of collaborative data.
80       * @return Current server version of collaborative data.
81       */
82      public CollaborativeData getCollaborativeData() {
83          return collaborativeData;
84      }
85
86      /**
87       * Returns current server shadow of collaborative data.
88       * @param websocketSessionId Websocket session ID.
89       * @return Current server shadow of collaborative data.
90       */
91      public CollaborativeData getCollaborativeDataShadow(
92              String websocketSessionId) {
93          return collaborationConnections.get(websocketSessionId)
94                  .getCollaborativeDataShadow();
95      }
96
97      /**
98       * Returns all Client connections.
99       * @return Map<String, CollaborationConnection>
100      */
101     public Map<String, CollaborationConnection> getCollaborationConnections() {
102         for (CollaborationConnection collaborationConnection
103                 : collaborationConnections.values()) {
104             if (collaborationConnection.getSession() == null) {
105                 unregister(collaborationConnection.getSessionId());
106             }
107         }
108         return collaborationConnections;
109     }
110
111     /**
112      * Patches the server version. Patches the server shadow version of the
113      * client. Creates a diff and patch between current server version and
114      * client shadow version. Sends this patch to all other clients.
115      * @param websocketSessionId WebSocket session ID.
116      * @param patchString String of patch.
117      * @param type Type of patch (e.g. json or text)
118      */
119     public synchronized void patchDiff(String websocketSessionId,
120             String patchString,
121             String type) {
122         if (patchString != null && !"".equals(patchString.trim())) {
123             if ("text".equals(type)
```

```
                    || "heading".equals(type)
                    || "checkbox".equals(type)) {
                patchDiffPlaintext(websocketSessionId, patchString, type);
            } else if ("json".equals(type)) {
                patchDiffJson(patchString, websocketSessionId);
            }
        }
    }

    private void patchDiffJson(String patchString, String websocketSessionId) {
        try {
            JsonPatch patch = mapper.readValue(patchString, JsonPatch.class);
            CollaborativeData collaborativeDataShadow
                    = collaborationConnections.get(websocketSessionId)
                    .getCollaborativeDataShadow();
            collaborativeDataShadow.patch(patch, false);
            collaborativeData.patch(patch, true);
            boolean fuzzy = patch.getOperations().size() == 1
                    && patch.getOperations().get(0) instanceof FuzzyOperation;
            for (CollaborationConnection collaborationConnection
                    : getCollaborationConnections().values()) {
                try {
                    Object patchObject
                            = collaborationConnection
                            .getCollaborativeDataShadow()
                            .diff(getCollaborativeData());
                    if (!(patchObject instanceof JsonPatch)) {
                        continue;
                    }
                    JsonPatch patch2 = (JsonPatch) patchObject;
                    if (patch2.getOperations() == null
                            || patch2.getOperations().isEmpty()) {
                        continue;
                    }

                    String patchString2 = patch2.toStringForClient();
                    if (patch2.getOperations().size() == 1 && fuzzy) {
                        patchString2 = fuzzyPatch(patch2,
                                collaborationConnection);
                    }
                    CollaborationCommand collaborationCommand
                            = new CollaborationCommand();
                    collaborationCommand.setCommand("patch");
                    collaborationCommand.setCollaborationUuid(
                            getCollaborativeData().getCollaborationUuid());
                    collaborationCommand.setArg1(patchString2);
                    collaborationCommand.setArg2("json");
                    collaborationConnection.getSession().getBasicRemote()
                            .sendText(mapper.writeValueAsString(
                                    collaborationCommand));
                    collaborationConnection.setCollaborativeDataShadow(
                            getCollaborativeData().clone());
                } catch (JsonProcessingException ex) {
                    Logger.getLogger(CollaborationSession.class.getName())
                            .log(Level.SEVERE, null, ex);
                } catch (IOException ex) {
                    Logger.getLogger(CollaborationSession.class.getName())
                            .log(Level.SEVERE, null, ex);
                }
            }
        } catch (IOException ex) {
            Logger.getLogger(CollaborationSession.class.getName())
```

```
186                     .log(Level.SEVERE, null, ex);
187         }
188     }
189 
190     private String fuzzyPatch(JsonPatch patch2,
191             CollaborationConnection collaborationConnection) {
192         String patchString2;
193         PathValueOperation pathValueOperation = (PathValueOperation) patch2
194                 .getOperations().get(0);
195         JsonNode collaborativeHtml =
196                 ((CollaborativeHtml) collaborationConnection
197                         .getCollaborativeDataShadow()).getHtml();
198         String oldValue = pathValueOperation.getPath()
199                 .path(collaborativeHtml).asText();
200         String newValue
201                 = pathValueOperation.getValue().asText();
202         diff_match_patch dmp = new diff_match_patch();
203         LinkedList<diff_match_patch.Diff> diffs
204                 = dmp.diff_main(oldValue, newValue, true);
205         if (diffs.size() > 2) {
206             dmp.diff_cleanupSemantic(diffs);
207         }
208         LinkedList<diff_match_patch.Patch> patches
209                 = dmp.patch_make(oldValue, newValue, diffs);
210         patchString2 = "[{"
211                 + "\"op\":\"fuzzy\","
212                 + "\"path\":\""
213                 + pathValueOperation.getPath().toString() + "\","
214                 + "\"value\":\""
215                 + dmp.patch_toText(patches).replaceAll("\\n", "\\\\n") + "\""
216                 + "}]";
217         return patchString2;
218     }
219 
220     private void patchDiffPlaintext(String websocketSessionId,
221             String patchString,
222             String type) {
223         CollaborativeData collaborativeDataShadow
224                 = collaborationConnections.get(websocketSessionId)
225                 .getCollaborativeDataShadow();
226         //Patch server shadow of client
227         collaborativeDataShadow.patch(patchString, false);
228         //Patch server version
229         collaborativeData.patch(patchString, true);
230         //Create diff between every server shadow and the server version.
231         //Create patch out of diff.
232         //Send the patch to every client.
233         for (CollaborationConnection collaborationConnection
234                 : getCollaborationConnections().values()) {
235             Object patchObject = collaborationConnection
236                     .getCollaborativeDataShadow().diff(getCollaborativeData());
237             if (!((String) patchObject).isEmpty()) {
238                 CollaborationCommand collaborationCommand
239                         = new CollaborationCommand();
240                 collaborationCommand.setCommand("patch");
241                 collaborationCommand.setCollaborationUuid(
242                         getCollaborativeData().getCollaborationUuid());
243                 collaborationCommand.setArg1((String) patchObject);
244                 collaborationCommand.setArg2(type);
245                 try {
246                     collaborationConnection.getSession().getBasicRemote()
247                             .sendText(
```

```
248                                     mapper.writeValueAsString(
249                                             collaborationCommand));
250                 } catch (JsonProcessingException ex) {
251                     Logger.getLogger(CollaborationSession.class.getName())
252                             .log(Level.SEVERE, null, ex);
253                 } catch (IOException ex) {
254                     Logger.getLogger(CollaborationSession.class.getName())
255                             .log(Level.SEVERE, null, ex);
256                 }
257             }
258             collaborationConnection.setCollaborativeDataShadow(
259                     getCollaborativeData().clone());
260         }
261     }
262
263     /**
264      * Sends command to all clients.
265      * @param collaborationCommand The command.
266      */
267     public void broadcast(CollaborationCommand collaborationCommand) {
268         for (CollaborationConnection collaborationConnection
269                 : getCollaborationConnections().values()) {
270             try {
271                 collaborationConnection.getSession().getBasicRemote().sendText(
272                         mapper.writeValueAsString(collaborationCommand));
273             } catch (IOException ex) {
274                 Logger.getLogger(CollaborationSession.class.getName())
275                         .log(Level.SEVERE, null, ex);
276             }
277         }
278     }
279
280 }
```

Quellcode 4.5: Quellcode der Klasse CollaborationSession.

Der Quellcode 4.6 auf Seite 55 zeigt den Quellcode der Klasse *CollaborationConnection*. Die Klasse hält die Server Shadow der Daten und symbolisiert eine Verbindung zu einem Client. Zusätzlich hält die Klasse eine Refernz auf die *WebSocket Session*. Über diese *Session* is es möglich Daten über das *WebSocket* an den Client zu senden. Beim Setzen der *WebSocket Session* wird zusätzlich die *WebSocket Session* ID als String gespeichert. Dies ist dazu notwendig, falls die Verbindung zum *WebSocket* abbricht. Dadurch wird die Referenz auf die *WebSocket Session null*. Versucht ein anderer Thread auf die *WebSocket Session* zuzugreifen und bemerkt, dass die *WebSocket Session* null ist, kann er die *sessionId* auslesen und über die *CollaborationSession* das Löschen der *CollaborationConnection* veranlassen. Die *sessionId* ist ja zugleich der Key für die Map in der die *CollaborativeConnection* gespeichert werden.

```
1  package at.reppeitsolutions.collaboration;
2
3  import at.reppeitsolutions.collaboration.model.CollaborativeData;
4  import javax.websocket.Session;
5
6  /**
7   * A CollaborationConnection stands for one client connection. Includes the
8   * server shadow version of a client.
9   * @author Mathias Reppe <mathias.reppe@gmail.com>
10  */
11 public class CollaborationConnection {
12
13     private CollaborativeData collaborativeDataShadow;
14     private Session session;
15     private String sessionId;
16     private CollaborationSession collaborationSession;
17
18     /**
19      * Creates a new CollaborationConnection
20      * @param collaborativeDataShadow Server shadow version of collaborative data.
21      * @param collaborationSession Reference to CollaborationSession to which
22      * this object belongs to.
23      */
24     public CollaborationConnection(CollaborativeData collaborativeDataShadow,
25             CollaborationSession collaborationSession) {
26         this.collaborativeDataShadow = collaborativeDataShadow;
27         this.collaborationSession = collaborationSession;
28     }
29
30     public CollaborativeData getCollaborativeDataShadow() {
31         return collaborativeDataShadow;
32     }
33
34     public void setCollaborativeDataShadow(
35             CollaborativeData collaborativeDataShadow) {
36         this.collaborativeDataShadow = collaborativeDataShadow;
37     }
38
39     /**
40      * Returns WebSocket session ID.
41      * @return WebSocket session ID.
42      */
```

```
43      public String getSessionId() {
44          return sessionId;
45      }
46
47      /**
48       * Return WebSocket session
49       * @return WebSocket session
50       */
51      public Session getSession() {
52          return session;
53      }
54
55      /**
56       * Set WebSocket session
57       * @param session WebSocket session
58       */
59      public void setSession(Session session) {
60          this.session = session;
61          this.sessionId = session.getId();
62      }
63
64      /**
65       * Returns reference to CollaborationSession to which this object
66       * belongs to.
67       * @return ColllaborationSession
68       */
69      public CollaborationSession getCollaborationSession() {
70          return collaborationSession;
71      }
72
73  }
```

Quellcode 4.6: Quellcode der Klasse CollaborationConnection.

4.1.4 Kollaborative JSF Komponenten

Um die Dokument Einstellungen zu synchronisieren, wurden folgende JSF Komponenten als kollaborative JSF Komponenten realisiert:

- *inputText*
  - Rendert den HTML Tag *input* mit dem Attribut *type* und zugehörigen Wert *input*
- *inputTextarea*
  - Render den HTML Tag *textarea*
- *booleanCheckbox*
  - Rendert den HTML Tag *input* mit dem Attribut *type* und zugehörigen Wert *checkbox*

Dieses Kapitel beschreibt die Implementierung der JSF Komponenten *collaborativeInputText* und *collaborativeTextarea*. In der Abbildung 4.3 auf Seite 58 wird das Klassendiagramm dargestellt. Die Komponente *collaborativeInputText* wird durch die Klasse *CollaborativeInputTextComponent* implementiert. Die Komponente *collaborativeTextarea* wird durch die Klasse *CollaborativeTextareaComponent* implementiert. Beide Klassen leiten von der abstrakten Klasse *CollaborativeTextComponent* ab. Diese Klasse leitet wiederum von der JSF Klasse *UIComponentBase* ab, dies wird im Klassendiagramm aus Gründen der Übersichtlichkeit nicht dargestellt. Realisiert man ein Custom JSF Component, muss die Komponenten Klasse von der Klasse UIComponentBase ableiben. In der Klasse *CollaborativeText* wird der Text der kollaborativen Textarea bzw. des kollaborativen Textfeldes gespeichert. Der Klasse *CollaborativeTextComponent* hält einen *CollaborativeText*. Die Klasse *CollaborativeText* leitet von der abstrakten Klasse *CollaborativeData* ab. Dadurch kann die Differential Synchronization Implementierung mit der Klasse *CollaborativeText* arbeiten. Wie der Text genau synchronisiert wird, wird im Kapitel 4.1.3 auf Seite 44 erklärt.
Sowohl die Klasse *CollaborativeTextareaComponent* und die Klasse *CollaborativeInputTextComponent* haben als Renderer die Klasse *CollaborativeTextRenderer*. Im *CollaborativeTextRenderer* wird das HTML und der *JavaScript* Code für den Client gerendert.

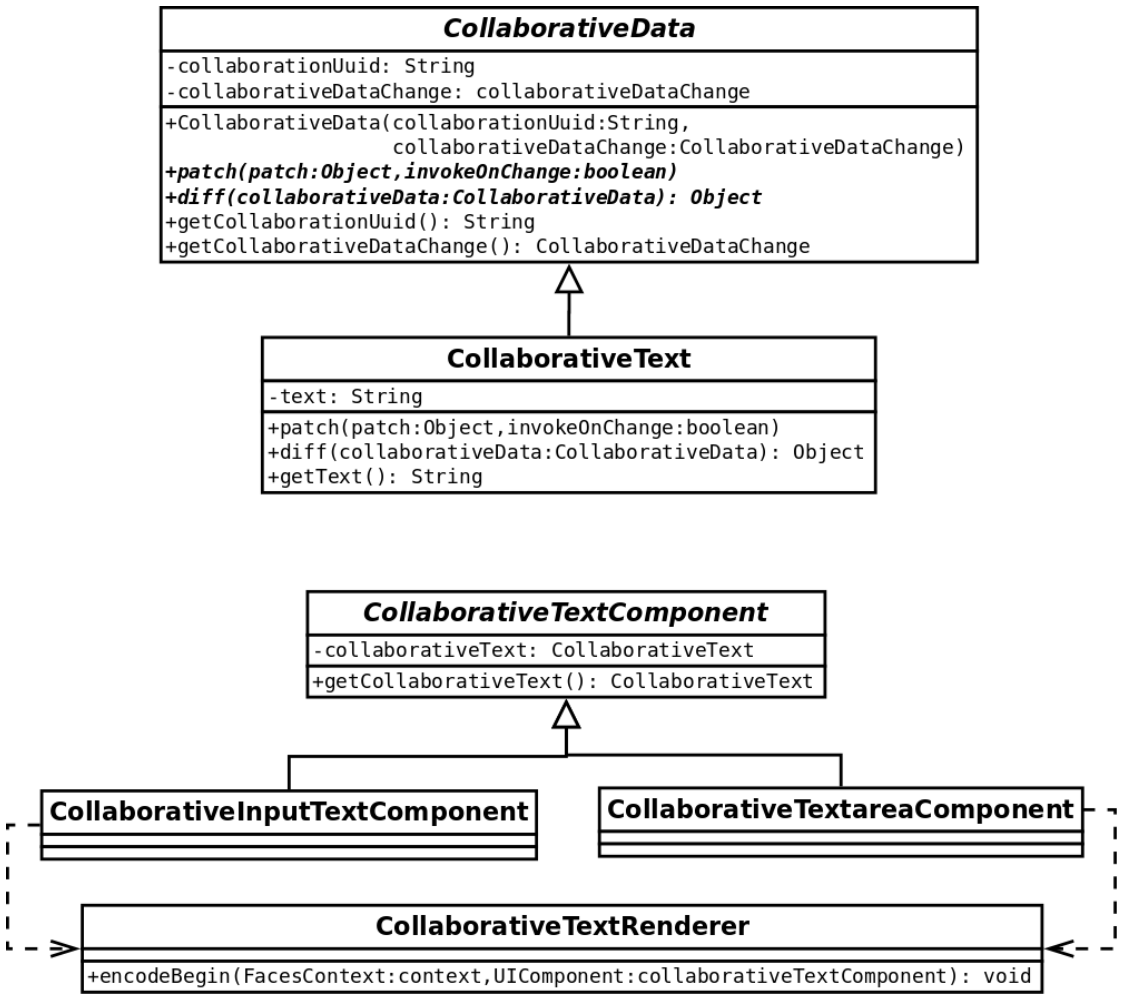


Abbildung 4.3: Klassendiagramm der kollaborativen JSF Komponenten *collaborativeText* und *collaborativeData*.

Der Quellcode 4.7 auf Seite 59 zeigt den Quellcode der Klasse *CollaborativeData*. Als Member Variable besitzt die Klasse den String *collaborationUuid*. Dies ist ein eindeutiger String, der einen zu synchronisierenden Datensatz identifiziert. Als weitere Member Variable besitzt die Klasse *CollaborativeData* eine Instanz einer Klasse die das Interface *CollaborativeDataChange* implementiert. Die *change* Methode des Interface wird jedes mal dann aufgerufen, wenn sich die Daten in den Subklassen von der Klasse *CollaborativeData* ändern.
Die zwei wichtigsten Methoden der Klasse sind die abstrakten Methoden *patch* und *diff*. Die Methode *diff* bekommt als Parameter eine Instanz der Klasse *CollaborativeData* und gibt als Rückgabewert ein Diff zurück. Der Typ des Diff ist nicht bestimmt, das Diff hängt von der *diff* Implementierung der Subklassen ab. Die Methode *patch* bekommt als Parameter einen Patch. Der Typ des Patch ist wiederum nicht bestimmt, er ist abhängig von der *patch* Implementierung der Subklassen. Über den Boolean Parameter *invokeOnChange* kann bestimmt werden, ob die *change* Methode von der Variable *collaborativeDataChange* aufgerufen wird.
Die abstrakte Methode *clone* muss implementiert werden, da beim Erzeugen der Server Shadow die Server Version der Daten kopiert werden muss.

```
1  package at.reppeitsolutions.collaboration.model;
2
3  /**
4   *
5   * @author Mathias Reppe <mathias.reppe@gmail.com>
6   */
7  public abstract class CollaborativeData {
8
9      protected String collaborationUuid;
10     protected CollaborativeDataChange collaborativeDataChange;
11
12     public CollaborativeData(String collaborationUuid,
13             CollaborativeDataChange collaborativeDataChange) {
14         if(collaborationUuid.contains("-")) {
15             throw new RuntimeException(
16                     "The collaborationUuid must not include a -.");
17         }
18         this.collaborationUuid = collaborationUuid;
19         this.collaborativeDataChange = collaborativeDataChange;
20     }
21
22     public String getCollaborationUuid() {
23         return collaborationUuid;
24     }
25
26     protected void setCollaborationUuid(String collaborationUuid) {
27         this.collaborationUuid = collaborationUuid;
28     }
29
30     public CollaborativeDataChange getCollaborativeDataChange() {
31         return collaborativeDataChange;
32     }
33
```

```
34      protected void setCollaborativeDataChange(
35              CollaborativeDataChange collaborativeDataChange) {
36          this.collaborativeDataChange = collaborativeDataChange;
37      }
38
39      public abstract void patch(Object patch, boolean invokeOnChange);
40
41      public abstract Object diff(CollaborativeData collaborativeData);
42
43      public abstract String toStringForDebug();
44
45      @Override
46      public abstract CollaborativeData clone();
47
48  }
```

Quellcode 4.7: Quellcode der Klasse CollaborativeData.

Der Quellcode 4.8 auf Seite 60 zeigt den Quellcode der Klasse *CollaborativeText*. Der zu synchronisierende Text wird in der Member Variable *text* gespeichert. Die *diff* und *patch* Methoden verwenden die Differential Synchronization Implementierung von Neil Fraser für Plaintext.

```
1   package at.reppeitsolutions.collaboration.model;
2
3   import at.reppeitsolutions.collaboration.misc.Constants;
4   import java.io.Serializable;
5   import java.util.LinkedList;
6   import name.fraser.neil.plaintext.diff_match_patch;
7   import org.jsoup.Jsoup;
8
9   /**
10   *
11   * @author Mathias Reppe <mathias.reppe@gmail.com>
12   */
13  public class CollaborativeText
14          extends CollaborativeData implements Serializable {
15
16      protected String text = "";
17      private diff_match_patch dmp = new diff_match_patch();
18
19      protected CollaborativeText() {
20          super(null, null);
21      }
22
23      public CollaborativeText(String collaborationUuid,
24              CollaborativeDataChange collaborativeDataChange) {
25          super(collaborationUuid, collaborativeDataChange);
26      }
27
28      public static final String LATEXBACKSLASH = "\\\\";
29      public static final String LATEXBREAK
30          = LATEXBACKSLASH + LATEXBACKSLASH + "\n";
31
32
33      public String getDecodedTextareaTextForLaTeX() {
34          return text.replaceAll("\n", LATEXBREAK);
35      }
```

```
36
37      public String getText() {
38          return text;
39      }
40
41      public void setText(String text) {
42          this.text = text;
43      }
44
45      @Override
46      public void patch(Object patch, boolean invokeOnChange) {
47          String oldText = text;
48          LinkedList<diff_match_patch.Patch> patch_fromText
49                  = (LinkedList<diff_match_patch.Patch>) dmp.patch_fromText(
50                          (String) patch);
51          Object[] patchResult = dmp.patch_apply(patch_fromText, text);
52          text = (String) patchResult[0];
53          if (collaborativeDataChange != null
54                  && !oldText.equals(text)
55                  && invokeOnChange) {
56              collaborativeDataChange.change();
57          }
58      }
59
60      @Override
61      public Object diff(CollaborativeData collaborativeData) {
62          String oldValue = getText();
63          String newValue = ((CollaborativeText) collaborativeData).getText();
64          LinkedList<diff_match_patch.Diff> diffs
65                  = dmp.diff_main(oldValue, newValue);
66          if (diffs.size() > 2) {
67              dmp.diff_cleanupSemantic(diffs);
68          }
69          LinkedList<diff_match_patch.Patch> patches
70                  = dmp.patch_make(oldValue, newValue, diffs);
71          return dmp.patch_toText(patches);
72      }
73
74      @Override
75      public CollaborativeData clone() {
76          CollaborativeText clone = new CollaborativeText(
77                  this.getCollaborationUuid(),
78                  this.getCollaborativeDataChange());
79          clone.setText(this.getText());
80          return clone;
81      }
82
83  }
```

Quellcode 4.8: Quellcode der Klasse CollaborativeText.

Der Quellcode 4.9 auf Seite 62 zeigt den Quellcode der Klasse *CollaborativeTextComponent*. Im Konstruktor der Klasse wird der Klasse der zugehörige JSF Renderer zugewiesen. Über die Methoden *getValue* und *setValue* kann auf den *CollaborativeText* zugegriffen werden. Die in dieser Klasse verfügbaren Getter und Setter definieren automatisch die Attribute des JSF Tags. Über dieses Attribut können beliebige CSS Anweisungen dem gerenderten HTML Tag hinzugefügt werden. Wie das Attribut *style* beim Rendern verwendet wird, ist im

Quellcode 4.12 auf Seite 65 ersichtlich.

```
package at.reppeitsolutions.collaboration.components;

import at.reppeitsolutions.collaboration.misc.Constants;
import at.reppeitsolutions.collaboration.model.CollaborativeText;
import at.reppeitsolutions.collaboration.renderer.CollaborativeTextRenderer;
import javax.faces.component.UIComponentBase;

/**
 *
 * @author Mathias Reppe <mathias.reppe@gmail.com>
 */

public abstract class CollaborativeTextComponent extends UIComponentBase {

    public CollaborativeTextComponent() {
        setRendererType(CollaborativeTextRenderer.RENDERTYPE);
    }

    @Override
    public String getFamily() {
        return Constants.FAMILY;
    }

    public CollaborativeText getValue() {
        return (CollaborativeText) getStateHelper().eval("value");
    }

    public void setValue(CollaborativeText value) {
        getStateHelper().put("value", value);
    }

    public String getStyle() {
        return (String) getStateHelper().eval("style");
    }

    public void setStyle(String style) {
        getStateHelper().put("style", style);
    }

}
```

Quellcode 4.9: Quellcode der Klasse CollaborativeTextComponent.

Der Quellcode 4.10 auf Seite 62 zeigt den Quellcode der Klasse *CollaborativeInputTextComponent*. Über die Annotation *FacesComponent* wird aus der Klasse ein JSF Tag. Das Attribut *tagName* der Annotation *FacesComponent* definiert den Namen des JSF Tags. Über die Annotation *ResourceDependencies* können die benötigten *JavaScript* Dateien am Client angegeben werden. Diese werden beim Rendern automatisch eingebunden. Die Klasse wird rein zur Definition des JSF Tags benötigt. Die benötigten Methoden sind alle in der Superklasse *CollaborativeTextComponent* definiert.

```
package at.reppeitsolutions.collaboration.components;

```

```
import at.reppeitsolutions.collaboration.misc.Constants;
import javax.faces.application.ResourceDependencies;
import javax.faces.application.ResourceDependency;
import javax.faces.component.FacesComponent;

/**
 *
 * @author Mathias Reppe <mathias.reppe@gmail.com>
 */
@FacesComponent(createTag = true,
        namespace = Constants.NAMESPACE,
        tagName = "collaborativeInputText")
@ResourceDependencies(value = {
    @ResourceDependency(library = "javax.faces", name = "jsf.js"),
    @ResourceDependency(library = "collaboration", name = "js/jquery.js"),
    @ResourceDependency(library = "collaboration", name = "js/collaboration.js"),
    @ResourceDependency(library = "collaboration", name = "js/diff_match_patch.js"
    )
})
public class CollaborativeInputTextComponent
        extends CollaborativeTextComponent {

}
```

Quellcode 4.10: Quellcode der Klasse CollaborativeInputTextComponent.

Der Quellcode 4.11 auf Seite 63 zeigt den Quellcode der Klasse *CollaborativeTextareaComponent*. Gleich wie die Klasse *CollaborativeInputTextComponent* wird die Klasse *CollaborativeTextareaComponent* nur zur Erstellung des JSF Tags benötigt. Einziger Zusatz der Klasse sind die Attribute *rows* und *cols*. Über diese Werte kann die Größe der gerenderten Textarea definiert werden.
Über die Superklasse *CollaborativeTextComponent* wird den Klassen *CollaborativeTextareaComponent* und *CollaborativeInputTextComponent* der Renderer *CollaborativeTextRenderer* zugewiesen. Wie der Renderer beim Rendern zwischen den beiden Klassen unterscheidet, wird im Quellcode 4.12 auf Seite 65 ersichtlich.

```
package at.reppeitsolutions.collaboration.components;

import at.reppeitsolutions.collaboration.misc.Constants;
import javax.faces.application.ResourceDependencies;
import javax.faces.application.ResourceDependency;
import javax.faces.component.FacesComponent;

/**
 *
 * @author Mathias Reppe <mathias.reppe@gmail.com>
 */
@FacesComponent(createTag = true,
        namespace = Constants.NAMESPACE,
        tagName = "collaborativeTextarea")
@ResourceDependencies(value = {
    @ResourceDependency(library = "javax.faces",
            name = "jsf.js"),
    @ResourceDependency(library = "collaboration",
```

```
19              name = "js/jquery.js"),
20      @ResourceDependency(library = "collaboration",
21              name = "js/collaboration.js"),
22      @ResourceDependency(library = "collaboration",
23              name = "js/diff_match_patch.js")
24  })
25  public class CollaborativeTextareaComponent
26          extends CollaborativeTextComponent {
27
28      public CollaborativeTextareaComponent() {
29          super();
30          setRows("" + 4);
31          setCols("" + 30);
32      }
33
34      public String getRows() {
35          return (String) getStateHelper().eval("rows");
36      }
37
38      public final void setRows(String rows) {
39          getStateHelper().put("rows", rows);
40      }
41
42      public String getCols() {
43          return (String) getStateHelper().eval("cols");
44      }
45
46      public final void setCols(String cols) {
47          getStateHelper().put("cols", cols);
48      }
49
50  }
```

Quellcode 4.11: Quellcode der Klasse CollaborativeTextareaComponent.

Der Quellcode 4.12 auf Seite 65 zeigt den Quellcode der Klasse *CollaborativeTextRenderer*. Der *CollaborativeTextRenderer* ist für das Rendern des HTML bzw. *JavaScript* Outputs für den Client verantwortlich. In der Methode *encodeBegin* wird zuerst die *collaborationWebsocketUrl* gesetzt. Dem Server ist genau bekannt, unter welchem Port er läuft bzw. unter welchem Context Path die Applikation deployed wurde. Ändert man Port oder Context Path is somit keine Anpassung der *WebSocket* URL in der JavaScript Datei *collaboration.js* notwendig.
Als nächstes wird in der Methode *encodeBegin* die *tmpSessionId* erzeugt und mit dieser wird der *CollaborativeText* im *CollaborationBackend* registriert. Dann wird der *JavaScript* Code gerendert. Über die *JavaScript* Methode *initCollaboration* wird falls noch nicht von einer anderen kollaborativen Komponente aufgebaut, die *WebSocket* Verbindung aufgebaut. Über die *JavaScript* Methode *initTextarea* wird über das *WebSocket* der Befehl *register* an den Server gesendet.
Zuletzt wird noch das HTML gerendert. Die Methode *encodeBegin* bekommt als Parameter eine *UIComponent*. Je nachdem ob diese *UIComponent* eine Instanz der Klasse *CollaborativeInputTextComponent* oder eine Instanz der Klasse *CollaborativeTextareaComponent* ist, wird ein HTML *input* Tag oder ein HTML *input* Tag

gerendert.

```
package at.reppeitsolutions.collaboration.renderer;

import at.reppeitsolutions.collaboration.CollaborationBackend;
import at.reppeitsolutions.collaboration.TmpSessionId;
import at.reppeitsolutions.collaboration.components.
    CollaborativeInputTextComponent;
import at.reppeitsolutions.collaboration.components.CollaborativeTextComponent;
import at.reppeitsolutions.collaboration.components.CollaborativeTextareaComponent
    ;
import at.reppeitsolutions.collaboration.misc.Constants;
import at.reppeitsolutions.collaboration.model.CollaborativeText;
import java.io.IOException;
import java.io.Writer;
import javax.faces.component.UIComponent;
import javax.faces.context.FacesContext;
import javax.faces.render.FacesRenderer;
import javax.faces.render.Renderer;

/**
 *
 * @author Mathias Reppe <mathias.reppe@gmail.com>
 */
@FacesRenderer(componentFamily = Constants.FAMILY,
        rendererType = CollaborativeTextRenderer.RENDERTYPE)
public class CollaborativeTextRenderer extends Renderer {

    public static final String RENDERTYPE = "CollaborativeTextRenderer";

    @Override
    public void encodeBegin(FacesContext context, UIComponent component)
            throws IOException {
        Writer writer = context.getResponseWriter();
        //Set WebSocket url if null
        writer.write("<script type=\"text/javascript\">\n");
        writer.write("if ("
                + "typeof window.collaborationWebsocketUrl === \"undefined\")\n"
                + "{\n"
                + "  window.collaborationWebsocketUrl = "
                + "'" + Constants.getCollaborationWebsocketUrl() + "';\n"
                + "}\n");
        writer.write("</script>\n");

        CollaborativeTextComponent collaborativeTextComponent
                = ((CollaborativeTextComponent) component);
        CollaborativeText collaborativeText
                = collaborativeTextComponent.getValue();
        String style = collaborativeTextComponent.getStyle();

        String collaborationUuid = collaborativeText.getCollaborationUuid();
        TmpSessionId tmpSessionId = new TmpSessionId();

        CollaborationBackend collaborationBackend
                = CollaborationBackend.getInstance();

        //Register tmpSessionId in CollaborationBackend
        collaborationBackend.registerCollaborationSession(
                collaborativeText, tmpSessionId);
        CollaborativeText currentServerVersion
                = (CollaborativeText) collaborationBackend.
```

```
58                  getCollaborativeData(collaborationUuid);
59
60          //Send register command via the WebSocket to the CollaborationBackend
61          //with tmpSessionId as parameter to set the WebSocket session
62          //reference in the CollaborationConnection
63          writer.write("<script type=\"text/javascript\">\n");
64          writer.write("$(function () {\n"
65                  + "    initCollaboration();\n"
66                  + "    initTextarea(\"" + collaborationUuid + "\",\n"
67                  + "      \"" + currentServerVersion.getDecodedTextareaTextHtml()
68                  + "      \",\n"
69                  + "      \"" + tmpSessionId + "\");\n"
70                  + "});\n");
71          writer.write("</script>\n");
72          String styleString = "";
73          if (style != null
74                  && !style.isEmpty()) {
75              styleString = " style=\"" + style + "\" ";
76          }
77          //Render HTML
78          if (component.getClass().equals(CollaborativeTextareaComponent.class)) {
79              CollaborativeTextareaComponent collaborativeTextareaComponent
80                      = (CollaborativeTextareaComponent) component;
81              writer.write("<textarea "
82                      + "id=\"pad" + collaborationUuid + "\" "
83                      + styleString
84                      + "cols=\""
85                      + collaborativeTextareaComponent.getCols() + "\" "
86                      + "rows=\""
87                      + collaborativeTextareaComponent.getRows() + "\">"
88                      + "</textarea>");
89          } else if (component.getClass().equals(
90                  CollaborativeInputTextComponent.class)) {
91              writer.write("<input "
92                      + "id=\"pad" + collaborationUuid + "\" "
93                      + styleString + " />");
94          }
95      }
96
97  }
```

Quellcode 4.12: Quellcode der Klasse CollaborativeTextRenderer.

Verwendung in externen Applikationen

Dieses Beispiel soll zeigen, wie einfach es durch die Implementierung von kollaborativen JSF Komponenten ist, beliebige JSF Applikationen mit kollaborativen Elementen zu erstellen. Im Beispiel wird ein kollaborativer HTML *input* Tag mit dem Attribut *type* und zugehörigen Wert *input* durch JSF gerendert. Abbildung 4.4 auf Seite 67 zeigt die Oberfläche der Demo Applikation. In der Abbildung werden zwei Browser nebeneinander dargestellt. Wird im ersten Browser ein Text getippt, wird dieser automatisch im 2. Browser dargestellt. Auch ein gleichzeitiges Tippen in beiden Browsern ist möglich.

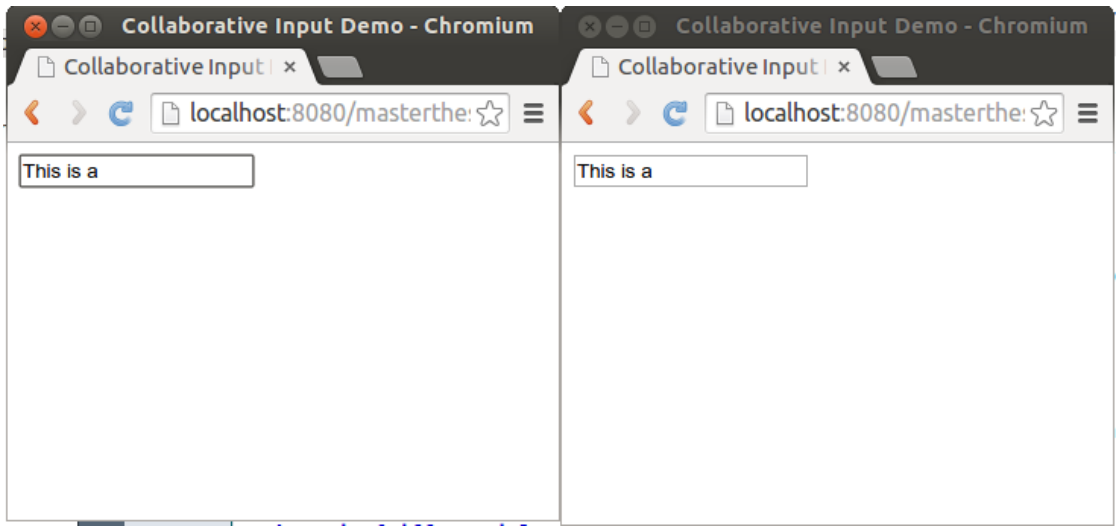


Abbildung 4.4: Oberfläche der Demo Applikation für kollaborative JSF Komponenten.

Die Demo Applikation besteht aus drei Teilen. Dem Frontend, implementiert durch die Datei *demo.xhtml*. Dem Backend, bestehend aus einer JSF Application Scoped Bean und der Klasse *DemoCollaborativeDataChange*. Der Quellcode 4.13 auf Seite 68 zeigt die Datei *demo.xhtml*. Das Frontend benötigt nur die Einbindung des Namespace und das Schreiben des JSF Tags. Der JSF Tag *collaborativeInputText* benötigt das Attribut *value*. Als Wert benötigt das Attribut *type* ein Objekt der Klasse *CollaborativeText*. Dieses Objekt wird in der JSF Application Scoped Bean *DemoBean* erzeugt und gehalten.

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/
    TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:rit="http://www.reppe-itsolutions.at">
    <h:head>
        <title>Collaborative Input Demo</title>
        <script src="js/libs/jquery/jquery.js"></script>
    </h:head>
    <h:body>
        <rit:collaborativeInputText value="#{demoBean.collaborativeText}" />
    </h:body>
</html>
```

Quellcode 4.13: Quellcode der xhtml Datei demo.xhtml.

Der Quellcode 4.14 auf Seite 68 zeigt den Quellcode der Klasse *DemoBean*. Die Bean, welche den *CollaborativeText* zur Verfügung stellt, muss Application Scoped sein. Beim Instanzieren der Klasse *CollaborativeText* muss eine ID vergeben werden. Diese ID darf nur einmal in der gesamten Applikation vorkommen. Dies ist am einfachsten über eine Application Scoped Bean zu realisieren. Als zweiten Parameter bekommt er Konstruktor der Klasse *CollaborativeText* eine Instanz einer Klasse, welche das Interface CollaborativeDataChange implementiert. In der Demo Applikation implementiert die Klasse *DemoCollaborativeDataChange* das Interface *CollaborativeDataChange*.

```
package at.reppeitsolutions;

import at.reppeitsolutions.collaboration.model.CollaborativeText;
import javax.enterprise.context.ApplicationScoped;
import javax.inject.Named;

/**
 *
 * @author Mathias Reppe <mathias.reppe@gmail.com>
 */
@Named(value = "demoBean")
@ApplicationScoped
public class DemoBean {

    private CollaborativeText collaborativeText;

    public DemoBean() {
        collaborativeText = new CollaborativeText("testid",
                new DemoCollaborativeDataChange());
    }

    public CollaborativeText getCollaborativeText() {
        return collaborativeText;
    }

}
```

Quellcode 4.14: Quellcode der Klasse DemoBean.

Der Quellcode 4.15 auf Seite 69 zeigt die Klasse *DemoCollaborativeDataChange*. Die Klasse implementiert das Interface *CollaborativeDataChange*. Auf Grund dessen gibt es eine *change* Methode. Diese wird jedes mal dann aufgerufen, wenn sich der Wert des Input Feldes ändert. Über das Singleton *CollaborationBackend* und der vergebenen ID für den *CollaborativeText* kann auf den aktuellen synchronisierten Wert des Input Feldes zugegriffen werden. Hier kann beispielsweise die Persistierung des Wertes in eine Datenbank erfolgen. Somit muss der Benutzer nach dem Ändern des Wertes nicht mehr auf einen Button zum Speichern klicken.

```
1  package at.reppeitsolutions;
2
3  import at.reppeitsolutions.collaboration.CollaborationBackend;
4  import at.reppeitsolutions.collaboration.model.CollaborativeDataChange;
5  import at.reppeitsolutions.collaboration.model.CollaborativeText;
6
7  /**
8   *
9   * @author Mathias Reppe <mathias.reppe@gmail.com>
10  */
11 public class DemoCollaborativeDataChange implements CollaborativeDataChange {
12
13     @Override
14     public void change() {
15         CollaborationBackend backend = CollaborationBackend.getInstance();
16         CollaborativeText collaborativeText
17                 = (CollaborativeText)backend.getCollaborativeData("testid");
18         String currentText = collaborativeText.getText();
19         System.out.println("Value of collaborative text changed. New value is: "
20                 + currentText);
21     }
22
23 }
```

Quellcode 4.15: Quellcode der Klasse DemoCollaborativeDataChange.

4.2 Kommunikations Modul

Dieses Kapitel beschreibt, wie realisiert wurde, dass der Benutzer mit anderen Benutzern über die Web-Oberfläche kommunizieren kann.

4.2.1 Video und Audio Chat

Damit Benutzer keine zusätzliche Voice-over-IP Software am Client installieren müssen, wurde der Video bzw. Audio Chat mit *WebRTC* realisiert. Das Herstellen eines Videostreams bei bestehender *WebRTC* Verbindung wurde anhand von Standard Beispielen für *WebRTC* realisiert [38]. Dies bedarf keiner besonderen Beschreibung. Wie im Kapitel 2.3.2 WebRTC auf Seite 26 beschrieben, benötigt man jedoch um eine Verbindung zwischen zwei Browsern herstellen zu können einen Signaling Server. Der Signaling Server sowie das Protokoll sind nicht standardisiert. Es folgt eine Beschreibung, wie der Signaling Server implementiert bzw. das Protokoll definiert wurde.

Der Signaling Server wurde mit Hilfe eines *JSR 356 WebSockets* realisiert. Die Abbildung 4.5 auf Seite 72 zeigt das Klassen Diagramm des Signaling Servers. Die Klasse *SignalingWebsocket* ist die Serverseitige Implementierung des *JSR 356 WebSockets*. Sie nimmt Nachrichten von den Clients entgegen und wird benachrichtigt, sobald ein Client eine Verbindung eröffnet bzw. schließt. Die Klasse *SignalingSingleton* ist das Kernstück des SignalingServers. Es ist wie der Name schon sagt nach dem Singleton Pattern implementiert. Die Klasse *SignalingSingleton* hält eine Map von *SignalingSession* Objekten. Der Key der Map ist die *signalingSessionUuid*. Für jedes Dokument existiert genau eine *SignalingSession*. Die *signalingSessionUuid* ist deshalb gleichzusetzen mit dem eindeutigen Namen eines Dokuments. Verwendet man das Kommunikations Modul alleinstehend ist die *signalingSessionUuid* gleichzusetzen mit dem eindeutigen Namen eines Chatrooms. Die Aufgabe der *SignalingSession* ist es, zwischen allen Benutzern die das Dokument editieren bzw. einem Chatroom beitreten eine *WebRTC* Verbindung herzustellen. Ist n die Benutzeranzahl entstehen also $\frac{n \cdot (n-1)}{2}$ Verbindungen. Ein Objekt der Klasse *SignalingConnection* stellt eine Verbindung zwischen zwei Browsern dar. Als Member Variablen besitzt die Klasse *SignalingConnection* für jeden Browser eine *Session*. Die *Session* wird vom *JSR 356 WebSocket* verwaltet. Über die *Session* können Daten an den Client gesendet werden.

Die Abbildung 4.6 auf Seite 73 zeigt den Protokoll Ablauf zwischen dem Signaling Server und zwei Clients. Zuerst meldet sich ein Client mit der Message

open am Server an. Als Parameter wird die *signalingSessionUuid* übergeben. Ist am Server noch keine *SignalingSession* für diese *signalingSessionUuid* instanziiert, wird dies durchgeführt. Nun wird gewartet, bis sich zusätzliche Clients mit der gleichen *signalingSessionUuid* am Server mit der Message *open* anmelden. Die folgenden Schritte werden für alle Client Paare durchgeführt. Das folgende Beispiel aus Abbildung 4.6 beinhaltet nur ein Paar.
Hat sich nun das Client Paar Client1 und Client2 am Server mit *open* angemeldet, wird eine *SignalingConnection* für das Paar erzeugt. Diese *SignalingConnection* fordert die Clients mit der Nachricht *createConnection* auf, eine *WebRTCConnection* herzustellen. Da jeder Client $n - 1$ *WebRTCConnections* besitzt, wird bei der Message *createConnection* als Parameter die *signalingConnectionUuid* übergeben, damit der Client die Verbindungen eindeutig zuordnen kann. Nach dem Erzeugen der *WebRTCConnection* benachrichtigen die Clients den Server mit der Message *webRTCConnectionCreated* darüber.
Hat der Server für beide Clients der *SignalingConnection* die Benachrichtigung *webRTCConnectionCreated* erhalten, beauftragt er mit der Message *startOffer* den ersten Client ein Angebot für seine Verbindungskonfiguration zu erstellen. Die Verbindungskonfiguration beinhaltet z.B. wie der Client mit dem Internet verbunden ist (STUN Server) und welche Streams (Audio, Video) bzw. Datenverbindung (RTCPeerConnection) die Verbindung enthält. Mit der Message *offer* und als Parameter die Verbindungskonfiguration wird die Verbindungskonfiguration über den Server an den 2. Client gesendet. Der 2. Client antwortet mit *offerAnswer* und fügt seine Verbindungskonfiguration hinzu. Die *offerAnswer* wird auch über den Server zurück an den 1. Client gesendet.
Da nun jeder Client die Verbindungskonfiguration des anderen Clients kennt, können die Clients nun peer-to-peer über *WebRTC* kommunizieren. Gib es Änderungen an der Verbindungskonfiguration (z.B. starten eines Videostreams), muss die Verbindungskonfiguration erneut ausgetauscht werden. Dieser Vorgang wiederholt sich bei jeder Änderung an der Verbindungskonfiguration in der *Negotiation Loop*. Ändert sich bei einem Client die Verbindungskonfiguration teilt er dies allen anderen Clients mit der Message *renegotiate* mit.

Adapter

Firefox und Google Chrome prefixen die Funktionen der *WebRTC* JavaScript API. Um die W3C Standard JavaScript Funktionen für WebRTC verwenden zu können, stellt Google einen Adapter bereit. Dieser Adapter stellt die W3C Standard JavaScript Funktionen zur Verfügung und ruft je nachdem mit welchem Browser die Funktionen aufgerufen werden die zugehörige Browserspezifische Funktion auf. Der Name dieses Adapters ist *adapter.js*. [27]

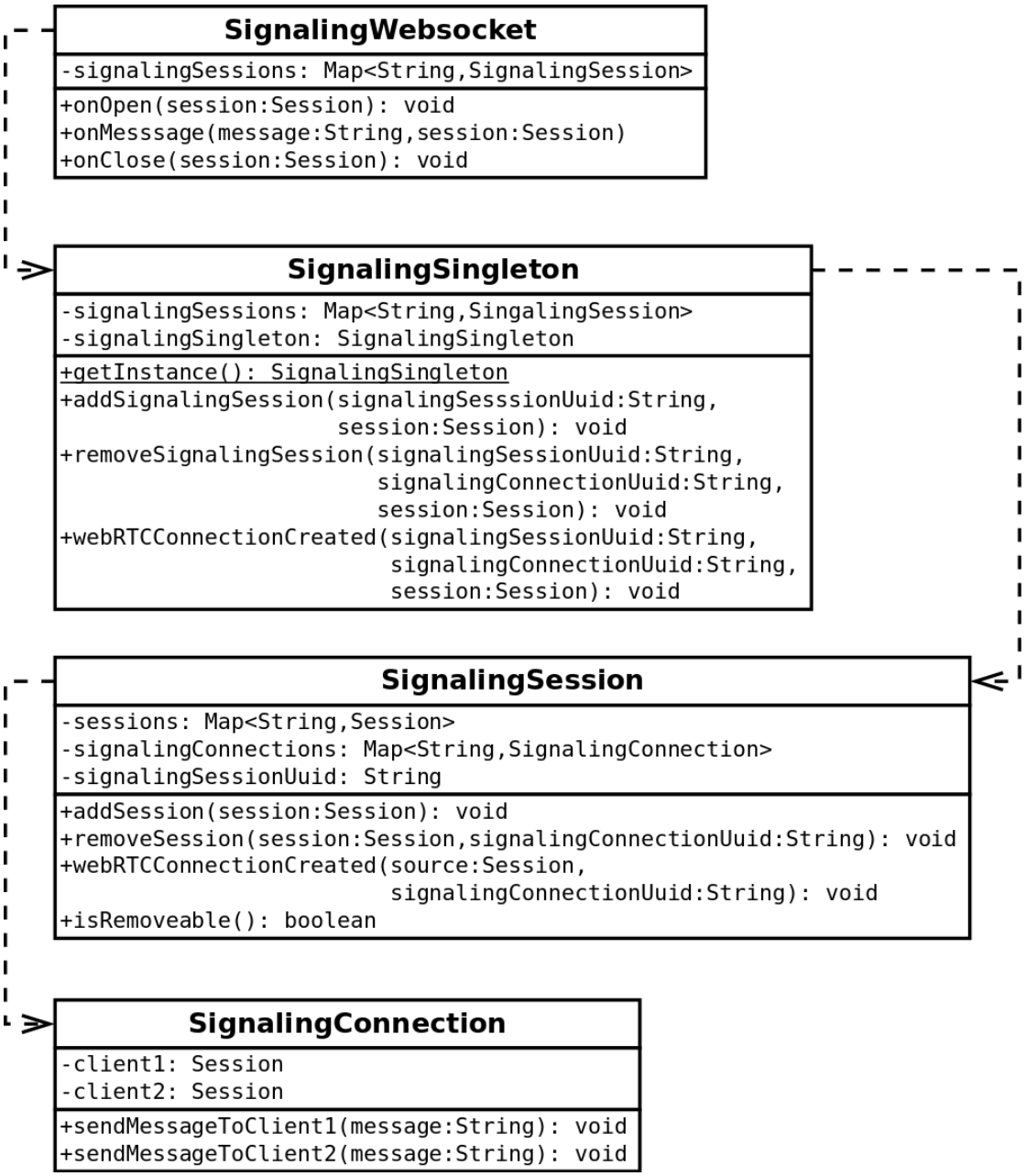


Abbildung 4.5: Klassen Diagramm des WebRTC Signaling Servers.

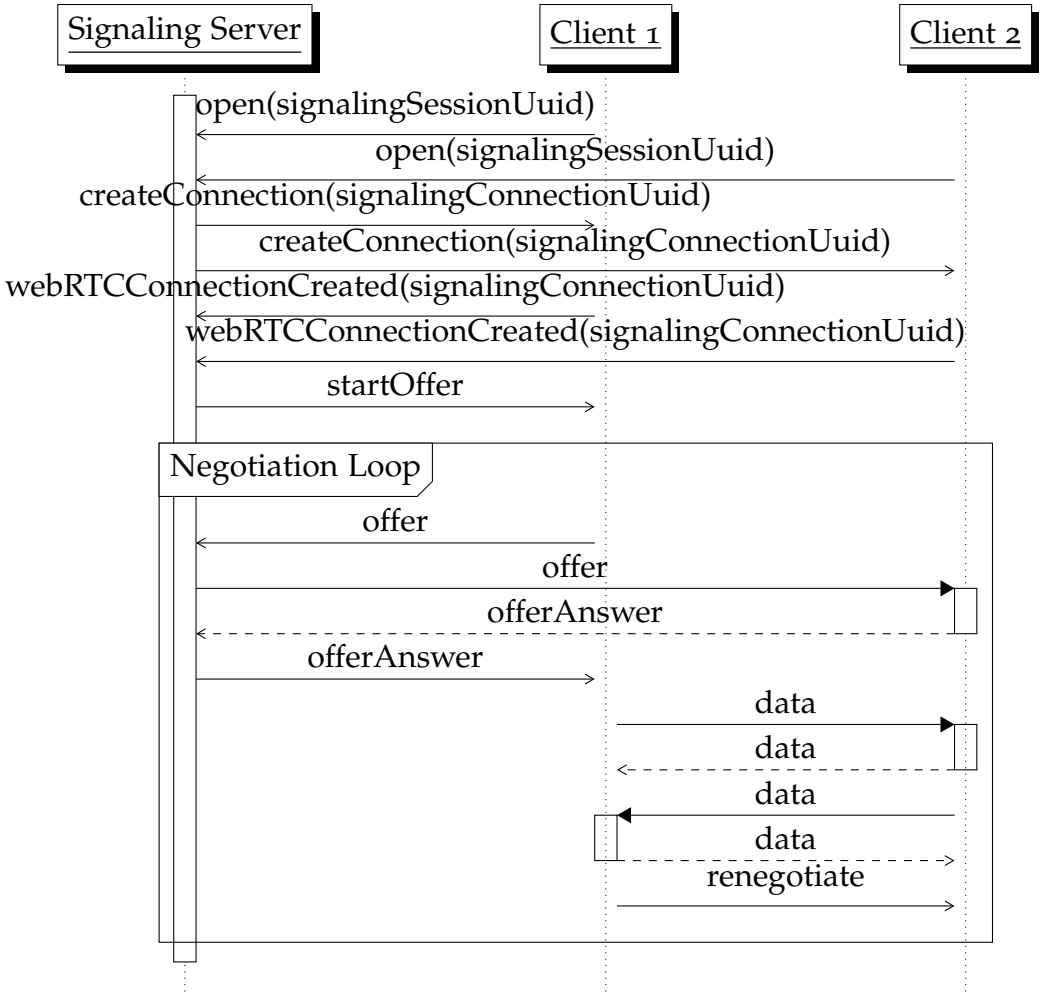


Abbildung 4.6: Protokoll Ablauf des WebRTC Signaling Servers.

4.3 Vorschau Modul

In diesem Kapitel wird beschrieben, wie realisiert wurde, dass der Benutzer immer eine aktuelle Vorschau der aktuellen PDF Version des Dokuments angezeigt bekommt.

4.3.1 PDF Darstellung

Um das PDF direkt auf der Web-Oberfläche darzustellen wird *pdf.js* verwendet. Bei *pdf.js* handelt es sich um eine komplett auf Basis von Web-Technologien entwickelten PDF Viewer. Entwickelt wurde die Software von Mozilla und kommt im Browser Firefox standardmäßig als PDF Viewer zum Einsatz. [42] Das PDF auf der Oberfläche aktualisiert sich automatisch und zeigt immer den aktuellsten PDF Output des LaTeX Dokuments.
Hierzu zeit Abbidlung 4.7 auf Seite 76 das Klassen Diagramm des Dokument Moduls mit den Komponenten für die PDF Erstellung am Server. Im Klassendiagramm werden auf Grund der Übersichtlichkeit nur Attribute und Methoden angezeigt, welche für die PDF Darstellung relevant sind. Das Interface *UIComponent* und die abstrakte Klasse *UIComponentBase* sind Teil von *JSF*. Alle Komponenten des Editors müssen von der abstrakten Klasse *DocumentComponentBase* ableiten. Über den einzigen Konstruktor wird sichergestellt, dass jede Komponente eine Referenz auf die Wurzel des Dokuments besitzt (Klasse *DocumentRoot*). Die Klasse *RootDocument* besitzt eine Member Variable vom Typ *CollaborativeDataChange*. Im Konstruktor der Klasse *DocumentRoot* wird der Variable *collaborativeDataChange* eine Instanz der Klasse *RenderLatex* zugewiesen. Pro Dokument gibt es nur eine *DocumentRoot* und dadurch auch nur eine Instanz der Klasse *RenderLatex*. Beliebige Änderungen an den verschiedenen Dokument Komponenten (*DocumentComponentX*) können somit über *getDocumentRoot().getCollaborativeDataChange().change()* eine Änderung am Dokument bekannt geben.
In der Klasse *RenderLatex* in der Methode *change* wird aus der *DocumentRoot* und dessen *children* mit LaTeX ein PDF des Dokuments erzeugt und im Anschluss für die Auslieferung an die Clients bereitgestellt. Änderungen können durch mehrere Clients über verschiedene Threads gleichzeitig passieren. Der Bereich in dem das PDF durch *latex* erzeugt wird, wird über die Variable *inProgress* geschützt. Wird zur Zeit ein PDF von einem Thread erzeugt, überspringen andere Threads das Erzeugen. Beendet der derzeit erzeugende Thread seine Arbeit, bekommt er über die Variable *threadTried* mit, dass andere Threads ebenfalls Änderungen gemeldet haben. Ist dies der Fall, startet er die PDF Erzeugung erneut. Das

Erzeugen des PDF's wird in einem eigenem Thread gestartet, um die restlichen Funktionen der Applikation während der Erstellung nicht zu blockieren. Nach dem erfolgreichen Erstellen eines PDF's werden alle Clients informiert, dass eine neue Version des PDF's verfügbar ist. Anschließend fordern die Clients die neue Version des PDF's über ein Servlet an.

4.3.2 PDF Übertragung

Beim Anfordern des neuen PDF's durch die Clients stellt sich die Frage, ob bei jeder Änderung das vollständige PDF heruntergeladen werden soll oder nur die veränderten Seiten. Wird die Variante mit den einzelnen Seiten gewählt, muss zusätzlich von jeder Seite am Server ein Hashwert erzeugt werden. Bei der Benachrichtigung an den Client, dass ein neues PDF verfügbar ist sendet der Server eine Liste der Hashwerte mit. Der Client kennt seine aktuell dargestellten Seiten mit den zugehörigen Hashwerten und fordert nur die Seiten mit veränderten Hashwerten beim Server an. Es stellt sich die Frage, ab wie vielen Seiten und bei welcher Übertragungsgeschwindigkeit zwischen Server und Client es performanter ist nur die einzelnen Seiten herunterzuladen anstatt des PDF's. Die Frage ist ab wann der Mehraufwand des Hashens durch den geringeren Aufwand der zu übertragenden Daten ausgeglichen wird.
Es gibt drei Schritte die bei der Auslieferung an den Client ausgeführt werden. Diese werden in der folgenden Aufzählung dargestellt:

1. PDF rendern
   - Wird beim Ausliefern der veränderten Seiten sowie beim Ausliefern des gesamten PDF's benötigt.
   - $t_{latex} = t_{latex_{base}} + t_{latex_{page}} * pages = 277ms + 4.32ms * pages$
2. PDF splitten und Hashwerte erzeugen
   - Nur nötig beim Ausliefern der veränderten Seiten.
   - $t_{splithash} = t_{splithash} * pages = 3.32ms * pages$
3. PDF an den Client übertragen
   - Beim Ausliefern des gesamten PDF's:
     - $t_{send_{full}} = \frac{size_{page} * pages}{speed} = \frac{25kB * pages}{speed}$
   - Beim Ausliefern der veränderten PDF Seiten:
     - $t_{send_{partial}} = \frac{size_{page} * users}{speed} = \frac{25kB * users}{speed}$
     - Hier wird angenommen das jeder Benutzer Änderungen an einer anderen Seite durchführt.

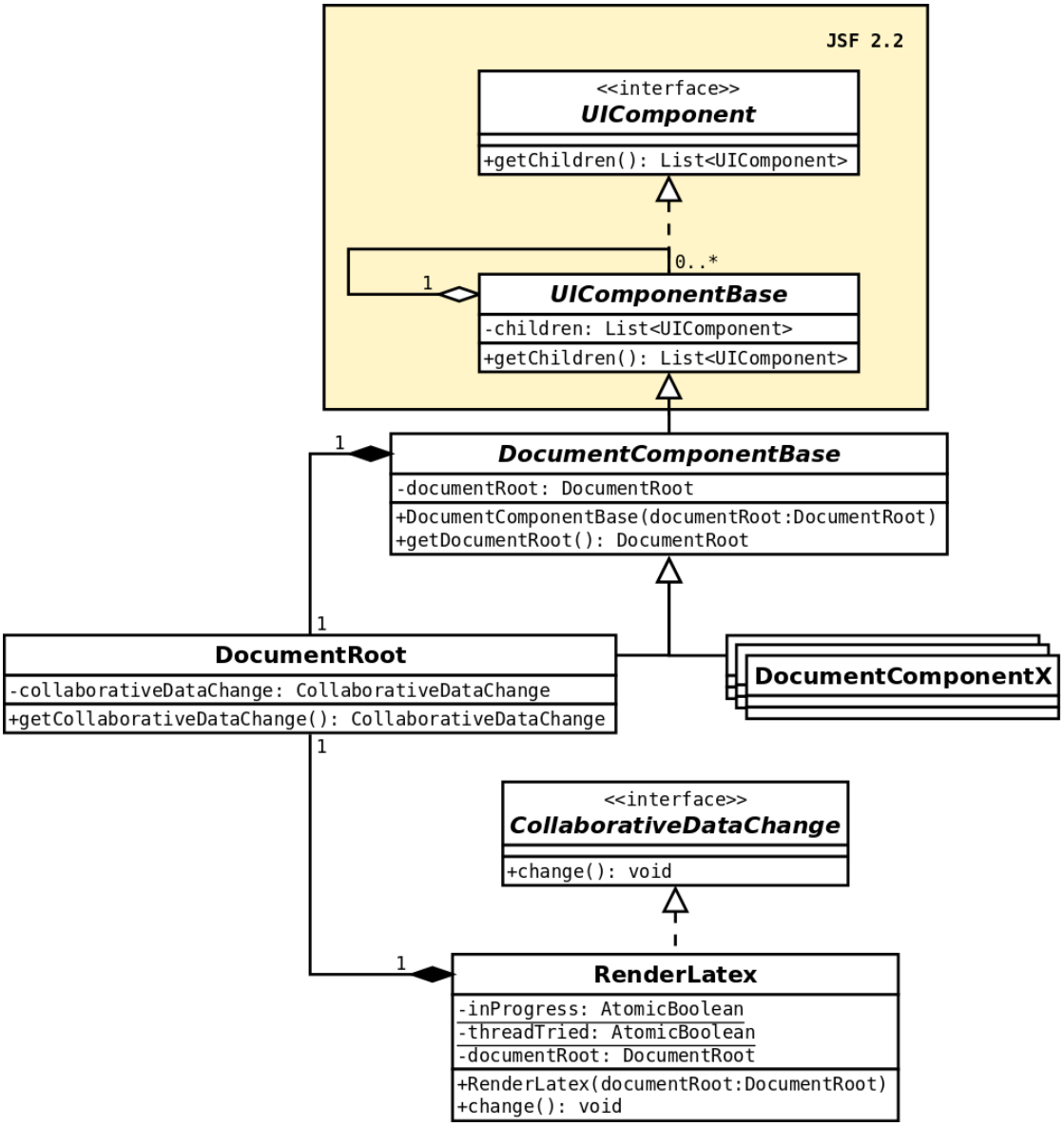


Abbildung 4.7: Auszug des Klassen Diagramms des Dokument Moduls mit den Komponenten für die PDF Darstellung.

users	speed (kbps)	pages
1	512	1.01
1	1024	1.02
1	2048	1.04
1	4096	1.07
2	512	2.02
2	1024	2.04
2	2048	2.07
2	4096	2.15
3	512	3.03
3	1024	3.05
3	2048	3.11
3	4096	3.22

Tabelle 4.1: Evaluierung welches Verfahren zur Auslieferung des PDF's geeigneter ist.

Die angegebenen Zeiten für die Schritte wurden über Durchschnittswerte mit verschieden großen PDF's per Software ermittelt. Die Anzahl der Tests für die Ermittlung des Durchschnitts betrug 1000 und die Seitenanzahl der PDF's variierte zufällig zwischen 10 und 200. Das PDF enthielt eine Titelseite mit Inhaltsverzeichnis sowie mit Überschriften und Absätzen gefüllte Seiten. Im Test PDF waren keine Bilder enthalten. Die Zeiten wurden auf einem Rechner mit 4GB Arbeitsspeicher und einer Intel(R) Core(TM) i5 CPU M560 @ 2.67GHz ermittelt. Bei der Zeit t_{latex} gibt es eine Basiszeit ($t_{latex_{base}}$) und einen Zeit in Abhängigkeit der Seiten ($t_{latex_{page}}$). Die Basiszeit beinhaltet Zeiten die unabhängig von der Seitenanzahl sind (z.B. *latex* Prozess starten und beenden). Für das Ausliefern des gesamten PDF's ergibt sich folgende Gesamtzeit:

$$t_{full} = t_{latex} + t_{send_{full}} \tag{4.1}$$

Für das Ausliefern der veränderten Seiten ergibt sich folgende Gesamtzeit:

$$t_{partial} = t_{latex} + t_{splithash} + t_{send_{partial}} \tag{4.2}$$

Setzt man $t_{full} = t_{partial}$ erhällt man die Seitenanzahl, ab welcher eine Auslieferung der veränderten Seiten performanter ist (*pages*). Abhängig ist diese Gleichung von der aktiven Benutzeranzahl (*users*) und von der Übertragungsgeschwindigkeit zwischen Server und Client (*speed*). Tabelle 4.1 auf Seite 77 zeigt das Ergebnis für verschiedene Benutzeranzahlen und Geschwindigkeiten.

Allgemein kann man aussagen, sobald die Benutzeranzahl gleich oder größer der Seitenanzahl ist, ist eine Auslieferung des gesamten PDF's sinnvoller, da anzunehmen ist, dass jede Seite verändert wird.

5 Ausblick

Diese Arbeit hat als Output einen kollaborativen Web-Editor zum Erstellen von wissenschaftlichen Arbeiten. In diesem Kapitel werden die Einsatzmöglichkeiten des Editors aufgezeigt. Des Weiteren sind durch die Implementierung des Editors Module entstanden, welche auch in anderen Web-Anwendungen integriert werden können. Dieses Kapitel beschreibt auch Module, welche in beliebigen Applikationen zum Einsatz kommen könnten.

5.1 Integration in SOP-Guard

Der *SOP-Guard* [12] ist ein von der *aiti-works EDV-Systemlösungen GmbH* [11] entwickeltes Dokumenten-Management System für die Pharmabranche. Im *SOP-Guard* ist es unter anderem möglich auf einer Web-Oberfläche Dokumente zu Erstellen. Im Anschluss durchläuft das Dokument einen speziellen Workflow gemäß den Richtlinien der Pharmabranche. Dabei müssen mehrere Personen das Dokument lesen und für in Ordnung befinden, damit z.B. das Dokument als Leitfaden für einen Produktionsprozess eingesetzt werden kann. Wird ein Teil des Dokuments nicht für in Ordnung befunden, muss der Verfasser des Dokuments die reklamierten Textteile erneut überarbeiten. Für das Reklamieren einzelner Textteile wäre die Kommentarfunktion des Editors bestens geeignet. Bekommt der Verfasser ein Dokument zurück gesendet, kann er direkt bei den Textteilen die aufgezeigten Fehler lesen und korrigieren.
Ein Dokument im *SOP-Guard* kann auch von mehreren Personen erstellt werden. Im SOP-Guard gibt es einen Verfasser und beliebig viele Überarbeiter eines Dokuments. Derzeit schließt der Verfasser seine Arbeiten ab und sendet das Dokument im Anschluss sequentiell an beliebig viele Überarbeiter. Überarbeiter können derzeit nicht gleichzeitig am Dokument arbeiten. Durch die Integration des Editors kann das Erstellen des Dokuments durch den Verfasser und die Überarbeiter erheblich beschleunigt werden, da ein Zuteilen des Dokuments zu den verschiedenen Personen entfällt.
Sollten sich die verschiedenen an der Erstellung beteiligten Personen an verschiedenen Firmenstandorten befinden, wäre auch das Kommunikations Modul

eine erhebliche Steigerung des Kundennutzen, des *SOP-Guards*. Virtuelle Diskussionen direkt über die Oberfläche des *SOP-Guard* wären ohne die Installation von zusätzlicher Software möglich.
Eine spezielle Herausforderung bei der möglichen Integration des Projekts in *SOP-Guard* wäre die Validierung gemäß *GAMP5* [30], um den Regularien der Pharmabranche zu entsprechen. Die Validierung nach *GAMP5* ist jedoch auch als Chance zu sehen, ein anerkanntes Qualitätsgütesiegel für diese Arbeit zu erhalten.

5.2 JSF Kollaboration Framework

Die einzelnen kollaborativen Web-Komponenten die im Zuge dieser Arbeit entstanden sind wurden als *JSF 2.2* Komponenten implementiert. Auf Grund der Implementierung als *JSF* Komponenten ergab sich der Vorteil der einfachen Wiederverwendbarkeit. Die Zusammenfassung aller entstandenen Komponenten zu einem Framework, welches in eine beliebige *JSF* Applikation integrierbar ist würde das Implementieren von kollaborativen Anwendungen mit JSF erheblich erleichtern. Zum Zeitpunkt der Verfassung dieser Arbeit war kein *JSF Kollaboration Framework* verfügbar.
Primefaces [45] ist ein populäres Oberflächenframework [32] für *JSF*. Die Popularität von Primefaces könnte genutzt werden um auch das *JSF Kollaboration Framework* bekannt zu machen. Das Projekt *Primefaces Extensions* [7] ist Open-Source und wird von der Community weiterentwickelt. In den *Primefaces Extensions* werden auf der Basis von Primefaces beliebige *JSF* Komponenten veröffentlicht. Über diese Community könnte das *JSF Kollaboration Framework* veröffentlicht und weiterentwickelt werden.

5.3 Online Nachhilfe

Verbindet man das JSF Kollaboration Framework mit dem Kommunikations Modul kann ein weiteres Produkt aus diesen Komponenten entwickeln werden. Es wäre möglich Nachhilfestunden online zu geben. Über das Kommunikations Modul könnte der Schüler mit dem Nachhilfelehrer kommunizieren als würden die Beiden nebeneinander sitzen. Mit dem JSF Kollaboration Framework wäre es möglich eine Oberfläche zu schreiben, die es jederzeit dem Nachhilfelehrer ermöglicht z.B. zu sehen wie der Schüler seine Aufgaben löst. Begeht der

Schüler bei einer Aufgabe einen Fehler kann der Nachhilfelehrer direkt einschreiten und mit dem Schüler besprechen warum die Antwort für eine Aufgabe falsch ist. Auf Grund dieser interaktiven Zusammenarbeit zwischen Schüler und Nachhilfelehrer wäre eine Online Nachhilfestunde qualitativ nicht schlechter als eine echte Nachhilfestunde. Dies wäre natürlich auf Grund von Tests zu beweisen.
Online Nachhilfe hätte jedoch weitere Vorteile. Der Schüler müsste nicht zum Lerninstitut anreisen. Dies wäre besonders für Schüler die ihren Wohnsitz in entlegenen Gebieten haben ein Vorteil. Es wäre möglich kurzfristig eine Nachhilfestunde zu nehmen. Der Schüler meldet z.B., dass er heute noch eine Nachhilfestunde benötigt. Steht hinter dem Online Nachhilfe System eine ganze Community von Nachhilfelehrern ist es wahrscheinlich, dass ein Nachhilfelehrer Zeit hat. Nach jeder Nachhilfestunde könnte man dem Schüler ein kurzes Feedback über den Nachhilfelehrer ausfüllen lassen. Dadurch könnte man die Qualität der Nachhilfestunden verbessern, da die Leistung der Nachhilfelehrer bewertet werden kann.
Der Markt in Österreich für Online Nachhilfe ist sehr klein aber rasch wachsend. In anderen Ländern wie z.B. den USA oder Südkorea ist der Markt schon seit mehreren Jahren groß. [1] Auf Grund dessen, dass ein Anbieter seine Nachhilfelehrer nicht global einsetzen kann, da sie den Landessprachen nicht mächtig sind können Anbieter aus z.B. den USA nicht so einfach den Markt in Österreich erobern. Ein Online Nachhilfe Produkt mit einer intuitiven Oberfläche, Tools zur Wissensweitergabe und guter Vermarktung könnte sich in Österreich und anderen Ländern mit wachsendem Online Nachhilfemarkt bewähren.

6 Resümee

Durch diese Arbeit ist der Grundstein gelegt worden, um ein Produkt zu implementieren, welches einen kollaborativen LaTeX Abstraktions Editor als Kernkomponente besitzt. Rund um diese Kernkomponente sind noch Implementierungen notwendig. Neue Benutzer müssen sich registrieren können. Benutzer müssen ihre Dokumente verwalten können. Es müssen LaTeX Vorlagen für die Benutzer zur Verfügung stehen. Benutzer müssen die Möglichkeit haben, dass Dokument mit anderen Verfassern zu teilen bzw. weitere Verfasser einladen zu können. Dies sind nur einige wenige Anforderungen. Gespräche mit Professoren, wissenschaftlichen Mitarbeitern und Studenten werden sicher noch unzählige nützliche Features hervorbringen.
Fragt man die Anwender nach ihren Wünschen entsteht ein maßgeschneidertes Produkt. Ein solches Produkt ist jedoch nicht automatisch innovativ. Zusätzlich ist bei der Konzeption des Produkts Querdenken gefragt. Steve Jobs hat in diesem Zusammenhang gesagt: *„Die Leute wissen gar nicht, was sie wollen, bis man es ihnen zeigt. Deshalb verlasse ich mich nicht auf Marktforschung. Unsere Aufgabe ist es, Dinge zu lesen, die noch gar nicht geschrieben sind.“* [29].
Das Entwickeln des Produkts rundherum ist keine große technische Herausforderung mehr. Die Herausforderung dabei besteht eher darin die Wünsche und Anforderungen der Benutzer zu erfüllen und auch innovativ zu sein. Die Usability und die Einfachheit der Oberfläche werden entscheidend sein, ob Benutzer diese neue Möglichkeit wissenschaftliche Arbeiten zu verfassen auch nutzen werden oder weiterhin ihrem Textverarbeitungssystemen treu bleiben.
Die wohl größte Herausforderung wird die Vermarktung des Produkts darstellen. Wie breit die Verwendung des technischen Fundaments also genauer gesagt der Output dieser Arbeit sein wird, hängt wohl vom Erfolg einer ausgeklügelten Marketing Strategie ab.

Appendix

Abbildungsverzeichnis

Tabellenverzeichnis

Literatur

[1] APA. *Online-Nachhilfe kommt in Österreich nur zögerlich vom Fleck*. 2014. URL: http://derstandard.at/1399507233580/Online-Nachhilfe-kommt-in-Oesterreich-nur-zoegerlich-vom-Fleck (besucht am 22. 07. 2015) (siehe S. 81).

[2] Inc Artifex Software. *dvipdf(1) - Linux man page*. 2013. URL: http://linux.die.net/man/1/dvipdf (besucht am 16. 06. 2015) (siehe S. 17).

[3] Inc. Authorea. *ShareLaTex*. 2015. URL: https://www.authorea.com/ (besucht am 20. 04. 2015) (siehe S. 4, 6).

[4] Tobias Berndt. *LaTeX Der typographische Einstieg*. first edition. ADDISON-WESLEY, 2008. Kap. Digitaler Textsatz (siehe S. 7, 11–13, 16).

[5] Klaus Braune, Joachim Lammarsch und Marion Lammarsch. „LaTeX Basissystem, Layout Formelsatz“. In: Springer, 2006. Kap. Gerätetreiber. (Besucht am 17. 06. 2015) (siehe S. 16).

[6] P. Bryan, Salesforce.com und M. Nottingham. *RFC6902 - JavaScript Object Notation (JSON) Patch*. 2013. URL: https://tools.ietf.org/html/rfc6902 (besucht am 31. 07. 2015) (siehe S. 43).

[7] Open Source Community. *PrimeFaces Extensions*. 2015. URL: http://primefaces-extensions.github.io/ (besucht am 23. 07. 2015) (siehe S. 80).

[8] Institut für Demoskopie Allensbach und Institut für Publizistik der Universität Mainz. *Welchen Anteil haben Text, Erscheinungsbild des Redners, Betonung und Gestik an der Gesamtwirkung eines Vortrags?* Techn. Ber. 2007 (siehe S. 35).

[9] Alexis Deveria und Lennart Schoors. *Can i use Web Sockets?* 2015. URL: http://caniuse.com/#feat=websockets (besucht am 23. 06. 2015) (siehe S. 25).

[10] Alexis Deveria und Lennart Schoors. *Can i use WebRTC?* 2015. URL: http://caniuse.com/#feat=WebRTC (besucht am 07. 07. 2015) (siehe S. 26).

[11] aiti-works EDV-Systemlösungen GmbH. *aiti-works*. 2015. URL: http://www.aiti.at/ (besucht am 16. 07. 2015) (siehe S. 79).

[12] aiti-works EDV-Systemlösungen GmbH. *SOP-Guard*. 2015. URL: http://www.sop-guard.com (besucht am 16. 07. 2015) (siehe S. 79).

[13] C. A. Ellis und S. J. Gibbs. „Concurrency Control in Groupware Systems“. In: *SIGMOD Rec.* 18.2 (Juni 1989), S. 399–407. ISSN: 0163-5808. DOI: 10.1145/66926.66963. URL: http://doi.acm.org/10.1145/66926.66963 (siehe S. 18, 20).

[14] I. Fette u. a. 2011. URL: https://www.websocket.org/quantum.html (besucht am 23. 06. 2015) (siehe S. 25).

[15] I. Fette u. a. *The WebSocket Protocol*. 2011. URL: http://tools.ietf.org/html/rfc6455 (besucht am 23. 06. 2015) (siehe S. 24, 25).

[16] R. Fielding u. a. *Hypertext Transfer Protocol – HTTP/1.1*. 1999. URL: https://tools.ietf.org/html/rfc2616 (besucht am 22. 06. 2015) (siehe S. 23).

[17] Neil Fraser. *Diff, Match and Patch libraries for Plain Text*. 2012. URL: https://code.google.com/p/google-diff-match-patch/ (besucht am 31. 07. 2015) (siehe S. 42).

[18] Neil Fraser. *Differential Synchronization*. 2009. URL: https://neil.fraser.name/writing/sync/eng047-fraser.pdf (besucht am 08. 07. 2015) (siehe S. 21, 22).

[19] Neil Fraser. *Plain Text vs. Structured Content*. 2010. URL: https://code.google.com/p/google-diff-match-patch/wiki/Plaintext (besucht am 31. 07. 2015) (siehe S. 42).

[20] Bibliographisches Institut GmbH. *Duden - Textverarbeitung*. 2015. URL: http://www.duden.de/rechtschreibung/Textverarbeitung (besucht am 15. 06. 2015) (siehe S. 12).

[21] Bibliographisches Institut GmbH. *Duden - Textverarbeitung*. 2015. URL: http://www.duden.de/rechtschreibung/Textverarbeitungssystem (besucht am 15. 06. 2015) (siehe S. 12).

[22] Google. *Google Docs*. 2015. URL: http://www.google.com/docs/about/ (besucht am 20. 04. 2015) (siehe S. 2, 4, 6).

[23] Peter Leo Gorski, Luigi Lo Iacono und Hoai Viet Nguyen. *WebSockets - Moderen HTML5-Echtzeitanwendungen entwickeln*. Hanser, 2015 (siehe S. 24).

[24] TeX Users Group. *TeX Live*. 2015. URL: https://www.tug.org/texlive/ (besucht am 20. 04. 2015) (siehe S. 16).

[25] Florian Heinz. *Schlechter Textsatz – ein stetes Ärgernis dank Word*. 2012. URL: http://www.theangrynerd.de/schlechter-textsatz-ein-stetes-argernis-dank-word/ (besucht am 15. 06. 2015) (siehe S. 14, 15).

[26] Ian Hickson und Inc. Google. *The WebSocket API*. 2012. URL: http://www.w3.org/TR/websockets/ (besucht am 23. 06. 2015) (siehe S. 24).

[27] Google Inc. *WebRTC - Interop Notes*. 2014. URL: http://www.webrtc.org/web-apis/interop (besucht am 26. 07. 2015) (siehe S. 71).

[28] Google Inc. *WebRTC project homepage*. 2014. URL: http://www.webrtc.org/ (besucht am 07. 07. 2015) (siehe S. 26).

[29] Walter Isaacson. *Steve Jobs: Die autorisierte Biografie des Apple-Gründers*. C. Bertelsmann, 2011 (siehe S. 83).

[30] ISPE. *A Risk-Based Approach to Compliant GxP Computerized Systems*. ISPE, 2007 (siehe S. 80).

[31] Henrik Joreteg. *simpleWebRTC*. 2015. URL: https://simplewebrtc.com/ (besucht am 07. 07. 2015) (siehe S. 28).

[32] Josh Juneau. *PrimeFaces in the Enterprise*. 2014. URL: http://www.oracle.com/technetwork/articles/java/java-primefaces-2191907.html (besucht am 23. 07. 2015) (siehe S. 80).

[33] Anne van Kesteren u. a. *XMLHttpRequest Level 1*. 1999. URL: https://tools.ietf.org/html/rfc2616 (besucht am 22. 06. 2015) (siehe S. 23, 24).

[34] Donald E. Knuth. *The DVItype processor*. 1995. URL: http://texdoc.net/texmf-dist/doc/generic/knuth/texware/dvitype.pdf (besucht am 16. 06. 2015) (siehe S. 16).

[35] LaTex Project. *LaTex -Project home page*. 2015. URL: http://latex-project.org/ (besucht am 20. 04. 2015) (siehe S. 1, 11).

[36] Philipp Lehman u. a. *The Biblatex Package*. 2015. URL: http://mirror.easyname.at/ctan/macros/latex/contrib/biblatex/doc/biblatex.pdf (besucht am 13. 07. 2015) (siehe S. 33).

[37] Writelatex Limited. *Google Docs*. 2015. URL: https://www.overleaf.com/ (besucht am 20. 04. 2015) (siehe S. 4, 6).

[38] Salvatore Loreto und Simon Pietro Romano. *Real-Time Communication with WebRTC: Peer-to-Peer in the Browser*. O'Reilly Media, Inc., 2014 (siehe S. 70).

[39] Rob Manson. *Getting Started with WebRTC*. Packt Publishing, 2013 (siehe S. 26–28).

[40] Jelica Nejasmic Markus Knauff. *An Efficiency Comparison of Document Preparation Systems Used in Academic Research and Development*. Dez. 2014. URL: http://dx.doi.org/10.1371/journal.pone.0115069 (besucht am 03. 06. 2015) (siehe S. 7, 8).

[41] Mojarra. *JSF 2.2 Javascript Documentation*. 2013. URL: https://javaserverfaces.java.net/docs/2.2/jsdocs/symbols/jsf.ajax.html (besucht am 30. 07. 2015) (siehe S. 40).

[42] Mozilla und individual contributors. *PDF.js*. 2012. URL: https://mozilla.github.io/pdf.js/ (besucht am 28. 07. 2015) (siehe S. 74).

[43] ownCloud.org. *ownCloud*. 2015. URL: https://owncloud.org/ (besucht am 20. 04. 2015) (siehe S. 4, 6).

[44] V. Pimentel und B.G. Nickerson. „Communicating and Displaying Real-Time Data with WebSocket". In: *Internet Computing, IEEE* 16.4 (Juli 2012), S. 45–53. ISSN: 1089-7801. DOI: 10.1109/MIC.2012.64 (siehe S. 24, 25).

[45] PrimeTek. *PrimeFaces - Ultimate JSF Framework*. 2015. URL: http://www.primefaces.org/ (besucht am 23. 07. 2015) (siehe S. 80).

[46] Roel und Job Zinkstok. *zink typography - latex*. 2015. URL: http://www.zinktypografie.nl/latex.php?lang=en (besucht am 15. 06. 2015) (siehe S. 13, 15).

[47] Tomas Rokicki. *dvips(1) - Linux man page*. 2013. URL: http://linux.die.net/man/1/dvips (besucht am 16. 06. 2015) (siehe S. 17).

[48] S. Sarin und I. Greif. „Computer-Based Real-Time Conferencing Systems". In: *Computer* 18.10 (Okt. 1985), S. 33–45. ISSN: 0018-9162. DOI: 10.1109/MC.1985.1662711 (siehe S. 18).

[49] Christian Schenk. *MiKTeX*. 2015. URL: http://miktex.org/ (besucht am 20. 04. 2015) (siehe S. 16).

[50] John Selbie. *STUNTMAN - Open source STUN server software*. 2012. URL: http://www.stunprotocol.org/ (besucht am 07. 07. 2015) (siehe S. 27).

[51] Andrii Sergiienko. *WebRTC Cookbook*. Packt Publishing, 2015 (siehe S. 27, 28).

[52] Amelie Solbrig. „Zweisprachige Mikrotypografie". Magisterarb. HTWK Leipzig (FH), 2008. URL: http://www.verlagsherstellung.de/fileadmin/fbmedien_bmp/downloads/Abschlussarbeiten/Zweisprachige_Mikrotypografie_Amelie_Solbrig_VH-02.pdf (siehe S. 15).

[53] BarD Software s.r.o. *PAPEERIA*. 2015. URL: http://papeeria.com (besucht am 20. 04. 2015) (siehe S. 4, 6).

[54] Statistik Austria. *Statistiken - Universitaeten, Studium*. 2015. URL: http://www.statistik.at/web_de/statistiken/bildung_und_kultur/formales_bildungswesen/universitaeten_studium/ (besucht am 16. 05. 2015) (siehe S. 1).

[55] Mark Stefik u. a. „Beyond the Chalkboard: Computer Support for Collaboration and Problem Solving in Meetings“. In: *Commun. ACM* 30.1 (Jan. 1987), S. 32–47. ISSN: 0001-0782. DOI: 10.1145/7885.7887. URL: http://doi.acm.org/10.1145/7885.7887 (siehe S. 18).

[56] Gnieh Development Team. *BlueLaTex*. 2015. URL: http://www.bluelatex.org/ (besucht am 20. 04. 2015) (siehe S. 4, 6).

[57] latex-lab Team. *latex-lab*. 2015. URL: http://docs.latexlab.org/ (besucht am 20. 04. 2015) (siehe S. 4, 6).

[58] LyX Team. *LyX*. 2015. URL: http://www.lyx.org/ (besucht am 20. 04. 2015) (siehe S. 3, 6).

[59] ShareLaTex Team. *ShareLaTex*. 2015. URL: https://de.sharelatex.com/ (besucht am 20. 04. 2015) (siehe S. 4, 6).

[60] Han The Thanh. *pdflatex(1) - Linux man page*. 2011. URL: http://linux.die.net/man/1/pdflatex (besucht am 16. 06. 2015) (siehe S. 17).

[61] Verbosus. *Verbosus*. 2015. URL: https://www.verbosus.com (besucht am 20. 04. 2015) (siehe S. 4, 6).

[62] Oliver Vogel u. a. *Software-Architektur: Grundlagen, Konzepte, Praxis*. German. 2. Aufl.; 2; 2. Auflage. Dordrecht: Spektrum Akademischer Verlag, 2009. ISBN: 3827419336; 9783827419330 (siehe S. 31).

[63] University of Washington. *SageMathCloud*. 2015. URL: https://cloud.sagemath.com/ (besucht am 20. 04. 2015) (siehe S. 4, 6).

[64] Chris Woolston. *Word-processing war flares up on social media*. Jan. 2015. URL: http://www.nature.com/news/word-processing-war-flares-up-on-social-media-1.16669 (besucht am 06. 06. 2015) (siehe S. 7, 9).