

Eberhard Karls Universität Tübingen  
Mathematisch-Naturwissenschaftliche Fakultät  
Wilhelm-Schickard-Institut für Informatik

## Masterarbeit Medieninformatik

### **Entwicklung einer Webanwendung zur Annotation spezifischer linguistischer Merkmale in Fließtexten**

Oliver Brehm

31.01.2018

#### **Gutachter**

Jun.-Prof. Dr. Alexandra Kirsch  
Wilhelm-Schickard-Institut für Informatik  
Universität Tübingen

Prof. Dr. Detmar Meurers  
Seminar für Sprachwissenschaft  
Universität Tübingen

#### **Betreuer**

M.Sc. Heiko Holz  
Wilhelm-Schickard-Institut für Informatik  
Universität Tübingen

**Brehm, Oliver:**

*Entwicklung einer Webanwendung zur Annotation spezifischer linguistischer Merkmale in Fließtexten*

Masterarbeit Informatik

Eberhard Karls Universität Tübingen

Bearbeitungszeitraum: August 2017 - Januar 2018

## **Zusammenfassung**

Viele Menschen haben Schwierigkeiten, flüssig zu Lesen und zu Schreiben. Man spricht von einer Lese- Rechtschreibschwäche (LRS), wenn keine erkennbaren äußeren Umstände die Entwicklung der Lese- und Schreibkompetenz beeinträchtigen. Es wurde gezeigt, dass gezielte Übungen, die das Erkennen von Silben und Sprachrhythmus fördern dazu führen können, diese Fähigkeiten zu verbessern. In der Therapie werden häufig Übungstexte verwendet, in denen Silben abwechselnd in verschiedenen Schriftfarben dargestellt werden.

In dieser Arbeit wurde eine Webapplikation entwickelt, mit der es ermöglicht werden sollte, solche farblich markierten Texte einfach und automatisch zu erstellen. Für die Verdeutlichung des Sprachrhythmus sollte auch die betonte Silbe im Wort speziell markiert werden können. Es wurde zunächst untersucht, wie aus einem beliebigen Text automatisch Silbentrennung und Wortbetonung bestimmt werden können. Dazu wurde eine auf einem Lexikon basierende Datenbank aufgebaut, die für jedes Wort diese Merkmale enthält. Die Datenbank ist durch NutzerInnen der Anwendung erweiterbar, hinzugefügte Einträge müssen jedoch von Experten oder anderen NutzerInnen verifiziert werden, damit sie für alle NutzerInnen verfügbar sind. Für die Applikation wurden zunächst die Anforderungen analysiert und anschließend zwei Hauptkomponenten entwickelt: Auf der einen Seite das Backend, welches Anfragen der Webapplikation, wie z.B. den zu analysierenden Text entgegen nimmt und beantwortet, sowie die erstellte Wortdatenbank und eine weitere Datenbank, die nutzerspezifische Einstellungen speichert, verwaltet. Auf der anderen Seite steht die Entwicklung des Frontends, der eigentlichen Webapplikation, welche mit NutzerInnen interagiert, Anfragen an das Backend schickt und die dadurch erhaltenen Informationen entsprechend darstellt.

Die abschließende Evaluation zeigt, dass mit Hilfe der Webapplikation Übungstexte erfolgreich erstellt werden können. Mit einem Nutzertest wurde festgestellt, dass die meisten Anwendungsfälle in der Web-Oberfläche intuitiv und zeiteffizient durchgeführt werden konnten. Die erkannten Probleme können durch eine Überarbeitung der Nutzeroberfläche mit wenig Aufwand behoben werden. Weitere Anmerkungen der ProbandInnen generierten interessante, weiterführende Ideen, die auf diese Arbeit aufbauend gut realisierbar sind.

# Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>7</b>
<b>2</b>	<b>Stand der Forschung</b>	<b>11</b>
2.1	Lese- Rechtschreibschwäche . . . . .	11
2.1.1	Dekodierung des Wortbildes . . . . .	11
2.1.2	Digitale Ansätze . . . . .	12
2.2	Technischer Hintergrund . . . . .	13
2.2.1	Linguistische Analyse . . . . .	13
2.2.2	Datenbank zur Wortbetonung . . . . .	14
<b>3</b>	<b>Anforderungsanalyse und Spezifikation</b>	<b>17</b>
3.1	Anforderungen an die Software . . . . .	17
3.1.1	User Interface . . . . .	19
3.2	Softwarestack . . . . .	20
3.2.1	Verwendete Technologien . . . . .	21
3.2.2	Backend . . . . .	22
3.2.3	Frontend . . . . .	23
3.2.4	Deployment . . . . .	25
<b>4</b>	<b>Entwicklung der Anwendung</b>	<b>27</b>
4.1	Erstellung einer Basisdatenbank . . . . .	28
4.1.1	Implementierung . . . . .	30
4.1.2	Ergebnisse . . . . .	31
4.2	Backend . . . . .	31
4.2.1	HTTP Routen . . . . .	31
4.2.2	Dictionary Service . . . . .	32
4.2.3	User Service . . . . .	35
4.2.4	Verification Service . . . . .	36
4.3	Frontend . . . . .	37
4.3.1	Hauptkomponente der Applikation und Routing . . . . .	38
4.3.2	Datenmodell . . . . .	39
4.3.3	Textanalyse . . . . .	40
4.3.4	Annotationseinstellungen . . . . .	45
4.3.5	Nutzerkonto . . . . .	49
4.3.6	Wort-Verifizierung . . . . .	50

<b>5</b>	<b>Evaluation</b>	<b>53</b>
5.1	Ergebnisse der Textannotation . . . . .	53
5.2	Nutzertest . . . . .	59
5.2.1	Vorbereitung und Durchführung . . . . .	59
5.2.2	Ergebnisse . . . . .	60
5.3	Diskussion . . . . .	64
5.4	Ausblick . . . . .	64
	<b>Literaturverzeichnis</b>	<b>67</b>
<b>6</b>	<b>Anhang</b>	<b>71</b>



# 1 Einleitung

In der vorliegenden Arbeit wurde eine Webapplikation entwickelt, die Fließtexte durch farbliche Annotationen so aufbereiten kann, dass diese als Übungstexte in der Therapie von Kindern mit Lese- Rechtschreibschwäche (LRS) eingesetzt werden können. Solche Übungstexte werden häufig von LerntherapeutInnen manuell, z.B. mit einem Textverarbeitungsprogramm mit viel Aufwand erstellt. Die Zielgruppe für die Benutzung der Applikation besteht also aus LerntherapeutInnen, Lehrkräften oder auch den Eltern betroffener Kinder, deren Arbeit durch die Automatisierung des Prozesses der Erstellung von Übungstexten erleichtert werden kann.

Die Lese- Rechtschreibschwäche (auch Legasthenie oder Dyslexie) ist eine der weit verbreitetsten Entwicklungsstörungen bei Kindern und Jugendlichen[21]. Bei der Therapie gibt es verschiedene Maßnahmen, um die Lese- und Schreibkompetenz von Betroffenen zu fördern. Das Lesen speziell bearbeiteter Texte zielt z.B. darauf ab die Dekodierfähigkeit zu verbessern, deren Unterentwicklung neben anderen Faktoren einen Teil der Ursachen für LRS abdeckt[14]. Es wurde gezeigt, dass beim fortgeschrittenen Lesen und Schreiben nicht Buchstaben und Laute die erfassten Elemente sind, sondern größere orthographische Einheiten, wie Silben oder ganze Wörter[12]. Eine zentrale Rolle spielt dabei also das Erkennen von Silben. Die Silbenmethode und Silbenfibeln sind auch im schulischen Umfeld im Einsatz. Neben dem Schrifterwerb durch einfache Zuordnung von Buchstaben und Lauten bilden Übungen, wie z.B. das Segmentieren der Wörter in Silben, sowie Rhythmusspiele eine Ergänzung zu herkömmlichen Unterrichtsmethoden[17]. Eine weitere Erkenntnis hat gezeigt, dass die Wahrnehmung von Sprachrhythmus ebenfalls eine Auswirkung auf die Lese- und Schreibkompetenz hat[9]. Es wird vermutet, dass ein gezieltes Training der Bewusstheit von Silbenbetonung zu höherem Lernerfolg führen kann[16] (s. Abschnitt 2.1.1). Im folgenden Kapitel werden auch einige Alternativen vorgestellt, die das Lesen Lernen mit der Silbenmethode unterstützen.

Basierend auf diesen Überlegungen wurde die Applikation mit der Motivation entwickelt, das Erstellen von Übungstexten zu vereinfachen und zu automatisieren. In den Übungstexten werden Wörter in Silben unterteilt, welche sich mit verschiedenen Farben hervorheben lassen. Im Gegensatz zu ähnlichen Ansätzen (sowohl Lehrmaterial in Papierform als auch digitale Ansätze) bietet die Applikation zusätzlich die Möglichkeit, die betonte Silbe im Wort besonders (z.B. in einer anderen Farbe) darzustellen. Weitere Merkmale, wie der Abstand zwischen Silben und Wörtern, sowie ein Silbentrennzeichen, lassen sich frei anpassen. Die Anwendung kann LerntherapeutInnen, Lehrkräften oder Eltern helfen, spezifisch zugeschnittene Übungen mit wenig Aufwand selbst zu erstellen.

Es können viele Gründe genannt werden, weshalb wir uns von der Entwicklung einer solchen

## 1 Einleitung

Anwendung Erfolg versprochen: Ein Grund ist die effizientere Nutzung von Ressourcen. Automatisierung von Aufgaben bringt immer einen Zeitgewinn für die Personen, die diese sonst erledigen mit sich. Zeit, welche die LerntherapeutInnen beim Erstellen von Übungsmaterial einsparen, kann z.B. für eine bessere Vorbereitung der Therapie oder zu mehr Betreuungszeit von SchülerInnen führen. Außerdem ist die Gestaltung von Übungsmaterial sehr flexibel, da in der Software viele Einstellungen für die Textdarstellung vorgenommen werden können. So können die LerntherapeutInnen einfach verschiedene Möglichkeiten beim Erstellen von Übungstexten ausprobieren und bei der Verwendung mit den SchülerInnen deren Präferenzen erkennen. Dadurch lassen sich eventuell direkt Erkenntnisse gewinnen, welche Art von Textannotation am besten funktioniert.

Bei der Recherche konnte nur wenig vergleichbare Software gefunden werden. Tools, die ebenfalls die Aufgabe der Annotation von Silben in Texten erledigen sind zum Teil veraltet oder bieten wenig Möglichkeiten zur Anpassung der Annotation (s. Abschnitt 2.1.2).

Die Entwicklung der Anwendung stellte verschiedene Herausforderungen in diversen Bereichen dar. Zuerst musste untersucht werden, wie in Wörtern Silbentrennung und Wortbetonung automatisch bestimmt werden können. Hierfür wurde eine eigene Datenbank aufgebaut, die diese und andere Merkmale speichert. Eine besondere Schwierigkeit stellte die Komplexität und der Umfang der deutschen Sprache dar. Als Grundlage für die Datenbank diente das Lexikon CELEX2[10], aus dem rund 360 000 Einträge, welche sowohl aus Grund- als auch Flexionsformen von Wörtern bestehen, extrahiert werden konnten. Damit ist es aber nicht möglich den deutschen Wortschatz lückenlos mit allen Flexionsformen abzubilden. Es wurde deshalb nach einer Möglichkeit gesucht, eine benutzerfreundliche Schnittstelle zu bieten, mit der korrekte Einträge von NutzerInnen und ExpertInnen manuell, mit wenig Aufwand der Datenbank hinzugefügt werden können. Es wurde untersucht, welche Ansätze es gibt, mit *Crowdsourcing* die Erweiterbarkeit der Datenbank auf die NutzerInnen aufzuteilen (s. Abschnitt 2.2.2). So könnten beispielsweise auch Eigennamen, die in als Übungstexte verwendeten Geschichten häufig auftauchen, Teil des Wortschatzes werden.

Vor der Entwicklungsphase mussten der Rahmen und Funktionsumfang der Software definiert und die einzelnen Komponenten spezifiziert werden. Dafür wurde zunächst eine Anforderungsanalyse durchgeführt. Hierbei wurden Ideen aus verschiedenen Quellen gesammelt, es wurden Brainstormings in der eigenen Forschungsgruppe durchgeführt, sowie Gespräche mit ExpertInnen aus der Computerlinguistik und der Lerntherapie geführt. Alle wichtigen Aspekte der Applikation wie Design, Arten der Textannotation, Funktionen der Wortdatenbank und des Nutzerkontos etc. wurden diskutiert. So entstanden viele Ideen und Szenarien woraus dann eine Spezifikation erarbeitet werden konnte. In einer späteren Phase der Entwicklung entstanden parallel noch weitere Ideen für Features, die am Anfang nicht vorgesehen waren. Daher wurden die ins Projekt involvierten Personen zusätzlich mit einem Fragebogen gebeten, die Wichtigkeit weiterer Features zu bewerten und anzugeben mit welcher Priorität diese noch entwickelt werden sollten.

Einen wichtigen Punkt für die Entwicklung der Webapplikation stellte die User Experience dar. Die Zielgruppe der NutzerInnen, welche die Applikation letztendlich bedienen sollen, wurde auf LerntherapeutInnen, Lehrkräfte und Eltern von betroffenen Kindern festgelegt. Diese



bringen sehr unterschiedliche Kenntnisse im Umgang mit Software bzw. Webanwendungen mit. Haben die NutzerInnen Schwierigkeiten das Programm zu bedienen, führt das schnell zu Frust und letztendlich dazu, dass die Software am Ende überhaupt nicht genutzt und die Aufgabe weiterhin manuell erledigt wird (s. Abschnitt 5.2). Das Ziel war also die Oberfläche so einfach und intuitiv wie möglich zu gestalten, um allen NutzerInnen einen reibungslosen Umgang mit der Applikation zu ermöglichen. Es wurde beschlossen eine Webbrowser-basierte Anwendung zu entwickeln, welche plattformunabhängig funktioniert. Damit wurden von Anfang an Hürden, wie die Bindung an ein bestimmtes Betriebssystem und die Notwendigkeit der Installation von Software beseitigt. Die Applikation kann von jedem Computer mit Internetverbindung und Webbrowser sofort benutzt werden. Durch die Verwendung von Nutzerkonten sind nutzerspezifische Einstellungen überall gleichermaßen abrufbar.

Im Evaluationsteil am Ende der Arbeit wurde die Benutzbarkeit des Gesamtsystems anhand eines Nutzertests bewertet. Hierfür wurden als ProbandInnen sowohl Expertinnen (Lerntherapeutinnen) als auch Laien herangezogen. Für die Auswahl der ProbandInnen aus der Gruppe der Laien gab es keine Beschränkungen, da in der Zielgruppe auch Elternteile von betroffenen Kindern mit LRS enthalten sind und diese aus den verschiedensten Berufsfeldern sowie unterschiedlichen Altersgruppen stammen können. Zunächst wurde ein Pilottest ausgeführt, um Mängel im Design des Tests, sowie Fehler der Webanwendung, die nicht im Zusammenhang mit der Usability standen im Vorfeld zu finden und zu beheben. Im Anschluss wurde der Nutzertest mit sieben ProbandInnen ausgeführt und die Ergebnisse zusammengetragen und bewertet.

In den folgenden Kapiteln werden zunächst die Grundlagen zur Lese-Rechtschreibschwäche, sowie zu linguistischen Begriffen und Technologien erklärt. Danach wird der Prozess der Planung und Entwicklung der Software beschrieben. Detaillierte Beschreibungen zu einzelnen Komponenten und zum Zusammenspiel dieser sollen verdeutlichen, wie das System funktioniert. Den Abschluss der Arbeit bildet die Evaluation anhand von Beispielen mit der Applikation produzierter Texte und der Beschreibung und Durchführung des Nutzertests. Hier wird gezeigt, wie der entwickelte Prototyp im tatsächlichen Einsatz zu bewerten ist. Die Evaluation zeigt sowohl Stärken als auch Schwächen des Systems auf, die in einem weiteren Einsatz noch zu beheben sind. Die Flexibilität des Gesamtsystems schaffte zudem viele anregende neue Ideen, die im Anschluss an diese Arbeit gut realisierbar sind.



## 2 Stand der Forschung

Die Motivation der Arbeit beruht auf Erkenntnissen zu Behandlungsformen für Lese- Rechtschreibschwäche (LRS). Es wurde untersucht, wie gut der Prozess des Erzeugens von Übungstexten automatisiert werden kann. Hierfür war das Ziel bei der Entwicklung der Applikation eine möglichst gute User Experience zu bieten. Die Zielgruppe, bestehend aus LerntherapeutInnen, Lehrkräften und Eltern betroffener Kinder, soll das Programm möglichst intuitiv bedienen können.

Im Folgenden werden daher erst die Grundlagen zu LRS erklärt und digitale Ansätze zur Erstellung von Übungstexten, z.B. auch ähnliche, bereits vorhandene Software beschrieben. Danach werden die technischen Grundlagen und computerlinguistische Begriffe erklärt, die in der Entwicklung der Applikation relevant waren.

### 2.1 Lese- Rechtschreibschwäche

Ein Ausgangspunkt zur Feststellung von LRS bietet die Lesekompetenz. Eine Definition aus der PISA Studie untergliedert die Lesekompetenz in die vier Teilbereiche *Kognitive Grundfähigkeit*, *Dekodierfähigkeit*, *Lernstrategiewissen* und *Leseinteresse*[14].

Der Teilbereich, zu dessen Verbesserung diese Arbeit einen Beitrag leisten kann ist die Dekodierfähigkeit. Diese stellt die Kompetenz dar, die Bedeutung von Wörtern, Sätzen und Texten zu erfassen und zu verstehen. Die Unterteilung in die drei Elemente Wort, Satz und Text stellt eine Dekodierungsfähigkeit auf verschiedenen Ebenen dar, die sich gegenseitig bedingen. Das Wort ist hier die Basisebene. So kann ein Text nur verstanden werden, wenn die Bedeutung der einzelnen Sätze erfasst wurde und diese wird wiederum nur erkannt, wenn eine ausreichende Dekodierfähigkeit für Wörter vorhanden ist. Somit ist eine Voraussetzung für die Verbesserung des Textverständnis, dass das Erkennen des Basiselements Wort beherrscht wird.

#### 2.1.1 Dekodierung des Wortbildes

Es ist bekannt, dass Kinder in der Entwicklung des Lesens und Schreibens verschiedene Phasen durchlaufen[12]. Es gibt eine alphabetische und eine orthographische Phase. Beim Lesen wird in der alphabetischen Phase ein Wort Buchstabe für Buchstabe dekodiert, *Grapheme* (die kleinste Einheit in der Schriftsprache, in der deutschen Sprache Buchstaben) werden einzeln in *Phoneme* (Laute, die kleinste Einheit der gesprochenen Sprache) umgewandelt, was aber bei nicht lautgetreuen Wörtern (Wörter, in denen sich nicht alle Grapheme korrekt in die

## 2 Stand der Forschung

zugehörigen Phoneme übersetzen lassen) nicht gelingt.

Daher wird später in der orthographischen Phase eine Strategie benutzt, die sich an größeren Bestandteilen orientiert. Wörter oder Wortteile werden hier aus dem Langzeitgedächtnis abgerufen, was bei richtig gelernten Silben und Wörtern zu korrekter Aussprache und Schreibung führt.

Einige schulische Ansätze wie das *ABC der Tiere*[6] oder die *Freiburger Rechtschreibschule (FRESCH)*[7] fördern das orthographische Lernen mit einer gezielten Hervorhebung von Silben. Hier sind z.B. Fördermaterialien erhältlich, die Silben farblich hervorheben um Lernerfolge beim Schrifterwerb zu erzielen.

Weiterführende Arbeiten legen auch nahe, dass der Sprachrhythmus und die Silbenbetonung eine zentrale Rolle, sowohl beim Lesen, als auch beim Schreiben spielt. Ein Wort hat in der deutschen Sprache eine oder mehrere Betonungen. Betonte Silben werden, im Gegensatz zu den unbetonten Silben, lauter und länger gesprochen. In der Dissertation von Brandelik[9] wurde gezeigt, dass sprachrhythmische Fähigkeiten, wie das Identifizieren von Betonungen einen Einfluss auf die Lese- und Rechtschreibleistung hat. An diese Ergebnisse angelehnt trainiert z.B. die Prosodiya App[16] für mobile Geräte bei LRS Kindern das Bewusstsein für linguistische Eigenschaften im Zusammenhang mit der Silbenbetonung.

In einer Studie von Rello[25] wurde zudem herausgefunden, dass die Art der Präsentation von Text einen starken Einfluss auf die Lesbarkeit für Menschen mit LRS hat. Es wurde gezeigt, dass Parameter wie verschiedene Fonts, Schriftgröße, Farbkombinationen und Hintergrundfarbe, sowie die Abstände zwischen Zeilen oder einzelnen Zeichen eine wesentliche Rolle für die Lesegeschwindigkeit spielen.

### 2.1.2 Digitale Ansätze

In der Recherche für die Entwicklung der Webapplikation wurde auch untersucht, welche ähnlichen Software-basierten Lösungen es schon gibt. Das im vorherigen Abschnitt erwähnte *ABC der Tiere* liefert auch einen *Silben-Generator*[1]. Dieser ist allerdings nur als Windows Programm erhältlich und die Möglichkeiten des Programms sind sehr begrenzt. So gibt es z.B. kein Textfeld, in das man selbst Text eingeben kann, es ist nur das Laden von Textdateien möglich. Es gibt hier auch nur wenige Einstellungen, so kann man zwar Silbenfarben und Zeilenabstand anpassen, nicht jedoch den Abstand zwischen Wörtern oder Silben. Einstellungen lassen sich auch nur global ändern und nicht, z.B. bezogen auf einzelne Texte speichern.

Ein weiteres zu erwähnendes Tool ist die *Celeco Druckstation*[5]. Auch hier können eigene Texte entweder mit farblich markierten Silben oder Silbenbögen gedruckt werden. Das Programm ist zusammen mit der *Celeco Software* (LRS Therapie und Leseübungen) erhältlich, welche für Windows verfügbar ist.

Um die Vorteile durch silben- und betonungsorientiertes Lernen auszunutzen ist man oft auf Materialien von Verlagen oder eine aufwendige manuelle Erstellung von Texten angewiesen. Dies bekräftigt die Motivation eine neue Lösung zu entwickeln, um eigenes Übungsmaterial

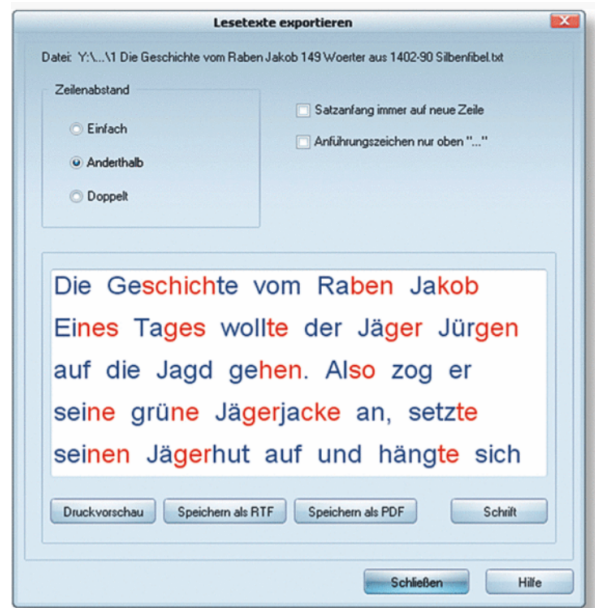


Abbildung 2.1: Bildschirmfoto des Silbengenerators auf der Internetseite von *ABC der Tiere*[1]

erstellen zu können, was zudem flexibel gestaltbar und z.B. auf bestimmte Schüler anpassbar ist.

## 2.2 Technischer Hintergrund

Um das Ziel einer automatischen Analyse und Annotation beliebiger Texte zu realisieren, wurden vor allem die folgenden zwei Schritte untersucht:

1. Linguistische Analyse (Zerlegung des Texts in Wörter)
2. Bestimmung von Silbentrennung und Betonungsmuster

### 2.2.1 Linguistische Analyse

Da die von der Applikation zu verarbeitende Eingabe ein Text ist, also eine Folge von alphanumerischen Zeichen, sowie Leerraumzeichen und Interpunktionen, muss diese Eingabe zunächst korrekt in Wörter zerteilt werden. Dieser Schritt wird **Tokenisierung** genannt. Zusätzlich werden auch von einem **Tagger** jedem Wort weitere Informationen, wie z.B. die Wortart (*Part-of-Speech-/POS-Tagger*) hinzugefügt. Diese werden üblicherweise mithilfe eines entsprechenden Lexikons nachgeschlagen[11]. Der für dieses Projekt verwendete Parser wird in Abschnitt 3.2.2 näher beschrieben. Beim Parsen des Texts können folgende wichtige Informationen extrahiert werden:

- **Textstruktur:** Zerlegung des Texts in Tokens, dadurch wird bestimmt, ob es sich bei den Zeichenfolgen um Wörter oder andere Strukturen wie Leerraum oder Interpunktionen handelt.
- **Wortart:** Die Werte der Wortart- (*Part-of-Speech*-) Tags geben an um was für ein Wort es sich handelt (mit dem *Spacy* Parser z.B. *NOM* für Nomen, *PROPN* für Eigennamen, *DET* für Artikel etc.).
- **Lemma:** Gibt die Grundform eines Wortes an, welche je nach Wortart unterschiedliche Ausprägungen annehmen kann. Bei Verben steht hier z.B. der Infinitiv, bei Pronomen die erste Person Singular etc.

### 2.2.2 Datenbank zur Wortbetonung

Liegt der Text als Liste von Wörtern (Tokens) vor, so können weitere Analysen auf dieser Ebene vorgenommen werden. Für die Annotation des Textes muss eine Repräsentation in Silben vorliegen, zusätzlich wird die betonte Silbe im Wort gesucht. Es ist durchaus möglich, dass in einem Wort mehrere Betonungen vorkommen (gerade bei Wortkompositionen, die im Deutschen häufig zu finden sind, z.B. *heraus+kommen*). Dies wird hier vernachlässigt, es wird nur eine Hauptbetonung, die erste vorkommende Betonung im Wort gesucht. Eine weitere Einschränkung, die getroffen wurde ist die Beschränkung auf die Annotation der Betonung im Wort-Kontext (und nicht im Zusammenhang des Satzes). Hier wäre es auch denkbar gewesen, für die Erlernung von Sprachrhythmus und Satzmelodie die Betonung über Wortgrenzen hinaus in Sätzen hervorzuheben. Folgen z.B. viele kurze Wörter aufeinander, so werden diese im natürlichen Sprachrhythmus nicht alle gleich betont gesprochen, sondern eine Betonung findet nur in bestimmten Wörtern im Satz statt. Im Rahmen der vorliegenden Arbeit wurde dieser Aspekt der Prosodie vernachlässigt. Das Ziel war es, bei Menschen mit LRS die Dekodierung des Wortbildes mithilfe von Silben zu erleichtern, daher wurde sich auf die Möglichkeit konzentriert, einzelne Silben und die betonte Silbe im Wort hervorzuheben.

Informationen zur Worttrennung in Silben und zur Wortbetonung können aus diversen Quellen bezogen werden. Für die Silbentrennung gibt es viele Tools, es können z.B. einfach die Hyphenator von Programmen wie *OpenOffice* oder *LaTeX*, sowie das Python Framework *Pyphen* verwendet werden. Angaben über Betonungen sind allerdings nicht ebenso trivial zu finden. Hierfür wurde untersucht, welche vorhandenen Lexika dafür infrage kamen.

Für die Verwendung in der zu entwickelnden Applikation wurde als Grundlage das Lexikon CELEX2 gewählt. Dabei handelt es sich um ein umfangreiches Sprachlexikon für die Sprachen Englisch, Deutsch und Niederländisch. Verarbeitet wurde hier zunächst nur die deutsche Sprache. Neben der gesuchten Worttrennung und -betonung enthält es auch weitere linguistische Informationen wie Wortart und Grundformen, sowie phonologische und orthographische Annotationen [10].

Als alternative wurde auch das *PHONOLEX* Lexikon des Bayrischen Archivs für Sprachsignale[4] untersucht. Diese Arbeit ist etwas aktueller (letzte Version erschienen 2013), mit etwa 1,6

Millionen Einträgen ist es als Aussprache-Lexikon zudem deutlich umfangreicher als der CELEX[27]. Da PHONOLEX allerdings nicht kostenfrei erhältlich ist wurde es in dieser Arbeit nicht näher untersucht. Für eine Erweiterung der verfügbaren Daten für die Wortbetonung in der Applikation wäre es aber denkbar, beide Lexika (CELEX und PHONOLEX) in der Zukunft als Grundlage zu verwenden.

Weiterführend wurden auch Text-To-Speech Systeme untersucht. Diese generieren bei gegebenem Eingabetext automatisch synthetische Sprache, wofür ein internes Modell aufgebaut wird. Damit sie funktionieren muss also zwangsweise eine phonologische Analyse durchgeführt werden, welche auch die Wortbetonung bestimmt. Beispielsweise liefern die Systeme MARY[28] oder BALLOON[24] auch textuell annotierte Formen ihrer phonologischen Analyse, die als Grundlage zur Extraktion von Betonung dienen können. MARY bietet zudem eine API an, die mit HTTP Requests angesprochen und somit in die zu entwickelnde Software eingebunden werden kann (s. Abschnitt 4.2.2).

Linguistische Datenbanken können aufgrund des Umfangs und der Flexibilität von Sprachgrammatiken niemals vollständig sein. Es wurde daher ein Weg gesucht die Datenbank auf unkomplizierte Weise erweiterbar zu machen. Dies kann mit einem geeigneten User-Interface durch ExpertInnen erfolgen, die unbekannte Einträge ergänzen und hinzufügen. Einige Arbeiten hatten bereits Erfolg damit, solche Aufgaben durch *Crowdsourcing* effektiv auch auf NutzerInnen zu verteilen, die keine ExpertInnen für den jeweiligen Anwendungsbereich sind. Da die zu entwickelte Applikation auch ein Nutzerverwaltungssystem beinhaltet, wurde untersucht, ob Datenbankeinträge mithilfe mehrerer Nicht-Experten NutzerInnen per einfachem Mehrheitsentscheid verifiziert werden können. Weiterführend könnten auf Plattformen wie Amazon Mechanical Turk oder CrowdFlower beispielsweise leicht ArbeiterInnen anonym engagiert werden, die solche Aufgaben erledigen[29]. Diese Plattformen wurden beispielsweise von Zaidan und Callison-Burch (für automatisierte Übersetzungen)[32] oder De Kuthy, Ziai und Meurers[18] (für Fokusannotation) benutzt, um computerlinguistische Aufgaben zu erledigen.





## 3 Anforderungsanalyse und Spezifikation

Während der Recherche und der Konzeption des Projekts ist klar geworden, dass die zu entwickelnde Software eine relativ komplexe Architektur erfordert. Da eine Browser-basierte Webapplikation entwickelt wurde, wurde untersucht, wie die Software am besten in Teilprojekte untergliedert werden konnte. Zum einen musste das User-Interface für die Web Applikation und zum Anderen die Programmlogik entwickelt werden, welche aufgrund der benötigten Funktionalitäten (wie Datenbankankbindung und Algorithmen zur Textverarbeitung) nicht sinnvoll im Web Client implementiert werden konnte.

Um eine Spezifikation für die Software zu entwerfen und den zu verwendenden Software-Stack zu definieren, wurde zunächst im Gespräch mit ExpertInnen (ComputerlinguistInnen und LerntherapeutInnen) eine Anforderungsanalyse durchgeführt, die im folgendem Teil beschrieben wird. Im darauffolgenden Kapitel gehe ich Schritt für Schritt auf die Entwicklung des Gesamtsystems ein.

### 3.1 Anforderungen an die Software

Zunächst wurde festgestellt, welche Funktionen die Applikation den NutzerInnen bieten sollte. Dazu wurden folgende mögliche Szenarien aufgestellt:

**Textanalyse** Die Nutzerin oder der Nutzer möchte einen Fließtext eingeben und von der Anwendung annotieren lassen. Nach der Eingabe soll das Ergebnis annotiert in der Applikation dargestellt werden.

**Anpassung der annotierten Darstellung** Die Nutzerin oder der Nutzer möchte die Einstellungen des annotierten Texts ändern. Die Texteigenschaften (Font, Zeilenabstand, Zeichenabstand) sollen angepasst werden können, sowie die Darstellung der Annotation (Farben für betonte und unbetonte Silbe, Silbenabstand, Trennzeichen zwischen Silben)

**Export der annotierten Darstellung** Die Nutzerin oder der Nutzer soll die Möglichkeit haben, verschiedene Formate des annotierten Texts zu exportieren, z.B. Druck, HTML oder Word.

**Verwaltung von User Accounts** Der Nutzerin oder dem Nutzer soll die Möglichkeit gegeben werden, ein Nutzerkonto zu erstellen um persönlich verwendete Daten (z.B. Texte oder Wortsegmentierungen) speichern zu können. Dazu sollen Funktionen und Interfaces zur Registrierung eines Nutzerkontos, zum Login und Logout, zum Bearbeiten der Nutzerinformationen und zum Löschen des Accounts bereitgestellt werden.

**Behandlung unbekannter Wörter** Die Nutzerin oder der Nutzer kann nacheinander die Segmentierung von Wörtern, die durch das System nicht eindeutig bestimmt werden konnten, selbst manuell festlegen.

**Bestimmung der Segmentierung unbekannter Wörter** Für ein unbekanntes Wort wählt die Nutzerin oder der Nutzer in einem neuen View die Segmentierung selbst aus. Dafür sollen folgende Möglichkeiten gegeben werden:

- Input aus G2P Systemen wie MARY TTS
- Manuelle Segmentierung mit geeignetem User Interface
- Eventuell weitere Quellen für die Segmentierung (z.B. Duden oder Wictionary)

**Speicherung von Nutzer-Segmentierungen** Von der Nutzerin oder dem Nutzer hinzugefügte Segmentierungen sollen (lokal für das Nutzerkonto) gespeichert werden können und beim nächsten Vorkommen in einem Text automatisch verwendet werden.

**Speicherung und Wiederverwendung von Annotationskonfigurationen für Texte** Die Einstellungen, die eine Nutzerin oder ein Nutzer an einem annotierten Text vorgenommen hat, sollen als Vorlage für andere Texte gespeichert werden können. In den Annotationseinstellungen eines Textes soll eine zuvor gespeicherte Konfiguration wiederverwendet werden können.

**Speicherung und Wiederverwendung von Nutzertexten** Analysierte Texte sollen von der Nutzerin oder dem Nutzer zusammen mit der verwendeten Konfiguration gespeichert werden können. Den Texten werden Metadaten zugeordnet, z.B. Titel, Thema, Niveau oder Zielgruppe. Im Nutzerbereich werden die Texte, die NutzerInnen hinzugefügt haben, geeignet strukturiert dargestellt. Es soll die Möglichkeit gegeben werden, den gespeicherten Text erneut analysieren zu lassen.

**Verifizierung unbekannter Wörter** Manuelle Segmentierungen von unbekannten Wörtern müssen auf Korrektheit überprüft werden können. Dazu sollen alle NutzerInnen aufgefordert werden, die Einträge, die mit anderen Nutzerkonten erstellt wurden, zu überprüfen. In einem Fenster, ähnlich zur manuellen Segmentierung, sollen die NutzerInnen fremde Einträge bestätigen oder Gegenvorschläge übermitteln können.

**Expertennutzer** ExpertInnen (z.B. LinguistInnen oder LerntherapeutInnen) sollen sich als Solche bei der Erstellung eines Accounts identifizieren können. Bei der Wort-Verifizierung zählt die Stimme der ExpertennutzerInnen mehr als die von NutzerInnen, die keine Experten sind.

Hieraus lassen sich folgende grundlegende Anforderungen an die Software spezifizieren:

- Service zur Textanalyse
- Datenbank für Wörter und deren Segmentierung
- Service zur Manipulation der Wortdatenbank
- Speicherung nutzerspezifischer Daten
- Service zur Nutzerverwaltung
- Service zur Verifizierung von manuell erstellten Einträgen
- Geeignetes User Interface zur Darstellung von Texten, Verwaltung der Nutzerdaten und zur Interaktion mit dem System

#### 3.1.1 User Interface

Im nächsten Schritt wurde, von den Szenarien und Anforderungen ausgehend, das Design für ein geeignetes User Interface erarbeitet. Tabelle 3.1 zeigt die für die Navigation wichtigsten Komponenten.

Hauptnavigationskomponente	Unterpunkte / Funktionen
Begrüßungsseite (Home)	Informationen zum Hintergrund der Applikation und zu verfügbaren Funktionen
Textanalyse	Eingabe und Analyse von Texten, Annotation und Darstellung des Textes, sowie Funktionen zum Druck und Export
Nutzerkonto	Login, Registrierung, Verwaltung des Nutzerkontos und nutzerspezifischer Daten
Verifizierung	Verifizierung manuell segmentierter Wörter

Tabelle 3.1: Hauptkomponenten der Navigationsstruktur

#### Informationsstruktur

Mithilfe dieser Überlegungen wurde festgelegt, dass es eine horizontale Navigationsleiste mit den vier aufgelisteten Hauptnavigationskomponenten geben soll. Der Link zum Nutzerkonto sollte aufgrund von Nutzergewohnheiten gesondert, rechtsbündig in der Navigationsleiste

### 3 Anforderungsanalyse und Spezifikation

platziert werden (Abbildung 3.1). Jeder Punkt der Navigation soll Einstiegspunkt in eine Hauptkomponente der Applikation sein, es gibt keine weiteren Ebenen in der Navigationshierarchie, da die Anzahl der Komponenten überschaubar ist. Daher wird auch auf andere Navigationsmöglichkeiten, wie z.B. eine seitenweite Suche verzichtet.

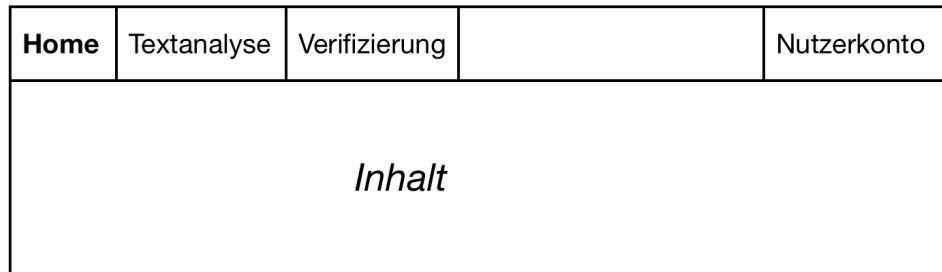


Abbildung 3.1: Navigationsstruktur

#### Farbschema

Als Nächstes wurde ein möglichst ansprechendes Farbschema gesucht. Hierfür wurden mit einem Farbrad (es wurde Adobe Color CC verwendet, <https://color.adobe.com>) zwei Komplementärfarben als Hauptfarben ausgesucht (s. Abbildung 3.2). Die blaue Farbe wurde vor allem für Links eingesetzt, eine hellere Version für Links im Hover Zustand. Eine weitere, ins Türkis gehende Abstufung wurde für Rahmen beim Gruppieren von Elementen auf der Seite verwendet. Der beige Farbton kommt als Hintergrundfarbe für die Navigationsleiste zum Einsatz.

Farbe				
Beschreibung	Primär	Primär hell	Primär variiert	Komplementär
Funktion	Links	Links hover	Rahmen	Navigation

Abbildung 3.2: Farbschema aus Komplementärfarben

## 3.2 Softwarestack

Bei der Vielseitigkeit der verschiedenen Anforderungen ist die Wahl der zu verwendenden Technologien nicht unerheblich. Programmiersprachen und Frameworks mussten sorgfältig ausgewählt werden, um für jedes Teilproblem eine passende Lösung zu entwickeln.

Die erste wichtige Unterteilung fand zum Einen in ein Frontend statt, welches die Schnittstelle zu den NutzerInnen darstellt und zum Anderen in ein Backend, das die vom System verwendeten Daten speichert, manipuliert und diese nach Aufbereitung mit entsprechenden Algorithmen

an das Frontend schickt (s. Abbildung 3.3). Eine genaue Beschreibung über den Aufbau und die Funktion von Front- und Backend wird in Kapitel 4 gegeben.

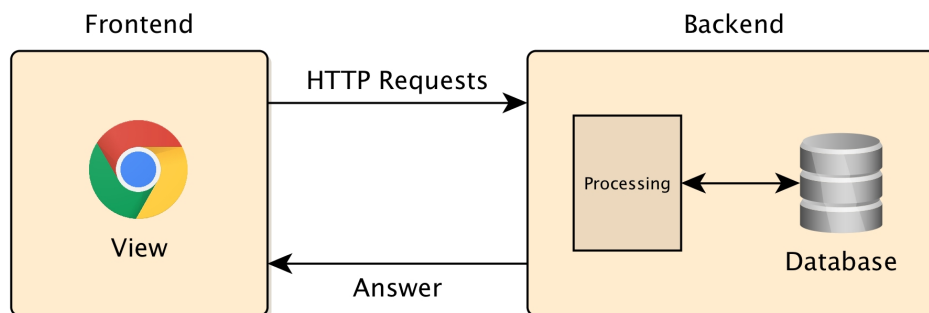


Abbildung 3.3: Kommunikation zwischen Front- und Backend

Diese Aufteilung ist in der Webentwicklung weit verbreitet[13] und bietet einige Vorteile gegenüber einem komplett integriertem Gesamtsystem:

- Klare Trennung von Darstellung und Datenverarbeitung
- Einfachere Fehlersuche durch kleinere Komponenten
- Bei der Entwicklung zusätzlicher Frontends (z.B. für Android oder iOS) muss nur ein neuer Teil des Softwaresystems entwickelt werden, während das Backend gleich bleibt.
- Arbeitsaufteilung im Entwicklerteam (relevant bei Weiterentwicklung der Applikation mit mehreren Personen)

### 3.2.1 Verwendete Technologien

Bei den verschiedenen zu realisierenden Zielen gibt es für jedes Teilziel Technologien, die Vorteile gegenüber Anderen bieten. Die Verwaltung von Datenbanken ist zum Beispiel besser mit einer server-seitigen Skriptsprache zu implementieren, als mit den Möglichkeiten, die das Frontend bietet. Im Web Frontend dagegen sind gewisse Technologien wie *HTML*, *CSS* und *JavaScript* Standard, die zwangsläufig verwendet werden müssen. Daher wird in der Applikation eine Vielzahl verschiedener Technologien verwendet, eine Übersicht ist hier gegeben:

- Backend: *Python* mit den Frameworks *Flask*, *Spacy* und *Sqllalchemy*
- Frontend: *HTML*, *CSS*, *JavaScript* und *AngularDart*
- Kommunikation zwischen Front- und Backend im *JSON* Format
- Deployment mit *Apache* auf einem *AWS* Server

Im Folgenden werden die einzelnen Technologien und die Gründe für deren Verwendung beschrieben.

#### 3.2.2 Backend

Um die Entwicklung des Backends möglichst einfach zu halten, wurde zunächst untersucht, ob ein fertiges Backend (Backend-as-a-Service) verwendet werden kann. Hierfür kämen z.B. fertige Services wie MongoDB oder Firebase mit integrierter NoSQL Datenbank infrage[8]. Zunächst wurde Firebase genauer untersucht und in Verbindung mit dem Frontend getestet. Prinzipiell war es möglich eine Verbindung mit dem Frontend herzustellen (Daten in der Firebase Datenbank können mit HTTP Requests geschrieben und gelesen werden). Die Möglichkeiten des Backends beschränkten sich allerdings auf Datenmanipulation und Cloud Code (server-seitiges JavaScript in Firebase).

Ein Problem dabei war, dass komplexere Logik wie das *Natural Language Processing* nur entweder im Frontend oder im Backend mit *JavaScript* implementiert werden konnte. NLP Frameworks, wie es sie z.B. in *Python* oder *Java* gibt, konnten somit nicht verwendet werden. Wurde der Text im Frontend in einzelne Wörter zerlegt und diese in der Firebase Datenbank nachgeschlagen, musste zudem für jedes Wort ein eigener HTTP Request geschickt werden, was zu unbefriedigenden Antwortzeiten führte.

Aufgrund der beschriebenen Probleme wurde parallel ein alternatives Backend in *Python* implementiert. Die Entwicklung der Softwarearchitektur wurde an den *Representational State Transfer* (REST) angelehnt. Dieser beschreibt Prinzipien und Restriktionen für die Konzeption von Webarchitekturen[13]. Diese Lösung brachte zwar wiederum eigene Schwierigkeiten und Probleme mit sich (s. Abschnitt 3.2.4), wurde aber für die kurzfristig bessere und schneller zu entwickelnde Variante erachtet und daher weiter verfolgt. Da in der Praxis die Entwicklung und Wartung eines eigenen Backends einen erheblichen Mehraufwand bedeuten kann, wäre weiterführend auch eine kombinierte Lösung denkbar. Hierfür könnte ein Backend-as-a-Service wie Firebase benutzt, aber zwischen Front- und Backend noch eine weitere Schicht eingesetzt werden, welche Programmlogik enthält und NLP Berechnungen durchführen kann.

Die Einhaltung der REST Restriktionen (z.B. muss das Backend zustandslos sein und die Anfragen müssen ressourcenorientiert formuliert werden) wurden so gut es im Rahmen der Arbeit möglich war eingehalten. Allerdings konnte die Konzeption einer konsequenten Softwarearchitektur nicht im Vordergrund stehen, da die entwickelte Applikation lediglich als Prototyp betrachtet werden muss. Im Folgenden werden die Technologien, die für das Python Backend benutzt wurden vorgestellt.

**Python** Die Verwendung von *Python*[26] im Backend brachte einige Vorteile für die Effizienz der Applikation, sowie für eine gute Strukturierung des Projekts, da ein großer Teil der Programmlogik im Backend abgebildet und somit die Komplexität des Frontends entlastet werden konnte. Zudem konnten nützliche Frameworks für NLP oder für die Verarbeitung der HTTP Requests benutzt werden.

```
@app.route ( '/ user / word / add ' , methods = [ ' POST ' ] )
def user_add_word ( ) :
...
```

Abbildung 3.4: Definition von *Flask* Routen

**Flask** ist ein *Python* Framework für die Webentwicklung. Es kann z.B. eingesetzt werden, um Webseiten an Clients auszuliefern, allerdings auch um eine API zu entwickeln, welche nur Daten an einen Client schickt anstatt ganzer Webseiten[15]. Bei der Entwicklung einer Flask Applikation wird zunächst im Python Skript ein Applikationsobjekt angelegt. Die Applikation kann in Python konfiguriert und mit einem Funktionsaufruf gestartet werden.

Alle beim Server ankommenden HTTP Requests werden von *Flask* automatisch verarbeitet. Das Verhalten für eigene Routen kann durch annotierte Funktionsdefinitionen angepasst werden. Abbildung 3.4 zeigt ein Beispiel, wie das Erstellen einer Ressource mit einem HTTP Request an eine selbst definierte Methode weitergeleitet wird. Zudem kann im `methods` Array definiert werden für welche HTTP Methoden die Route gilt.

In den zu den Routen definierten Methoden können Backend Services (s. Abschnitt 4.2) aufgerufen werden, welche die Daten der Applikation abfragen und entsprechend aufbereitet zurückgeben.

**Spacy** Für die Erledigung der linguistischen Aufgaben in der Applikation wurde nach einem effektiven NLP Parser gesucht. Hierfür wurde erst das *Natural Language Toolkit* (NLTK)[20] ausprobiert. Hierbei handelt es sich um eine Reihe von Python Modulen für die Durchführung von NLP Aufgaben. Es wurde Open Source und vor allem für akademische Zwecke entwickelt und wurde für die Verwendung in der Applikation als geeignet befunden.

Zusätzlich wurde der *Spacy* Parser untersucht, ein weiteres Python NLP Framework. Bei der Erstellung von *Spacy* wurde unter anderem der Fokus auf Performanz gelegt, entwickelt wurde es in *Cython* (eine zu Python kompatible Sprache, welche aber bei der Kompilierung nach C übersetzt wird), was zu schnellerer Ausführung im Vergleich zum Python Interpreter führt.

Eigenes Testen und andere Quellen haben gezeigt, dass der *Spacy* Parser aufgrund seiner Effizienz[30] und Benutzbarkeit für die Aufgabe sehr gut geeignet ist. Er wurde daher in der Applikation zum Parsen des Eingabetexts, als Tokenizer und zur Bestimmung der Part-of-Speech Tags anstelle von NLTK benutzt.

### 3.2.3 Frontend

Für die Entwicklung einer Webapplikation sind viele Technologien schon vorgegeben. Der Einsatz von *HTML*, *CSS* und *JavaScript* auf der Client Seite ist zur Darstellung der Applikation im Browser notwendig und praktisch unverzichtbar. Im Gegensatz zu einer normalen Website wurde hier eine Single-View-Applikation entwickelt, d.h. anstatt bei Nutzerinteraktionen eine neue HTML Datei zu laden, verändert die Applikation den Inhalt der Seite (das Document

Object Model, DOM) dynamisch mit Skripten[22].

JavaScript könnte als einzige Frontend Sprache verwendet werden. Das Modell der Applikation würde dann mit verschiedenen Skripten abgebildet werden und die Kommunikation mit dem Backend würde über AJAX Anfragen erfolgen. Da JavaScript aber über keine starke Typisierung verfügt und eine Strukturierung der Anwendung durch Klassen nur bedingt möglich ist, wurden Alternativen für die Frontendsprache gesucht. Eine Möglichkeit hier ist die Generierung von JavaScript aus anderen Sprachen. Hierbei wird Code in einer anderen Hochsprache geschrieben und für die Ausführung im Browser mit einem Compiler nach JavaScript übersetzt. Dieser Ansatz wird von einigen Sprachen oder Frameworks unterstützt, z.B. CoffeeScript, TypeScript, GWT (Java) oder AngularDart. Nach dem Erwerb eines Überblicks über diese Technologien entschied ich mich dafür, die von Google entwickelte Sprache *Dart* und das *AngularDart* Framework zu verwenden, deren Funktionsweise im folgenden Abschnitt näher erklärt wird.

#### AngularDart

Dart ist eine von Google entwickelte Programmiersprache, die z.B. in einigen Google Tools (z.B. AdWords oder interne Tools für Marketing und Sales) verwendet wird. Sie wird nicht in erster Linie für die Webentwicklung benutzt, sondern kann durch entsprechende Kompilierung z.B. auch für die Backend Entwicklung oder für iOS und Android Applikationen verwendet werden.

Für die Erstellung einer Webanwendung mit Dart wird das Framework *AngularDart*[3] benutzt. Dieses wurde nach dem Vorbild der Technologien *AngularJS* und *Angular* entwickelt, wobei Ersteres mit reinem JavaScript programmiert wird und bei *Angular* die Programmiersprache *TypeScript* zum Einsatz kommt. Bei *AngularDart* wird der in *Dart* erstellte Code beim Build mit dem *dart2js* Compiler nach *JavaScript* übersetzt.

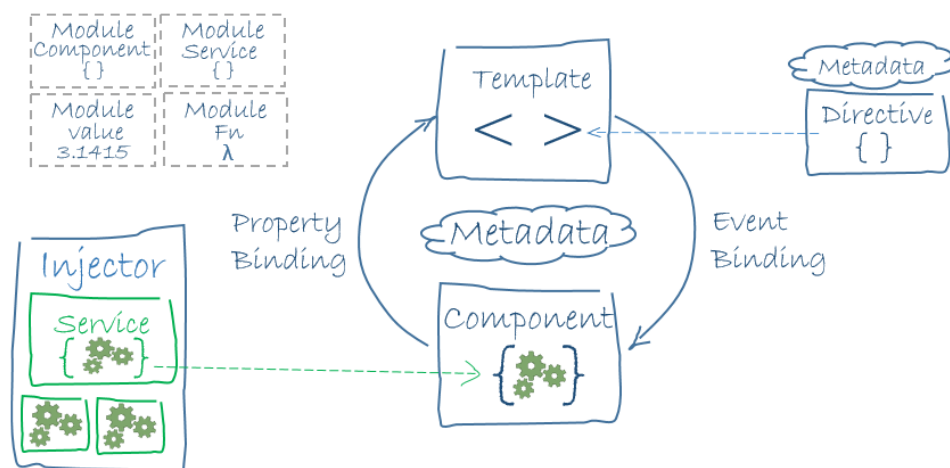


Abbildung 3.5: Funktionsweise einer AngularDart Webanwendung[3]



Abbildung 3.5 gibt einen Überblick über die Funktionsweise einer AngularDart Webanwendung. Bei dieser Architektur wird die Programmlogik in *Services* abgebildet welche durch *Dependency Injection* in anderen Programmteilen verwendet werden können. Das Layout der Website wird von den eingebundenen *Komponenten* bestimmt. Eine AngularDart Komponente besteht aus einem HTML *Template* und einer (Dart) Komponentenklasse und kann in jedem anderen HTML Template eingebunden werden. Es gibt sowohl von AngularDart vordefinierte Komponenten als auch die Möglichkeit, eigene Komponenten zu entwickeln.

Das Template verwendet einen eigenen Syntax bei dem Angular *Direktiven* in HTML aufgerufen werden können. Der Template Syntax bietet z.B. die Möglichkeiten mit *Property Binding* auf Daten der Komponentenklasse zuzugreifen, oder mit den Direktiven *ngIf* oder *ngFor* Daten bedingt darzustellen.

Werden in der Komponentenklasse Daten verändert, wird dies automatisch von AngularDart detektiert. Infolgedessen wird durch das *Property Binding* die Anzeige in der HTML Seite automatisch angepasst. Eine genaue Beschreibung der Funktionsweise liefert die AngularDart Dokumentation[3].

### 3.2.4 Deployment

Der Output der AngularDart Applikation ist eine Website aus HTML und JavaScript. Um diese verfügbar zu machen, muss sie auf einem Webserver gehostet werden, der mittels Namensauflösung über DNS mit einer URI erreichbar ist. Hier wurde für den Prototyp der Applikation über die Amazon Web Services (AWS) ein kostenloser Zugang angelegt, mit dem eine virtuelle Maschine (VM) erstellt werden konnte (EC2 Service[2]), die man z.B. über SSH erreichen kann. Sowohl die AngularDart Applikation, als auch das Python Backend wurden auf die VM übertragen, jedoch auf verschiedenen Ports gestartet, das Frontend mit Apache2 auf Port 8001, das Backend mit Python3 auf Port 8000. Alle im Frontend ausgehenden Anfragen rufen das Backend über die Server URI des Amazon Servers auf Port 8000 auf.

Adresse	Port	Applikation
http://ec2-<...>.compute.amazonaws.com:8001	8001	Frontend, mit Apache2 gehostet
http://ec2-<...>.compute.amazonaws.com:8000	8000	Python3 Flask Backend

Tabelle 3.2: Die URIs beginnen mit „ec2“ und enden mit der Adresse von Amazon AWS. Front- und Backend wurden mit unterschiedlichen Ports auf dem gleichen Server gehostet.

Der beschriebene Weg wurde gewählt um die Webapplikation schnell und einfach verfügbar zu machen. Netzwerksicherheit wurde im Rahmen der Arbeit vernachlässigt, da der Schwerpunkt auf der Entwicklung der User Interfaces und auf den NLP Aufgaben lag. So wird z.B. nur das HTTP Protokoll verwendet, nicht *HTTPS*. Die Verwendung von zwei verschiedenen Ports für die Anwendung führt zu Problemen, wenn man Sicherheitskonzepte wie die HTTP Authentifizierung umsetzen will. Die meisten Webbrowser implementieren die *Same Origin Policy* und beschränken die Kommunikation zwischen verschiedenen Quellen (*Cross-Origin*

### 3 Anforderungsanalyse und Spezifikation

*Resource Sharing*[31]). Als Behilfslösung wurden für die Authentifizierung Mailadresse und Passwort als Form Data von HTTP POST Requests mit übertragen. Dies führte auch dazu, dass in allen Anfragen, bei denen die Authentifizierung übermittelt werden musste, die POST Methode benutzt wurde (s. Tabelle 4.2) und nicht die für die REST Architektur üblichen Methoden wie *PUT* oder *DELETE* eingesetzt werden konnten.

Bei einer Weiterentwicklung der Applikation wäre hier denkbar, z.B. entweder auch das mit AngularDart erstellte Frontend als Teil der *Flask* Applikation zu hosten (und somit beides von einer Quelle aus und auf einem Port auszuliefern), oder ein Backend-as-a-Service wie Firebase (s. Abschnitt 3.2.2) für die sicherheitskritischen Ressourcen zu verwenden, da diese Backends von Haus aus Authentifizierung anbieten.

## 4 Entwicklung der Anwendung

Im Rahmen dieser Arbeit wurde die beschriebene Webapplikation prototypisch entwickelt. Die dazu nötige Planung und die verwendeten Technologien wurden in den vorherigen Kapiteln geschildert. Nun wird auf den Prozess der Entwicklung und auf die detaillierte Struktur der Applikation eingegangen. Ein Bildschirmfoto des fertigen Prototyps ist in Abbildung 4.1 zu sehen.

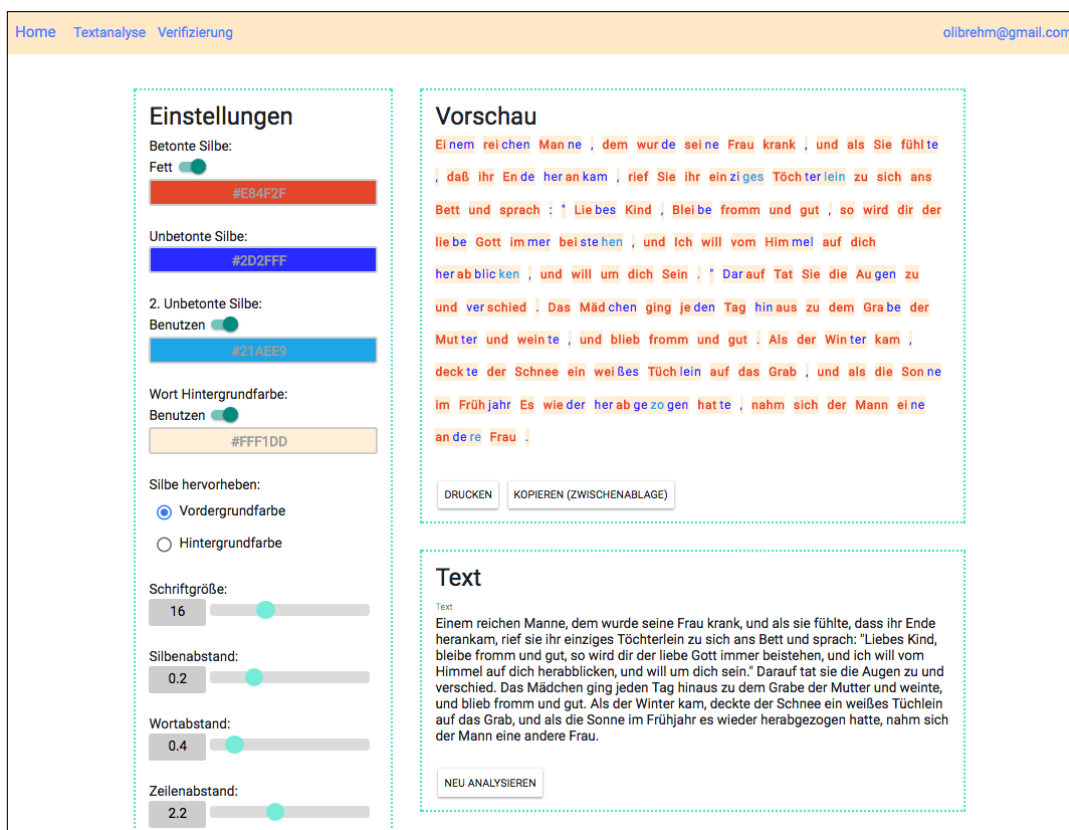


Abbildung 4.1: Bildschirmfoto der Textanalyse im fertigen Prototyp

Die Entwicklung des Systems wurde in drei Teilprojekte untergliedert:

1. Erstellung der Basisdatenbank aus dem Wörterbuch CELEX2
2. Entwicklung des Backends mit Python
3. Entwicklung des Frontends mit gängigen Webtechnologien und AngularDart

Dabei wurde Schritt 1 als Erstes durchgeführt, die Verfügbarkeit der Basisdatenbank war Voraussetzung, um mit der Entwicklung von Front- und Backend zu beginnen. Als eine erste Version der Datenbank vorlag wurde zunächst mit der Entwicklung eines minimalen Backends begonnen, welches die Kernfunktionalität - das Analysieren von Texten - beherrschte. Damit konnte eine erste Version des Frontends erstellt werden, um einen ersten Eindruck zu bekommen, wie alle Technologien miteinander funktionieren.

Im Anschluss konnten Front- und Backend parallel entwickelt werden. Es wurden immer wieder Funktionen im Backend entwickelt, woraufhin das Frontend angepasst wurde, um den NutzerInnen diese Funktionen anzubieten. Zu diesem Zeitpunkt wurde auch die Basisdatenbank einige Male überarbeitet, so ist das Gesamtsystem Schritt für Schritt gewachsen. Im Folgenden wird näher auf die drei Teilprojekte eingegangen.

### 4.1 Erstellung einer Basisdatenbank

Für die Segmentierung von Wörtern in Silben mit Betonung wird eine SQLite Datenbank verwendet, deren Entwicklung hier beschrieben wird. Als Grundlage für die Datenbank dient das digitale Lexikon CELEX2, wie in Kapitel 2.2.2 beschrieben. Folgende Informationen wurden dem CELEX entnommen und in der Datenbank gespeichert:

**Worttext** Hier wurde darauf geachtet, eine einheitliche orthographische Form der Wörter durchzusetzen. Wörter bestehen nur aus Kleinbuchstaben (auch die Anfänge von Nomen und Eigennamen). Außerdem wird hier auf Umlaute verzichtet, „ä“ wird durch „ae“ ersetzt, „ö“ durch „oe“, „ü“ durch „ue“ und „ß“ durch „ss“. Beispielsweise wird das Wort *Häuser* zu *haeuser*. So kann beim Nachschlagen garantiert werden, dass ein Wort nicht wegen uneinheitlicher Schreibweise nicht gefunden wird, sofern alle Umlaute vorher ersetzt wurden.

**Wortart** Um Doppeldeutigkeiten zu vermeiden wird ein Eintrag nur eindeutig über das Paar aus Worttext und Wortart identifiziert. Viele Grundformen von Verben kommen zum Beispiel auch als Nomen vor, werden aber dann groß geschrieben („sie sollte die Maschine *verwenden*“ gegen „das *Verwenden* der Maschine“). Die Bezeichner für Wortarten sind in verschiedenen Systemen wie dem CELEX und dem NLP Parser *Spacy* unterschiedlich, daher wird hier eine eigene - im Gesamtsystem konsistente - Benennung der Wortart eingeführt (die Benennung orientiert sich an den Namen der Spacy Tags, verwendet jedoch zur Unterscheidung Klein- statt Großschreibung, s. Abbildung 4.1).

**Silbentrennung** Die Silbentrennung enthält die einzelnen Silben, getrennt durch einen Querstrich („-“). Im Gegensatz zum Worttext erhält dieser Eintrag Groß- und Kleinschreibung, sowie alle Umlaute. Die Silbentrennung des Wortes *Möglichkeit* ist also *Mög-lich-keit*.

Wortart	Benennung/Tag
Nomen	<i>noun</i>
Eigenname	<i>propn</i>
Verb	<i>verb</i>
Hilfsverb	<i>aux</i>
Adjektiv	<i>adj</i>
Adverb	<i>adv</i>
Artikel	<i>det</i>
Adposition	<i>adp</i>
Pronomen	<i>propn</i>
Konjunktion	<i>conj</i>
Zahl	<i>num</i>
Partikel	<i>part</i>
Zahl	<i>num</i>

Tabelle 4.1: Benennung der Wortarten in der Applikation

**Betonungsmuster** Das Betonungsmuster identifiziert eine Silbe als Hauptbetonung im Wort. Dieser Eintrag ist ein String aus aneinandergereihten „0“ und „1“ Werten. Die „1“ bezeichnet dabei die betonte Silbe, die „0“ stellt eine unbetonte Silbe dar. Nebenbetonungen werden im Rahmen dieser Arbeit vernachlässigt (s. Abschnitt 2.2.2). Das Betonungsmuster besteht nur aus einer einzigen „1“ - der Hauptbetonung - während alle anderen Silben unbetont („0“) bleiben. Beispielsweise resultiert das Wort *hervorragend* mit der Silbentrennung *her-vor-ra-gend* in das Betonungsmuster „0100“.

**Lemma** Das Lemma ist die Grundform eines Wortes. Diese unterscheidet sich je nach Wortart, beispielsweise ist das Lemma eines Verbs dessen Infinitiv und das Lemma eines Pronomens die erste Person Singular.

Neben den Worteinträgen sollen in der Datenbank noch Informationen über die Herkunft der Wörter gespeichert werden. Neben den Basiseinträgen, die aus dem CELEX extrahiert werden, wird den NutzerInnen später die Möglichkeit gegeben, Einträge zur Datenbank hinzuzufügen. Um diesen Vorgang nachvollziehbar zu machen gibt es eine zweite Tabelle, in welcher Einträge für neu hinzugefügte Wörter mit Zeitstempel angelegt werden. Das folgende Schema 4.2 zeigt die Struktur der Wortdatenbank.

Alle in der Worttabelle beschriebenen Einträge sind auch im CELEX vorhanden. Der folgende Abschnitt beschreibt, wie die Daten aus dem Lexikon extrahiert und in der Wortdatenbank gespeichert wurden.

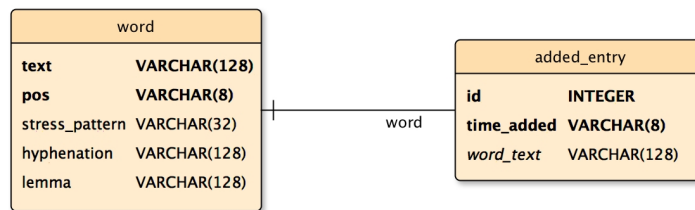


Abbildung 4.2: Schema der Wortdatenbank (*word.db*)

### 4.1.1 Implementierung

Für die Generierung der Datenbank wurde das python Projekt *celex2db* erstellt, welches in „/backend/celex2db“ zu finden ist (alle Pfadangaben in diesem Abschnitt beziehen sich auf dieses Verzeichnis). Die Daten im CELEX sind über mehrere Dateien verteilt. So gibt es eine Teilung in Sprachen (für die Datenbank werden nur deutsche Wörter benutzt, zu finden in „celex2/german“), Wörter und Lemmata sowie in Orthographie, Phonologie und Morphologie. Die Teilung in Wort und Lemma spart Speicherplatz. Das Lemma eines Verbs zum Beispiel, ist dessen Infinitiv. Zur Flexionsform *gehst* gehört also das Lemma *gehen*. Viele Eigenschaften des Verbs *gehen* beziehen sich auf alle Flexionsformen, so müssen diese nicht für alle Formen redundant gespeichert werden.

Die Einträge in den Wortlisten des CELEX verweisen jeweils mit einer ID auf den zugehörigen Eintrag in der Lemma-Liste. Da in der zu entwickelnden Datenbank auch das Lemma gespeichert werden sollte, mussten sowohl Wort- als auch Lemma-Listen geparkt werden.

Das Hauptskript *celex2db.py* führt die folgenden Schritte aus:

1. Parsen der orthographischen Lemma-Liste
2. Parsen der orthographischen und phonologischen Wort-Listen
3. Generierung von Einträgen aus Wort- und Lemma-Listen und Schreiben in die Datenbank

Während der Ausführung des Skripts werden schrittweise Fortschrittsangaben mit Fehlern ausgegeben, da das Parsen und Erstellen der Datenbank einige Zeit benötigt (mehr dazu im Abschnitt 4.1.2).

Die Klasse *CelexDictionary* im Skript *celex.py* bietet die Methoden zum Parsen der Wort- und Lemma-Listen. In der Methode zum Parsen der Lemmata wird die CELEX Datei *gml.cd* verarbeitet, die morphologische Informationen zu den Lemmata enthält. Jede Zeile enthält ein Lemma, welches mit ID, Text und Wortart gespeichert wird.

Beim Parsen der Wort-Listen werden die beiden Dateien *gpw.cd* und *gow.cd* verarbeitet, welche phonologische bzw. orthographische Informationen zu allen enthaltenen Flexionsformen der Wörter enthalten. Die IDs und Reihenfolge in beiden Wort-Listen stimmen überein, sodass beide Listen gleichzeitig verarbeitet werden konnten. Aus der phonologischen Liste wurden der Text des Wortes, der Verweis auf das zugehörige Lemma sowie die phonologische Darstellung entnommen. Die orthographische Liste lieferte lediglich die Silbentrennung.

Die geparsten Lemma- und Wortstrukturen wurden zusammengeführt und daraus Einträge für die Datenbank erstellt, welche dann mit dem Skript „writeSQLite.py“ in die Datenbank übertragen wurden. In einem frühen Entwicklungsstadium wurden hier SQL Instruktionen verwendet, um die Datenbankeinträge zu erstellen. Nach der Entwicklung des `DictionaryService` im Backend (genauer beschrieben im Abschnitt 4.2.2) wurde später direkt auf diesen zugegriffen, um redundanten Code zu vermeiden. Ein einfacher Funktionsaufruf der Methode `add_word` des `DictionaryService` fügt den Eintrag der Wortdatenbank hinzu.

### 4.1.2 Ergebnisse

Ein Wort sollte durch den Worttext und die Wortart eindeutig identifizierbar sein. Daher wurde in der Datenbank die Kombination der Felder *text* und *pos* (Part-of-Speech/Wortart) als Primärschlüssel gewählt. Ein doppeltes Einfügen eines Wortes in die Datenbank verletzt den SQL *UNIQUE Constraint* und ist somit nicht möglich. Die dadurch von Python geworfene Exception wird in diesem Fall abgefangen und die `add_word` Methode liefert `None` zurück. Trotz der Wortidentifizierung anhand von zwei Merkmalen beinhaltet der CELEX teilweise redundante Informationen. 4 356 Einträge verursachen aufgrund des *UNIQUE Constraint* Fehler, da sie mit gleichem Text und gleicher Wortart schon in der Datenbank vorhanden waren und wurden somit übersprungen. Bei einem kompletten Neuaufbau der Datenbank aus dem CELEX werden 360 243 Einträge (Flexionsformen von Wörtern) hinzugefügt. Die Datenbank wurde in der SQLite Datei „/backend/db/words.db“ angelegt.

## 4.2 Backend

Das in Python mit dem Web-Framework Flask entwickelte Backend verarbeitet sämtliche, vom Frontend gesendeten HTTP Requests (s. Abschnitt 3.2.2). Es sendet entweder eine Antwort mit den geforderten Daten im JSON Format oder einen HTTP Fehlercode mit Fehlermeldung. Das Backend Projekt befindet sich in „/backend/API“. Es wurde in die drei Services *DictionaryService*, *UserService* und *VerificationService* strukturiert. Das Hauptskript „LinguisticTextAnnotation.py“ verwaltet diese Services, startet die Flask Applikation und definiert die einzelnen Routen für HTTP Requests (diese werden im folgenden Abschnitt aufgelistet). Abbildung 4.3 zeigt eine Übersicht der Struktur des Backends.

### 4.2.1 HTTP Routen

Tabelle 4.2 bietet eine Übersicht über die verfügbaren Routen, welche die HTTP Requests verarbeiten. Zudem werden kurz der Zweck und die Parameter der Route beschrieben. Ein X in der Spalte Authentifizierung bedeutet, dass die Mailadresse und das Passwort der Nutzerin oder

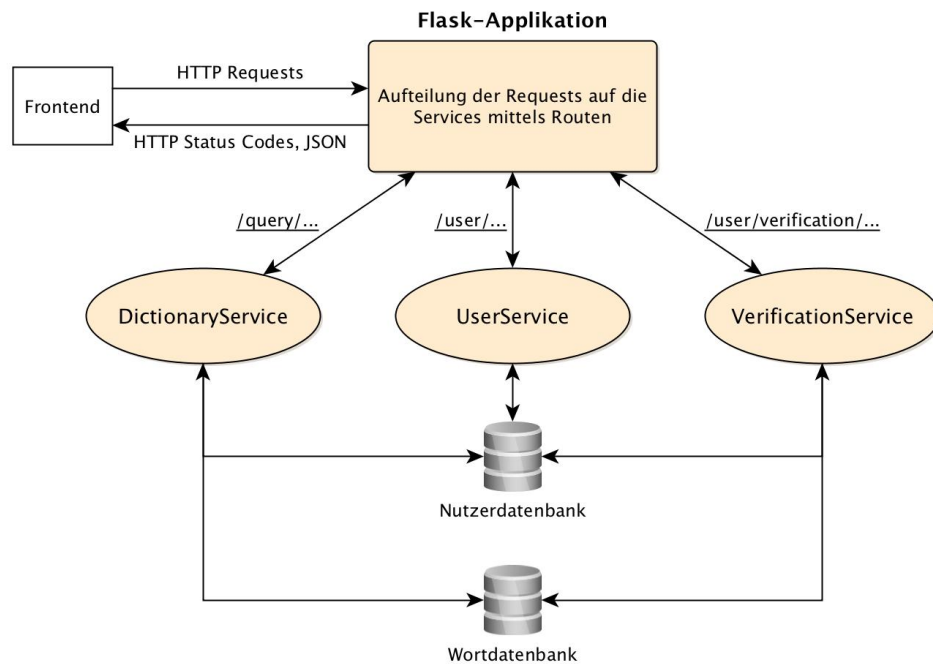


Abbildung 4.3: Aufbau und Kommunikation des Backends, mit Services und Datenbanken

des Nutzers übergeben werden muss. Diese werden nicht extra in der Spalte Parameter aufgeführt (außer in der „/user/authenticate“ Route, diese dient ausschließlich dazu, die übermittelten Anmeldedaten der NutzerInnen zu prüfen).

Die bei „user/configuration/“ nicht aufgeführten *options* bestehen aus allen für die Annotationsvorlage gespeicherten Einstellungen, z.B. aus den Parametern *stressed\_color*, *unstressed\_color*, *word\_background*, *alternate\_color*, *syllable\_separator* oder *line\_height*.

### 4.2.2 Dictionary Service

Das Modul *DictionaryService.py* verarbeitet Anfragen zur Textanalyse, verwaltet die Wortdatenbank und generiert Vorschläge für die manuelle Segmentierung von Wörtern. Das Schema der Wortdatenbank zeigt Abbildung 4.2. Folgende Klassen werden definiert:

- **Word**: Entspricht Tabelle *word* in der Wortdatenbank (Abbildung 4.2) und speichert die Merkmale eines Wortes.
- **AddedEntry**: Enthält einen Verweis (Fremdschlüssel) auf ein hinzugefügtes Wort und beinhaltet dessen Erstellungszeitpunkt.
- **Segmentation**: Modelliert einen Segmentierungsvorschlag. Die Klasse enthält Worttext, Segmentierung und Betonungsmuster. Außerdem speichert sie den Namen und die Herkunft der Quelle des Vorschlags (z.B. MARY TTS).



Route	Parameter	Methode	Auth.
/query/word	text, pos	GET	
/query/text	text	POST	
/query/segmentation	word	GET	
/user/register	email, password, first_name, last_name, is_expert	POST	
/user/authenticate	<i>email, password</i>	POST	X
/user/delete	id	POST	X
/user/word/add	text, stress_pattern, hyphenation, pos	POST	X
/user/word/delete	id	POST	X
/user/word/list	-	POST	X
/user/configuration/add	name, <i>options</i>	POST	X
/user/configuration/update	id, name, <i>options</i>	POST	X
/user/configuration/list	-	POST	X
/user/configuration/delete	id	POST	X
/user/text/add	title, text	POST	X
/user/text/delete	id	POST	X
/user/text/list	-	POST	X
/user/verification/query	-	POST	X
/user/verification/submit	id, stress_pattern, hyphenation	POST	X

Tabelle 4.2: Backend Routen für HTTP Requests mit Parametern und Methode

## Funktionen der Wortdatenbank

- Die Methode `add_word` bekommt Worttext, Betonungsmuster, Wortteilung, Lemma und Wortart übergeben (s. Abschnitt 4.1), fügt das Wort der Datenbank hinzu und liefert eine `Word` Instanz zurück. Im Falle eines Fehlers oder falls das Wort schon existiert, gibt die Methode `None` zurück.
- `query_word` schlägt ein Wort mit den Parametern Worttext und Wortart in der Datenbank nach und gibt das annotierte Wort im JSON Format (*text, stress\_pattern, hyphenation, lemma, pos*) zurück. Als Parameter kann optional die eingeloggte Nutzerin oder der eingeloggte Nutzer übergeben werden. Ist das der Fall, wird zuerst überprüft, ob im Nutzeraccount (s. Abschnitt 4.2.3) eine manuelle Segmentierung für das Wort gespeichert ist und falls ja, diese zurückgegeben. Werden in der Wortdatenbank mehrere Wörter gefunden, die dem gesuchten Worttext entsprechen, wird zusätzlich die Wortart geprüft und der passende Eintrag (bzw. der erste Eintrag, falls kein Wort mit der entsprechenden Wortart gefunden wird) zurückgegeben.

### Textanalyse

Die Methode `query_text` bekommt den gesamten zu analysierenden Text übergeben (sowie optional die eingeloggte Nutzerin oder den eingeloggten Nutzer für manuelle Worteinträge). Der Text wird zunächst mit dem Spacy Parser analysiert. Dieser zerlegt den Text in einzelne Wörter und Trennzeichen und gibt eine Liste von Tokens zurück. Nun wird über die Liste der Tokens iteriert und dabei wiederum eine Liste von JSON Einträgen generiert, welche am Schluss von der Methode zurückgegeben wird. Jedes spacy Token liefert den Worttext `text`, die Wortart `pos_` und das Lemma `lemma_`. Das Token wird wie folgt verarbeitet:

1. In jedem Fall wird ein neuer JSON Eintrag angelegt und die Felder `text`, `pos` und `lemma` mit den entsprechenden Inhalt des Spacy Tokens gefüllt.
2. Enthält der Worttext einen Linebreak „`\n`“, so wird das JSON Feld `type` mit `linebreak` belegt.
3. Enthält der Worttext einen Trennstrich „`-`“, so handelt es sich um ein zusammengesetztes Wort (z.B. *100-Millionen-Einwohner-Staat*). Hier werden alle Trennstriche durch Leerzeichen ersetzt. Der resultierende String wird dann erneut mit Spacy analysiert und die Tokens dann entsprechend verarbeitet. Hinter jedem Teilwort (außer dem Letzten) wird manuell der Trennstrich als eigenes Token eingefügt.
4. Entspricht der Wortart-Tag einem String aus `[ 'X' , ' PUNCT' , ' NUM' , ' SPACE' ]`, so handelt es sich um eine dem Parser unbekannte Zeichenfolge, eine Zahl, ein Satz- oder Leerzeichen. Diese Tags werden nicht nachgeschlagen sondern mit dem JSON `type` Feld `ignored` der Rückgabeliste hinzugefügt.
5. Trifft keiner dieser Sonderfälle zu, wird das Wort mit der Methode `query_word` nachgeschlagen. Wird dieses gefunden (Rückgabewert nicht `None`), wird der `type` auf `annotated_word` gesetzt. Das JSON Feld `annotation` enthält dann die Informationen des annotierten Wortes (von `query_word` zurückgegeben). Ist das Wort nicht in der Datenbank, so enthält das Feld `type` den Wert `not_found`.

### Generierung von Segmentierungsvorschlägen

Der *DictionaryService* bietet auch die Möglichkeit, automatisch Segmentierungsvorschläge zu generieren. Diese werden im Frontend dazu benutzt, den NutzerInnen bei der manuellen Wortsegmentierung die Arbeit zu erleichtern, indem das Wort schon in Silben zerlegt und versucht wird, automatisch die Wortbetonung zu bestimmen. Der Standardvorschlag, den das System liefert ist einfach die Silbentrennung, die der Python Silbentrenner *Pyphen* liefert. Diese ist in vielen Fällen korrekt und bietet in jedem Fall einen Anhaltspunkt für die Silbentrennung des Wortes. Zudem wird beim Standardvorschlag ein Betonungsmuster generiert, welches immer die erste Silbe im Wort betont. Auch das ist in der deutschen Sprache oft der Fall, so können Betonung und Silbentrennung bei der manuellen Analyse einfach bestätigt werden.

Bei der Abfrage der Vorschläge im Backend wird eine Liste von Segmentierungsvorschlägen zurückgeliefert. Der Standardvorschlag mit *Pyphen* ist nur eine Möglichkeit. Hier wurde überlegt, aus mehreren externen Systemen, wie z.B. MARY oder BALLOON (s. Abschnitt 2.2.2), sowie später auch Duden und Anderen, Vorschläge zu sammeln und den NutzerInnen in einer Liste zu präsentieren, aus welcher der passendste Vorschlag ausgewählt werden kann. Exemplarisch wurde hier eine Anfrage an MARY geschickt und der resultierende Vorschlag in der Antwort an das Frontend mit eingebunden. Dieses System kann mit HTTP Requests angefragt werden. Das entsprechende Interface ist unter <http://mary.dfki.de:59125/process> zu erreichen. Tabelle 4.3 zeigt die bei der Anfrage verwendeten GET Parameter. MARY liefert eine Antwort im XML Format zurück, welche die phonologische Silbentrennung und die Betonung enthält. Das XML wird entsprechend geparst, womit dann ein neuer Segmentierungsvorschlag bestimmt werden kann.

Parameter	Wert
INPUT_TEXT	<i>das abzufragende Wort</i>
INPUT_TYPE	TEXT
OUTPUT_TYPE	ACOUSTPARAMS
LOCALE	de

Tabelle 4.3: GET Parameter für Anfrage an MARY TTS

### 4.2.3 User Service

Der *UserService* dient der Verwaltung der Nutzerdaten und der Authentifizierung. Nutzerkonten bestehen aus Mailadresse, Passwort und Vor- und Nachnamen der Nutzerin oder des Nutzers. Daneben werden die Ressourcen *user\_text*, *user\_word*, *text\_config* und *pos\_config* verwaltet. Welche Funktionen für die jeweilige Ressource zur Verfügung stehen, wird im Folgenden erklärt. Abbildung 4.4 zeigt das Schema der Nutzerdatenbank.

- Zur **Authentifizierung** wird überprüft, ob ein Nutzerkonto für die als Parameter übergebene Mailadresse und das übermittelte Passwort existiert.
- **Nutzertexte** (*user\_text*) können mit Text und Titel erstellt, sowie mit entsprechenden Anfragen abgefragt (als Liste aller Texte eines Nutzerkontos) oder entfernt werden.
- Auch für **Nutzerwörter** (*user\_word*, Einträge die durch die manuelle Segmentierung entstehen), können mit Text, Silbentrennung, Betonungsmuster und optional Wortart angelegt, sowie abgefragt (ebenfalls als Liste aller Wörter eines Nutzerkontos) oder gelöscht werden.
- Für **Annotationskonfigurationen** (*text\_config*) gibt es vier Funktionen: Sie können als Liste abgefragt oder mit der zugehörigen ID gelöscht werden, sowie angelegt und aktualisiert werden. Beim Anlegen und Aktualisieren werden sämtliche Attribute der Konfiguration (Name, Silbenfarben, Text Einstellungen, wie Silben- oder Wortabstand

## 4 Entwicklung der Anwendung

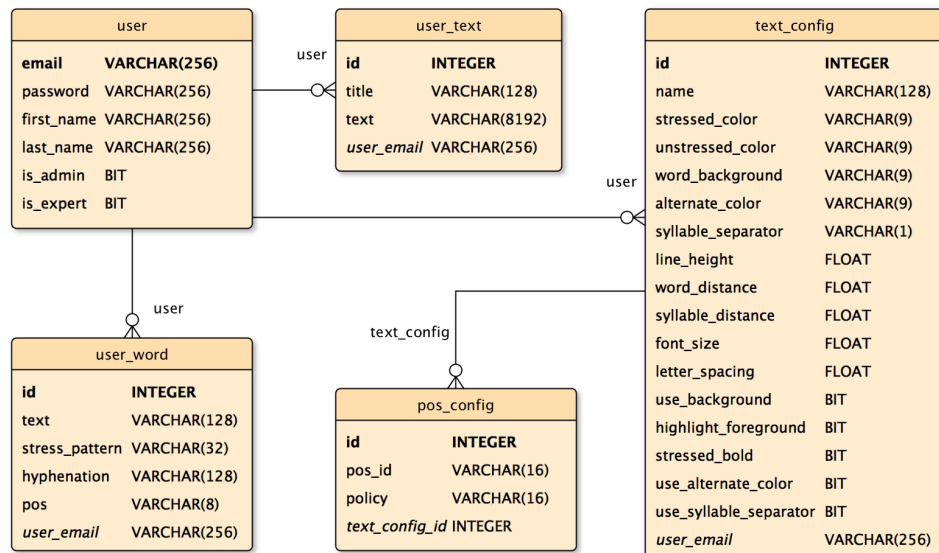


Abbildung 4.4: Schema der User Datenbank (*user.db*)

etc.) in der Anfrage mit übergeben. Auch die Konfigurationsliste für die Wortarten (*pos\_config*), die für jede Wortart die Einstellung, ob diese annotiert oder ignoriert wird speichert, wird in der Anfrage benötigt.

### 4.2.4 Verification Service

Von den NutzerInnen manuell erstellte Segmentierungen für Wörter können nicht direkt in die globale Wortdatenbank für alle NutzerInnen übernommen werden, da bei der Segmentierung Fehler gemacht werden können. Eine Verifizierung der Einträge durch mehrere NutzerInnen soll die Anzahl der fehlerhaften Einträge, die der Datenbank hinzugefügt werden minimieren. Der *VerificationService* definiert die Funktionen, mit denen zu verifizierende Segmentierungen abgefragt oder Verifizierungsvorschläge abgeschickt werden können (s. *Crowdsourcing* in Abschnitt 2.2.2).

Ein zu verifizierender Eintrag ist hierbei direkt ein *user\_word* (s. Abbildung 4.4), dass von einer Nutzerin oder einem Nutzer bei der manuellen Segmentierung erstellt wurde. Die Verifizierung (Klasse *VerificationProposal*) bezieht sich auf diesen Eintrag und enthält ebenfalls Silbentrennung und Betonungsmuster für das Wort. Nach der Übermittlung eines Verifizierungsvorschlags prüft ein Algorithmus, ob ausreichend Verifizierungen unterschiedlicher NutzerInnen für das Wort übereinstimmen (also Silbentrennung und Betonung identisch sind). Ist dies der Fall, wird das Wort in die globale Wortdatenbank übernommen und alle Verifizierungsvorschläge werden wieder entfernt. Im Folgenden wird der Verifizierungsalgorithmus (dieser wird dann verwendet, wenn ein neuer Verifizierungsvorschlag übermittelt wird) kurz vorgestellt:

1. Für den übermittelten Vorschlag wird ein *VerificationProposal* Objekt in der

Nutzerdatenbank angelegt.

2. Der Vorschlag erhält initial den `score` 0. Es wird über alle Vorschläge, die sich auf dasselbe Wort beziehen iteriert, dabei werden Silbentrennung und Betonungsmuster verglichen. Bei Übereinstimmung wird der `score` erhöht, die Gewichtung ergibt sich je nachdem, ob die Autorin oder der Autor des Vorschlags ein normaler Nutzer, Experte oder Administrator ist. Vorschläge von Administratoren erhalten drei Punkte, Experten-vorschläge zwei Punkte und die Vorschläge normaler NutzerInnen einen Punkt.
3. Wird die Zielwertung von 3 nicht erreicht, so wird abgebrochen. Bei der Übermittlung des nächsten Vorschlags kann so erneut überprüft werden, ob genug Verifizierungen einstimmig sind.
4. Bei einem Score von 3 wird das Wort in die globale Wortdatenbank übertragen. Bei der gegebenen Punkteverteilung geschieht das, sobald entweder ein Administrator, ein Experte und mindestens ein anderer, beliebiger Nutzer oder drei normale NutzerInnen identische Vorschläge übermitteln.
5. Sowohl die manuelle Segmentierung für das Wort, sowie alle Verifizierungsvorschläge werden nun nicht mehr benötigt und daher aus der Datenbank gelöscht.
6. Zudem wird ein Eintrag `AddedEntry` angelegt, der einen Verweis auf das verifizierte, neue Wort in der Wortdatenbank und einen Zeitstempel mit dem Erstellungszeitpunkt enthält. So kann nachvollzogen werden welche Wörter wann durch den Verifizierungsprozess der Datenbank hinzugefügt wurden.

## 4.3 Frontend

Das Frontend ist der den Nutzerinnen und Nutzern sichtbare Teil des Softwaresystems. Die Webapplikation ist zunächst wie eine normale Webseite aufgebaut, Inhalte der Komponenten werden aber dynamisch geladen (d.h. bei Nutzerinteraktionen wird keine HTML Seite neu aufgerufen, sondern der Inhalt der Seite wird direkt angepasst). Neben den standardmäßig im Web verwendeten Sprachen *HTML*, *CSS* und *JavaScript* wurde die Applikation größtenteils in *AngularDart* entwickelt (eine Beschreibung zu dieser Technologie, der Unterteilung von Komponenten in Klassen, Templates und Services sowie der Templatesprache von Angular, liefert Abschnitt 3.2.3).

Die Features des *AngularDart* Frameworks ermöglichen eine strukturierte, objektorientierte Architektur des Frontends. Die folgenden Abschnitte zeigen zunächst den Aufbau der Applikation und das verwendete Datenmodell, danach die Struktur aller größeren Komponenten.

### 4.3.1 Hauptkomponente der Applikation und Routing

Die Hauptseite *index.html* enthält nur einen Platzhalter (Kasten mit dem Text *Prosodiya Online Text Analyse wird geladen...*) und bindet notwendige CSS Styles sowie die Hauptkomponente *app\_component* ein. Das Einbinden dieser Komponente veranlasst Angular diese beim Aufruf der Webseite zu laden und liefert so den Einstiegspunkt in die *AngularDart* Applikation.

Der zur Komponente zugehörige *AppService* definiert lediglich die Funktionen zum Anzeigen und Löschen von Info- und Fehlermeldungen. Das Template der Hauptkomponente *app\_component.html* enthält folgende Elemente:

- Die Navigationsleiste mit den Einträgen *Home*, *Textanalyse*, *Verifizierung* und *Login*
- Container, in denen Info- und Fehlermeldungen angezeigt werden
- Die `router-outlet` Direktive, die dafür sorgt, dass die in der Navigation ausgewählte Komponente angezeigt wird

In der Komponentenklassse *AppComponent* werden folgende Metadaten der Applikation per Klassenannotationen definiert:

- **Angular Direktiven in *@Component*:** Alle Basiskomponenten (`CORE_DIRECTIVES`, `ROUTER_DIRECTIVES`, `materialDirectives`, `formDirectives`) und eigene Subkomponenten (`TextAnalysisComponent`, `UserAccountComponent`, `WordReviewComponent`, `WordVerificationComponent`, `HomeComponent`, `UserRegisterComponent`) werden hier eingebunden und können damit in allen Templates verwendet werden.
- **Angular Providers in *@Component*:** Alle vordefinierten (`ROUTER_PROVIDERS`, `materialProviders`) und eigenen Provider (`TextAnalysisService`, `UserAccountService`, `AppService`, `SegmentationProposalService`, `SegmentationVerificationService`) werden hier der Applikation bekannt gemacht, sodass sie von allen Komponenten verwendet werden können.
- **Routen** für den Angular Router werden in *@RouteConfig* zu den entsprechenden Komponenten zugeordnet (s. Tabelle 4.4).

#### Begrüßungsseite

Die Begrüßungsseite (Abbildung 4.5) *home\_component* beinhaltet keine Programmlogik. Sie ist der Einstiegspunkt der Webapplikation und stellt eine Übersicht über die gebotenen Funktionen dar. In den oberen zwei Kästen findet die Nutzerin oder der Nutzer eine kurze Einführung in die Software und ihre Anwendung, sowie eine Erklärung der Textanalyse Funktion. Darunter befinden sich nebeneinander zwei Kästen, in denen die Nutzerkontoverwaltung und der Verifizierungsprozess für die Wortdatenbank erklärt werden.

Route	Name	Komponente
<i>/home</i>	<i>Home</i>	<i>HomeComponent</i>
/text_analysis	TextAnalysis	TextAnalysisComponent
/text_analysis/:text	TextAnalysisParam	TextAnalysisComponent
/word_review/:wordIndex	WordReview	WordReviewComponent
/word_verification	WordVerification	WordVerificationComponent
/user_account	UserAccount	UserAccountComponent
/user_register	UserRegister	UserRegisterComponent
/**	NotFound	HomeComponent

Tabelle 4.4: Routen für den Angular Router, mit Name und zugehöriger Komponente. Die kursiv dargestellte *Home* Route ist der Default Wert und wird bei Programmstart angezeigt.

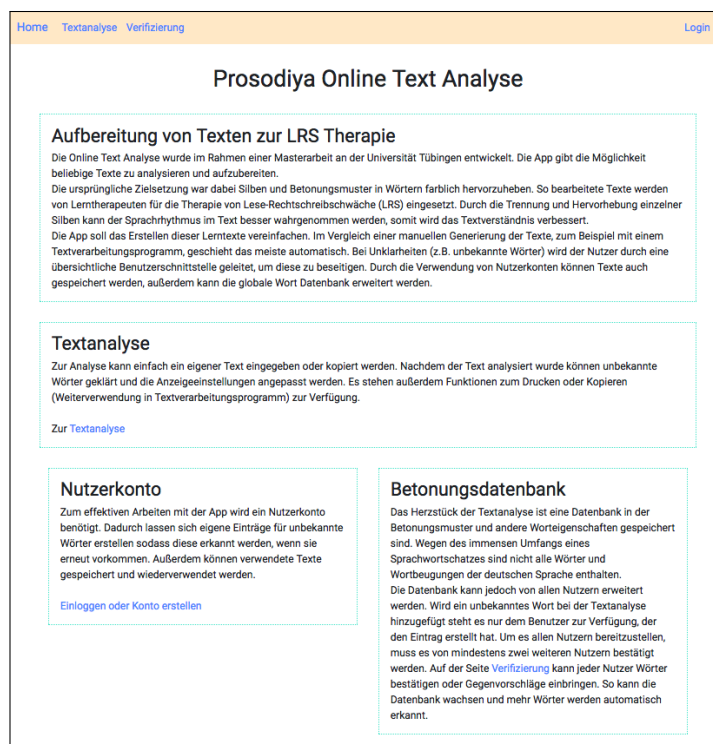


Abbildung 4.5: Bildschirmfoto der Begrüßungsseite

### 4.3.2 Datenmodell

Im Folgenden wird ein Überblick über das Datenmodell gegeben, dessen Klassen diverse Komponenten der Applikation benutzen. Die Klasse `AnnotationText` wird instantiiert wenn ein neuer Text analysiert wird. Sie enthält eine Liste von Wörtern (Klasse `Word`), welche wiederum eine Liste von Silben (Klasse `Syllable`) speichern (Abbildung 4.6).



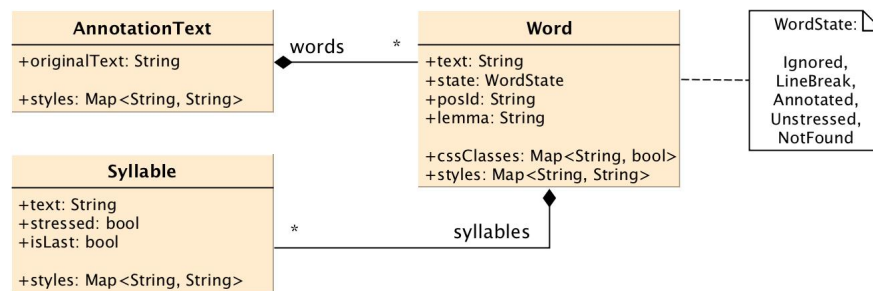


Abbildung 4.6: UML Klassendiagramm des Textmodells mit Wörtern und Silben

Diese Klassen enthalten auch Informationen zu HTML Klassen und CSS Styles. Durch entsprechende Zugriffe können diese dann direkt im Angular HTML Template ausgelesen werden. Mit dieser Vorgehensweise ist es möglich, die Textvorschau direkt und verzögerungsfrei nach einer Nutzereingabe anzupassen. Wird zum Beispiel eine Silbenfarbe geändert, wird erst über alle Wörter und Silben im `AnnotationText` iteriert und die entsprechenden CSS Farben (z.B. `color` oder `background-color`) auf die neue Farbe gesetzt. Im Angular Update Cycle wird dann die Veränderung im Modell detektiert und das *Document Object Model* (DOM) der Webseite automatisch angepasst. Es musste kein manueller Code geschrieben werden, durch *Data Binding* werden die aktuellen Werte des Modells (*Dart* Klassen) sofort automatisch in das DOM übernommen.

Die Klasse `TextConfiguration` (Abbildung 4.7) wird zur Verwaltung von Vorlagen zur Einstellung der Annotation verwendet. Sie enthält alle Einstellungen, die auf einen Text angewandt werden können, z.B. die Farben von betonten und unbetonten Silben, Silben- und Wortabstände oder das Silbentrennzeichen. Außerdem beinhaltet sie eine Instanz der Klasse `PartOfSpeech`, welche spezielle Annotationseinstellungen für die verschiedenen Wortarten speichert.

Die Klasse `UserEntry` bildet die manuell segmentierten Wörter ab, die dem System unbekannt waren und von den NutzerInnen selbst erstellt wurden. Ein `UserEntry` enthält Informationen über Worttext, Wortart, Segmentierung der Silben und Wortbetonung. Vom Nutzer angelegte, eigene Texte werden in der Klasse `UserText` (mit dem Wortlaut des Texts und einem Titel) abgebildet.

### 4.3.3 Textanalyse

Die Textanalyse (Komponente *text\_analysis\_component*) stellt das Herzstück der Applikation dar. Sie bindet mehrere Subkomponenten ein, die in den folgenden Abschnitten vorgestellt werden. Es gibt Nutzeroberflächen für

- die Eingabe neuer Texte,
- die Vorschau des aktuell analysierten Texts,
- sowie die manuelle Analyse dem System unbekannter Wörter.



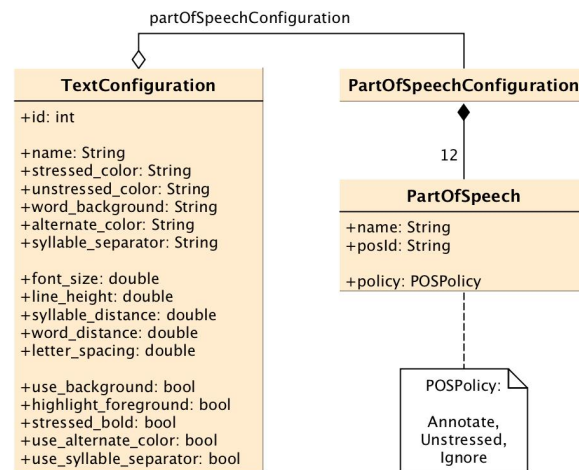


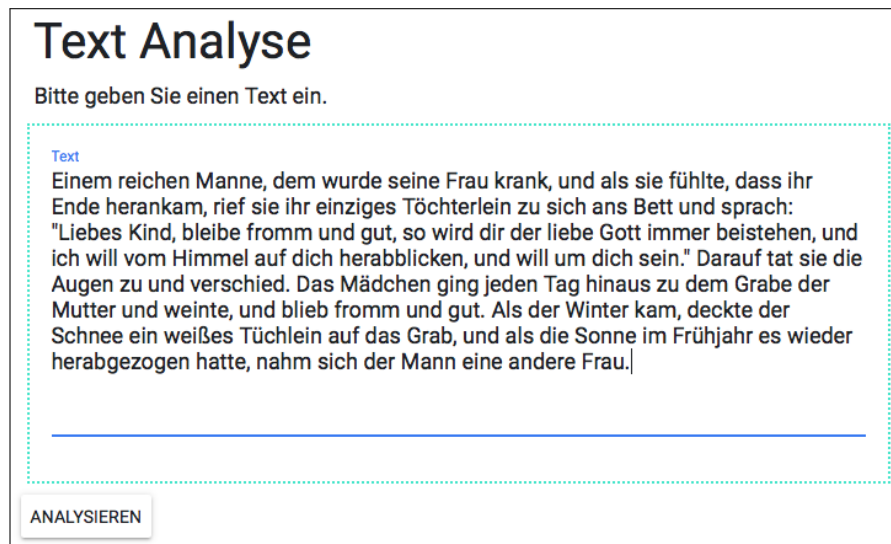
Abbildung 4.7: UML Klassendiagramm der Textkonfiguration mit der Wortartkonfiguration (bestehend aus den Einstellungen für alle 12 Wortarten)



Abbildung 4.8: Bildschirmfoto der Textanalyse-Vorschau

## Texteingabe

Ist noch kein Text analysiert, so zeigt die Textanalyse ein leeres Eingabetextfeld (in Komponente *text\_analysis\_component*). Hier kann ein beliebiger Text eingetippt und analysiert werden.



**Text Analyse**

Bitte geben Sie einen Text ein.

Text

Einem reichen Manne, dem wurde seine Frau krank, und als sie fühlte, dass ihr Ende herankam, rief sie ihr einziges Töchterlein zu sich ans Bett und sprach: "Liebes Kind, bleibe fromm und gut, so wird dir der liebe Gott immer beistehen, und ich will vom Himmel auf dich herabblicken, und will um dich sein." Darauf tat sie die Augen zu und verschied. Das Mädchen ging jeden Tag hinaus zu dem Grabe der Mutter und weinte, und blieb fromm und gut. Als der Winter kam, deckte der Schnee ein weißes Tüchlein auf das Grab, und als die Sonne im Frühjahr es wieder herabgezogen hatte, nahm sich der Mann eine andere Frau.

ANALYSIEREN

Abbildung 4.9: Bildschirmfoto der Texteingabe

### TextAnalysisService

Der *TextAnalysisService* wird von mehreren Komponenten verwendet und verwaltet vor allem eine Instanz der Klasse *AnnotationText*. Er beinhaltet auch die Logik, um einen Eingabetext zu analysieren. Dazu wird ein HTTP Request mit dem Eingabetext an das Backend geschickt. Aus der Antwort kann das Modell des annotierten Texts aufgebaut und in der Vorschau angezeigt werden. Der Service bietet zudem die Funktionen zum Anwenden der aktuellen Annotationseinstellung auf den Text.

### Textvorschau

Die Textvorschau (Komponente *text\_preview\_component*, gezeigt in Abbildung 4.8) stellt das Ergebnis der Textannotation so dar, wie es auch im Druck aussehen würde. Die Informationen über die Darstellung (HTML Klassen, CSS Attribute) sind jeweils schon im Datenmodell (in den Wörtern und Silben des *AnnotationText*) gespeichert und werden im Template der Komponente nur abgerufen. Im Template wird mithilfe von Angular Direktiven über alle Wörter und Silben des Texts iteriert, je nach Typ des Wortes werden verschiedene HTML Tags eingesetzt:

- Für jedes annotierte Wort und jede Silbe wird ein `span` erstellt. Die Klassen und CSS Attribute der Spans werden mit *Data Binding* (`[ngClass]` und `[ngStyle]`) dem Modell entnommen.
- Für im Text vorkommende Linebreaks (diese werden als `Word` mit speziellem Typ abgebildet) wird jeweils ein `<br>` Tag eingesetzt.
- Für Wörter, welche bei der Analyse nicht in der Datenbank gefunden wurden, wird

ein Link (span mit (click) Event) eingesetzt, welcher zur manuellen Wortanalyse (Abschnitt 4.3.3) führt.

Jedes annotierte Wort enthält außerdem eine Subkomponente `word_detail_component`, die beim Klicken auf das Wort als Popup sichtbar gemacht wird. Dieses Popup (Abbildung 4.10) enthält folgende Einstellungen, die für jedes Wort manuell angepasst werden können (eine Änderung hier verändert die grundlegenden Texteingstellungen nicht und hat auch keine Auswirkungen auf die Wortdatenbank oder manuelle Nutzereinträge).

- Wort annotieren oder nicht (wird diese Option ausgeschaltet, so wird das Wort ohne Silbentrennung und in der Farbe der unbetonten Silben dargestellt)
- Manuelles Betonungsmuster (gilt nur für dieses Wort)
- Manuelle Silbentrennung (gilt nur für dieses Wort)
- Gleiche Wörter übernehmen (beim Klicken dieses Buttons werden in der Vorschau alle Wörter mit dem gleichen Text genauso annotiert, das ist z.B. bei häufig vorkommenden Eigennamen sinnvoll)
- Wortart und Lemma (diese werden nur angezeigt und können nicht verändert werden)

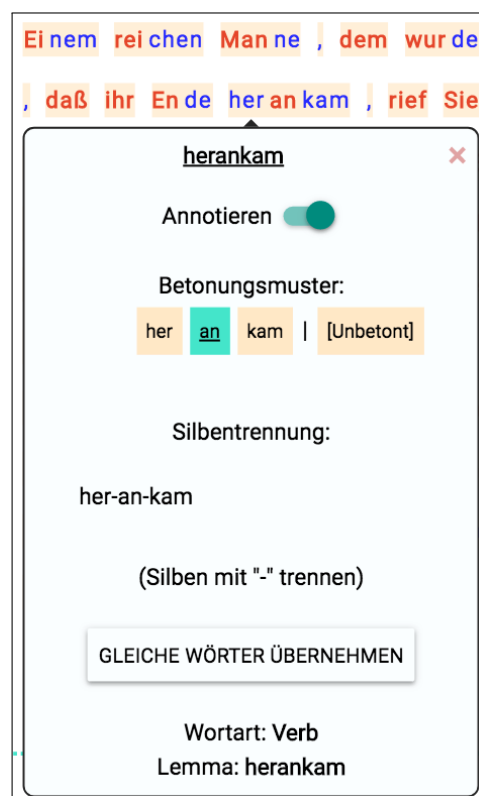


Abbildung 4.10: Bildschirmfoto der spezifischen Worteinstellungen

Zusätzlich sind im Vorschauenfenster Buttons für das Drucken und Kopieren des Texts vorhanden.

- Der *Drucken* Button öffnet den Drucken Dialog des verwendeten Browsers. Damit nur der Text in der Vorschau gedruckt wird, wird hier ein eigener CSS Style verwendet. Mit der `@media print` CSS Anweisung werden dafür alle nicht im Druck benötigten HTML Tags auf `display: none` gesetzt und somit ausgeblendet. Zudem wird die Größe des Textvorschauelements verändert. Ist auf dem Nutzersystem ein PDF Drucker installiert, kann der Text so auch als PDF gespeichert werden.
- Der Button *Kopieren (Zwischenablage)* kopiert automatisch den Vorschautext samt Annotation in die Zwischenablage, sodass dieser z.B. in einem externen Textverarbeitungsprogramm weiter bearbeitet werden kann.

### Manuelle Wortanalyse

Ist bei der Analyse dem System ein Wort nicht bekannt, so wird es in der Textvorschau rot hinterlegt. Beim Klicken auf das Wort öffnet sich die manuelle Wortanalyse (Komponente `word_review_component`), in welcher die Nutzerin oder der Nutzer für dieses Wort das Betonungsmuster und die Silbentrennung selbst bestimmen kann. Hierfür gibt es zum Einen ein Auswahlfenster für die betonte Silbe (durch Klicken kann diese bestimmt werden und wird bei Auswahl in einer anderen Farbe dargestellt), und zum Anderen ein Textfeld für die Silbentrennung. Hier können manuell Trennstriche gesetzt werden, wobei der Text des Wortes selbst nicht verändert werden darf. Ändert sich bei entsprechender Eingabe die Silbentrennung, passt sich die Eingabe für das Betonungsmuster automatisch an. Mit entsprechenden Buttons kann die Nutzerin oder der Nutzer dann die vorgenommene Analyse entweder speichern, das Wort im Text ignorieren oder ohne zu speichern zum Text zurückkehren.

Um die manuellen Eingaben zu erleichtern und zu minimieren werden außerdem automatische Vorschläge generiert. Das Backend sendet für jedes zu bearbeitende Wort eine Liste von Vorschlägen, die aus verschiedenen Quellen stammen können (s. Abschnitt 4.2.2). Durch Anklicken eines Vorschlags wird dieser in den Eingabefenstern auf der linken Seite übernommen.

### Textoptionen

Unter dem Textvorschaufenster befinden sich weitere Fenster zur Verwaltung des aktuellen Texts (Komponente `text_input_component`). Folgende Funktionen (in drei Kästen, von oben nach unten) werden hier angeboten:

- Der aktuelle Text wird in einem Eingabefeld angezeigt. Dieser kann hier verändert und neu analysiert werden.
- Der aktuelle Text kann im Nutzerkonto (s. Abschnitt 4.3.5) gespeichert werden. Dazu muss ein Titel eingegeben werden. Wird anschließend der *Speichern* Button gedrückt, wird der Text mit zugehörigem Titel vom `UserAccountService` gespeichert.
- Die aktuelle Analyse kann vollständig verworfen werden, sodass wieder das ursprüngliche, leere Textfeld angezeigt wird.

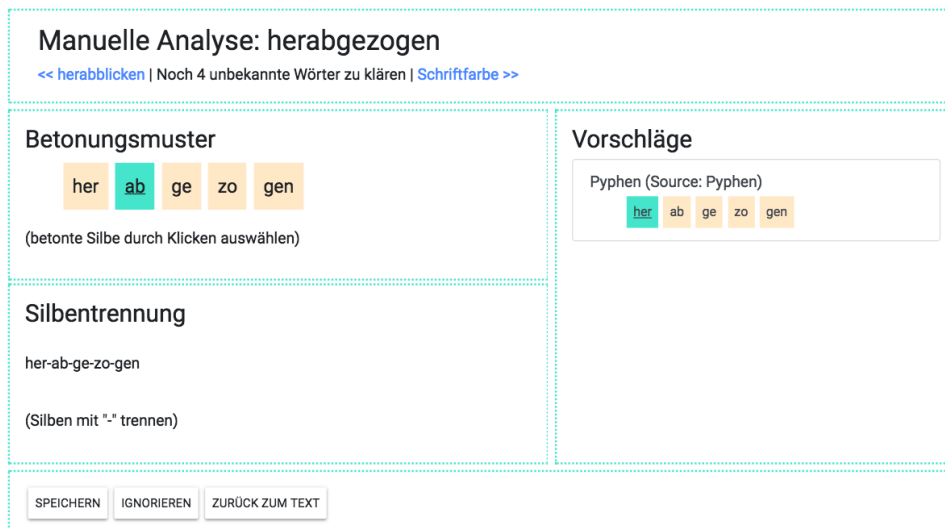


Abbildung 4.11: Bildschirmfoto der manuellen Wortanalyse

### 4.3.4 Annotationseinstellungen

Im Kasten links neben der Textvorschau können die Einstellungen verändert werden, welche die Art der Annotation bestimmen. Die Einstellungen werden zentral in einem Objekt der Klasse `TextConfiguration` im `TextAnalysisService` gespeichert und nach Nutzereingaben auf den aktuellen Text angewandt. Wird hier eine Farbe oder ein Abstand verändert, so ist dies sofort in der Textvorschau sichtbar. Außerdem kann die aktuelle Konfiguration jederzeit als Vorlage gespeichert werden, so kann sie später (z.B. für einen anderen Text) wiederverwendet werden. Das User Interface für die Einstellungen ist in der Komponente `text_settings_component` definiert.

#### User Interface für Einstellungen

Ganz oben können zunächst Farben für die Annotation und die Art, wie diese verwendet werden eingestellt werden (Abbildung 4.12):

- **Betonte Silbe:** Hier wird die Farbe für die betonte Silbe in jedem Wort festgelegt. Außerdem kann gewählt werden, ob der Text der Silbe fett dargestellt werden soll. Für die Auswahl der Farben wurde ein entsprechend farbiger Kasten verwendet, der beim Anklicken den Farbwahldialog des Browsers öffnet.
- **Unbetonte Silbe:** Alle unbetonten Silben werden in dieser Farbe dargestellt.
- **2. Unbetonte Silbe:** In längeren Wörtern folgen mehrere unbetonte Silben aufeinander. Wird diese Option aktiviert, werden aufeinanderfolgende unbetonte Silben abwechselnd in den Farben der 1. und der 2. unbetonten Silbe dargestellt.

Betonte Silbe:  
Fett ☒  
#E84F2F

Unbetonte Silbe:  
#2D2FFF

2. Unbetonte Silbe:  
Benutzen ☒  
#21AEE9

Wort Hintergrundfarbe:  
Benutzen ☒  
#FFF1DD

Silbe hervorheben:  
☒ Vordergrundfarbe  
☐ Hintergrundfarbe

Abbildung 4.12: Bildschirmfoto der Farbeinstellungen

- **Worthintergrundfarbe:** Ist diese Option aktiviert erhält zusätzlich jedes Wort (über Silbengrenzen hinaus) eine Hintergrundfarbe. Dies ist z.B. dann sinnvoll, wenn man große Abstände zwischen den einzelnen Silben wählt, so kann das Wort trotzdem als Einheit erfasst werden.
- **Silbe hervorheben:** Hier kann gewählt werden, ob die Silbenfarben als Schriftfarbe (*Vordergrundfarbe*) oder Hintergrundfarbe benutzt werden sollen.

Beispiele zu den verschiedenen Einstellungen sind in der Evaluation in Abschnitt 5.1 zu finden.

Unterhalb der Farbeinstellungen lassen sich verschiedene Größen und Abstände im Text definieren (Abbildung 4.13):

- Schriftgröße
- Silbenabstand (Abstand zwischen einzelnen Silben)
- Wortabstand (Abstand zwischen ganzen Wörtern)
- Zeilenabstand
- Zeichenabstand (Abstand zwischen den einzelnen Buchstaben eines Wortes)

The image shows a settings panel for text formatting. It contains five sliders and one toggle switch, all with a light gray track and a teal-colored knob. The settings are as follows:

Einstellung	Wert
Schriftgröße:	16
Silbenabstand:	0.2
Wortabstand:	0.4
Zeilenabstand:	2.2
Zeichenabstand:	1
Silbentrennzeichen:	Benutzen (aktiviert)

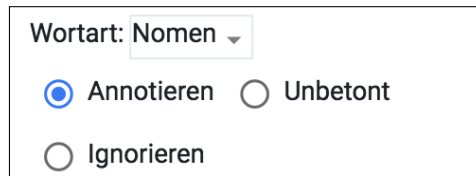
Abbildung 4.13: Bildschirmfoto der TextEinstellungen

Alle Werte lassen sich hier mithilfe eines Sliders einstellen. Außerdem gibt es die Möglichkeit, ein beliebiges Zeichen als Silbentrennzeichen zu benutzen, das dann im Zwischenraum zwischen allen Silben im Text erscheint.

An letzter Stelle in den Einstellungen steht ein Dropdown Menü, in dem man eine Wortart auswählen kann (Abbildung 4.14). Für jede Wortart lassen sich folgende Einstellungen vornehmen, die dann auf den gesamten Text angewandt werden:

- **Annotieren:** Aktiviert die Annotation für Wörter dieser Wortart, d.h. Silben werden getrennt und in den gewählten Farben dargestellt.
- **Unbetont:** Die Annotation ist zwar aktiv und Silben werden getrennt dargestellt, allerdings wird die betonte Silbe nicht besonders hervorgehoben.
- **Ignorieren:** Das Wort wird ohne Trennung von Silben in der Farbe der unbetonten Silbe dargestellt.

Ändert die Nutzerin oder der Nutzer eine Einstellung, so wird im `AnnotationText` über sämtliche Wörter iteriert und deren Einstellung entsprechend angepasst.



Wortart: Nomen ▼

☒ Annotieren ☐ Unbetont

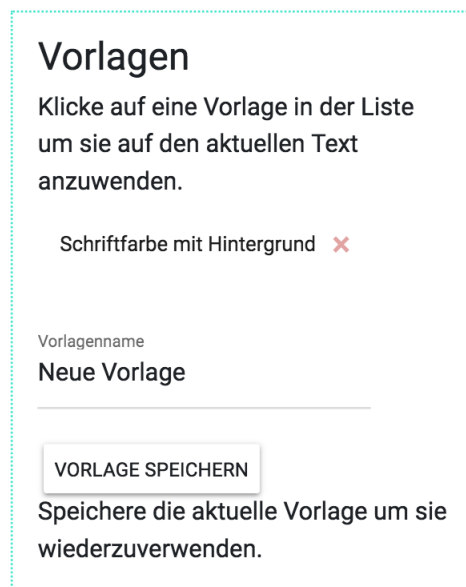
☐ Ignorieren

Abbildung 4.14: Bildschirmfoto der für die Wortart spezifischen Einstellungen

### Annotationsvorlagen

Im Kasten unterhalb der Annotationseinstellungen können Konfigurationen verwaltet werden. Jedem Nutzerkonto sind Vorlagen zugeordnet, d.h. jede Nutzerin und jeder Nutzer verwaltet und sieht nur ihre oder seine eigenen Vorlagen. In der Vorlage werden sämtliche Einstellungen gespeichert, das ganze `TextConfiguration` Objekt wird hier an das Backend geschickt und serialisiert.

Sind bereits Vorlagen vorhanden, werden diese hier aufgelistet. Sie können ausgewählt (und damit auf den aktuellen Text angewandt) werden oder durch Klicken auf das kleine X rechts neben dem Vorlagennamen gelöscht werden. Darunter befindet sich ein Textfeld für den Namen einer neu anzulegenden Vorlage. Beim Klicken des *Vorlage speichern* Buttons wird diese dann unter dem gewählten Namen (und mit den momentan aktiven Einstellungen) neu gespeichert oder, wenn die Vorlage schon existiert überschrieben.



**Vorlagen**

Klicke auf eine Vorlage in der Liste um sie auf den aktuellen Text anzuwenden.

Schriftfarbe mit Hintergrund ✕

Vorlagenname  
**Neue Vorlage**

VORLAGE SPEICHERN

Speichere die aktuelle Vorlage um sie wiederzuverwenden.

Abbildung 4.15: Bildschirmfoto des User Interface zur Verwaltung und Verwendung von Annotationsvorlagen



### 4.3.5 Nutzerkonto

Die Nutzerkonto Seite zeigt entweder das Formular zum Einloggen oder Konto erstellen an, oder falls die Nutzerin oder der Nutzer schon eingeloggt ist, Informationen und Inhalte des Nutzerkontos. Die Komponente `user_account_component` lagert einige Funktionalitäten wie die Registrierung oder die Liste von Nutzertexten in Subkomponenten aus.

#### UserAccountService

Der `UserAccountService` speichert die Informationen zum aktiven Nutzerkonto, z.B. Mailadresse und Passwort, sowie eine Liste von Nutzertexten und manuell segmentierten Wörtern. Er bildet alle Funktionen ab (z.B. Anlegen, Auflisten oder Löschen), die zum Verwalten von Nutzerkonten, Nutzertexten und Wörtern notwendig sind. Dafür werden jeweils HTTP Requests an das Backend gesendet und die erhaltenen Antworten weiter verarbeitet.

#### Login und Registrierung

Ist keine Nutzerin oder kein Nutzer eingeloggt, so wird zuerst das Login Formular angezeigt. Außerdem lautet der Link rechts oben in der Navigationsleiste *Login*, sodass der Login schnell gefunden werden kann. Loggt sich eine Nutzerin oder ein Nutzer mit einem bereits bestehenden Konto ein, so enthält der Navigationslink die Mailadresse des Nutzerkontos.

Unter dem Login Formular führt ein Link mit der Aufschrift *Neues Konto erstellen* auf die Registrierungsseite. Hier muss für die erfolgreiche Registrierung eine Mailadresse sowie ein Passwort eingegeben werden. Außerdem speichert das Nutzerkonto Vor- und Nachname, sowie die Information, ob es sich bei dem neu anzulegenden Konto um ein Expertenkonto handelt, was in einer Checkbox angegeben werden kann. Expertenkonten sollen von SprachtherapeutInnen oder NachhilfelehrerInnen genutzt werden. Diese verfügen bei der Verifizierung über eine schwerer gewichtete Stimme (s. Abschnitt 4.2.4). Wird die Registrierung abgeschickt, so wird das neue Nutzerkonto im Backend erstellt und die Nutzerkontoseite kehrt zum Login Formular zurück. Schlägt das Registrieren fehl (z.B. wenn es schon ein Konto mit der angegebenen Mailadresse gibt, oder weil Eingaben fehlen), wird eine entsprechende Fehlermeldung angezeigt.

#### Nutzerkonto Seite

Ist aktuell eine Nutzerin oder ein Nutzer eingeloggt, werden Informationen zum Konto, sowie Listen von gespeicherten Texten und manuell segmentierten Wörtern angezeigt (Abbildung 4.16).

Die Liste der gespeicherten Nutzertexte zeigt zunächst nur deren Titel als Links an (Komponente `user_textlist_component`). Werden diese angeklickt, sieht man den Text und hat die

## 4 Entwicklung der Anwendung

Möglichkeit, diesen für die Analyse zu verwenden oder zu löschen. Klickt man auf den Button *Verwenden*, so wechselt die Applikation zur Textanalyse und analysiert den gewählten Text. Unter der Textliste ist die Liste manuell segmentierter Wörter zu finden, deren User Interface in der Komponente `user_wordlist_component` definiert ist. Die Einträge zeigen den Worttext, die Silbentrennung und das Betonungsmuster der Segmentierung. Die Einträge können auch gelöscht werden, z.B. falls ein Fehler bei deren Erstellung unterlaufen ist. Ganz unten auf der Seite sind Buttons zum Ausloggen und Löschen des Kontos vorhanden.

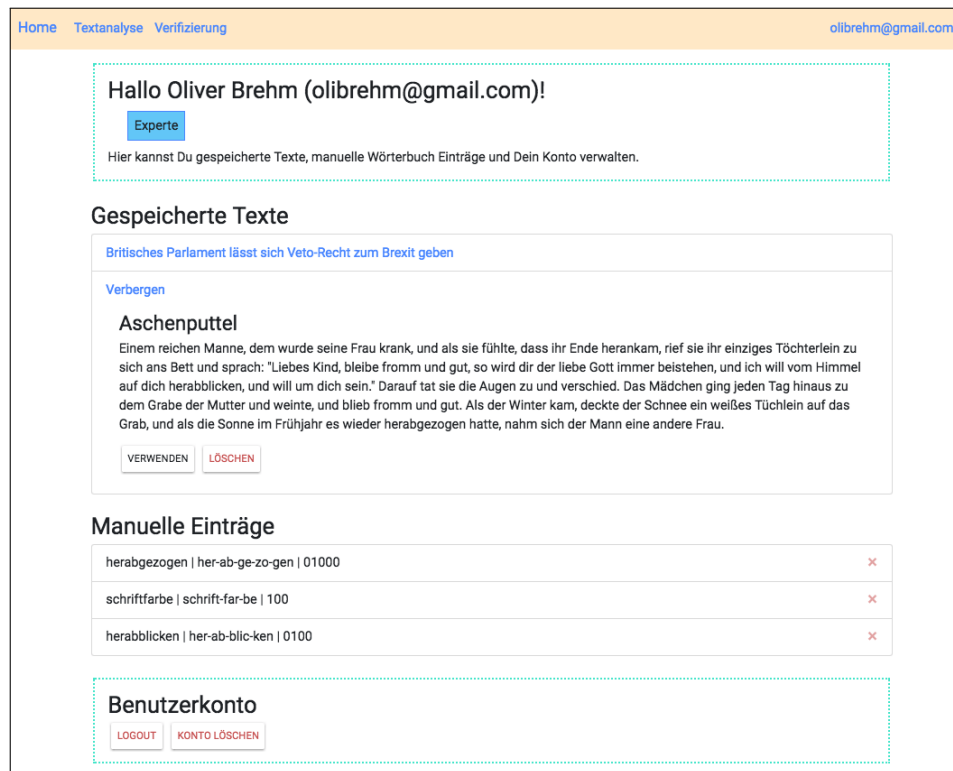


Abbildung 4.16: Bildschirmfoto der Nutzerkontoseite

### 4.3.6 Wort-Verifizierung

Die Wort-Verifizierung dient der Bestätigung von manueller Segmentierung von NutzerInnen durch andere NutzerInnen, sodass diese in die globale Wortdatenbank aufgenommen werden können (s. Abschnitt 4.2.4). Das User Interface für diesen Prozess ist in der Komponente `word_verification_component` implementiert.

Ist eine Nutzerin oder ein Nutzer eingeloggt, werden nacheinander zu verifizierende Wörter geladen, das User Interface ähnelt dem, das zur manuellen Analyse von unbekannten Wörtern verwendet wird (s. Abschnitt 4.3.3). Auch hier gibt es jeweils eine Eingabemöglichkeit zur Auswahl der betonten Silbe und zur Bestimmung der Silbentrennung. Auf der rechten Seite

befinden sich allerdings neben den automatisch generierten Vorschlägen auch die Segmentierungsvorschläge anderer NutzerInnen. Automatisch wird der Vorschlag der Nutzerin oder des Nutzers ausgewählt, welche oder welcher die Segmentierung zuerst erstellt hat. Haben weitere NutzerInnen den Vorschlag schon verifiziert, so erscheinen auch deren Segmentierungen in der Liste.

Zur eigentlichen Verifizierung kann nun ein Eintrag in der Liste der Vorschläge ausgewählt werden oder, falls alle Vorschläge als falsch empfunden werden, eine eigene, andere Segmentierung auf der linken Seite eingestellt werden. Danach wird die Verifizierung mit dem Button *Abschicken* gespeichert.

Sind noch weitere zu verifizierende Einträge vorhanden, wird sofort das nächste Wort geladen. Ein Label unter der Überschrift der Komponente zeigt an, wie viele zu verifizierende Wörter es noch gibt.

### Verifizierung: herabblicken

Noch zu verifizierende Wörter: 9

#### Betonungsmuster

her

ab

blic

ken

(betonte Silbe durch Klicken auswählen)

#### Silbentrennung

her-ab-blic-ken

---

(Silben mit "-" trennen)

#### Vorschläge

(Benutzer)

her

ab

blic

ken

(Benutzer)

her

ab

blic

ken

Pyphen (Source: Pyphen)

her

abbli

cken

ABSCHICKEN

Abbildung 4.17: Bildschirmfoto der Verifikation von manuell hinzugefügten Wörtern



# 5 Evaluation

Die Applikation wurde mit dem Hauptziel entwickelt, eine Nutzeroberfläche zu bieten, mit der sowohl Personen, die in der LRS Therapie arbeiten (wie LerntherapeutInnen oder Lehrkräfte) als auch Nicht-ExpertenInnen (z.B. die Eltern betroffener Kinder) intuitiv und effizient benötigtes Arbeitsmaterial erstellen können. Außerdem wurde untersucht wie eine Datenbank für Betonungsmuster in Wörtern erstellt und benutzerfreundlich erweitert werden kann.

Die Evaluation untersucht daher folgende Punkte:

- Erstellung von Arbeitsmaterial aus einem Fließtext
- Intuitivität und Einfachheit der Bedienung
- Effizienz des Prozesses der Erstellung von Arbeitsmaterial
- Nutzerfreundliche Erweiterung der Wortdatenbank

Zunächst wird durch Beispiele eines annotieren Textes mit verschiedenen Einstellungen demonstriert, welche vielseitigen Möglichkeiten die Textannotation für das Erstellen von Arbeitsmaterial bietet. Für die Untersuchung von Intuitivität, Einfachheit und Effizienz bei der Bedienung der Applikation wurde ein Nutzertest durchgeführt. Dieser fand als Interview statt, um qualitative Daten durch Nutzerbefragung zu erheben. Außerdem wurden quantitative Daten durch das Ausfüllen von Ankreuzfragen während des Interviews erhoben.

## 5.1 Ergebnisse der Textannotation

Um zu zeigen, dass die Applikation das Ziel des automatischen Erstellens von Arbeitsmaterial erfüllt, werden im Folgenden verschiedene Texte präsentiert, die mit Hilfe des Web Interfaces erstellt wurden. Die Ergebnisse zeigen, dass das Programm dieses Ziel nicht nur erfüllt, sondern im Vergleich zu herkömmlichen Methoden auch noch einen hohen Grad an Flexibilität bietet. Die Einstellungen und damit das Erscheinungsbild des resultierenden Textes können von der Nutzerin oder dem Nutzer frei an die Bedürfnisse des jeweiligen Falls angepasst werden.

Im Folgenden wurde der Beispieltext *Aschenputtel* (s. Abschnitt 6 im Anhang) mit der Webapplikation analysiert, sodass Silbentrennung und Betonung farblich markiert dargestellt wurden. Die in den Beispielen verwendeten Texte wurden mit der Druckfunktion der Textvorschau als PDF Dateien exportiert. Jedes Bild entspricht einer DIN A4 Seite.

### Beispiel 1: Hervorhebung der Silben mit zwei alterierenden Farben

Im ersten Beispiel wird die Schriftfarbe für das Hervorheben der Silben genutzt. Die betonte Silbe wird immer rot dargestellt, in Wörtern mit vielen Silben werden diese dann alterierend rot und blau hervorgehoben. Die Schriftgröße wurde auf 22 gesetzt, der Wortabstand auf den Wert 0.4. Zusätzlich sorgt die Einstellung des Zeilenabstands mit dem Wert 1.5 für bessere Lesbarkeit.

Einem reichen Manne , dem wurde seine Frau krank , und als  
sie fühlte , dass ihr Ende herankam , rief sie ihr einziges  
Töchterlein zu sich ans Bett und sprach : " Liebes Kind ,  
bleibe fromm und gut , so wird dir der liebe Gott immer  
beistehen , und Ich will vom Himmel auf dich herabblicken ,  
und will um dich sein . " Darauf Tat sie die Augen zu und  
verschied . Das Mädchen ging jeden Tag hinaus zu dem  
Grabe der Mutter und weinte , und blieb fromm und gut . Als  
der Winter kam , deckte der Schnee ein weißes Tüchlein auf  
das Grab , und als die Sonne im Frühjahr es wieder  
herabgezogen hatte , nahm sich der Mann eine andere Frau .

Abbildung 5.1: Hervorhebung der Silben mit zwei alterierenden Farben

## Beispiel 2: Worthintergrund und Silbenabstand

In diesem Beispiel sorgt die Worthintergrundfarbe für eine bessere Wahrnehmung des Wortes als Einheit. Dadurch kann auch ein Silbenabstand (hier *0.1*) eingestellt werden ohne dass die Gefahr besteht, dass dadurch nicht mehr klar ist, welche Silben zusammen gehören. Wegen des Worthintergrunds nehmen die Wörter mehr Platz in der Zeile ein, daher wurde auch der Zeilenabstand auf 2.2 erhöht. Zudem wurde hier ein Zeichenabstand von 2 (statt 1 in Beispiel 1) gewählt.

Ei nem rei chen Man ne , dem wur de sei ne Frau krank ,  
und als sie föhl te , dass ihr En de her an kam , rief sie  
ihr ein zi ges Töch ter lein zu sich ans Bett und sprach :  
" Lie bes Kind , blei be fromm und gut , so wird dir der  
lie be Gott im mer bei ste hen , und Ich will vom  
Him mel auf dich her ab blic ken , und will um dich sein  
. " Dar auf Tat sie die Au gen zu und ver schied . Das  
Mäd chen ging je den Tag hin aus zu dem Gra be der  
Mut ter und wein te , und blieb fromm und gut . Als der  
Win ter kam , deck te der Schnee ein wei ßes Tüch lein  
auf das Grab , und als die Son ne im Früh jahr es  
wie der her ab ge zo gen hat te , nahm sich der Mann  
ei ne an de re Frau .

Abbildung 5.2: Worthintergrund und Silbenabstand

### Beispiel 3: Silbentrennzeichen und verschiedene Silbenfarben

In Beispiel 3 wurde wieder auf die Worthintergrundfarbe verzichtet. Das Trennzeichen „=“ soll hier das Silbentrennen erleichtern sowie zusammenhängende Silben zu einem Wort gruppieren. Der Zeichenabstand beträgt wieder 1, der Silbenabstand 0, da das Gleichheitszeichen hier als Silbentrenner dient. Zusätzlich wurde eine weitere Farbe für alterierende Silben gewählt. Hier wird nur noch die betonte Silbe rot dargestellt, danach alterieren die Silben bei langen Wörtern in den Farben blau und violett.

Ei=nem rei=chen Man=ne , dem wur=de sei=ne Frau krank ,  
und als sie fühl=te , dass ihr En=de her=an=kam , rief sie  
ihr ein=zi=ges Töch=ter=lein zu sich ans Bett und sprach : "  
Lie=bes Kind , blei=be fromm und gut , so wird dir der  
lie=be Gott im=mer bei=ste=hen , und Ich will vom Him=mel  
auf dich her=ab=blic=ken , und will um dich sein . "  
Dar=auf Tat sie die Au=gen zu und ver=schied . Das  
Mäd=chen ging je=den Tag hin=aus zu dem Gra=be der  
Mut=ter und wein=te , und blieb fromm und gut . Als der  
Win=ter kam , deck=te der Schnee ein wei=ßes Tüch=lein  
auf das Grab , und als die Son=ne im Früh=jahr es wie=der  
her=ab=ge=zo=gen hat=te , nahm sich der Mann ei=ne  
an=de=re Frau .

Abbildung 5.3: Silbentrennzeichen und verschiedene Silbenfarben



**Beispiel 4: Hervorhebung des Silbenhintergrunds**

In diesem Beispiel wird ein anderer Ansatz der farblichen Hervorhebung gewählt. Die eingestellten Farben bestimmen hier den Hintergrund der Silben und nicht die Schriftfarbe (diese ist immer schwarz). Das Beispiel hebt vor allem die betonten Silben hervor, die grün dargestellt werden. Unbetonte Silben bekommen alterierend zwei verschiedene Grautöne als Hintergrund. Der Silbenabstand ist hier wieder 0, somit wird die Einheit des Wortes deutlich erkennbar.

Einem reichen Manne , dem wurde seine Frau krank , und  
als sie fühlte , dass ihr Ende herankam , rief sie ihr  
einziges Töchterlein zu sich ans Bett und sprach : " Liebes  
Kind , bleibe fromm und gut , so wird dir der liebe Gott  
immer beistehen , und Ich will vom Himmel auf dich  
herabblicken , und will um dich sein . " Darauf Tat sie die  
Augen zu und verschied . Das Mädchen ging jeden Tag  
hinaus zu dem Grabe der Mutter und weinte , und blieb  
fromm und gut . Als der Winter kam , deckte der Schnee  
ein weißes Tüchlein auf das Grab , und als die Sonne im  
Frühjahr es wieder herabgezogen hatte , nahm sich der  
Mann eine andere Frau .

Abbildung 5.4: Hervorhebung des Silbenhintergrunds

### Beispiel 5: Manuelles Deaktivieren der Hervorhebung in kurzen Wörtern

In Beispiel 5 wird der Sprachrhythmus mit Wortbetonungen noch deutlicher hervorgehoben. Die betonte Silbe wird zusätzlich zur grünen Farbe auch fett dargestellt. Außerdem wurde die Betonung in manchen Wörtern deaktiviert, so werden im ganzen Text beispielsweise Artikel, Präpositionen oder manche einsilbigen Wörter nicht betont markiert. Dieser Text soll damit einen Fokus auf den Sprachrhythmus innerhalb ganzer Sätze legen.

Ei-nem **rei**-chen **Man**-ne , dem wur-de sei-ne **Frau** **krank** ,  
und als sie **fühl**-te , dass ihr **En**-de her-**an**-kam , rief sie ihr  
**ein**-zi-ges **Töch**-ter-lein zu sich ans **Bett** und **sprach** : "  
**Lie**-bes **Kind** , **blei**-be **fromm** und **gut** , so **wird** dir der **lie**-be  
**Gott** **im**-mer **bei**-ste-hen , und **ich** will vom **Him**-mel auf dich  
her-**ab**-blic-ken , und will **um** dich sein . " Dar-**auf** tat sie  
die **Au**-gen zu und ver-**schied** . Das **Mäd**-chen ging **je**-den  
**Tag** hin-**aus** zu dem **Gra**-be der **Mut**-ter und **wein**-te , und  
blieb **fromm** und **gut** . Als der **Win**-ter **kam** , **deck**-te der  
**Schnee** ein **wei**-ßes **Tüch**-lein auf das **Grab** , und als die  
**Son**-ne im **Früh**-jahr es wie-der her-**ab**-ge-zo-gen hat-te ,  
**nahm** sich der **Mann** ei-ne **an**-de-re **Frau** .

Abbildung 5.5: Manuelles Deaktivieren der Hervorhebung in kurzen Wörtern

## 5.2 Nutzertest

Während die Beispiele im vorherigen Abschnitt verdeutlichen, dass die entwickelte Applikation generell dazu in der Lage ist, Texte anforderungsgemäß zu annotieren, sollte der Nutzertest zeigen, ob die Applikation von der Zielgruppe (LerntherapeutInnen, Lehrkräfte und Eltern betroffener Kinder) problemlos bedient werden kann. Vor allem Intuitivität und Zeiteffizienz sollten dabei berücksichtigt werden, da Software, die diese Kriterien unzureichend erfüllt, ungenutzt oder im schlechtesten Fall überhaupt nicht benutzt wird.

Vor dem eigentlichen Nutzertest wurde zunächst ein Pilottest durchgeführt. Dieser half dabei, Mängel im Design des Tests an sich aufzudecken und Fehler in der Webapplikation zu finden, welche keinen direkten Einfluss auf die Usability hatten und die noch vor dem Nutzertest behoben werden konnten.

### 5.2.1 Vorbereitung und Durchführung

Der Nutzertest beinhaltete neben den statistischen Daten zu Alter und Beruf bzw. Studiengang fünf Szenarien, die die NutzerInnen bearbeiten sollten. Um detaillierte Informationen zur Vorgehensweise der ProbandInnen zu erhalten, wurde im Interview mit der *Thinking Aloud* Methode gearbeitet, d.h. die Testperson wurde aufgefordert bei jeder Aktion, die sie durchführt, möglichst genau zu beschreiben was sie damit bezweckt und warum sie es tut[23]. In der schriftlichen Testbeschreibung wurden die NutzerInnen daher informiert, alle Arbeitsschritte der Szenarien laut vorzulesen und Kommentare und Kritik jederzeit zu äußern. Außerdem wurde in jedem Test explizit mündlich darauf hingewiesen, alle Handlungen möglichst genau zu beschreiben. In der Testbeschreibung wurde auch verdeutlicht, dass es sich um ein Test des Softwaresystems handelt und nicht der Nutzerin oder des Nutzers.

Um konsistente Testbedingungen zu schaffen, wurde bei allen ProbandInnen mit möglichst aktuellen Versionen von Google Chrome gearbeitet. Es wurde zudem sichergestellt, dass den Testpersonen ein möglichst gewohntes Umfeld geboten wird. Im besten Fall benutzten die NutzerInnen ihre eigenen Computer. Falls dies nicht möglich war, wurde vorher überprüft ob beim Testgerät alles genauso wie gewohnt bedient werden konnte, d.h. Einstellungen für Peripherie wie Maus, Tastatur oder Touchpad wurden vorher, der Nutzerin oder dem Nutzer entsprechend angepasst.

Nach jedem der fünf Szenarien wurde von den ProbandInnen ein *After Scenario Questionnaire*[19] bearbeitet. Es beinhaltete die folgenden drei Aussagen:

- **Intuitivität:** „Die Aufgabe konnte ich intuitiv und problemlos erledigen.“
- **Zeitaufwand:** „Ich halte die Zeit, die ich gebraucht habe um die Aufgabe zu erledigen, für angemessen.“
- **Dokumentation:** „Ich bin mit den Informationen, die ich während der Bearbeitung in der App erhalten habe (Beschreibungen, Rückmeldungen) zufrieden.“

## 5 Evaluation

Zustimmung oder Ablehnung konnte für jede Aussage in den fünf Optionen *trifft zu*, *trifft eher zu*, *weder noch*, *trifft eher nicht zu* oder *trifft nicht zu* ausgedrückt werden.

Die Software wurde mit insgesamt 7 ProbandInnen getestet, wobei bei der Auswahl darauf geachtet wurde, dass die Applikation den Testpersonen unbekannt war (sie also die Applikation nicht vorher schon bedient hatten). Die Befragten wurden in zwei Gruppen eingeteilt, zum Einen die Expertengruppe, bestehend aus Lerntherapeutinnen, und zum Anderen die Laien, die exemplarisch für Menschen (z.B. Elternteile) aus dem Umfeld der Betroffenen stehen. Durch die Befragung von ExpertInnen wurde eine besonders kritische, realistische und fachbezogene Einschätzung der Software erhofft. Es wurden 3 Expertinnen (3 Frauen zwischen 40 und 51 Jahren, Durchschnitt 45 Jahre) und 4 Laien (2 Frauen und 2 Männer zwischen 22 und 27 Jahren, Durchschnitt 25 Jahre) befragt.

### 5.2.2 Ergebnisse

Im Folgenden werden die einzelnen Szenarien beschrieben und deren Ergebnisse dargestellt. Von den ProbandInnen erkannte und während des Szenarios geäußerte Schwierigkeiten und Probleme werden hier jeweils kurz zusammengefasst und im Anhang (Abschnitt 6) tabellarisch, zusammen mit eventuellen Vorschlägen zu deren Behebung aufgelistet.

#### Szenario 1: Nutzerkonto

Zuerst sollte die Testperson ein Nutzerkonto mit Mailadresse und Passwort erstellen (hier wurde, damit die Testperson kein sicherheitskritisches Passwort verwendet und sich das Passwort während des Tests merken konnte, „12345678“ vorgegeben). Anschließend loggte sie sich ein und wieder aus.

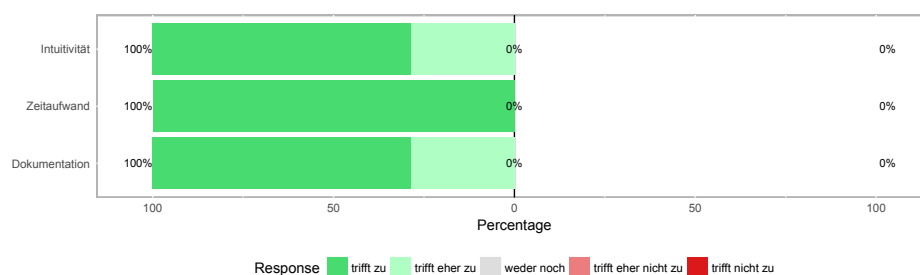


Abbildung 5.6: Quantitative Ergebnisse von Szenario 1 (Nutzerkonto)

Die Testpersonen konnten die Aufgabe meist problemlos lösen. Mehrmals wurde vor dem Erstellen des neuen Kontos erst Mailadresse und Passwort in das Login Fenster eingetragen, was zu einer Fehlermeldung führte (da das Konto noch nicht existierte). Die Testpersonen kamen aber in jedem Fall selbst darauf, dass man dem Link *Neues Konto erstellen* folgen

musste. Hier könnte die Unterscheidung zwischen Login und Konto erstellen noch deutlicher gemacht werden, bzw. könnte der Link zum Konto erstellen besser sichtbar platziert werden (s. Tabelle 6.1 im Anhang).

## Szenario 2: Textanalyse

Als Nächstes loggte sich die Testperson wieder ein und sollte einen gegebenen Text in das Textfeld der Textanalyse kopieren und diesen dann analysieren lassen. Alle unbekannten Wörter (in der Anwendung rot hinterlegt) sollten anschließend manuell geklärt werden. Die NutzerInnen klickten so lange durch das User Interface der manuellen Analyse, bis alle Wörter annotiert waren. Als Letztes sollte mit den Einstellungen der Annotation experimentiert werden bis eine Einstellung gefunden wurde, welche die Testperson ansprach. Hier wurde noch mündlich hinzugefügt, dass alle Einstellungen ausprobieren werden sollten. Damit sollten die Testpersonen sich mit den Einstellungsmöglichkeiten vertraut machen.

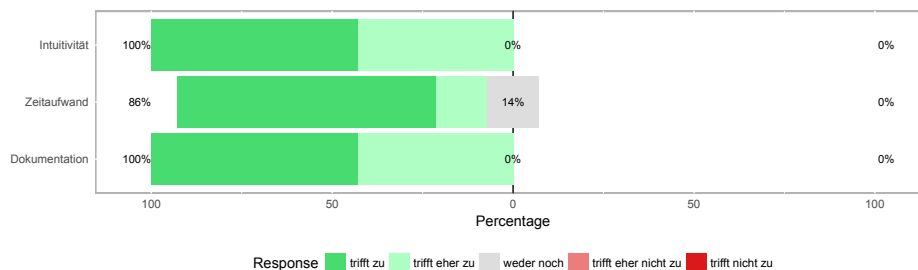


Abbildung 5.7: Quantitative Ergebnisse von Szenario 2 (Textanalyse)

Auch hier konnten alle Testpersonen die Aufgabe bewältigen. Es wurde teilweise das Layout bemängelt. Es war umständlich, Einstellungen zu verändern, wenn man dafür weit runter scrollen musste und so den Vorschautext nicht mehr im Blick hatte. Hier wurde eine kompaktere Darstellung der Einstellungen gewünscht. Es wurde außerdem vorgeschlagen, die Farbauswahl zu vereinfachen, sodass dafür nicht jedes mal ein extra Fenster aufgeht (s. Tabelle 6.2 im Anhang).

## Szenario 3: Annotationsvorlagen

Im dritten Szenario wurden der Testperson zweimal die erste Zeile des Textes, jeweils mit verschiedenen Einstellungen annotiert, vorgegeben. Sie sollte nun nacheinander versuchen, die Annotation so einzustellen, dass es etwa so wie im gegebenen Ausschnitt aussah. Zudem sollten diese Einstellungen als Vorlagen mit den Namen „Vorlage 1“ und „Vorlage 2“ gespeichert werden. Zum Schluss wechselte die Nutzerin oder der Nutzer zwischen Vorlage 1 und Vorlage 2 hin und her.

Das Speichern und Wechseln der Vorlagen bereitete den Testpersonen hier größtenteils keine Probleme. Fast alle NutzerInnen konnten allerdings den Text für Vorlage 2 (farbiger Hintergrund

## 5 Evaluation

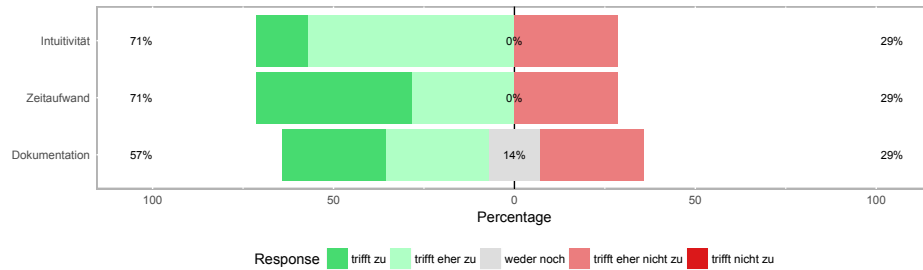


Abbildung 5.8: Quantitative Ergebnisse von Szenario 3 (Annotationsvorlagen)

mit schwarzer Schrift) nicht ohne Hilfe entsprechend anpassen. Hier war nicht klar, was die Option *Silbe betonen* (Radio Buttons, welche festlegen, ob die Schrift- oder Hintergrundfarbe bei den Silben angepasst wird) bedeutete und wurde oft mit der *Wort Hintergrundfarbe* (Farbe, mit der ganze Wörter hinterlegt werden) verwechselt. Erst durch längeres Ausprobieren oder Hilfestellungen konnten die Testpersonen die Aufgabe lösen. Hier sollte zum Einen der Aufbau und die Beschreibung der Elemente der Einstellungen überarbeitet werden, zum Anderen legen die Probleme bei den Einstellungen nahe, eine kurze Einführung (evtl. in Form einer Kurzdokumentation oder Online-Hilfe in der Applikation) bereitzustellen (s. Tabelle 6.3 im Anhang).

### Szenario 4: Texte wiederverwenden

Hier sollte die Testperson den aktuellen Text mit Titel in ihrem Nutzerkonto speichern. Anschließend wurde auf die Nutzerkonto-Seite gewechselt und der gespeicherte Text neu analysiert.

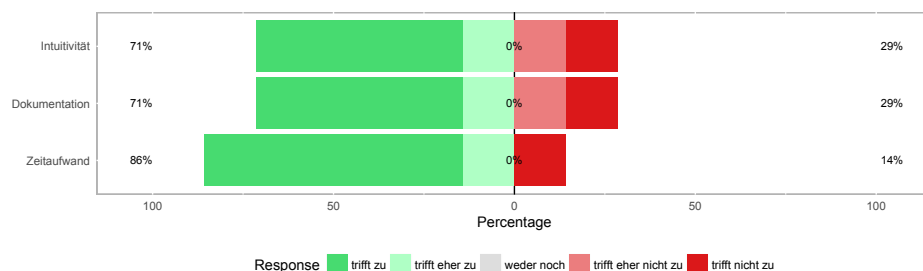


Abbildung 5.9: Quantitative Ergebnisse von Szenario 4 (Texte wiederverwenden)

Die meisten Testpersonen konnten diese Aufgabestellung problemlos lösen. Zwei Probandinnen konnten die Aufgabe allerdings nicht ohne Hilfestellung bewältigen, da die Nutzerseite nicht gefunden wurde. Hier war nicht klar, dass der Link rechts oben in der Navigation zur Nutzerkonto Seite führt und dort Nutzertexte gespeichert sind. Die Navigation könnte hier noch deutlicher und übersichtlicher gestaltet werden (s. Tabelle 6.4 im Anhang).

### Szenario 5: Wort-Verifizierung

Im letzten Szenario wechselte die Nutzerin oder der Nutzer auf die Verifizierungsseite. Hier sollten vier Einträge anderer NutzerInnen bestätigt oder verbessert werden.

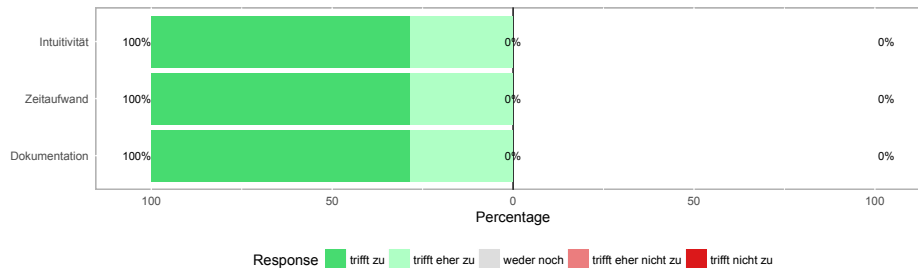


Abbildung 5.10: Quantitative Ergebnisse von Szenario 5 (Wort-Verifizierung)

Bei der Durchführung der Verifizierung gab es keine größeren Probleme. In einigen Vorschlägen bemerkten die Testpersonen, dass der Prozess zum Beispiel verbessert werden könnte, indem mehrere Wörter auf einmal in einer Liste „abgehakt“ werden könnten oder für Änderungen bei der Segmentierung die Texteingabe für die Silbentrennung vereinfacht werden könnte (s. Tabelle 6.5 im Anhang).

### Allgemeine Anmerkungen

Zum Schluss des Nutzertests wurden die ProbandInnen zu allgemeinen Äußerungen von Kritik und Vorschlägen zur gesamten Applikation aufgefordert. Wichtige Ideen und Kommentare sind im Folgenden festgehalten:

- Alte, nicht mehr aktuelle Fehlermeldungen verschwinden oft nicht (oder nur, wenn man sie selbst schließt). Diese sollten sich beim Ausführen neuer Interaktionen automatisch schließen.
- Die *Home* Seite sollte die Beschreibungen besser gruppieren (z.B. nebeneinander mit Bildern) und die Breite besser ausnutzen.
- Die türkise Farbe für die Umrandung der Kästen wirkte unpassend. Diese könnte in die blaue Farbe der Links geändert werden.
- Es wurde eine Option gewünscht, bei der die Wortbetonung in der Annotation vollständig ausgeschaltet wird und stattdessen ein einfaches, abwechselndes Einfärben der Silben (über Wortgrenzen hinaus) ermöglicht wird.
- Beim Navi-Link zum Nutzerkonto (Mailadresse rechts oben) könnte ein Bücherregal Symbol verdeutlichen, dass dort Nutzertexte zur Wiederverwendung gespeichert werden.

### 5.3 Diskussion

In der Evaluation wurde geprüft inwiefern die Ergebnisse der mit der Applikation erstellten Texte mit den gesetzten wissenschaftlichen Zielen übereinstimmen und wie Nutzerinnen und Nutzer die Bedienbarkeit der Applikation einschätzten. Ziel der Entwicklung war es eine Applikation zu schaffen, mit der Übungstexte für die LRS Therapie einfach erstellt werden können.

Zum Einen zeigen die erstellten Beispieltex-te (s. Abschnitt 5.1), dass solche Texte mit der Applikation erstellt und exportiert werden können. Im Vergleich zu bereits existierenden Lösungen (s. Abschnitt 2.1.2) bietet die entwickelte Webanwendung mehr Einstellungsmöglichkeiten und ist somit flexibler in der Gestaltung der Textannotation. Neben dieser Erkenntnis sollte der Nutzertest zeigen, dass die Texte einfach und mit wenig Zeitaufwand erstellt werden können. Der Test deckte zwar einige Probleme bei der User Experience auf, allerdings gab bei jedem Szenario die Mehrheit der NutzerInnen an, die Aufgabe intuitiv und zeiteffizient erledigt zu haben. Beschreibungen und Rückmeldungen, die von der Applikation bereitgestellt wurden, wurden ebenfalls mehrheitlich als zufriedenstellend empfunden. Dennoch würde die Weiterentwicklung und der Einsatz der Anwendung eine Überarbeitung des User Interfaces erfordern. Während des Tests gab es nützliche Anmerkungen der ProbandInnen, deren Umsetzung größtenteils mit wenig Aufwand verbunden wäre und zu einer verbesserten User Experience führen kann. Laut Aussagen der Expertinnen bietet die entwickelte Webapplikation eine Unterstützung in ihrer Arbeit als Lerntherapeutinnen und sie könnten sich vorstellen, die Applikation zur Erstellung von Unterrichtsmaterial zu verwenden.

Die Anforderungen, die in der Spezifikation erarbeitet worden sind wurden größtenteils bei der Entwicklung umgesetzt. Einige Features wie z.B. die Annotation von mehreren betonten Silben in einem Wort, oder das Generieren von Segmentierungsvorschlägen aus weiteren Quellen, wurden aufgrund des zunehmenden Umfangs der Applikation nicht weiter verfolgt. Die Features wurden als optional eingestuft und bilden, zusammen mit anderen, weiterführenden Ideen eine Möglichkeit, die Applikation in zukünftiger Arbeit zu erweitern und zu verbessern.

### 5.4 Ausblick

Einige Einschränkungen, die im Rahmen dieser Arbeit gemacht wurden, führen direkt zu Ansätzen, wie die Applikation verbessert und erweitert werden könnte:

- Der Grundwortschatz zur Analyse des Textes könnte durch die Verwendung weiterer Wörterbücher wie das PhonoLex Core ausgeweitet werden.
- Weitere linguistische Merkmale könnten in der Applikation abgebildet werden, um noch mehr Möglichkeiten für die Annotation anzubieten. Beispielsweise können in einem Wort mehrere betonte Silben auftreten und es könnte eine Unterscheidung zwischen offenen Silben (Silbe endet in Vokal, *Schu-le*) sowie geschlossenen Silben (Silbe endet in Konsonant, *trin-ken*) gemacht werden. Außerdem könnte durch weitere linguistische Analyse



die Betonung im Satz über Wortgrenzen hinaus annotiert werden um sprachrhythmische Übungen zu generieren.

- Für die Erstellung von einfachen Lesetexten wurde von den Expertinnen eine einfache Farbannotation der Silben ohne Betonung gewünscht. Silben könnten für diese Methode immer abwechselnd (auch über Wortgrenzen hinaus) eingefärbt werden. Die Implementierung dieses Features sollte beim gegebenen Stand keine große Herausforderung darstellen.
- Mithilfe einer Sprachauswahl und mit anderssprachigen Wörterbüchern könnte das Konzept leicht auf andere Sprachen wie z.B. Englisch übertragen werden.
- Einige kleinere Zusatzfeatures wie das Importieren von Textdateien oder ein direkter PDF Export (ohne den Umweg über das Druckmenü des Browsers) können die User Experience noch weiter abrunden.
- Obwohl beim Verifizierungsprozess durch das Verteilen der Verantwortung auf mehrere NutzerInnen eine Qualitätskontrolle für die zum Wörterbuch hinzugefügten Einträge stattfindet, kann nicht garantiert werden, dass diese immer korrekt sind. Auch können Fehler in der Basisdatenbank (aus dem CELEX Wörterbuch entnommen) enthalten sein. Wird ein solcher Fehler während der Annotation durch die Nutzerin oder den Nutzer entdeckt, könnte ein geeignetes User Interface die Möglichkeit geben, den fehlerhaften Eintrag zu melden und aus dem Wörterbuch zu entfernen bzw. ihn zu verbessern.

Im Web-Frontend wurde ein umfangreiches Datenmodell für den Text und die darin enthaltenen Wörter und Silben aufgebaut. Mit den vorliegenden Daten wäre es auch mit nicht allzu großem Aufwand möglich weitere Anwendungen neben der annotierten Darstellung des Texts zu finden. Es könnten weitere Module entwickelt werden, in denen der Text beispielsweise vorgelesen wird, während die zugehörigen Silben in der visuellen Darstellung hervorgehoben werden. Weiterhin wären Touch-basierte Komponenten denkbar, in denen Kinder z.B. durch das Anklicken von Wörtern Informationen zu diesen erhalten. Weiterhin könnten sie beim Lesen des Texts unterstützt werden indem sie den Text, von der Applikation geleitet, Wort für Wort im eigenen Tempo durchgehen können.

Mit dem aktuellen Stand macht es die entwickelte Web-Architektur möglich, solche Module zusätzlich zu entwerfen. Ohne große Veränderungen könnte das Frontend der Applikation außerdem auch auf mobilen Endgeräten wie Tablets zum Einsatz kommen.

Die Webanwendung stellt ein aktuelles Beispiel für Software im Bildungsbereich dar. Die durchgeführte Evaluation zeigt, dass sie in Zukunft z.B. in Lerntherapiezentren von TherapeutInnen eingesetzt werden und somit deren Arbeit erleichtern kann.



# Literaturverzeichnis

- [1] *ABC der Tiere – Silben-Generator*. <http://www.abc-der-tiere.de/index.php?id=388>, besucht: 2018-01-26.
- [2] *Amazon EC2*. <https://aws.amazon.com/de/ec2/>, besucht: 2018-01-29.
- [3] *AngularDart Documentation*. <https://webdev.dartlang.org/angular/guide>, besucht: 2018-01-25.
- [4] *Bayerisches Archiv für Sprachsignale - Aussprache-Lexikon PHONOLEX*. <http://www.phonetik.uni-muenchen.de/forschung/Bas/BasPHONOLEXdeu.html>, besucht: 2018-01-29.
- [5] *celeco Druckstation*. <https://www.celeco.de/?page=LRS{ }Texte{ }mit{ }Silbenboegen{ }drucken>, besucht: 2018-01-26.
- [6] *Eine Einführung in die Silbenmethode*. Mildeberger Verlag, 2013, ISBN 9783619501144.
- [7] *Richtig schreiben von Anfang an: mit FRESCH*. In: *Jo-Jo Sprachbuch - Aktuelle allgemeine Ausgabe: 2. Schuljahr - Schülerbuch*. Cornelsen Schulverlage GmbH, 2016, ISBN 9783060826001.
- [8] Almootassem, Omar, Syed Hamza Husain, Denesh Parthipan und Qusay H Mahmoud: *A Cloud-based Service for Real-Time Performance Evaluation of NoSQL Databases*. 2017.
- [9] Brandelik, Katharina: *Sprachrhythmische Fähigkeiten im Schriftspracherwerb*. Dissertation, 2014.
- [10] Burnage, Gavin: *CELEX: A guide for users*. 1990.
- [11] Carstensen, Kai Uwe (Herausgeber): *Computerlinguistik und Sprachtechnologie: Eine Einführung*. Elsevier, Spektrum Akad. Verl., München, 2. Auflage, 2004, ISBN 3-8274-1407-5.
- [12] Claudia Steinbrink und Thomas Lachmann: *Lese-Rechtschreibstörung*. Springer VS, Berlin, Heidelberg, 2014, ISBN 9783642418419.
- [13] Fielding, Roy Thomas: *Architectural Styles and the Design of Network-based Software Architectures*. Dissertation, 2000, ISBN 0-599-87118-0.

- [14] Findeisen, Uwe, Reinhard Kargl, Christian Purgstaller, Andreas Fink, Martin Schöfl, Liane Kaufmann, David Gerlach, Stefan Heim, Marion Grande, Maria Klatte, Claudia Steinbrink, Alexander Pröhl, Barbara Estner, Corinna Christmann, Thomas Lachmann, Katharina Brandelik, Jens Holger Lorenz, Edeltraud Koschay, Susanne Trauzettel-Klosinski, Sandra Ohlenforst, Ursula Fischer, Korbinian Moeller, Annemarie Fritz, Antje Ehlert, Carola Reuter-Liehr, Alexander Geist, Claudia Mähler, Kirsten Schuchardt und Gerd Schulte-Körne: *Legasthenie und Dyskalkulie: Neue Methoden zu Diagnostik und Förderung*. Winkler, Dieter, Bochum, 2014, ISBN 978-3-89911-213-9 KT.
- [15] Grinberg, Miguel: *Flask Web Development: Developing Web Applications with Python*. O'Reilly Media, Inc., 1st Auflage, 2014, ISBN 1449372627, 9781449372620.
- [16] Holz, Heiko, Katharina Brandelik, Jochen Brandelik, Alexandra Kirsch, Jürgen Heller und Detmar Meurers: *Prosodiya - A Mobile Game for German Dyslexic Children*. In: Dias, João, Pedro A Santos und Remco C Veltkamp (Herausgeber): *Games and Learning Alliance: 6th International Conference, GALA 2017, Lisbon, Portugal, December 5–7, 2017, Proceedings*, Seiten 73–82. 2017, ISBN 9783319719405.
- [17] Krauß, Andrea: *Orthographieerwerb von Beginn an - Ein silbenorientiertes Konzept für den Anfangsunterricht*. de Gruyter, Berlin [u.a.], 2010, ISBN 978-3-11-023224-0.
- [18] Kuthy, Kordula De, Ramon Ziai und Detmar Meurers: *Learning what the crowd can do : A case study on focus annotation*. In: *Proceedings of the 6th Conference on Quantitative Investigations in Theoretical Linguistics*, Tübingen, 2015. Universität Tübingen.
- [19] Lewis, James R.: *IBM Computer Usability Satisfaction Questionnaires: Psychometric Evaluation and Instructions for Use*. International Journal of Human-Computer Interaction, 7(1):57–78, 1995, ISSN 15327590.
- [20] Loper, Edward und Steven Bird: *NLTK: The Natural Language Toolkit*. In: *Proceedings of the ACL-02 Workshop on Effective tools and methodologies for teaching natural language processing and computational linguistics -*, Band 1, Seiten 63–70, Morristown, NJ, USA, 2002. Association for Computational Linguistics, ISBN 1932432841.
- [21] Mayer, Andreas: *Lese-Rechtschreibstörungen (LRS)*. 2016, ISBN 9783838586625.
- [22] Mikowski, Michael S und Josh C Powell: *Single page web applications*. B and W, 2013.
- [23] Rauterberg, Matthias: *Usability Engineering*. Morgan Kaufmann, 1996, ISBN 978-0-08-052029-2.
- [24] Reichel, Uwe: *PermA and Balloon: Tools for string alignment and text processing*. Proc. Interspeech, (January 2012), 2012.
- [25] Rello, Luz und Ricardo Baeza-Yates: *How to Present More Readable Text for People with Dyslexia*. Univers. Access Inf. Soc., 16(1):29–49, 2017, ISSN 1615-5289.
- [26] Rossum, Guido van und Fred L Drake: *The Python Language Reference Manual*. Network Theory Ltd., 2011, ISBN 1906966141, 9781906966140.

- [27] Schiel, Florian, Christoph Draxler, Phil Hoole und Hans G Tillmann: *New resources at BAS: acoustic, multimodal, linguistic*. Eurospeech, Seiten 2271–2274, 1999.
- [28] Schröder, M. und J. Trouvain: *The German Text-to-Speech Synthesis System MARY: A Tool for Research, Development and Teaching*. International Journal of Speech Technology, 6:365–377, 2003, ISSN 1381-2416.
- [29] Snow, Rion, Brendan O’Connor, Daniel Jurafsky und Andrew Y Ng: *Cheap and Fast—but is It Good?: Evaluating Non-expert Annotations for Natural Language Tasks*. In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP ’08*, Seiten 254–263, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics.
- [30] Stent, Amanda, Jinho D Choi, West St, West St und New York: *It Depends : Dependency Parser Comparison Using A Web-based Evaluation Tool*. Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), Seiten 387–396, 2015.
- [31] Van Kesteren, Anne: *Cross-origin resource sharing*. W3C Working Draft WD-cors-20100727, latest version available at <http://www.w3.org/TR/cors>, 2010.
- [32] Zaidan, Omar F und Chris Callison-Burch: *Crowdsourcing Translation: Professional Quality from Non-professionals*. In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1, HLT ’11*, Seiten 1220–1229, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics, ISBN 978-1-932432-87-9.



# 6 Anhang

## Nutzertest

### Beispieltext

Einem reichen Manne, dem wurde seine Frau krank, und als sie fühlte, dass ihr Ende herankam, rief sie ihr einziges Töchterlein zu sich ans Bett und sprach: "Liebes Kind, bleibe fromm und gut, so wird dir der liebe Gott immer beistehen, und ich will vom Himmel auf dich herabblicken, und will um dich sein." Darauf tat sie die Augen zu und verschied. Das Mädchen ging jeden Tag hinaus zu dem Grabe der Mutter und weinte, und blieb fromm und gut. Als der Winter kam, deckte der Schnee ein weißes Tüchlein auf das Grab, und als die Sonne im Frühjahr es wieder herabgezogen hatte, nahm sich der Mann eine andere Frau.

Abbildung 6.1: Beispieltext *Aschenputtel*

Problem	Lösungsvorschlag
Statt ein neues Konto zu erstellen gab die Testperson Mailadresse und Passwort in das Login Formular ein.	Separate Links in der Navigation für Login/Konto erstellen oder <i>Neues Konto erstellen</i> Link auf der Login Seite vor dem Login platzieren.
Email und Passwort wurden beim Konto erstellen vergessen, da der Cursor beim Laden der Seite zum Feld <i>Name</i> springt.	Cursor ganz nach oben, ins Email Feld setzen.
Beim Feld <i>Name</i> verwirrte die Beschreibung <i>Vor- und Nachname werden bei der Verifizierung angezeigt, wenn Sie einen Vorschlag abschicken</i> . Es war nicht klar, was hier ein <i>Vorschlag</i> ist.	Besser beschreiben, was hier mit Vorschlag gemeint ist (Vorschlag für die Wortsegmentierung bei der Verifizierung).
Auf der <i>Home</i> Seite wurde Login/Konto erstellen nicht gefunden, da es nicht offensichtlich war, das man nach unten scrollen kann.	-

Tabelle 6.1: Nutzeranmerkungen zu Szenario 1



Problem	Lösungsvorschlag
Die Einstellungsleiste links geht so weit nach unten, dass beim Ändern der Einstellungen immer wieder hoch und runter gescrollt werden musste. Die Testperson hätte bei den Änderungen lieber den Vorschautext immer im Blick gehabt.	Einstellungen kompakter darstellen, Farbauswahl verkleinern.
Das Farbauswahl Fenster war zu umständlich (Fenster musste immer manuell geschlossen werden, Klicken außerhalb funktionierte nicht).	-
Die Beschreibungen zu den Einstellungen <i>Wort Hintergrundfarbe</i> und <i>Silbe hervorheben</i> sind unzureichend und verwirrend. Es wurde nicht selbständig verstanden, was diese tun.	Verständlichere Beschriftung der Optionen (z.B. Schriftfarbe und Hintergrundfarbe). Kurze Erklärung, z.B. Popup bei Hover über ein Info-Symbol.
Bei der manuellen Segmentierung lieferten die Vorschläge Wörter ohne Betonung, die es nicht geben sollte.	Nur Vorschläge mit Betonung erlauben.
Bei der manuellen Segmentierung wurde der <i>Speichern</i> Button nicht gefunden, da nicht klar war, dass nach unten gescrollt werden kann.	-
Eine Testperson aus der Expertengruppe vermisste eine schnelle Einstellung, sodass einsilbige Wörter immer in abwechselnden Farben dargestellt werden können.	-
Die Satzzeichen waren falsch gesetzt, z.B. müssten Punkte und Kommas immer direkt (ohne Lücke) an das vorherige Wort anschließen.	-

Tabelle 6.2: Nutzeranmerkungen zu Szenario 2

Problem	Lösungsvorschlag
Bei der zweiten Einstellung (Hintergrundfarbe pro Silbe und schwarzer Text) wurden die Silbenfarben auf Schwarz gestellt (wegen der Textfarbe). Dass man mit der Option <i>Silbe hervorheben - Hintergrundfarbe</i> umstellen kann, ob sich die Silbenfarben auf Schrift- oder Hintergrundfarben beziehen, war nicht klar.	Die Option <i>Silbe betonen</i> sollte besser beschrieben werden. Sie sollte an den Anfang der Einstellungen gesetzt werden und ihre Wichtigkeit sollte betont werden (etwa als <i>Modus</i> - Schriftfarbe vs. Hintergrundfarbe). Bei den beiden Optionen von <i>Silbe betonen</i> sollte jeweils ein hervorgehobenes Beispielwort neben dem Menü verdeutlichen, wie es ungefähr aussehen wird.
Statt die Vorlage zu speichern, speicherten Testpersonen den Text mit dem Namen <i>Vorlage 1</i> .	-
Der Kasten für die Vorlagen befand sich zu weit unten auf der Seite.	-
Die Meldung <i>Vorlage angelegt</i> verschwindet nicht beim Wechseln der Vorlagen, was verwirrend war, weil sie sich auf eine vorherige Aktion bezog.	Meldungen sollten sich automatisch schließen, wenn eine neue Interaktion stattfindet.
Die Einstellungen wurden als schlecht gruppiert empfunden.	Farbeinstellungen nebeneinander platzieren (etwa so, wie es im Wort aussieht). Das erleichtert das Verständnis und würde weniger Platz wegnehmen.
Eine Testperson aus der Expertengruppe hätte gerne sowohl Schriftfarben als auch Hintergrundfarben individuell angepasst.	-
Beim Speichern der Vorlagen war nicht ersichtlich, dass der Vorlagename in einem Textfeld steht und verändert werden kann.	Textfeld besser als solches kennzeichnen.

Tabelle 6.3: Nutzeranmerkungen zu Szenario 3

Problem	Lösungsvorschlag
Die Nutzerkonto Seite wurde erst nach längerem Ausprobieren oder nach einem Hinweis gefunden.	Auf der <i>Home</i> Seite war der Link immer noch mit <i>Einloggen oder Konto erstellen</i> beschriftet. Dieser sollte dann <i>Zum Nutzerkonto</i> lauten.
Es war nicht ersichtlich, dass Texte im Nutzerkonto gespeichert wurden.	-
Es war nicht klar, welche <i>Seite</i> in der Navigation gerade ausgewählt ist.	In der Navigation sollte die aktuelle Seite hervorgehoben dargestellt werden (z.B. unterstrichen).
Die Unterscheidung von Vorlagen und Nutzertexten war nicht klar. Manche Testpersonen dachten, dass die Texte zusammen mit den Vorlagen gespeichert werden.	-
Die Beschreibung des Buttons <i>Verwenden</i> beim ausgewählten Nutzertext ist nicht einheitlich mit der Textanalyse.	Ändern in <i>Analysieren</i> .

Tabelle 6.4: Nutzeranmerkungen zu Szenario 4

Problem	Lösungsvorschlag
Die Beschreibung des Buttons <i>Abschicken</i> gefällt nicht.	Ändern in <i>Bestätigen/Verifizieren</i> .
Bei vielen Vorschlägen ist der Button <i>Abschicken</i> zu weit unten. Es war nicht klar, was abgeschickt wird (dass der linke Teil die eigene Segmentierung darstellt).	Der Button sollte zusammen mit der Betonung und Silbentrennung gruppiert sein und direkt darunter stehen.
Es wurde als umständlich empfunden, dass immer nur ein Wort auf einmal verifiziert werden kann.	Eventuell Liste mit Wörtern zum „Abhaken“ anzeigen. Vorschläge nur bei Bedarf durch Aufklappen anzeigen.
Die manuelle Anpassung der Silbentrennung wurde als umständlich empfunden, da sich der Text nicht verändern lässt (neu schreiben des Textes ist nicht möglich).	Erlauben, das Wort mit Trennstrichen neu zu schreiben, aber kontrollieren, dass der Worttext danach nicht verändert ist. Erklärung, dass Text nicht verändert werden kann, sondern nur Trennstriche gesetzt werden können.

Tabelle 6.5: Nutzeranmerkungen zu Szenario 5

### Aufgabe 1: Nutzerkonto

- Legen sie ein neues Nutzerkonto mit einer beliebigen Email Adresse und Ihrem Namen an, verwenden Sie als Passwort "12345678".
- Loggen Sie sich mit dem neu erstellten Konto ein und anschließend wieder aus.

---

Bitte kreuzen Sie für die folgenden Aussagen Zutreffendes an:

**Die Aufgabe konnte ich intuitiv und problemlos erledigen.**

☐ trifft zu    ☐ trifft eher zu    ☐ weder noch    ☐ trifft eher nicht zu    ☐ trifft nicht zu

**Ich halte die Zeit, die ich gebraucht habe um die Aufgabe zu erledigen, für angemessen.**

☐ trifft zu    ☐ trifft eher zu    ☐ weder noch    ☐ trifft eher nicht zu    ☐ trifft nicht zu

**Ich bin mit den Informationen, die ich während der Bearbeitung in der App erhalten habe (Beschreibungen, Rückmeldungen) zufrieden.**

☐ trifft zu    ☐ trifft eher zu    ☐ weder noch    ☐ trifft eher nicht zu    ☐ trifft nicht zu

### Aufgabe 2: Textanalyse

- Loggen Sie sich ein und gehen Sie zur Textanalyse. Kopieren Sie einen vom Interviewer vorgegebenen Text in das Textfeld und analysieren Sie diesen.
- Klären Sie alle unbekannten Wörter (rot hinterlegt) manuell.
- Experimentieren Sie kurz mit den Einstellungen für die Vorschau bis Sie eine Einstellung finden, die Sie anspricht.

---

Bitte kreuzen Sie für die folgenden Aussagen Zutreffendes an:

**Die Aufgabe konnte ich intuitiv und problemlos erledigen.**

☐ trifft zu    ☐ trifft eher zu    ☐ weder noch    ☐ trifft eher nicht zu    ☐ trifft nicht zu

**Ich halte die Zeit, die ich gebraucht habe um die Aufgabe zu erledigen, für angemessen.**

☐ trifft zu    ☐ trifft eher zu    ☐ weder noch    ☐ trifft eher nicht zu    ☐ trifft nicht zu

**Ich bin mit den Informationen, die ich während der Bearbeitung in der App erhalten habe (Beschreibungen, Rückmeldungen) zufrieden.**

☐ trifft zu    ☐ trifft eher zu    ☐ weder noch    ☐ trifft eher nicht zu    ☐ trifft nicht zu

Abbildung 6.2: Nutzertest: Szenarien 1 und 2

### Aufgabe 3: Annotationsvorlagen

- Versuchen Sie, die Textannotation so einzustellen, dass es etwa wie im Folgenden aussieht:

Ei = nem rei = chen Man = ne , dem wur = de sei = ne Frau krank , und als

- Speichern Sie die Einstellungen als Vorlage mit dem Namen "Vorlage 1".
- Versuchen Sie nun, die Textannotation so einzustellen, dass es etwa wie im Folgenden aussieht:

Einem reichen Manne , dem wurde seine Frau krank , und als

- Speichern Sie die Einstellungen als Vorlage mit dem Namen "Vorlage 2"
- Wechseln Sie zwischen "Vorlage 1" und "Vorlage 2" hin und her.

---

Bitte kreuzen Sie für die folgenden Aussagen Zutreffendes an:

**Die Aufgabe konnte ich intuitiv und problemlos erledigen.**

☐ trifft zu ☐ trifft eher zu ☐ weder noch ☐ trifft eher nicht zu ☐ trifft nicht zu

**Ich halte die Zeit, die ich gebraucht habe um die Aufgabe zu erledigen, für angemessen.**

☐ trifft zu ☐ trifft eher zu ☐ weder noch ☐ trifft eher nicht zu ☐ trifft nicht zu

**Ich bin mit den Informationen, die ich während der Bearbeitung in der App erhalten habe (Beschreibungen, Rückmeldungen) zufrieden.**

☐ trifft zu ☐ trifft eher zu ☐ weder noch ☐ trifft eher nicht zu ☐ trifft nicht zu

Abbildung 6.3: Nutzertest: Szenario 3

**Aufgabe 4: Texte wiederverwenden**

- Zur Wiederverwendung lässt sich der Text im Nutzerkonto speichern. Wählen Sie einen Titel und speichern Sie den Text.
- Wechseln Sie auf die Nutzerkonto Seite und analysieren Sie den gespeicherten Text neu.

---

Bitte kreuzen Sie für die folgenden Aussagen Zutreffendes an:

**Die Aufgabe konnte ich intuitiv und problemlos erledigen.**

☐ trifft zu    ☐ trifft eher zu    ☐ weder noch    ☐ trifft eher nicht zu    ☐ trifft nicht zu

**Ich halte die Zeit, die ich gebraucht habe um die Aufgabe zu erledigen, für angemessen.**

☐ trifft zu    ☐ trifft eher zu    ☐ weder noch    ☐ trifft eher nicht zu    ☐ trifft nicht zu

**Ich bin mit den Informationen, die ich während der Bearbeitung in der App erhalten habe (Beschreibungen, Rückmeldungen) zufrieden.**

☐ trifft zu    ☐ trifft eher zu    ☐ weder noch    ☐ trifft eher nicht zu    ☐ trifft nicht zu

**Aufgabe 5: Wort Verifizierung**

- Helfen Sie mit bei der Verifizierung neuer Wörter für die Datenbank. Wechseln Sie dazu auf die Verifizierungsseite und bestätigen Sie vier Einträge.

---

Bitte kreuzen Sie für die folgenden Aussagen Zutreffendes an:

**Die Aufgabe konnte ich intuitiv und problemlos erledigen.**

☐ trifft zu    ☐ trifft eher zu    ☐ weder noch    ☐ trifft eher nicht zu    ☐ trifft nicht zu

**Ich halte die Zeit, die ich gebraucht habe um die Aufgabe zu erledigen, für angemessen.**

☐ trifft zu    ☐ trifft eher zu    ☐ weder noch    ☐ trifft eher nicht zu    ☐ trifft nicht zu

**Ich bin mit den Informationen, die ich während der Bearbeitung in der App erhalten habe (Beschreibungen, Rückmeldungen) zufrieden.**

☐ trifft zu    ☐ trifft eher zu    ☐ weder noch    ☐ trifft eher nicht zu    ☐ trifft nicht zu

Abbildung 6.4: Nutzertest: Szenarien 4 und 5

# **Erklärung**

Hiermit erkläre ich, dass ich diese schriftliche Abschlussarbeit selbständig verfasst habe, keine anderen als die angegebenen Hilfsmittel und Quellen benutzt habe und alle wörtlich oder sinngemäß aus anderen Werken übernommenen Aussagen als solche gekennzeichnet habe.

Ort, Datum

Unterschrift