h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

Hochschule Darmstadt

– Fachbereich Informatik –

Design und Konzeption von Suchfunktionalitäten in digitalen historisch-kritischen Editionen

Abschlussarbeit zur Erlangung des akademischen Grades

Master of Science (M.Sc.)

vorgelegt von

Matthias Fabinski

Matrikelnummer: 753681

Referentin	:	Prof. Dr. Uta Störl
Korreferent	:	Tobias Holstein, Lic., M.Sc.

Matthias Fabinski: *Design und Konzeption von Suchfunktionalitäten in digitalen historisch-kritischen Editionen,* © 01. Dezember 2020

ERKLÄRUNG

Ich versichere hiermit, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die im Literaturverzeichnis angegebenen Quellen benutzt habe.

Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder noch nicht veröffentlichten Quellen entnommen sind, sind als solche kenntlich gemacht.

Die Zeichnungen oder Abbildungen in dieser Arbeit sind von mir selbst erstellt worden oder mit einem entsprechenden Quellennachweis versehen.

Diese Arbeit ist in gleicher oder ähnlicher Form noch bei keiner anderen Prüfungsbehörde eingereicht worden.

Viernheim, 01. Dezember 2020

Matthias Fabinski

Matthias Fabinski

ABSTRACT

This master thesis aims to identify the requirements and problem statements of users searching in digital (scholarly) editions and to develop and evaluate user-centric solutions.

The thesis is based on the double-diamond design process method (British Design Council 2005). To define the problem statements, a competitor analysis of 42 digital (scholarly) editions was conducted. As a starting point, the criteria for reviewing scholarly digital editions by Patrick Sahle (2014) was extended by attributes linked to searching. Furthermore, an online survey was conducted to identify the user requirements. Based on the consolidated requirements precise problem statements were deducted.

Afterwards, solutions for the identified problem statements were developed using sketching and prototyping. They were then implemented within a high fidelity prototype as a web application based on the digital scholarly edition for correspondences of Frank Wedekind. Guerilla Testing and a remote usability test among other methods were used to refine and evaluate the solutions in an iterative manner.

The results of this thesis include the identified problem statements as well as the solutions which were implemented and evaluated as part of a high fidelity prototype. To ensure the correct functionality of the prototype while combining the various problem statements was particularly challenging. A configurable full text search with entity and metadata filters was implemented with an user-friendly results page. Moreover, the architecture of the prototype allows for a fast delivery of solution approaches and is available for further usage as open source.

ZUSAMMENFASSUNG

Ziel dieser Masterarbeit ist es, die Anforderungen und Problemstellungen zu ermitteln, die Anwender bei der Verwendung einer Suchfunktion in digitalen (historisch-kritischen) Editionen haben, hierfür nutzerzentriert Lösungsansätze zu entwickeln und diese zu evaluieren.

Die vorliegende Arbeit orientiert sich am Double-Diamond Design Prozess (British Design Council 2005). Zur Definition der Problemstellungen wurde eine Marktforschung in Form einer Konkurrenzanalyse von 42 digitalen (historisch-kritischen) Editionen durchgeführt. Grundlage bildet dabei der Kriterienkatalog für die Besprechung digitaler Editionen von Patrick Sahle (2014), welcher um Attribute mit Suchbezug ergänzt wurde. Außerdem wurde eine Nutzerrecherche mittels einer Onlineumfrage durchgeführt, um die Bedürfnisse der Benutzer zu identifizieren. Basierend auf den konsolidierten Anforderungen wurden konkrete Problemstellungen abgeleitet.

Anschließend wurden für diese identifizierten Problemstellungen mittels Sketching und Protoyping Lösungsansätze entwickelt. Am Beispiel der Briefedition Wedekind wurden diese in einem High-Fidelity-Prototypen als Web-Applikation umgesetzt und unter anderem mit Guerilla Testing und einem Remote-Usability-Test iterativ weiterentwickelt und evaluiert.

Ergebnisse der Arbeit sind sowohl die identifizierten Problemstellungen als auch die dafür entwickelten Lösungsansätze, die in einem performanten High-Fidelity-Prototypen umgesetzt und evaluiert wurden. Die Kombination der unterschiedlichen Problemstellungen zu einem funktionierenden Prototyp stellte eine der zentralen Herausforderungen dieser Arbeit dar. Es wurde eine konfigurierbare Volltextsuche mit Entitäts- und Metadatenfiltern und einer benutzerfreundlichen Ergebnisdarstellung implementiert. Die Architektur des Prototyps erlaubt die schnelle Auslieferung neuer Lösungsansätze und steht unter Open-Source Lizenz zur weiteren Verwendung zur Verfügung.

INHALTSVERZEICHNIS

I THESIS

1 EINLEITUNG 2

2 GRUNDLAGEN 4

2.1 Digitale Editionen 4

2.2 Information Retrieval 5

2.3 Volltextsuche in Datenbanken 8

2.3.1 Rudimentäre Textsuche 8

2.3.2 Datentypen und Operatoren der Volltextsuche 8

2.3.3 Funktionen der Volltextsuche 10

2.3.4 Konfiguration der Volltextsuche 12

2.4 Nutzerzentriertes Design 14

2.4.1 Usability 14

2.4.2 Anforderungsanalyse 15

2.4.3 Design-Methoden für User Interfaces 16

2.4.4 Methoden zur Evaluation von User Interfaces 17

2.4.5 Design-Prozessmodell 20

3 DISCOVER UND DEFINE 23

3.1 Design der Konkurrenzanalyse 24

3.2 Auswertung der Konkurrenzanalyse 26

3.3 Aufbau der Umfrage 30

3.4 Auswertung der Umfrage 32

3.5 Zusammenfassung der Analysen 35

3.6 Definition der Problemstellung 36

3.7 Übersicht der Problemstellungen 43

4 DEVELOP UND DELIVER 48

4.1 Beschreibung der Evaluations Methodik 48

4.2 Übersicht über die Iterationen 50

4.3 Erste Iteration 51

4.3.1 Problemstellung 51

4.3.2 Develop 51

4.3.3 Deliver 54

4.4 Zweite Iteration 55

4.4.1 Problemstellung 55

4.4.2 Develop 56

4.4.3 Deliver 61

4.5 Dritte Iteration 61

4.5.1 Problemstellung 61

4.5.2 Develop 62

4.5.3 Deliver 67

4.6 Vierte Iteration 69

4.6.1 Problemstellung 69

4.6.2 Develop und Deliver 69
4.7 Fünfte Iteration 77
4.7.1 Problemstellung 77
4.7.2 Develop 77
4.7.3 Deliver 83
4.8 Sechste Iteration 85
4.8.1 Problemstellung 85
4.8.2 Develop 85
4.8.3 Deliver 88
4.9 Siebte Iteration 94
4.9.1 Problemstellung 94
4.9.2 Develop 94
4.9.3 Deliver 97
5 FAZIT 104

II ANHANG
A ANHANG A 108
A.1 Anschreiben 108
A.2 Fragebogen Deutsch 109
A.3 Fragebogen Englisch 116
B ANHANG B 123
B.1 Notizen Feedback persönliches Gespräch zu Iteration 1 123
B.2 Notizen Feedback Telefonat zu Iteration 1 124
B.3 Feedback zu Iteration 3 124
C ANHANG C 127
D ANHANG D 132
D.1 Interview und Aufgabenstellungen 132
D.2 Usabilitytest mit P-7 133
D.3 Usabilitytest mit P-8 136
D.4 Usabilitytest mit P-9 139
D.5 Usabilitytest mit P-10 141
E ANHANG E 144

LITERATUR 145

ABBILDUNGSVERZEICHNIS

Abbildung 2.1 IR-System als Blackbox 5
Abbildung 2.2 Interaktion des Informationssuchenden mit dem IR-System 6
Abbildung 2.3 Architektur eines Text-IR Systems 7
Abbildung 2.4 Verschiedene Ausprägungen von Protoypen 17
Abbildung 2.5 Prototyping as Iterative Incremental Refinement 20
Abbildung 2.6 Design as Branching Exploration and Comparison . . . 21
Abbildung 2.7 Die Phasen des Double-Diamond Prozesses 21
Abbildung 3.1 Hervorhebung des Discover und Define Diamanten . . 23
Abbildung 3.2 Demographie der Umfrage 32
Abbildung 3.3 Hintergrund und Erfahrung der Befragten 33
Abbildung 3.4 Was die Editionen der Befragten an Funktionalitäten anbieten 45
Abbildung 3.5 Erwartung an Funktionalität in Editionen 46
Abbildung 3.6 Relevanz von Metadaten in der Darstellung der Suchergebnisse 46
Abbildung 3.7 Mindmap mit Zusammenfassung der aufgenommenen Themen 47
Abbildung 4.1 Hervorhebung des Develop und Deliver Diamanten . . 48
Abbildung 4.2 Die Iterationen im zeitlichen Verlauf 50
Abbildung 4.3 Ergebnis Darstellung 52
Abbildung 4.4 Ergebnis Darstellung 53
Abbildung 4.5 Möglichkeiten Filter umzusetzen 54
Abbildung 4.6 Komponenten der Webapplikation 57
Abbildung 4.7 Screenshot des ersten High-Fidelity-Prototyps 60
Abbildung 4.8 Screenshot des weiterentwickelten High-Fidelity-Prototyps 67
Abbildung 4.9 Entitätsfilter in unterschiedlichen Rollen PS-04 71
Abbildung 4.10 Sketch1 72
Abbildung 4.11 Suche in Kommentaren PS-06 73
Abbildung 4.12 Flexible Fehlertoleranz PS-08 74
Abbildung 4.13 Vorschlag alternativer Suchbegriffe PS-10 75
Abbildung 4.14 Erschließung von Hintergrundinformationen PS-12 . . 76
Abbildung 4.15 Phrasensuche PS-14 77
Abbildung 4.16 Mockup des Layouts für Iteration 5 78
Abbildung 4.17 Sketch der Selektion von Entitäten in unterschiedlichen Rollen 79
Abbildung 4.18 Mockup der Filter für Iteration 5 80
Abbildung 4.19 Sketch der konfigurierbaren Volltextsuche 82
Abbildung 4.20 Screenshot des weiterentwickelten High-Fidelity-Prototyps 84
Abbildung 4.21 Mockup der Explain Component 85
Abbildung 4.22 Screenshot des Layout Entwurfs 88

Abbildung 4.23 Version des Prototyps beim Usability Test mit P-6 . . . 89
Abbildung 4.24 Aktive Volltextsuche behindert Entitätssuche 89
Abbildung 4.25 Performancetest Virtual Private Server 93
Abbildung 4.26 Performancetest Dedicated Server 94
Abbildung 4.27 Hinweis auf aktive Volltextsuche 96
Abbildung 4.28 Das finale Layout der Suchfunktion 97

TABELLENVERZEICHNIS

Tabelle 3.1 Erläuterung der Metadaten über die Editionen der Konkurrenzanalyse 24
Tabelle 3.2 Erläuterung der Felder in der Konkurrenzanalyse . . . 26
Tabelle 3.3 Quantitative Auswertung der (Such-)funktionalitäten in untersuchten Editionen 27
Tabelle 3.4 Quantitative Auswertung des Vorkommens von Registern in den untersuchten Editionen 28
Tabelle 3.5 Quantitative Auswertung der Filtermöglichkeiten in den untersuchten Editionen 29
Tabelle 3.6 Quantitative Auswertung des Vorkommens von Darstellung der Metadaten in den untersuchten Editionen 29
Tabelle 3.7 Quantitative Auswertung der Sortieroptionen in den untersuchten Editionen 30
Tabelle 3.8 Ausprägungen der verwendeten fünfstufigen Likert-Skala 31
Tabelle 3.9 Funktionalitäten mit Priorisierung 36
Tabelle 3.10 Übersicht über die Problemstellungen 44
Tabelle 4.1 Übersicht über die an der Evaluation beteiligten Personen 49
Tabelle 4.2 Wahrheitstabelle der Volltextsuchkombinationen 82
Tabelle 4.3 Übersicht über die Testumgebungen 91

LISTINGS

Listing 2.1 Ein Text umgewandelt in *tsvector* Darstellung 8
Listing 2.2 Die *tsvector* Darstellung mit Position und Gewichtung . 9
Listing 2.3 Die Rangfolge bei einer *tsquery* mit mehreren Operatoren 9
Listing 2.4 Beispiel für *websearch_to_tsquery* 11
Listing 2.5 Ein Text umgewandelt in *tsvector* Darstellung 14
Listing 4.1 Die WHERE Bedingung für Volltextsuche 58
Listing 4.2 Bestimmung des Treffers im Kontext 58
Listing 4.3 Ranking 59
Listing 4.4 SearchView Template 59
Listing 4.5 *State* des Frontends 64
Listing 4.6 Ausführen der Suche mit *fetchSearchResults* 65
Listing 4.7 Definition einer eigenen immutable *unaccent* Funktion 66
Listing 4.8 Erstellung des Indexes 66
Listing 4.9 JSON-Repräsentation der Entitäten im Filterarray . . . 80
Listing 4.10 Dictionaries mit Reduktion auf deutschen Wortstamm und mit Entfernung von Stoppworten 82
Listing 4.11 Textsuchkonfiguration mit Reduktion auf deutschen Wortstamm 83
Listing 4.12 JSON-Repräsentation der Entitäten im Lookup Objekt . 86
Listing 4.13 Anreicherung der Daten für die Filterdarstellung . . . 87
Listing 4.14 Use-Case Script 92
Listing 4.15 Optimierung Autovervollständigung Performance . . . 95
Listing 4.16 Leeres Ergebnis Template 95

Teil I

THESIS

1 EINLEITUNG

„Frank Wedekind (1864-1918) zählt heute wie kein anderer zu den bahnbrechenden Autoren der literarischen Moderne (..)“ [6]. Wedekind war zu seinen Lebzeiten gut vernetzt und so entstand eine große Anzahl von Korrespondenzen von und an Wedekind [6]. Von der Editions- und Forschungsstelle Wedekind wurden rund 3600 Korrespondenzstücke ermittelt und als Kopien gesammelt, wobei etwa 2800 noch unveröffentlicht sind und im Rahmen des Forschungsprojektes erschlossen werden [11]. Die überlieferten Korrespondenzen werden dabei nicht nur transkribiert, ediert und editionswissenschaftlich kommentiert, sondern es wird auch der historische Kontext durch Kommentare erschlossen [6].

Diese Arbeit beschäftigt sich mit der Entwicklung einer webbasierten Suchfunktion im Kontext von digital (historisch-kritischen) Editionen. Ziel der Arbeit ist es, die Anforderungen zu ermitteln, die Anwender an eine Suchfunktion in digitalen (historisch-kritischen) Editionen haben. In einem nutzerzentrierten Ansatz werden Lösungsansätze entwickelt und evaluiert. Zusätzlich wird ein Prototyp entwickelt, der die Evaluation und Weiterentwicklung der Lösungsansätze ermöglicht.

Zunächst werden Grundlagen zu digitalen Editionen erläutert und beschrieben was diese auszeichnet. Danach wird auf den Aufbau und die Methoden des Information Retrieval eingegangen, im Speziellen auf den Prozess des Informationssuchenden, welcher sich mit dem Auffinden von Dokumenten beschäftigt. Danach folgen die technischen Grundlagen der Volltextsuche mit Datenbanken und anschließend wird auf den nutzerzentrierten Designprozess eingegangen, welcher den methodischen Leitfaden für den weiteren Aufbau der Arbeit darstellt.

Im Kapitel *Discover und Decide* wird erläutert, wie die Konkurrenzanalyse und die Umfrage zur Ermittlung der Anforderungen und Problemstellungen durchgeführt wurde. Anschließend werden die Ergebnisse hiervon kategorisiert und dargestellt. Basierend darauf können Lösungen entwickelt und evaluiert werden.

Das Kapitel *Develop und Deliver* beschreibt die Entwicklung von Lösungsansätzen in einem iterativen Ansatz, mit dem Ziel nutzerzentriert Lösungen zu bestimmen und diese für den Nutzer optimal abzubilden. Hierzu werden Methoden wie Sketching und Prototyping eingesetzt und es wird die Konzeption und Entwicklung des High-Fidelity-Prototyps beschrieben, welcher anschließend zur Evaluation der Lösungsansätze mit Anwendern verwendet wird. Außerdem wird die Anwendungsperformance mit einem Stresstest evaluiert.

Die Arbeit wird mit einem Fazit abgeschlossen, welches die verwendete Methodik und die Ergebnisse der Arbeit zusammenfasst. Anknüpfend an

die Ergebnisse wird im Ausblick darstellt, welche weiteren Perspektiven untersucht werden sollten.

2 GRUNDLAGEN

In diesem Kapitel werden die Grundlagen für das Verständnis dieser Arbeit beschrieben. Zunächst werden digitale Editionen erläutert, in deren Kontext die Suchfunktionalitäten untersucht werden.

Aus dem Information Retrieval wird der Prozess des Informationssuchenden beschrieben. Der Prozess gibt einen guten Überblick über den Aufbau und die Komponenten eines Information Retrieval Systems, und ist somit für das Verständnis der Arbeit hilfreich.

Danach wird auf die Volltextsuche in Datenbanken eingegangen, welche die technische Grundlage für die Umsetzung der Suchfunktion bilden. Es werden die benötigten Funktionalitäten erläutert, die zur Umsetzung mit PostgreSQL benötigt werden.

In Abschnitt 2.4 auf Seite 14 werden Begriffe und Methoden des nutzerzentrierten Designs vorgestellt. Für die Anforderungsanalyse und die iterative, nutzerzentrierte Entwicklung der Suchfunktion wird der Double-Dimond Designprozess beschrieben.

2.1 DIGITALE EDITIONEN

In dieser Arbeit werden die Suchmöglichkeiten in digitalen (historisch-kritischen) Editionen betrachtet. Daher soll zunächst beschrieben werden, was eine digitale (historisch-kritische) Edition ist.

Eine „Edition ist die erschließende Wiedergabe historischer Dokumente“, lautet eine allgemeine Definition von Sahle[44, S. 138]. Sahle definiert weiter, dass Dokumente hierbei nicht zwingend schriftlich sein müssen, sondern es kann grundsätzlich alles Gegenstand einer Edition werden, worauf sich ein Erkenntnissinteresse einer Forschung gründen lässt. Alles, was also systematisch beschreibbar und inhaltlich codierbar ist, kann zu einem Dokument werden, indem es ediert wird. [44, S. 138]

Für eine Edition muss eine Zielstellung gesetzt werden. Nach Sahle ist aber spätestens bei digitalen Edition wichtig, die Richtlinien an den überlieferten Dokumenten auszurichten. Diese sollen nicht mit einer engen Zielsetzung bearbeitet werden, sondern als quellnahe Wiedergabeform Benutzern die Untersuchung von Forschungsfragen ermöglichen. [44, S. 139]

Eine Herausforderung digitaler Editionen liegt in der notwendigen historischen Distanz zu den Dokumenten, welche für eine erschließende Kritik erforderlich ist. Es muss sowohl die Nähe zum historischen Dokument (Quellennähe) gewahrt werden, als auch der einfache Zugang für den Benutzer hergestellt werden (Nutzernähe). Kritisch wird die Edition durch die Erschließung, das bedeutet durch eine Einordnung des Dokuments in den Kontext. [44, S. 140 f.]

Rohrschneider schreibt in [43], dass in einer historisch-kritischen Edition danach gestrebt wird, sämtliche Materialien zur zu edierenden Quelle zu rekonstruieren und dem Nutzer zur Verfügung zu stellen, mit dem Anspruch, die zugehörigen Quellen abschließend zu edieren.

Caria und Mathiak untersuchten, inwiefern digitale historisch-kritische Editionen einen Mehrwert für die Anwender bieten [4]. Dazu haben sie ein Experiment durchgeführt, in dem eine Fokusgruppe verschiedene Aufgabenstellungen zu lösen hatte. Untersucht haben sie drei digitale historisch-kritische Editionen. Hierbei haben sie herausgefunden, dass die Usability der Edition eine wesentlich größere Rolle spielt als die vorhandenen Funktionalitäten. Denn die Nutzer bevorzugten das effizienteste und effektivste System vor dem System, welches die erwarteten Funktionalitäten am besten bereitstellte. Eine schlechte Usability benachteiligt die Verwendung von digitalen historisch-kritischen Editionen auch bei denjenigen, die mit positiven Erwartungen an sie herangehen. Problematisch war für die Benutzer vor allem die Einstiegshürde („Learnability“).

2.2 INFORMATION RETRIEVAL

Information Retrieval (IR) ist das Fachgebiet der Informatik welches sich unter anderem mit dem Auffinden von Dokumenten beschäftigt. Die Fachgruppe IR der Gesellschaft für Informatik definiert, dass im IR „Informationssysteme in Bezug auf ihre Rolle im Prozess des Wissenstransfers vom menschlichen Wissensproduzenten zum Informations-Nachfragenden betrachtet“[15] werden. In dieser Definition wird bereits deutlich, dass es im IR Prozess mehrere Rollen gibt. Der Prozess beginnt damit, dass Dokumente bereitgestellt werden (hier: transkribierte Kommunikation wie beispielsweise Briefe) und in einem Informationssystem abgelegt werden. Aus diesem Informationssystem sollen die Informationen nun einem Informations-Nachfragenden zur Verfügung gestellt werden. Zum IR wird die Rolle des hier beschriebenen Informations-Nachfragenden als *information seeker* bezeichnet [25, 55], was in dieser Arbeit mit dem deutschen Begriff *Informationssuchender* übersetzt wird.

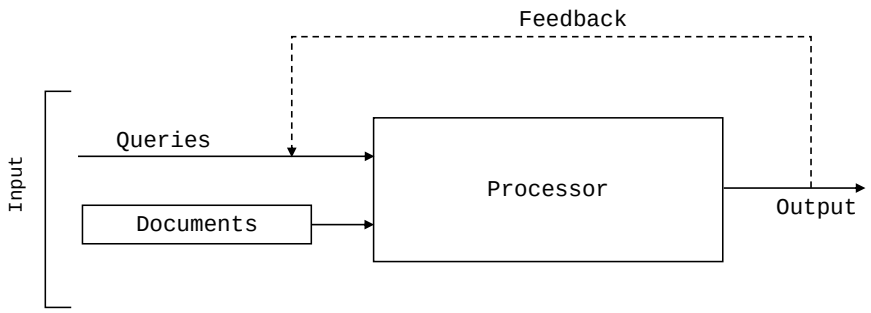


Abbildung 2.1: Eigene Darstellung des IR-System als Blackbox in Anlehnung an [55, Fig. 1.1. S. 4]

Das IR-System als Blackbox wird in Abbildung 2.1 auf der vorherigen Seite dargestellt und besitzt, wie bereits beschrieben, als ersten Input zunächst die Dokumente. Häufig werden die Dokumente als Repräsentation und nicht im Original gespeichert. Ein Dokument könnte beispielsweise durch alle relevanten Schlagworte repräsentiert werden. Der zweite Input sind die Abfragen, welche entweder in natürlicher Sprache oder in einer künstlichen Abfragesprache bestehen. Die künstliche Sprache muss vom Abfragenden zunächst erlernt werden. Der Output besteht nach Van Rijsbergen typischerweise aus Zitaten und Dokumentnummern und kann vom Informationssuchenden verwendet werden, um seine Abfrage zu verfeinern, dies wird als *Feedback* bezeichnet. [55, S. 4-5]

Der Prozess beginnt zwar mit einem Informationsbedarf, dieser ist häufig aber nicht so konkret, dass er mit einem *retrieval task* beantwortet werden kann. Der *retrieval task* besteht in einer für das IR-System verständlichen Abfrage. Neben der Durchsicht der Ergebnisse und dem Verfeinern der Abfrage, um den konkreten Informationsbedarf zu decken, kann es ebenso sein, dass ein Nutzer ein eher generelles Informationsinteresse hat und keine konkrete Anfrage formulieren kann. In diesem Fall lässt der Benutzer sich von den Ergebnissen inspirieren, welche sein Interesse für andere Informationen wecken. Ein solcher, eher zielloser Zugriff auf weitere Informationen wird im IR als *browsing* bezeichnet. Diese Aufgaben sollen in der Abbildung 2.2 veranschaulicht und anhand des nachfolgenden Beispiels erläutert werden. [1, S. 3-4]

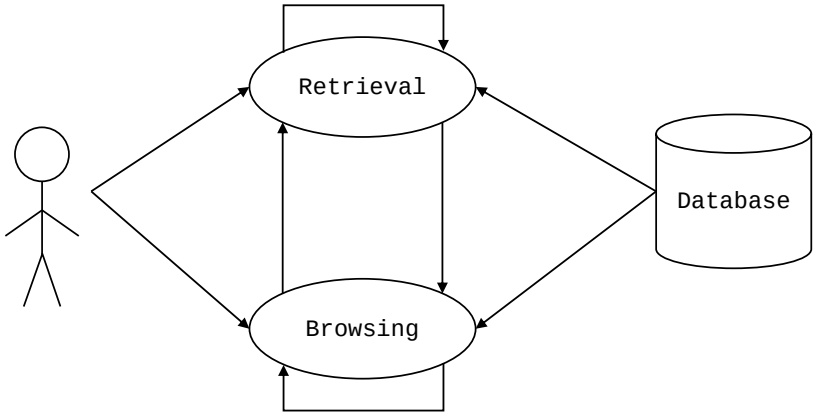


Abbildung 2.2: Eigene Darstellung der Interaktion des Informationssuchenden mit dem IR-System in Anlehnung an [1, Fig. 1.1. S. 4]

Im Kontext der Briefedition Wedekind [12] könnte beispielsweise ein *retrieval task* mit einem Interesse für Wedekinds Reisen beginnen. Bei der Sichtung seiner Korrespondenzen mit dem Begriff *Reise* fällt dem Anwender auf, dass er sich unter anderem in Zürich aufhielt. Nun fragt sich der Nutzer, was Wedekind in Zürich tat und wird somit im Sinne des *browsing* versuchen, weitere Informationen hierüber zu erlangen. Beide Prozesse sind voneinander unabhängig, der Informationssuchende kann aber beliebig zwischen ihnen

wechseln. Sowohl der *retrieval task* als auch das *browsing* haben Einfluss auf die Anforderungen der Wedekind Suche.

Die Architektur eines typischen textbasierten IR-Systems wird in Abbildung 2.3 verdeutlicht. Bei der Verarbeitung der Abfrage werden üblicherweise die nachfolgenden Prozessschritte ausgeführt [5, S. 13-14]:

1. Der Informationssuchende stellt seine Anfrage q_u (typischerweise Schlüsselworte)[1]
2. Auf die Anfrage q_u werden dieselben Operationen angewendet, die auch bei der Indexierung angewandt wurden. Ergebnis ist die normalisierte Anfrage q'_u
3. Die Anfrage wird nun weiter verarbeitet und in eine technische Abfrage q_s umgewandelt
4. Die technische Abfrage q_s wird von der Dokumentendatenbank verarbeitet
5. Die Ergebnisse R werden nach ihrer geschätzten Relevanz sortiert
6. Der Informationssuchende untersucht die genannten Dokumente auf für ihn relevante Informationen

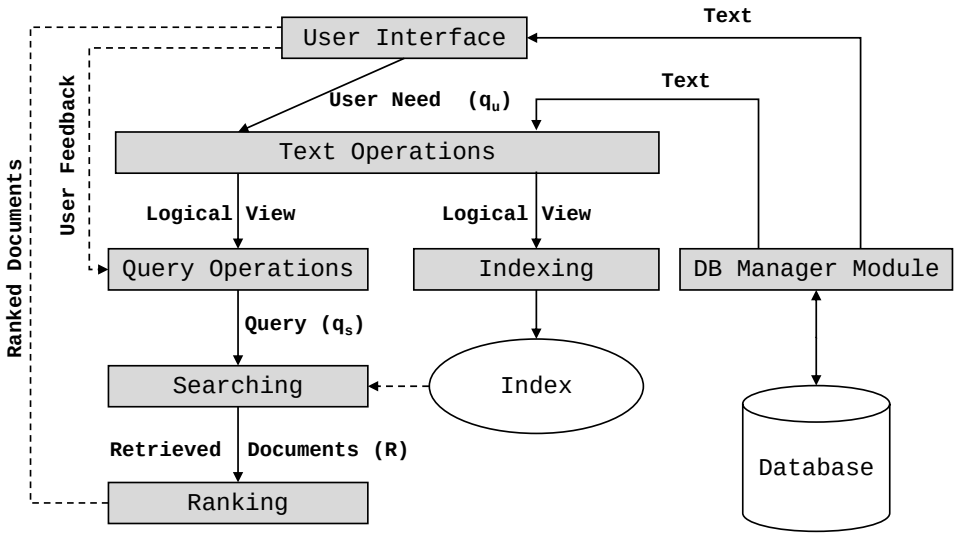


Abbildung 2.3: Eigene Darstellung der Architektur eines Text-IR Systems in Anlehnung an [5, Fig. 2.2. S. 14]

Auf die Methoden zur Aufbereitung, Indexierung und Sortierung nach Relevanz wird im nachfolgenden Kapitel eingegangen.

1 Der User Need ist im Kontext des nutzerzentrierten Designs zwischen dem Benutzer und dem User Interface zu sehen. In der Grafik wird aber bereits eine technische Repräsentation der Abfrage gemeint (siehe Abschnitt 2.4 auf Seite 14)

2.3 VOLLTEXTSUCHE IN DATENBANKEN

In diesem Kapitel wird beschrieben, wie man die Prozessschritte des IR-Systems in Datenbanken abbilden kann. Dabei werden die gängigen Begriffe erläutert und im Speziellen darauf eingegangen, wie diese Funktionalität von PostgreSQL bereitgestellt wird. Die Edition Wedekind setzt auf PostgreSQL als Datenbankmanagementsystem (DBMS) [11]. PostgreSQL ist eine „quelloffenes, objektrelationales DBMS [...] mit einem guten Ruf für Zuverlässigkeit, Stabilität und Performance“ [34]. Die Volltextsuche bezeichnet in PostgreSQL erweiterte Möglichkeiten zur Textsuche mit der Unterstützung linguistischer Methoden, welche für moderne IR-Systeme genutzt werden können [35].

2.3.1 *Rudimentäre Textsuche*

Operatoren für die Textsuche in Datenbanken existieren seit vielen Jahren. In PostgreSQL stehen für die Textsuche die Operatoren *LIKE* und *ILIKE* aus dem SQL-Standard zur Verfügung, sowie ~ und ~* für die Suche mit regulären Ausdrücken[2]. Diese Operatoren haben jedoch Einschränkungen, da sie keine Unterstützung linguistischer Methoden bieten. Beispielsweise werden Wortstämme nicht automatisch gefunden, obwohl diese möglicherweise für den Suchenenden relevant sein könnten. Darüber hinaus gibt es keine Möglichkeit nach Relevanz zu sortieren und durch fehlende Indexunterstützung sind Abfragen nicht effizient. [35]

2.3.2 *Datentypen und Operatoren der Volltextsuche*

Die Volltextsuche in PostgreSQL erlaubt die effiziente Suche in Dokumenten. Ein Dokument ist typischerweise ein Textfeld oder die Kombination mehrerer Felder einer Datenbanktabelle. PostgreSQL besitzt zwei Datentypen, die die Volltextsuche unterstützen. Die Dokumente werden für die Suche in den sogenannten *tsvector* umgewandelt und Abfragen werden als *tsquery* repräsentiert. Beide Datentypen haben gemein, dass sie den Input als eine Menge von Lexemen darstellen. Ein Lexem ist ein Wort, welches normalisiert wurde, um die unterschiedlichen Varianten desselben Wortes zu repräsentieren. [35, 36]

Der *tsvector* ist ein spezieller Datentyp für die Volltextsuche. Er repräsentiert ein Dokument in einer für die Volltextsuche optimierten Form. Er enthält eine sortierte Liste der eindeutigen Lexeme, die im Dokument vorkommen. [36]

In Listing 2.1 wird die einfache Typumwandlung eines Textes in *tsvector* dargestellt. Wie man im Beispiel sieht, werden die Worte alphabetisch sortiert und, erkennbar an „Beispiel“ und „einfaches“, eindeutig repräsentiert.

2 Reguläre Ausdrücke sind Zeichenketten mit bestimmten Steuerzeichen, welche die Beschreibung von und die Suche nach Textmustern erlauben. [9, S. 151]

```
SELECT 'ein einfaches Beispiel mit Worten wie einfaches und Beispiel'::
    tsvector;
                        tsvector
-------------------------------------------------------
 'Beispiel' 'Worten' 'ein' 'einfaches' 'mit' 'und' 'wie'
(1 Zeile)
```

Listing 2.1: Ein Text umgewandelt in *tsvector* Darstellung

Optional kann den Lexemen im *tsvector* auch eine Position zugeordnet werden. Die Positionsinformation repräsentiert im Regelfall die Position(en), an denen das Lexem im Dokument vorkommt. Die Position wird mit einer Nummer zwischen 1 und 16383 repräsentiert (alle späteren Positionen werden auf 16838 gesetzt). Zusätzlich können die Lexeme mit einem Gewicht versehen werden. In der menschenlesbaren Ausgabe wird nach jedem Lexem, separiert durch einen Doppelpunkt, zunächst die Position und dann das Gewicht (*A, B, C oder D*) ausgegeben, siehe Listing 2.2. [36]

```
SELECT 'ein:1B Wort:2B mit:3B Gewicht:4A'::tsvector;
                 tsvector
-----------------------------------------
 'Gewicht':4A 'Wort':2B 'ein':1B 'mit':3B
(1 Zeile)
```

Listing 2.2: Die *tsvector* Darstellung mit Position und Gewichtung

Wie man den Beispielen entnehmen kann, wird der Input durch die Typumwandlung in *tsvector* nicht normalisiert. Für die Umwandlung von Dokumenten in die *tsvector*-Darstellung gibt es die Funktion *to_tsvector* (siehe Abschnitt 2.3.3 auf der nächsten Seite). [36]

Die *tsquery* wird genutzt um eine Suche zu repräsentieren. Die *tsquery* unterstützt boolesche Operationen „&“ (und), „|“ (oder) und „!“ (nicht). Außerdem wird die Phrasensuche unterstützt, also ob ein Lexem auf ein anderes Lexem folgt. Dies wird durch den Operator „<*N*>“ ausgedrückt. „*N*“ bezeichnet dabei die exakte Distanz zwischen den Lexemen, *N=1* kann vereinfacht als „<->“ dargestellt werden. Die Rangfolge, wenn sie nicht durch Klammern definiert wird, berücksichtigt zunächst das „!“, gefolgt von „<->“, „&“ und zuletzt „|“. [36]

Die Zuordnung eines Gewichts zu Lexemen in der *tsquery* ist ebenfalls möglich und hat zur Folge, dass nur Lexeme mit diesen Gewichten berücksichtigt werden. Bei der *tsquery* ist es möglich, einem Lexem mehrere Gewichte zuzuordnen. Außerdem besteht die Möglichkeit des Prefixmatching, bei dem nur die ersten Zeichen eines Lexems übereinstimmen müssen. Dazu wird ein Lexem mit einem $*$ markiert. [36]

Die verschiedenen Operatoren werden in Listing 2.3 angewendet. Die *tsquery* liefert einen Treffer entweder bei Vorhandensein des Lexems „ein“ oder wenn das Dokument das Lexem „Wort“ mit dem Gewicht A und zusätzlich entweder das Lexem „mit“ enthält oder ein Lexem welches mit „Gew“ beginnt.

```
SELECT 'ein | Wort:A & ( mit | Gew:* ) '::tsquery;
                tsquery
----------------------------------------
 'ein' | 'Wort':A & ( 'mit' | 'Gew':* )
(1 Zeile)
```

Listing 2.3: Die Rangfolge bei einer *tsquery* mit mehreren Operatoren

Es gibt mehrere Operatoren zur Interaktion mit den Datentypen der Volltextsuche. Der wohl wichtigste ist der Matching-Operator @@. Die Operation *tsvector* @@ *tsquery* gibt einen booleschen Wert zurück, der wahr ist, wenn die *tsquery* zutrifft. Somit können die Dokumente selektiert werden, die für eine Abfrage relevant sind. [37]

2.3.3 *Funktionen der Volltextsuche*

Neben den Datentypen und Operatoren zur Volltextsuche gibt es noch weitere Funktionen, die für die Umsetzung einer Volltextsuche verwendet werden können. Sie werden genutzt, um Dokumente und Abfragen in *tsvector* und *tsquery* Darstellung umzuwandeln und sie zu manipulieren.

Um Dokumente mit der Volltextsuche suchen zu können, müssen diese in die *tsvector* Darstellung umgewandelt werden. Die Funktion *to_tsvector* nimmt ein Dokument (Text) entgegen und gibt es in *tsvector* Darstellung zurück. Intern werden hierbei mehrere Schritte ausgeführt, welche durch die Textsuchkonfiguration gesteuert werden können. Zunächst wird der übergebene Text mit einem *Parser* in *Token* zerlegt. Der Parser von Postgres erkennt verschiedene Arten von Token und gibt diese entsprechend typisiert zurück. Je nach Art des Tokens und abhängig von der Konfiguration werden unterschiedliche *Dictionaries* angewandt, um die *Lexeme* zu erhalten. Wird ein Token von keinem Dictionary erkannt oder ist es in einer Stopp-Liste, so entsteht kein Lexem. [38]

Um Abfragen in Form von *tsquery* zu erstellen, gibt es mehrere Funktionen. Sie werden nachfolgend erläutert. Am mächtigsten ist hierbei *to_tsquery*, da sie alle bereits im letzten Abschnitt eingeführten booleschen Operatoren und den Phrasenoperator unterstützt. Hierbei muss die Anfrage mit korrekter Syntax übergeben werden, da es ansonsten bei der Umwandlung zu einem Syntaxfehler kommt. [38]

plainto_tsquery wandelt die übergebene Anfrage so in die *tsquery* um, dass alle Wörter vorkommen müssen (in beliebiger Reihenfolge). [38]

Mit *phraseto_tsquery* werden alle genannten Wörter mit dem Phrasenoperator verknüpft. Wenn dabei Stoppworte in der Suchanfrage vorkommen, werden diese im Phrasenoperator berücksichtigt, indem die Stoppworte an dieser Stelle der Phrase entfernt werden. [38]

websearch_to_tsquery ist eine vereinfachte Version von *to_tsvector*, welche mit einer an Websuchen angelehnten Syntax funktioniert. In einer Suchanfrage ohne Anführungszeichen, wird der Text wie in *plainto_tsquery* umgewandelt. Wenn Text in doppelten Anführungszeichen gesetzt ist, wird dieser wie

mit *phraseto_tsquery* als Phrase umgesetzt. Mit *OR* können Teile der Abfrage per „oder“ verknüpft werden. Mit einem Minus kann der Nicht-Operator verwendet werden, um Dokumente auszuschließen, die dieses Wort enthalten. Ein Vorteil von *websearch_to_tsquery* ist, dass keine Syntaxfehler durch Abfragen erzeugt werden können, somit kann der Text ohne eine Bereinigung direkt vom Benutzer übernommen werden. [38]

In Listing 2.4 sind zwei Beispiele dargestellt um die Funktion von *websearch_to_tsquery* zu verdeutlichen. Im ersten Beispiel wird gezeigt, wie die Anfrage „wundervoller Morgen“ in die *tsquery* umgewandelt wird. Man sieht, dass alle Worte der Abfrage im Dokument vorkommen müssen. Das zweite Beispiel soll verdeutlichen, wie die Operatoren angewendet werden können. Es werden Dokumente gesucht, die entweder die Phrase „Feiner Garten“ oder „Rosen“ aber nicht „Tulpen“ enthalten.

```
1  select websearch_to_tsquery('wundervoller Morgen');
2     websearch_to_tsquery
3  ---------------------------
4   'wundervoller' & 'morgen'
5  (1 Zeile)
6
7  select websearch_to_tsquery('"Feiner Garten" OR Rosen -Tulpen');
8              websearch_to_tsquery
9  --------------------------------------------
10  'feiner' <-> 'garten' | 'rosen' & !'tulpen'
11 (1 Zeile)
```

Listing 2.4: Beispiel für *websearch_to_tsquery*

Wie in Abschnitt 2.2 auf Seite 5 aufgezeigt, sollen die Dokumente nach ihrer Relevanz sortiert werden. Dazu gibt es in PostgreSQL die Funktionen *ts_rank* und *ts_rank_cd*. Bei *ts_rank* werden die Dokumente höher bewertet, je mehr Treffer die Abfrage im Dokument hat. Mit der Funktion *ts_rank_cd* („cover density“) wird zusätzlich die Nähe der Suchbegriffe zueinander berücksichtigt. Das Ranking kann außerdem mit den in Abschnitt 2.3.3 auf der vorherigen Seite vorgestellten Gewichten beeinflusst werden. Diese werden mit Standardwerten gewichtet oder es kann jedem Gewicht ein eigener Faktor zugewiesen werden. [38]

Weiterhin ist es möglich, verschiedene Normalisierungen zu konfigurieren. Dazu gibt es die nachfolgenden Möglichkeiten, welche als Bitmaske (Integer) beliebig kombiniert werden können. Wenn mehr als ein Bit gesetzt ist, werden die Funktionen in aufsteigender Reihenfolge angewandt [38]:

0 Ignorieren der Dokumentlänge (Standard)

1 Der Ranking-Wert wird ins Verhältnis gesetzt mit 1 + dem Logarithmus der Dokumentlänge

2 Der Ranking-Wert wird ins Verhältnis gesetzt mit der Dokumentlänge

4 Der Ranking-Wert wird ins Verhältnis gesetzt mit einem harmonischen Mittelwert der Distanz der vorkommenden Treffer (nur für *ts_rank_cd* implementiert)

8 Der Ranking-Wert wird ins Verhältnis gesetzt mit der Anzahl eindeutiger Worte im Dokument

16 Der Ranking-Wert wird ins Verhältnis gesetzt mit 1 + dem Logarithmus der Anzahl eindeutiger Worte im Dokument

32 Der Ranking-Wert wird geteilt durch 1 + den Ranking-Wert

Um dem Informationssuchenden eine Vorschau der Treffer anzuzeigen gibt es die Funktion *ts_headline*. Aus Dokument und Abfrage generiert sie eine Vorschau der Treffer, indem diese im Kontext zurückgegeben werden. Das Verhalten von *ts_headline* kann mit den nachfolgenden Optionen konfiguriert werden [38]:

MAXWORDS, MINWORDS Damit kann die Anzahl der Worte der kürzesten und längsten Zeile gesteuert werden. Standardwerte sind maximal 35 und minimal 15.

SHORTWORD Die Worte dieser Länge werden am Anfang und am Ende einer Zeile verworfen, wenn sie nicht abgefragt wurden. Standardwert ist 3, somit werden kurze Worte wie Artikel verworfen

HIGHLIGHTALL Statt Trefferzeilen im Kontext zu generieren, wird das gesamte Dokument mit den hervorgehobenen Treffern zurückgegeben

MAXFRAGMENTS Die maximale Anzahl an Trefferzeilen wird limitiert. Der Standardwert ist 0 und bedeutet, dass alle Treffer zurückgegeben werden.

STARTSEL, STOPSEL Die Zeichenketten, die zur Hervorhebung genutzt werden, werden dadurch definiert. Hier können HTML Tags verwendet werden, standardmäßig nutzt *ts_headline* „
“ und „“

FRAGMENTDELIMITER Eine Zeichenkette um mehrere Trefferzeilen zu separieren. Dabei ist der Standard „...“

Bei allen in diesem Abschnitt vorgestellten Funktionen kann eine Suchkonfiguration optional mit angegeben werden, ansonsten wird die Standardsuchkonfiguration verwendet. [35] Im nächsten Abschnitt wird erläutert, wie man verschiedene Suchkonfigurationen definieren kann.

2.3.4 *Konfiguration der Volltextsuche*

Alle Funktionen der Volltextsuche aus dem letzten Abschnitt haben einen optionalen *regconfig* Parameter, mit dem die Suchkonfiguration ausgewählt werden kann. Eine Suchkonfiguration besteht dabei aus einem *Parser* und einem Set an *Dictionaries*. [35]

Ein *Parser* zerlegt ein Dokument in Token und weist diesen einen Tokentyp zu. Ein Token ist dabei eine Zeichenfolge. Die möglichen Typen an Tokens sind vom *Parser* abhängig. Der Parser zerlegt das Dokument, er verändert den Text aber nicht. Er identifiziert nur mögliche Wortgrenzen und erzeugt daraus eine Liste von Token. Dabei können auch sich überlappende Tokens erstellt werden. Vom *pg_catalog.default* Parser werden 23 Tokentypen erkannt. Für diese Arbeit ist nur relevant, dass natürlichsprachlichen Wörtern unterschiedliche Typen zugewiesen werden können, was in der Textsuchkonfiguration beim Zuordnen von *Dictionaries* von Bedeutung ist. [39]

Wie im Abschnitt zum IR (Abschnitt 2.2 auf Seite 5) ausgeführt, sollen die Dokumente für die Indexierung und Suche normalisiert werden. Hierzu werden in PostgreSQL die *Dictionaries* verwendet. Ein *Dictionary* wird genutzt um Stoppworte zu entfernen oder Wörter unterschiedlicher Formen in eine zu normalisieren. Diese normalisierte Form eines Wortes bezeichnet man als Lexem. Ein Dictionary ist ein Programm, welches Token als Eingabe erhält und als Rückgabe eine der folgenden Ausgaben besitzt [40]:

ARRAY VON LEXEMEN Ein oder mehrere Lexeme, wenn das Eingabe-Token bekannt ist

EINZELNES LEXEM MIT TSL_FILTER FLAG Ersetzt das Token durch das neue Token und gibt es an das nächste *Dictionary* weiter

LEERES ARRAY Das Eingabe-Token wurde erkannt, aber ist ein Stoppwort

NULL Das Token wurde vom *Dictionary* nicht erkannt

Mit PostgreSQL werden (seit 2007 unverändert) 231 Füllwörter für die deutsche Sprache ausgeliefert, die in der Regel nicht zu einem besseren Ergebnis beitragen, da sie sehr häufig vorkommen oder keinen Informationsgehalt besitzen und daher nicht geeignet sind, um Dokumente voneinander abzugrenzen. [40] Die vollständige Liste der Stoppworte kann in [27] eingesehen werden. Sie umfasst beispielsweise bestimmte Artikel (der, die, das), unbestimmte Artikel (ein, eine), sowie Pronomen (ich, mein, mich, etwas).

Zur Normalisierung von Worten kann eine Reduktion auf den Wortstamm eingesetzt werden. PostgreSQL liefert mehrere *Dictionaries* aus, darunter eines, das auf dem Snowball-Stemmer von Martin Porter basiert [40]. Der Snowball-Stemmer entfernt mittels eines Regelwerks Wortendungen, um Worte auf ihren Wortstamm zu reduzieren. Dabei kann es allerdings einerseits zum „Overstemming“ kommen, das heißt es wird ein zu großer Teil des Wortes abgeschnitten. Andererseits kann auch zu wenig abgeschnitten werden („Understemming“). Es handelt sich bei der Verkürzung nicht immer um eine Reduktion auf linguistisch korrekte Wortstämme („Blüte“ zu „Blut“), daher muss dies bei der Interpretation der Suchergebnisse berücksichtigt werden. [33]

Eine Textsuchkonfiguration wird erstellt indem Parser, Dictionaries und Mappings für Tokentypen definiert werden. Ein Beispiel kann Listing 2.5 auf der nächsten Seite entnommen werden. In der ersten Zeile wird ein *Dictionary* mit dem Snowball-Stemmer für die Sprache Deutsch angelegt. Da-

nach wird eine Textsuchkonfiguration mit der Standardsuchkonfiguration „pg_catalog.german“ als Template angelegt. In der dritten Zeile wird für die natürlichsprachlichen Worte (anhand der Tokentypen) definiert, dass das in Zeile eins definierte *my_dictionary* für diese Tokens verwendet werden soll. Die Verwendung der Textsuchkonfiguration wird in Zeile vier verdeutlicht, wobei den Textsuchfunktionen die Konfiguration als Parameter mitgegeben wird. In der *tsquery* ist ersichtlich, dass das definierte Wörterbuch mit Snowball-Stemmer die Worte auf ihren Wortstamm reduziert hat.

```
1 CREATE TEXT SEARCH DICTIONARY my_dictionary (TEMPLATE = snowball,
      LANGUAGE = german);
2 CREATE TEXT SEARCH CONFIGURATION my_search_config (copy = pg_catalog.
      german);
3 ALTER TEXT SEARCH CONFIGURATION my_search_config ALTER MAPPING FOR
      asciiword, asciihword, hword_asciipart, hword, hword_part, word WITH
       my_dictionary;
4 select websearch_to_tsquery('my_search_config','Feiner Garten mit Tulpen
      ');
5        websearch_to_tsquery
6 ----------------------------------
7  'fein' & 'gart' & 'mit' & 'tulp'
8 (1 Zeile)
```

Listing 2.5: Ein Text umgewandelt in *tsvector* Darstellung

2.4 NUTZERZENTRIERTES DESIGN

Im folgenden Kapitel werden Begriffe und Methoden des nutzerzentrierten Designs vorgestellt sowie der in dieser Arbeit verwendete Double-Diamond Designprozess beschrieben.

2.4.1 *Usability*

Die Suchfunktion soll Benutzern ermöglichen, einfach und effektiv ihre Suchergebnisse zu erhalten und relevante Dokumente zu identifizieren. Da das Problemfeld einer Suchmaschine sehr groß ist und es vielfältige Umsetzungsmöglichkeiten für einzelne Funktionalitäten gibt, soll das Design nutzerzentriert erstellt werden. Das Ziel kann formal auch beschrieben werden als: die Suchfunktion soll eine hohe Usability aufweisen.

Die Usability (Deutsch Gebrauchstauglichkeit) wird in der ISO Norm *Ergonomie der Mensch-System-Interaktion* (ISO 9241) definiert als „Ausmaß, in dem ein System [...] durch bestimmte Benutzer in einem bestimmten Nutzungskontext genutzt werden kann, um bestimmte Ziele effektiv, effizient und zufriedenstellend zu erreichen“ [50, S. 9].

Effektivität ist dabei als die Genauigkeit und Vollständigkeit definiert, mit der ein Ziel erreicht wird. Die Effizienz setzt die eingesetzten Ressourcen zum erreichten Ergebnis ins Verhältnis. Die Zufriedenstellung des Benutzers

beeinflusst sowohl sein Verhalten als auch seine Leistung und wird in der Regel durch die eigene Beurteilung des Benutzers evaluiert. [50, S. 11, 20]

Die Usability hat nicht nur einen Einfluss auf die Effizienz und Effektivität, mit dem der Anwender das Produkt einsetzen kann, sondern beeinflusst auch stark die Akzeptanz durch die Anwender und den Erfolg von Softwareprodukten [48, S. 160]. „Ein wichtiger Aspekt des User Interface Engineering ist daher eine Analyse von Aufgaben und Prozessen mit dem Ziel, valide Anforderungen an die Entwicklung abzuleiten.“ [42, S. 4 ff.] Durch eine sorgfältige Anforderungsanalyse und das regelmäßige Einholen sowie Berücksichtigen von Anwenderfeedback und Verbesserungsvorschlägen soll eine möglichst gute Usability sichergestellt werden.

2.4.2 *Anforderungsanalyse*

Am Anfang der Entwicklung interaktiver Systeme steht die Phase der Anforderungsanalyse. Ziel ist dabei, die Anforderungen an die Entwicklung auf strukturierte Weise zu sammeln und erfassen, zu priorisieren und anschließend zu bewerten. Dabei werden unterschiedliche Methoden genutzt. Bei der Entwicklung interaktiver Systeme kann zusätzlich eine Analyse der Benutzer des Systems relevant sein. [42, S. 45]

Eine Anforderung ist dabei eine „Bedingung oder Fähigkeit, die von einem Benutzer zur Lösung eines Problems oder zur Erreichung eines Ziels benötigt wird“ und damit eine „Fähigkeit, die ein System erfüllen oder besitzen muss“. [42, S. 99]

Mögliche Methoden zur Sammlung von Anforderungen sind beispielsweise die Durchführung von Interviews oder schriftliche Befragungen, bei denen potentiellen Anwendern gezielt Fragen gestellt werden. Dabei werden geschlossene Fragen mit eingeschränkten Antwortmöglichkeiten (z. B. Ja / Nein) genutzt, um die Meinung zu bestimmten Aspekte zu erfassen. Offene Fragen bieten zwar breitere Möglichkeiten zur Antwort, erfordern für den Befragten aber mehr Denk- und Zeitaufwand. [42, S. 63]

Schriftliche Befragungen stellen ein einfaches Mittel dar, um die Bedürfnisse und Prioritäten von Anwendern aufzunehmen und Anforderungen an ein System abzuleiten. [42, S. 66]

Die Aussagekraft einer Umfrage hängt davon ab, ob für diese eine repräsentative Stichprobe gewählt wurde. Dabei ist eine möglichst hohe Rücklaufquote wichtig. Die Herausforderung dabei ist die Motivation der Befragten zur Beantwortung der Umfrage. Dies kann durch verschiedene Aspekte beeinflusst werden, die bei der Erstellung des Fragebogens berücksichtigt werden müssen, wie der Umfang des Fragebogens und die Einfachheit, diesen auszufüllen. Weiterhin ist die Ansprache der Befragten wichtig, da sie wahrscheinlicher antworten, wenn sie die Befragung als wichtig empfinden. [42, S. 67-68]

Weitere Möglichkeiten bestehen in der Verwendung von Kreativitätstechniken. Beispielsweise können in einer Gruppe mehrerer Personen mittels *Brainstorming* wertungsfrei Ideen gesammelt und erst anschließend disku-

tiert und bewertet werden. Das unreflektierte Sammeln von Ideen dient dazu, möglichst viele und zusätzlich neue Ideen und Lösungsmöglichkeiten zu erzeugen. [32, S. 36-37]

2.4.3 *Design-Methoden für User Interfaces*

Im nächsten Schritt der Entwicklung nutzerzentrierter User Interfaces erfolgt der Entwurf von Designs, mit denen die zuvor definierten Anforderungen umgesetzt werden könnten.

Dabei werden die Methoden *Prototyping* und *Sketching* genutzt, um Umsetzungsideen zu repräsentieren und damit möglichst schnell Feedback von Anwendern einzuholen. [42, S. 111]

Prototyping

Prototypen werden bei der Entwicklung von User Interfaces genutzt, um Umsetzungmöglichkeiten in der direkten Verwendung zu testen und verschiedene Varianten vergleichbar zu machen. Der Prototyp muss dabei noch keinen vollständigen Funktionsumfang, sondern kann auch nur einzelne Aspekte umfassen. Bei der Genauigkeit der Prototypen kann man zwischen *Low-Fidelity-* und *High-Fidelity-Prototypen* unterscheiden. Low-Fidelity bezeichnen noch sehr ungenaue Prototypen wie Papierskizzen oder Wireframes, High-Fidelity-Prototypen dagegen solche, die das finale Produkt sehr genau abbilden. [42, S. 122] Je nach Ziel des Prototyps wird eine Ausprägung für folgende Dimensionen festgelegt [42, S. 73-74]:

- Funktionsumfang: Welche und wieviele Funktionen umfasst der Prototyp?
- Funktionstiefe: Wie genau werden die Funktionalitäten umgesetzt?
- Darstellungstreue: Wie ähnlich ist die Benutzeroberfläche dem Ergebnis?
- Interaktivität: Welche Interaktionsmöglichkeiten bestehen für den Anwender?

In Abbildung 2.4 auf der nächsten Seite sind Beispiele mit unterschiedlich genauen Prototypen dargestellt. [42, S. 122]

Sketching

Beim Sketching werden Skizzen zu Funktionalitäten oder dem Verhalten eines Systems erstellt. Sie bieten in einer frühen Phase den Vorteil, dass diese schneller und damit kostengünstiger als High-Fidelity-Prototypen erstellt werden können. Dadurch können zum einen mehr Ideen dargestellt werden, zum anderen bieten diese eine geringere Hürde, wenn nicht erfolgreiche Ideen wieder verworfen werden sollen. [42, S. 116][2, S. 139]

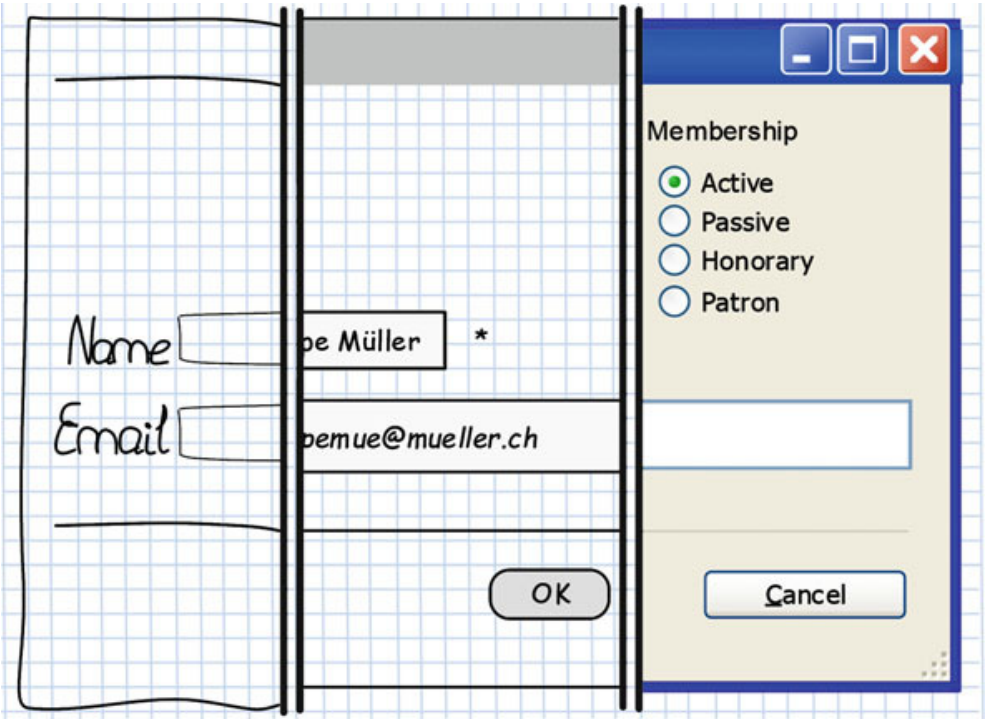


Abbildung 2.4: Verschiedene Ausprägungen von Protoypen [42, S. 122]

Dabei können sogenannte *Wireframes*, abstrahierte Darstellungen grafischer Elemente, zur Darstellung der grundsätzlich zur Verfügung stehenden Oberflächenelemente verwendet werden. Mittels dieser kann die Anordnung, Größe und Platzierung von Elementen diskutiert werden. Das konkrete Aussehen wie Farbe und Schriftart ist zu diesem Zeitpunkt noch nicht relevant. Ein Beispiel für ein solches Wireframe ist in Abbildung 4.19 auf Seite 82 zu sehen. [42, S. 116-119]

2.4.4 *Methoden zur Evaluation von User Interfaces*

Bei der nutzerzentrierten Entwicklung von User Interfaces sind die Rückmeldungen der Anwender ein besonders wichtiges Werkzeug zur Verbesserung. Methoden zur Evaluation können unterschiedlich formell sein und reichen von der Auswertung von Daten wie Logdateien, über die strukturierten Inspektion durch Experten (Cognitive Walkthrough) bis zur Durchführung empirischer Evaluierung mit Endbenutzern. Tests durch Anwender und Befragungen sind dabei gängige Methoden zur Ermittlung der Usability von Webseiten. [54]

Die empirische (erfahrungsbezogene) Evaluierung testet Prototypen mit Anwendern und sammelt deren Rückmeldungen zum Gesamteindruck oder zu einzelnen Aspekten wie dem Verhalten einzelner Funktionen, der verwendeten Terminologie oder der Verständlichkeit von Fehlermeldungen. [42, S. 160]

Dazu können beispielsweise Usability-Tests durchgeführt werden, bei denen die Anwender mit dem Prototyp vordefinierte Aufgaben lösen sollen. Dabei ist es nicht ausschlaggebend, eine möglichst große Zahl von Anwendern in den Test einzubeziehen, sondern im Fokus steht die Problementdeckung, damit diese zur Verbesserung des Produkts genutzt werden können.

Subjektive Kriterien können beispielsweise mithilfe einer schriftlichen oder mündlichen Befragung erfasst werden. Als ergänzende Methoden können weiterhin Beobachtungen von Anwendern oder Diskussionen mit mehreren Benutzern dienen. Bei Gruppendiskussionen besteht der Vorteil, dass in diesen recht schnell erkannt werden kann, ob es sich um vereinzelte oder typische Meinungen handelt.

Inspektionsmethoden

Eine Klasse von Evaluationsmethoden wird unter dem Oberbegriff „Usability-Inspektion“ zusammengefasst. Im Gegensatz zum Usability-Test wird die Benutzerschnittstelle nicht von Anwendern sondern von Experten getestet. Die Evaluation kann die Entwicklung begleiten, dann wird sie als formative Evaluierung bezeichnet. Wird das vollendete Produkt getestet, bezeichnet man dies als summative Evaluierung. [42, S. 154 f.]

Die Usability-Inspektion verfolgt dabei das Ziel, mögliche Probleme der Benutzbarkeit aufzudecken, diese zusammenzufassen, zu klassifizieren und zu priorisieren. Ein Vorteil bei der Arbeit mit Experten liegt darin, dass Experten eher in der Lage sind, auf Basis weniger detaillierten Prototypen und Skizzen die Benutzbarkeit zu diskutieren. [42, S. 155]

Eine Inspektionsmethode ist die heuristische Evaluierung. Nach Preim und Dachselt ist die heuristische Evaluierung unter anderem auf eine Vielzahl von veröffentlichten Fallstudien zurückzuführen, die belegen, dass die Methode ein gutes Aufwand-Nutzen-Verhältnis bietet. Der Fokus liegt dabei auf der Beschreibung des Gesamteindrucks, aber dennoch können auch Detailprobleme entdeckt werden. [42, S. 156 f.]

Mit der heuristischen Evaluierung können vor allem Probleme identifiziert werden, die [42, S. 157]:

- gravierend für die Handhabbarkeit eines Systems sind oder
- die leicht beseitigt werden können.

Experten greifen bei der Evaluierung auf Heuristiken zurück. Die nachfolgenden Fragestellungen von Preim und Dachselt können hier zur Orientierung genutzt werden [42, S. 156]:

- Welche Bedienhandlungen erfordern einen hohen Lernaufwand?
- Welche Aspekte erschweren das Lernen?
- Welche Fehler können leicht entstehen?
- Welche Fehler haben gravierende Folgen?

- Welche Bedienelemente sind ungünstig gestaltet (unübliche Beschriftungen oder grafische Darstellungen, mangelhafte Strukturierung von Dialogen oder Menüs)?
- Welche Systemnachrichten sind unangemessen formuliert oder nicht aussagekräftig genug?

Die Evaluierung durch den Experten sollte sich an allgemeinen Richtlinien und Styleguides orientieren. Kritik und Meinungen sollten dabei stets objektiv begründet werden, sodass sie nicht als subjektive Eindrücke wahrgenommen werden. Typische Gründe sind Kontrastprobleme oder Probleme in der Lenkung der Aufmerksamkeit. Da bei komplexen Systemen nicht alle Kombinationen und Aktionen getestet werden können, kann die Inspektion durch Szenarien fokussiert werden. Die Inspektion wird durch den Experten alleine durchgeführt, die Probleme werden strukturiert und dann anschließend mit Entwicklern, weiteren Experten und der Projektleitung diskutiert. Der Usability-Experte gibt Empfehlungen ab, die finale Entscheidung trifft aber die Projektleitung. [42, S. 157]

(Remote) Usability-Test

Bei einem Usability-Test erhält der Benutzer klar spezifizierte Aufgaben, die er mit dem System zu lösen hat. Sie werden genutzt, um den Gesamteindruck einer Anwendung zu testen. Da hierbei meist die Entdeckung von Problemen und nicht eine hohe statistische Sicherheit entscheidend ist, sind auch geringe Nutzerzahlen ausreichend. [42, S. 161 ff.]

Befinden sich Testperson und Tester nicht am selben Ort, so wird dies als *Remote-Usability-Tests* bezeichnet [42, S. 161 f.]. Unterschieden werden kann der Remote-Usability-Test darin, ob er moderiert oder unmoderiert durchgeführt wird [29].

Im unmoderierten Remote-Usability-Test wird der Test durch die Testperson alleine durchgeführt. Er erhält Aufgaben und es wird aufgezeichnet, wie er die Aufgaben löst. Somit wird erhoben, ob und wie die Testperson die Aufgabe ohne Hilfestellung lösen konnte. Vorteile des unmoderierten Tests liegen darin, dass keine Moderation des Tests notwendig ist und dass sie schnell aufzusetzen sind. Allerdings sind die Daten eines unmoderierten Tests weniger detailliert und der Test sollte normalerweise nicht länger als 20 Minuten dauern. Außerdem fehlt die Möglichkeit, Rückfragen an die Testperson zu stellen. [29, 63]

Im moderierten Remote-Usability-Test befinden sich Testperson und Tester in einer gemeinsamen Konferenz, somit ist die Interaktion des Testers mit der Testperson während des Tests möglich. Durch die Freigabe des Bildschirms der Testperson ist der Tester in der Lage, die Interaktion der Testperson zu beobachten. Weitere Vorteile bestehen darin, dass die Aufgabenstellungen verändert, übersprungen oder die Reihenfolge verändert werden kann. Nachteile des moderierten Tests liegen im höheren Aufwand der Organisation (Rekrutierung und Terminfindung). Außerdem sollte für das Starten der Konferenz zusätzliche Zeit eingeplant werden. [29]

2.4.5 *Design-Prozessmodell*

Der Herausforderung, die relevanten Probleme zu identifizieren und diese dann zu lösen, kann mit einem Design-Prozess begegnet werden.

„Getting the Design Right“

Nach Buxton dient das Prinzip „Getting the Design Right“ dazu, eine einzelne Idee kontinuierlich weiterzuentwickeln bis das bestmögliche Design erreicht ist. Ein Problem bei diesem Vorgehen ist, dass dies häufig die erste Idee ist, welche dann optimiert wird. Die Lösung, die dabei entsteht, kann maximal so gut sein, wie die erste Idee die dem Design Zugrunde liegt. Man erreicht dadurch also ein lokales Maximum, aber möglicherweise nicht die bestmögliche Lösung. Dieses Problem ist auch als „local hill climbing“ bekannt. [17, S. 8][42, S. 113]

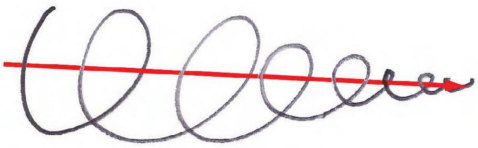


Abbildung 2.5: Prototyping as Iterative Incremental Refinement [3, Figure 149, S. 388]

Die schrittweise Annährung an das Optimum ist in Abbildung 2.5 visualisiert. Der Zielzustand wird durch die rote Linie repräsentiert, an welche sich die Lösung spiralförmig immer weiter annährt. Die Konvergenz besteht darin, diesem Zielzustand iterativ durch Testen und Verfeinern immer näher zu kommen. [3, S. 387]

„Getting the Right Design“

Um dieses Problem zu umgehen, sollte man möglichst viele Ideen generieren, die das Problem lösen könnten. Anschließend wählt man die vielversprechendesten Ideen aus, entwickelt sie parallel weiter und ergänzt sie um neue Ideen, die im weiteren Prozess entstehen. Man verfolgt dabei nur die Ideen weiter, die weiterhin erfolgreich erscheinen. [17, S. 9]

Dies ist in Abbildung 2.6 auf der nächsten Seite visualisiert. Die rote Linie repräsentiert einen Zeitpunkt, an dem mehrere Alternativen untersucht wurden. Am rechten Rand verbleibt ein Ast und an diesem Ast beginnt dann die schrittweise Annährung an das Optimum. [3, S. 387]

Der Double-Diamond Prozess

Eine Methode, mit der die beiden beschriebenen Prinzipien berücksichtigt werden, ist der Double-Diamond Prozess, welcher vom British Design Council entwickelt wurde. Für die Entwicklung untersuchte das British Design Council in einer Studie die Designabteilungen von elf großen Weltmarktführern mit einzelnen Fallstudien. Auch wenn sich die Terminologie und das

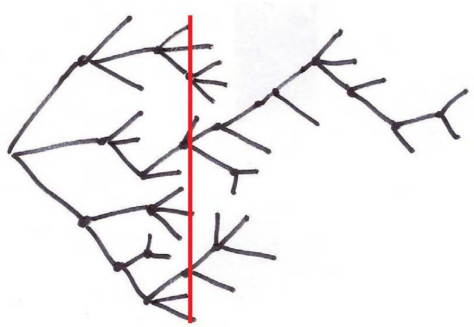


Abbildung 2.6: Design as Branching Exploration and Comparison [3, Figure 150, S. 388]

Management der Designabteilungen in den Unternehmen unterschieden, so hatten die beteiligten Unternehmen doch die Grundprinzipien des Design-Prozesses gemeinsam. [53, S. 6-8]

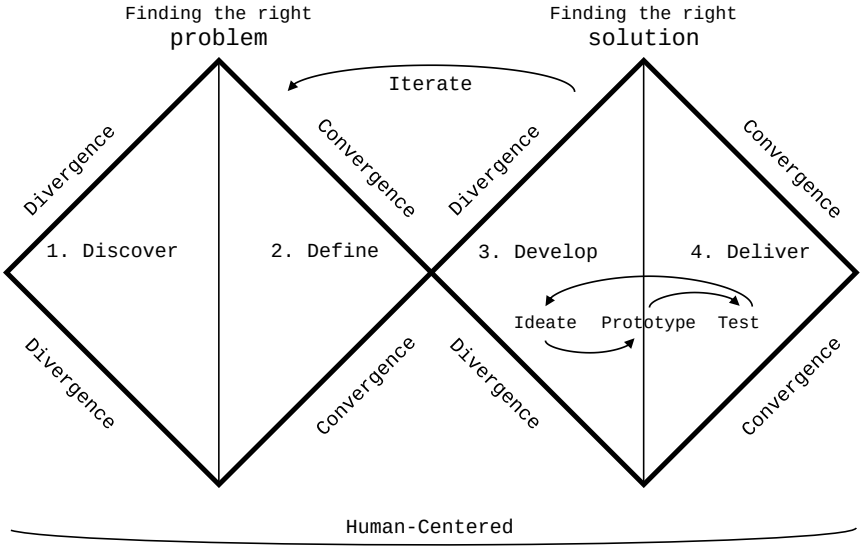


Abbildung 2.7: Eigene Darstellung der Phasen des Double-Diamond Prozesses in Anlehnung an den ursprünglichen Prozess des British Design Councils [52, S. 6] und Liu [24]

Der Double-Diamond Prozess wird in Abbildung 2.7 dargestellt. Er besteht aus den vier Phasen *Discover*, *Define*, *Develop* und *Deliver*, welche die beiden Stadien Divergenz und Konvergenz abbilden [53, S. 6]. Er modelliert dabei die Phasen der Divergenz („Getting the Right Design") und der Konvergenz („Getting the Design Right"). Der Prozess hilft dabei, die Phasen zu balancieren und miteinander zu verknüpfen [52, S. 6].

Die erste Phase des Double-Diamond Prozesses ist die Discover Phase. In dieser verschafft man sich einen Überblick, versucht die Bedürfnisse des Benutzers herauszufinden und es werden erste Ideen generiert. Mögliche

konkrete Methoden dafür sind beispielsweise Marktforschung oder Nutzerrecherche. [52, S. 7]

Ziel der ersten Phase ist es, die Probleme zu erforschen und einen Lösungsraum zu definieren. Dabei soll eine möglichst große Menge an Einblicken gewonnen und eine Ausgangsbasis für Inspiration geschaffen werden. [52, S. 8]

Die nächste Phase ist die Define Phase. In der ersten Phase lag der Fokus darauf, so offen wie möglich zu sein, anschließend geht es dagegen um eine klare Priorisierung [8]. Es werden konkrete Problemstellungen identifiziert, für die im nächsten Diamanten Lösungsmöglichkeiten betrachtet werden sollen [52, S. 8]. Die Problemstellung kann man beispielsweise als „Wie könnte man...“ Frage formulieren, welche man dann in der nächsten Phase zu beantworten versucht [30].

Die Entwicklungsphase des Design Prozesses nimmt die definierte Problemstellung auf und durch Protoyping und Testen werden die Problemstellungen oder Konzepte weiterentwickelt [52, S. 9]. Mit den zuvor bereits vorgestellten Methoden des nutzerzentrierten Designs werden Lösungsansätze entwickelt und untersucht. Mit Low-Fidelity-Prototypen kann zu diesen Ideen Feedback eingeholt werden. Somit kann die beste Problemlösung identifiziert und es können Unklarheiten beim Benutzer aufgedeckt werden. Geeignete Methoden hierfür sind das Guerilla Testing oder Remote User Testing. [8]

Die Auslieferung der Lösung wird in der Deliver Phase betrachtet. Das Produkt wird am Markt getestet und mit Endanwendern evaluiert. [53, S. 6]

Wie Liu in seinem aktualisierten Double Diamond Prozess beschreibt, ist der Prozess iterativ, aber nicht zwingend sequentiell [24]. Jede Lösung ist ein Kompromiss und die perfekte Lösung existiert nicht. Während der Entwicklung und dem Testen werden neue Probleme aufgedeckt und Lücken in den Lösungen werden transparent. Wie in Abbildung 2.7 auf der vorherigen Seite dargestellt, ist eine Anpassung der Problemstellungen ebenfalls iterativ möglich.

3 DISCOVER UND DEFINE

Im Analysekapitel wird der erste Diamant des Double-Diamond Prozess (blau hervorgehoben in Abbildung 3.1) mit dem Ziel bearbeitet, die Probleme zu erheben und definieren, die Suchfunktionen in digitalen (historisch-kritischen) Editionen lösen sollen.

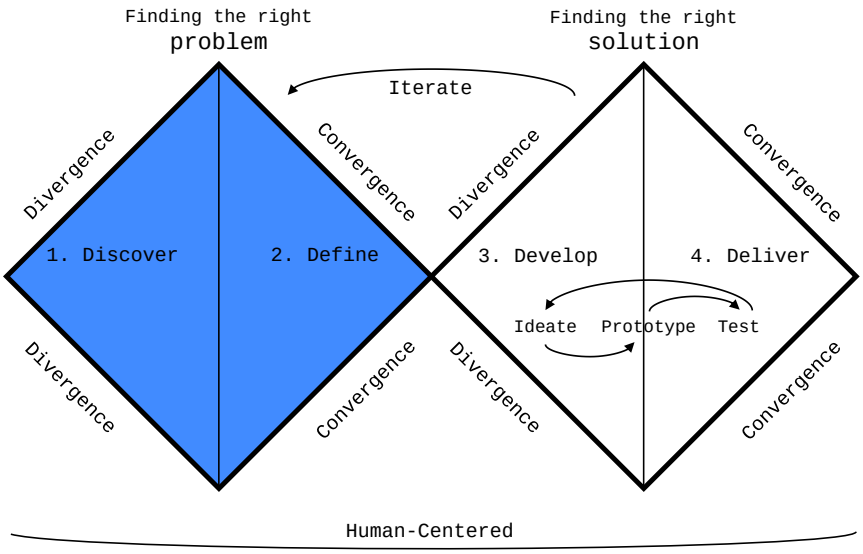


Abbildung 3.1: Hervorhebung des Discover und Define Diamanten (Abbildung in Anlehnung an [52, S. 6, 24])

Wie bereits in Abschnitt 2.4.5 auf Seite 20 erläutert ist eine Methode der Discover Phase die Marktforschung. Dazu wird eine Konkurrenzanalyse durchgeführt, die einen Überblick über den aktuellen Stand der Suchfunktionalität in anderen digitalen (historisch-kritischen) Editionen erhebt. Hierzu wird auf einen existierenden Kriterienkatalog von Patrick Sahle aufgesetzt, welcher um spezifische auf Suche bezogene Attribute erweitert wird. [46]

Neben der Marktforschung ist eine weitere Methode der Discover Phase die Nutzerrecherche. Mit dem Überblick aus der Konkurrenzanalyse, kann eine Nutzerrecherche durchgeführt werden. Die Nutzerrecherche wird in dieser Arbeit in Form einer Onlineumfrage durchgeführt und soll die tatsächlichen Anforderungen und Erwartungen der Benutzer an Suchfunktionen in digitalen (historisch-kritischen) Editionen erheben.

Die so erhobenen Ergebnisse werden zusammengefasst und kategorisiert. Das Ziel des Kapitels wird dann mit der Define Phase erreicht, indem aus den Erkenntnissen konkrete Problemstellungen identifiziert werden, die für Suchfunktionen in digitalen (historisch-kritischen) Editionen relevant sind.

Diese Problemstellungen stellen den Ausgangspunkt für den zweiten Diamanten des Prozesses [52, S. 7.].

3.1 DESIGN DER KONKURRENZANALYSE

In diesem Abschnitt wird der Aufbau der Konkurrenzanalyse erläutert. Zunächst wird dargestellt, wie die Konkurrenzanalyse durchgeführt wird. Die Ergebnisse werden anschließend in Abschnitt 3.2 auf Seite 26 ausgewertet.

Basis für die Konkurrenzanalyse ist der Kriterienkatalog für die Besprechung digitaler Editionen von Patrick Sahle, veröffentlicht vom Institut für Dokumentologie und Editorik. In seinen Ausführungen geht Sahle darauf ein, dass es allgemeine und spezielle Kriterien gibt, die es bei der Besprechung von digitalen Editionen zu analysieren gilt. In der Konkurrenzanalyse dieser Masterarbeit wurde der Fokus auf die funktionalen Kriterien gelegt. Diese sind im Abschnitt Umsetzung und Präsentation von Sahle näher ausgeführt.

Bei der Umsetzung und Präsentation einer digitalen (historisch-kritischen) Edition kann analysiert werden, welche technische Architektur die Edition nutzt. Weiterhin können Fragestellungen hinsichtlich der Oberfläche untersucht werden, z. B.: Wie kann auf die Dokumente zugegriffen werden? Wie ist die Suchfunktion ausgeprägt (wenn eine vorhanden ist)? Hier kann untersucht werden, ob es eine allgemeine oder eine spezielle Suche gibt. Werden Indizes, Register oder Autovervollständigung angeboten? Ein weiteres Kriterium ist, ob es technische Schnittstellen für die Extraktion der Dokumente in andere Werkzeuge gibt. Sahle nennt hier auch die Möglichkeit weitere Features von Editionen, die beispielsweise besonders nützlich oder ungewöhnlich sind, zu analysieren. [46]

Die Felder, die in der Konkurrenzanalyse erhoben werden, können grob in Metadaten und Daten aufgeteilt werden. Die erfassten Metadaten werden in Tabelle 3.1 beschrieben.

Feld	Beschreibung	Datentyp
Editionsgruppe	Die Editionsgruppe beschreibt, durch welches Selektionskriterium diese Edition für die Analyse ausgewählt wurde	Schlüssel
Eindeutige ID	Als eindeutige ID wird die Franzini-Handle-PID genutzt, die im Datensatz von Franzini [13] vergeben wird. Beispiel: *21.11115/0000-000B-D094-2*[1]	Schlüssel
Edition name (Franzini)	Die Bezeichnung der Edition	Text
URL (Franzini)	Die URL zur Startseite der Edition	URL
Ausschluss Grund	Der Grund für den Ausschluss aus der Analyse, sollte eine Edition nicht weiter analysiert werden können	Text

Tabelle 3.1: Erläuterung der Metadaten über die Editionen der Konkurrenzanalyse

Die untersuchten Funktionalitäten der Editionen sind in Tabelle 3.2 auf Seite 26 dargestellt. Sie sind nach Kategorien zusammengefasst.

1 Ein Aufruf des Handles (`http://hdl.handle.net/21.11115/0000-000B-D094-2`) führt in den Katalog von Franzini auf die jeweilige Detailseite zu dieser Edition

Feld	Beschreibung	Datentyp
	Startseite	
Verlinkungen auf Startseite	Untersucht die Einstiegsmöglichkeit, ob direkt von der Startseite auf Werke oder Register zugegriffen werden kann	Wahrheitswert
Suchfeld direkt auf Startseite	Untersucht die Einstiegsmöglichkeit direkt per Suche	Wahrheitswert
Einstiegspunkte	Nennung der Möglichkeiten auf Dokumente zuzugreifen	Liste
	Suchfunktionalitäten	
Volltextsuche	Untersucht ob die vorhandene Suche eine Volltextsuche ist	Wahrheitswert
Erweiterte Suchoptionen	Untersucht ob eine Erweiterte Suche mit Filtern zur Verfügung steht	Wahrheitswert
Filtermöglichkeiten	Eine Liste aller Filtermöglichkeiten in der Suchfunktion	Liste
Fehlertolerante Suche	Untersucht ob die Suche fehlertolerant ist. Suche per Wortstamm oder Ergebnisse auch bei Schreibfehler	Wahrheitswert
Art der Suche	Die Suche beschreibende Features, wie Volltextsuche, Erweiterte Suche mit Filtern, Registersuche, Keine, Stichwortsucheo der CQP (Corpus Query Protocol)	Liste
Suchbegriffe	Welche Suchbegriffe können eingegeben werden? Möglichkeiten sind hier Volltext, aber auch Wortstamm oder Registerbegriffe	Liste
Wortstammsuche	Untersucht ob eine Suche über den Wortstamm eines Suchbegriffs durchgeführt wird	Wahrheitswert
Autocomplete	Besitzt die Suche eine Autovervollständigung in der Suchleiste	Wahrheitswert
Besondere Suchoptionen	Hier können Besonderheiten der Suchfunktion genannt werden	Text
	Register	
Anzahl der Register	Wie viele Register bietet die digitale (historisch-kritische) Edition	Ganzzahl
Register	Eine Liste aller Register der digitalen (historisch-kritischen) Edition	Liste
	Ergebnisdarstellung	
Anzeige der Suchergebnisse	Beschreibung der Anzeige der Suchergebnisse in Prosa	Text
Inhalt Vorschau	Wird eine Vorschau des Inhalts in den Suchergebnissen dargestellt	Wahrheitswert
Treffer hervorgehoben	Wird der Treffer / Werden die Treffer des Suchergebnisses dargestellt	Wahrheitswert
Treffer verlinkt	Wird beim Öffnen des Dokuments die Stelle des Treffers im geöffneten Dokument verlinkt (direkter Sprung an diese Stelle oder Hervorhebung?)	Wahrheitswert
Relevanz Dargestellt	Ist ein Score dargestellt, wie relevant das Suchergebnis ist	Wahrheitswert
Miniaturansicht	Wird eine Miniaturansicht des Faksimiles im Suchergebnis dargestellt	Wahrheitswert
Metadaten	Werden Metadaten zu den Suchergebnissen dargestellt	Wahrheitswert
Welche Metadaten	Eine Liste der Metadaten die dargestellt werden	Liste
Anzahl Treffer dargestellt	Wird dargestellt wie viele Suchergebnisse gefunden wurden	Wahrheitswert
Anzahl angezeigte Treffer	Wie viele Treffer werden auf einmal angezeigt	Zahl
Suchergebnisanzahl konfigurierbar	Ist konfigurierbar wie viele Treffer auf einmal angezeigt werden sollen	Wahrheitswert + Zahl
Treffer sortierbar	Sind die Suchergebnisse sortierbar	Wahrheitswert
Sortieren nach	Liste wonach die Suchergebnisse sortierbar sind	Liste
	Sonstiges	
Parallelansicht auf Dokumente	Untersucht ob bei der Darstellung der Ergebnisse eine parallelansicht von Faksimile und Transkription möglich ist	Wahrheitswert
Trefferliste exportieren	Ist die Suchergebnissliste exportierbar	Wahrheitswert
Formate für Suchergebnisse	In welchem Format können die Suchergebnisse exportiert werden	Liste

Feld	Beschreibung	Datentyp
Technologie	Welche Technologie setzt die digitale (historisch-kritische) Edition ein	Liste

Tabelle 3.2: Erläuterung der Felder in der Konkurrenzanalyse

Nach der Erläuterung der zu untersuchenden Features soll nun auf die Selektion der Editionen für die Analyse eingegangen werden. Hierzu wird die Herkunft der Datengrundlage vorgestellt und anschließend erläutert, wie die Datengrundlage ausgewählt wurde.

In *A catalogue of digital editions* [14, S. 161 ff.] beschreiben Franzini, Mahony und Terras die Zusammenstellung eines Katalogs von digitalen Editionen. Der Katalog verfolgt das Ziel, eine zugängliche, aktuelle Sammlung von Editionen zur Verfügung zu stellen, die in aufbauender Forschung und Weiterentwicklung eingesetzt werden kann. Der Katalog wird als CSV [13] bereitgestellt. Zu jeder Edition werden URLs, Editionsname, Sprachen, Dokumentarten, Förderung und ontologische, technologische und philologische Aspekte erfasst. Die Menge der Editionen im Katalog wurde als Grundlage der Konkurrenzanalyse genutzt.

Um die Menge für die weitere Analyse einzuschränken, wurden zwei Selektionen durchgeführt. Diese Masterarbeit wird im Rahmen der Weiterentwicklung einer durch die DFG geförderten Edition durchgeführt. Daher wurde die erste Selektion so gewählt, dass alle Editionen mit einer genannten DFG-Förderung weiter untersucht werden. Die zweite bezieht sich auf die Dokumentart, da die Suchfunktion in dieser Masterarbeit für eine Briefedition implementiert werden soll. Daher wurde so selektiert, dass alle Editionen analysiert werden, die als unterstütztes Dokument auch Briefe anbieten.

3.2 AUSWERTUNG DER KONKURRENZANALYSE

Von den 42 selektierten Editionen wurden 15 Editionen nicht in der Auswertung betrachtet. Die Gründe dafür sollen zunächst dargelegt werden. Zwei Editionen wurden doppelt selektiert und wurden daher nur einfach berücksichtigt. Vier weitere Editionen wurden ausgeschlossen, da sie keine Suchfunktion implementieren. Zwei Editionen hatten jeweils identische Suchschnittstellen und wurden daher nicht weiter untersucht. Zwei Editionen in der Brief-Selektion enthielten keine Briefe und wurden daher auch nicht weiter untersucht. Vier Editionen waren nicht erreichbar oder es kam zu Fehlern bei der Suche, eine weitere war durch ein Login geschützt und konnte nicht verwendet werden. Insgesamt wurden somit 9 DFG-geförderte Editionen und 18 Briefeditionen ausgewertet.

Für die Auswertung der Wahrheitswerte wird für jede Edition ermittelt, ob ein Feature zur Verfügung steht (*WAHR* oder (*FALSCH*) und anschließend die Ausprägungen gezählt. Daneben wurden Listen mit Details zu einigen Features (beispielsweise zu den Entitäten mit Filtermöglichkeit) erhoben, die mehrere Werte pro Edition erfassen. Im nächsten Schritt wurden die Werte harmonisiert, da die ausgewerteten Editionen teilweise verschiedene Spra-

chen und unterschiedliche Bezeichnungen für die semantisch selben Werte haben. Daher wurde jedem Wert ein harmonisierter Wert zugeordnet. Beispielsweise wurden das englische *author* auf *Autor* zugeordnet. Die genaue Zuordnung bei der Harmonisierung kann in dem Exceldokument („konkurrenzanalyse.xlsx" in Anhang E auf Seite 144) eingesehen werden. Danach wurde gezählt, wie oft die Einzelwerte in den Editionen vorkommen. In dieser Ausarbeitung werden alle Werte dargestellt, die in mindestens zwei Editionen vorkamen. Die Auswertung der Features mit Wahrheitswert werden in der Tabelle 3.3 dargestellt und werden nachfolgend beschrieben. Gezählt wurde jeweils, wieviele der Editionen eine gewisse Funktionalität zur Verfügung (*WAHR*) oder nicht zur Verfügung stellen (*FALSCH*). Die Liste wurde absteigend nach der Spalte *WAHR* sortiert. Die Anzahl der Editionen bei denen keine klare Entscheidung möglich war, sind in der Spalte *n/a* dargestellt.

Feature	**WAHR**	**FALSCH**	**n/a**
Volltextsuche	26	1	0
Darstellung Trefferanzahl	26	1	0
Inhaltsvorschau	24	2	1
Trefferhervorhebung	22	5	0
Register	21	6	0
Erweiterte Suchoptionen	20	7	0
Metadaten	19	8	0
Suchfeld direkt auf Startseite	16	11	0
Verlinkungen auf Startseite	15	12	0
Treffer verlinkt	13	13	1
Parallelansicht auf Dokumente / Briefe	11	15	1
Treffer sortierbar	9	17	1
Fehlertolerante Suche	6	20	1
Wortstammsuche	6	21	0
Miniaturansicht	6	21	0
Suchergebnisanzahl konfigurierbar	6	21	0
Autocompletion	5	22	0
Suchergebnisse exportierbar	4	23	0
Darstellung der Relevanz	1	26	0

Tabelle 3.3: Quantitative Auswertung der (Such-)funktionalitäten in untersuchten Editionen

Zunächst fällt auf, dass bis auf eine, alle Editionen eine Volltextsuche besitzen, da Editionen ohne Suche nicht weiter untersucht und ausgewertet wurden. Der Ausreißer in der Analyse ist darauf zurückzuführen, dass in dieser Edition zwar eine Suche vorhanden, die Volltextsuche selbst aber nicht funktional war. Darüber hinaus lässt sich feststellen, dass bis auf eine Ausnahme stets die Anzahl der gefundenen Treffer ausgegeben wird. Eine Vorschau auf den Inhalt des Treffers wird bei den meisten Editionen ebenfalls angeboten,

eine Hervorhebung des Suchtreffers kommt dabei zwar sehr häufig, aber nicht immer vor.
Über drei Viertel der ausgewerteten Editionen besitzen ein oder mehrere Register. Bei mehr als der Hälfte bestehen erweiterte Suchoptionen oder Filter, eine Darstellung von Metadaten und ein Suchfeld direkt auf der Startseite der Edition. Einen Absprung vom Suchergebnis auf die Stelle des Treffers im Dokument bietet die Hälfte der Editionen. Bei einem Drittel der Editionen sind die Suchergebnisse außerdem sortierbar. Eine fehlertolerante Suche und Wortstammsuche sind bei etwa einem Fünftel der ausgewerteten Editionen noch vertreten. Eine Autovervollständigung im Suchfeld bieten nur fünf der 27 Editionen. Der Export der Suchergebnisse und die Darstellung der Relevanz kommen ebenso nur selten vor.

Das Aufrufen von Dokumenten über Register ist in den untersuchten Editionen sehr häufig möglich. Welche Register angeboten werden, unterscheidet sich aber stark zwischen den Editionen. In Tabelle 3.4 wird die Häufigkeit der verschiedenen Register dargestellt. Am häufigsten sind dabei die Register zu Personen, geographischen Merkmale und Werken.

Register	**Anzahl**
Personen	15
Geographisch	12
Werke	10
Zeit	6
Biblographie	4
Themen	3
Absender	3
Briefe	3
Ereignis	3
Organisation	2
Glossar	2
Tagebücher	2
Empfänger	2
Dokumente	2
Institutionen	2

Tabelle 3.4: Quantitative Auswertung des Vorkommens von Registern in den untersuchten Editionen

Zur weiteren Einschränkung der Suchergebnisse können Filter genutzt werden, welche in Tabelle 3.5 auf der nächsten Seite ausgewertet sind. Typische Filtermöglichkeiten sind hierbei über den Autor, die Art des Dokuments und Korrespondenten. Das häufige Vorkommen von briefbezogenen Attributen liegt in der Auswahl der untersuchten Editionen begründet. Der Unterschied zwischen Datum und Zeitraum besteht darin, dass bei einem Datumsfilter das exakte Datum selektiert und bei einem Zeitraum ein Datumsbereich ausgewählt werden kann.

Filtermöglichkeit	Anzahl
Autor	9
Dokumententyp	8
Korrespondent	7
Datum	5
Empfänger	5
Absendeort	5
Empfangsort	5
Ort	4
Zeitraum	4
Sprache	3
Titel	2
Briefart	2
Jahr	2
Biblographie	2

Tabelle 3.5: Quantitative Auswertung der Filtermöglichkeiten in den untersuchten Editionen

Untersucht wurde außerdem die Darstellung der Suchergebnisse. Hierbei wurde erhoben, welche Metadaten zu einem Ergebnis angezeigt werden. Die Ergebnisse werden in Tabelle 3.6 dargestellt. Am häufigsten waren hierbei die Darstellung von Datum, Empfänger, Autor und Ort.

Metadaten	Anzahl
Datum	12
Empfänger	11
Autor	8
Ort	7
Absender	6
Titel	5
Briefnummer	5
Zusammenfassung	3
Dokumententyp	3
Klassifikation	2
ID	2
Brieftitel	2

Tabelle 3.6: Quantitative Auswertung des Vorkommens von Darstellung der Metadaten in den untersuchten Editionen

In den untersuchten Editionen ist die Möglichkeit zur Sortierung der Suchergebnisse nicht so weit verbreitet. Für die Editionen, die dies unterstützen, konnte nach den in Tabelle 3.7 auf der nächsten Seite dargestellten Attributen sortiert werden. Am häufigsten sind dies Datum und Relevanz.

Sortieren nach	Anzahl
Datum	6
Relevanz	4
Titel	3
Empfänger	2
Autor	2
Treffer Anzahl	2

Tabelle 3.7: Quantitative Auswertung der Sortieroptionen in den untersuchten Editionen

In diesem Abschnitt wurde die mittels der durchgeführten Konkurrenzanalyse ermittelte Auswertung beschrieben und dadurch dargestellt, welche Verbreitung die Funktionalitäten in den untersuchten Editionen haben. Insbesondere die erhobenen Listen mit konkreten Ausprägungen der Register, Filter, Metadaten und Sortierfunktion unterstützen beim Design der Umfrage, welches im nächsten Abschnitt behandelt wird.

3.3 AUFBAU DER UMFRAGE

Zusätzlich zur Konkurrenzanalyse wird eine Nutzerrecherche durchgeführt mit dem Ziel die Erwartungen der Nutzer an die Suchfunktion einer digitalen (historisch-kritischen) Edition zu ergründen. Die Nutzerrecherche wird in Form einer Onlineumfrage durchgeführt, deren Aufbau in diesem Kapitel beschrieben wird. Der Fragebogen befindet sich in Anhang A auf Seite 108 und wurde sowohl in deutscher als auch in englischer Sprache bereitgestellt. Für die Durchführung der Onlineumfrage wurde das OpenSource Online-Umfragetool *LimeSurvey* [23] verwendet.

Der Fragebogen ist gegliedert in sechs Abschnitte:

1. Allgemeine Angaben
2. Bezug zu digitaler (historisch-kritischen) Edition
3. Befragung zur Suchfunktion der eigenen digitalen (historisch-kritischen) Edition
4. Fragen zu den Erwartungen an die Suchfunktion einer digitalen (historisch-kritischen) Edition
5. Befragung zur Forschung mit digitalen (historisch-kritischen) Editionen
6. Abschluss mit offenen Fragen zur Erhebung von Anmerkungen und Kommentaren

Die allgemeinen Angaben dienen zur Erhebung von Informationen über die Befragten wie Alter und Geschlecht.

Die zweite Fragengruppe dient der Bestimmung, welchen Bezug der Befragte zu digitalen (historisch-kritischen) Editionen hat. Hier wird beispielsweise befragt, in welcher Rolle man mit der Edition arbeitet und welche Dokumenttypen (Manuskripte, Briefe, etc.) in der Edition vorkommen.

Die dritte Fragengruppe „Befragung zur Suchfunktion der eigenen digitalen (historisch-kritischen) Edition“ wird dem Befragten, basierend auf seinen Antworten, nur angezeigt, wenn er in Fragengruppe zwei angegeben hat, dass er mit Editionen arbeitet. Nur in diesem Fall kann erhoben werden, welche Suchfunktionalitäten die Edition des Befragten bereitstellt. Hierzu wurden geschlossene Fragen mit Mehrfachnennung genutzt, sowie die Möglichkeit, weitere zu nennen. Die Antwortmöglichkeiten wurden auf Basis der Konkurrenzanalyse festgelegt.

Die vierte Fragengruppe dient dazu, die Erwartungen und Bedürfnisse des Benutzers in Erfahrung zu bringen. Hier wird mithilfe einer *Likert-Skala* gemessen, welche Funktionalitäten von einer Suchfunktion erwartet werden. Die Likert-Skala wird genutzt, um Befragten mehrstufige Antwortmöglichkeiten zur Zustimmung oder Ablehnung anzubieten. [21, S. 397] Da die Umfrage in deutscher und englischer Sprache angeboten wird und die Ergebnisse vergleichbar sein sollen, wird die Skala in Tabelle 3.8 mit der jeweiligen Übersetzung genutzt.

Englisch	**Deutsch**
Strongly Agree	Stimme voll und ganz zu
Agree	Stimme zu
Neither Agree Nor Disagree	Weder noch
Disagree	Stimme nicht zu
Strongly Disagree	Stimme überhaupt nicht zu

Tabelle 3.8: Ausprägungen der verwendeten 5-stufigen Likert-Skala [56, Table 5.1. S. 125]

Die vorletzte Fragengruppe zielt darauf ab, zu erfragen, wozu die Suchfunktion in der Forschung eingesetzt wird. Hierbei wird mit offenen Fragen wie z. B. „Beschreiben Sie eine Forschungsfrage die Sie mit einer digitalen (historisch-kritischen) Edition untersucht haben.“ und mit darauf bezugnehmenden Fragen gearbeitet. Dies soll helfen, das Nutzerbedürfnis noch besser einschätzen zu können.

Die letzte Fragengruppe enthält offene Fragen, um Ergänzungen und frei definierbares Feedback zum Fragebogen zu erhalten.

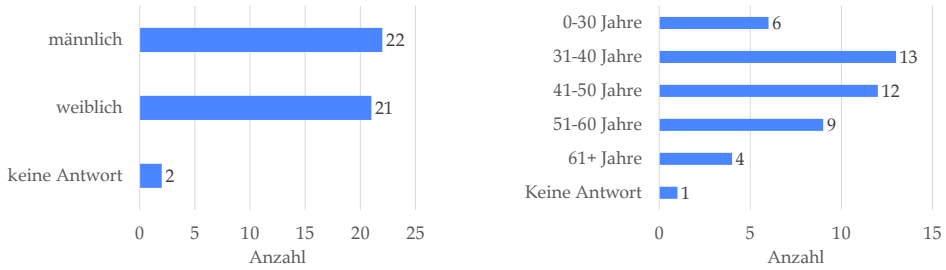
Die Umfrage wurde auf Social Media und über direkte Anschreiben beworben. Auf Social Media wurde in Facebookgruppen von Digital Humanities, auf Twitter mit den Hashtags *#opendata* und *#digitaledition* geworben. Zusätzlich wurden Projektbeteiligte an den digitalen (historisch-kritischen) Editionen aus Abschnitt 3.1 auf Seite 24 direkt angeschrieben.

Für die Bewerbung der Umfrage per E-Mail wurde die Annahme getroffen, dass sich eine höhere Beteiligung an der Umfrage erreichen lässt, wenn die Personen persönlich angeschrieben werden. Aus diesem Grund wurde

das Anschreibung per Serienbrieffunktion personalisiert. Das verwendete Anschreiben kann Anhang A.1 auf Seite 108 entnommen werden. Neben der persönlichen Ansprache wurde zusätzlich noch der Grund erläutert, warum eine Person angeschrieben wurde. Hierbei wurde der Name der Edition genutzt. Das Anschreiben an Projektbeteiligte der Bernoulli-Edition enthielt dabei beispielsweise den nachfolgenden Satz: „Ich schreibe Sie an, wegen Ihrer Beteiligung an der Briefedition des Mathematikers Bernoulli.“ Somit sollte es für die angeschriebene Person nachvollziehbarer werden, warum genau sie an der Umfrage teilnehmen soll.

3.4 AUSWERTUNG DER UMFRAGE

Die Umfrage wurde insgesamt 69 mal ausgeführt. 45 Datensätze konnten ausgewertet werden, 34 davon waren vollständig ausgefüllt, die restlichen elf nur teilweise. In Abbildung 3.2 sind Geschlecht und Altersstruktur der Befragten dargestellt. Hinsichtlich des Geschlechts ist die Gruppe der Befragten ausgeglichen.

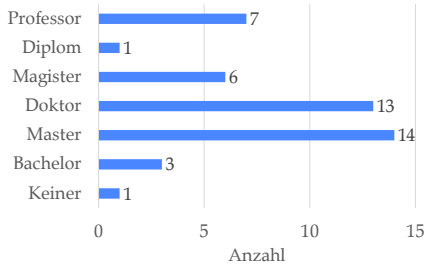


(a) Geschlecht (b) Alter

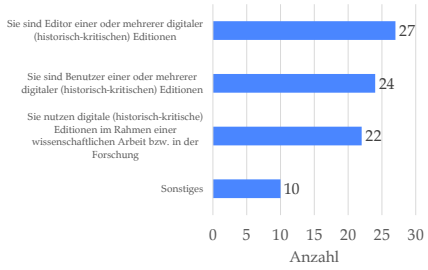
Abbildung 3.2: Demographie der Umfrage

Der Bezug der Befragten zu digitalen (historisch-kritischen) Editionen wird in Abbildung 3.3 auf der nächsten Seite dargestellt. Die meisten der Befragten haben einen akademischen Abschluss, darunter überwiegend Masterabschlüsse oder höhere. Bei der Nennung der eigenen Rolle im Umgang mit Editionen waren Mehrfachnennungen möglich. Die meisten sind Editoren von Editionen (n=27), weniger kennzeichneten sich als Benutzer (n=24) und manche nutzen Editionen im Rahmen ihrer wissenschaftlichen Arbeit (n=21). Den Freitext für sonstige Beteiligung nutzten einige für Ergänzungen, andere exklusiv. Hier gab es Beteiligungen als Entwickler (n=5), Koordinator (n=1), Ersteller und Anbieter (n=1), Autor (n=1), Masterarbeit (n=1) und Mitarbeiter (n=1). Es gab Antworten aus mehreren Forschungsfeldern, die meisten aus der Geschichte (n=13) und der Literatur (n=8). Fünf der Sechs *Anderes*-Angaben des Forschungsfeldes wurden mit Digital Humanities beantwortet, somit sollte dies in zukünftigen Umfragen als Option mit

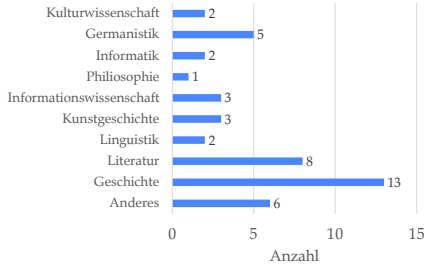
angeboten werden. Die meisten der Befragten haben mehrjährige Erfahrung in der Arbeit mit digitalen (historisch-kritischen) Editionen.



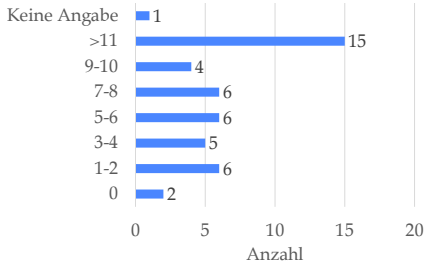
(a) Höchster akademischer Grad



(b) Rolle im Umgang mit Editionen



(c) Forschungsfeld



(d) Erfahrung im Umgang mit Editionen in Jahren

Abbildung 3.3: Hintergrund und Erfahrung der Befragten

Alle Befragten, die angegeben haben mit digitalen (historisch-kritischen) Editionen zu arbeiten, wurden nach ihren Editionen befragt. Die Antworten sind in den Diagrammen in Abbildung 3.4 auf Seite 45 dargestellt. In der Umfrage stark repräsentiert sind Editionen, die sich mit Briefen (n=36), Manuskripten (n=33) und Tagebüchern (n=25) beschäftigen. Dies liegt vermutlich in der Bewerbung der Umfrage begründet und war beabsichtigt, um einen Fokus auf Brief-Editionen zu bekommen. Bei den Einstiegsmöglichkeiten in die Editionen besitzen die meisten eine Suchfunktion (n=27). Mit absteigender Anzahl wurden als Einstiegsmöglichkeiten Register (n=23), Verlinkung (n=20) und Schlagworte (n=11) genannt.
Besonders interessant sind die Nennungen sonstiger Einstiegsmöglichkeiten. Diese werden in der nachfolgenden Liste aufgezählt:

- Zusammenstellung der Dokumente als Forschungsdossiers, die diskursiv den Zusammenhang verschiedener Dokumente herstellen (und auf diese verlinken)
- Zusammenstellung der Dokumente nach Themenschwerpunkten
- Zusammenstellung der Dokumente mit biographischem Bezug

- Ordnung nach Briefkonvoluten[2]
- Darstellung der Dokumente auf einer Karte
- Darstellung der Dokumente als Zeitleiste
- Bereitstellung redaktioneller Vorschläge
- Inventar in einem Handschriftenkatalog der Universitätsbibliothek
- Zugriff über aus der Forschung bekannte Manuskript-ID
- Sequentielles Lesen von vorne nach hinten

Betrachtet man die Suchfunktionalitäten in den Editionen (Abbildung 3.4 auf Seite 45), so ist erkennbar, dass die meisten der Befragten eine Volltextsuche anbieten (n=25), überwiegend mit einem Suchfeld direkt auf der Startseite (n=19). Bereits weniger Editionen bieten erweiterte Suchoptionen und Filter (n=13) oder die Suche mit Platzhaltern (n=13). Noch weniger verbreitet sind sortierbare Trefferlisten (n=7), die Suche nach ähnlichen Suchbegriffen (n=6), die Darstellung der Trefferrelevanz (n=5) und Autovervollständigung im Suchfeld (n=4). Eine Exportmöglichkeit der Ergebnisse (n=4) und eine Worstammsuche (n=3) kommen ebenfalls sehr selten vor. Als weitere Möglichkeit wurde ein Suchbaum genannt, der hierarchisch in der ersten Stufe alle Autoren listet und in der nächsten alle Empfänger.

Ebenfalls aus Abbildung 3.4 auf Seite 45 geht hervor, welche Filtermöglichkeiten angeboten werden. Auch hier lässt sich der Fokus auf Briefeditionen erkennen. Sehr häufig werden Filter nach einem Datum (n=18) oder Zeitraum (n=16) angeboten. Empfänger (n=17), Absender (n=16) und Ort (n=16) sind ebenso häufig vertreten. Gibt es Dokumente in unterschiedlicher Sprache, so wird auch ein Filter nach der Sprache (n=9) angeboten. Interessant ist, dass der Ort als allgemeines Filterkriterium häufiger vorkommt (n=16) als die spezielleren Absende- (n=8) und Empfangsorte (n=7), sich dies jedoch umkehrt bei Empfänger (n=17) und Absender (n=16), welche häufiger vorkommen als der allgemeinere Korrespondent (n=9).

Es wurden auch weitere Filterkriterien genannt:

- Überlieferungsstrang
- Redaktion
- Erwähnte Personen
- Briefwechsel
- Systemnummer
- Wahlperiode
- Dokumenttyp

2 Konvolut ist ein Sammelband / Bündel von Schriftstücken

In Abbildung 3.5 auf Seite 46 wird dargestellt, welche Funktionalitäten mit Suchbezug von digitalen Editionen erwartet werden. Die größte Zustimmung wird der Volltextsuche zuteil. Nach und nach nimmt die Relevanz der Funktionalitäten über Filter, Sortierbarkeit der Ergebnisse, dem Angebot von Platzhaltersuchen und dem Export von Sucherergebnissen ab. Gegenüber der Fehlertoleranz, Autovervollständigung, Wortstammsuche und Darstellung der Trefferrelevanz gibt es vereinzelt Nicht-Zustimmung und teilweise sogar Ablehnung.
Betrachtet man die Relevanz der Darstellung von Metadaten (Abbildung 3.6 auf Seite 46) so sieht man, dass Datum, Ort und Autor mit gleicher Priorität als relevant angesehen werden. Etwas dahinter liegen Titel, Absender und Empfänger. Für die Darstellung des Dokumententyps wird weniger häufig *Stimme voll und ganz zu*, dafür aber oft *Stimme zu* gewählt. Briefnummer und Brieftitel wurden nach Userfeedback zu Beginn der Umfrage entfernt, da ihre Differenzierung zu Titel und Dokument ID unklar waren und werden daher nicht weiter betrachtet.

3.5 ZUSAMMENFASSUNG DER ANALYSEN

In diesem Abschnitt werden die Ergebnisse der Discover Phase zusammengefasst, um die gesammelten Informationen zu strukturieren und die relevanten Problemstellungen zu identifizieren.

Einen Überblick zu den aufgenommenen Themen bietet die Mindmap in Abbildung 3.7 auf Seite 47. Aus den Informationen wurden die Grobkategorien Ergebnisdarstellung, Filter, Volltextsuche, Einstieg und Navigation identifiziert und die Informationen wurden jeweils in Stichworten zugeordnet.
Die nachfolgende Tabelle 3.9 auf der nächsten Seite stellt vereinheitlicht auf grober Ebene dar, welche Funktionalitäten identifiziert wurden. Basierend auf den Erkenntnissen dieses Kapitels werden die Funktionalitäten mit einer Skala von 1 bis 5 bewertet. Der Mittelwert aus Konkurrenzanalyse und Umfrage bildet die Priorität.

Die Priorisierung, basierend auf der Konkurrenzanalyse, wird berechnet aus dem Verhältnis der Vorkommen (Tabelle 3.3 auf Seite 27) in anderen Editionen zu den gesamten untersuchten Editionen (siehe Gleichung (3.1)):

$$5 - \left\lceil \left(\frac{n_{WAHR}}{n_{GESAMT}} * 4\right) \right\rceil \tag{3.1}$$

Der Score für die Erwartung der Funktionalität (Abbildung 3.5 auf Seite 46) aus der Umfrage (Abschnitt 3.4 auf Seite 32) wird dabei so berechnet, dass den ordinalskalierten Antworten von *Stimme voll und ganz zu* $i = 1$ bis *Stimme überhaupt nicht zu* $i = 5$ die Werte $a_1 = 1$ mit 1 bis $a_5 = 5$ zugeordnet werden, jeweils mit ihrer Häufigkeit n_i multipliziert und durch die Gesamt-

summe an gegebenen Antworten geteilt und dann kaufmännisch gerundet (*nint*) wird (siehe Gleichung (3.2)):

$$nint(\frac{\sum_{i=1}^{5} a_i * n_i}{\sum_{i=1}^{5} n_i}) \tag{3.2}$$

Nr.	Funktionalitäten in Suchfunktionen	UMFR	KA	Prio
F-01	Volltextsuche	1	1	1
F-02	Anzahl Treffer dargestellt		1	1
F-03	Inhalt Vorschau		1	1
F-04	Treffer hervorgehoben		1	1
F-05	Filter	2	2	2
F-06	Wildcards (Platzhalter, häufig * oder ?)	2	2	2
F-07	Suchfeld direkt auf Startseite		2	2
F-08	Verlinkungen von Register/Werke auf Startseite		2	2
F-09	Metadaten im Ergebnis darstellen		2	2
F-10	Treffer sortierbar	2	3	3
F-11	Suchergebnisse exportierbar	2	4	3
F-12	Fehlertoleranz der Suche	2	4	3
F-13	Autovervollständigung im Suchfeld	2	4	3
F-14	Wortstammsuche	2	4	3
F-15	Treffer verlinkt		3	3
F-16	Parallelansicht auf Dokumente / Briefe		3	3
F-17	Darstellung der Trefferrelevanz	3	4	4
F-18	Suchergebnisanzahl konfigurierbar		4	4
F-19	Miniaturansicht		4	4
F-20	Trefferliste exportieren		4	4

Tabelle 3.9: Funktionalitäten mit Priorisierung

Die wichtigste Funktion einer Suchfunktion ist nach Einschätzung der Befragten die Volltextsuche. Wenn man die Erwartung von Funktionalitäten der Benutzer damit vergleicht, inwieweit erwartete Funktionalitäten in Editionen verbreitet sind, so zeigt sich, dass hier eine Abweichung beim Sortieren nach Suchergebnissen, Fehlertoleranz in der Suche, Autovervollständigung und Wortstammsuche vorhanden ist, welche zwar von Editionen erwartet werden, aber wenig verbreitet sind in den untersuchten Editionen.

Über die Hälfte aller Befragten erwartet einen Export der Suchergebnisse, dieser wird aber nur von den wenigsten Editionen angeboten.

3.6 DEFINITION DER PROBLEMSTELLUNG

Wie die bisherige Analyse gezeigt hat, sind die Anforderungen an die Suchfunktion vielfältig. In diesem Abschnitt werden nun die Problemstellungen

herausgearbeitet, welche es im Design-Prozess weiter zu analysieren gilt. Diese werden dann fokussiert betrachtet und im Sinne von „Getting the Right Design“ werden verschiedene Lösungsansätze beschrieben und evaluiert. Die Problemstellungen werden sowohl aus der quantitativen als auch aus der qualitativen Analyse hergeleitet.

Dieser Abschnitt listet diese Problemstellung auf und ordnet zur weiteren Referenzierung in dieser Arbeit ein Kürzel zu. Das Kürzel *PS* steht dabei für Problemstellung. Die Reihenfolge der Problemstellungen besitzt keine Semantik und besitzt somit auch keine Aussagekraft über die Priorität.

Die Antworten zu offenen Fragen der Umfrage sind im digitalen Anhang dieser Arbeit abgelegt, welcher in Anhang E auf Seite 144 zu finden ist. Die Verweise auf Antworten werden mit der Fragen-Id, gefolgt von den jeweils relevanten Antwort-Ids angegeben. Mit (Q0016: 8, 26) wird somit auf die Frage mit der Fragen-Id *Q0016* und die Antworten *8* und *26* verwiesen. Da es auch Fragen gibt, die gemeinsam betrachtet werden müssen, können auch mehrere Fragen zu einer Antwort referenziert werden, dabei werden alle Ids separiert aufgeführt.

Teilweise wurden neue Problemstellungen in den Iterationen des zweiten Diamanten (Kapitel 4 auf Seite 48) identifiziert, diese werden in der nachfolgenden Aufzählung ebenfalls beschrieben, da der Double Diamond Designprozess zwar iterativ, aber nicht sequentiell ist und somit erlaubt, neue Erkenntnisse in die Definition der Problemstellungen einfließen zu lassen. Um den Lesefluss dieser Arbeit beizubehalten, wird dabei auf die Iteration verwiesen, in der die Problemstellung ergänzt wurde. In der Übersicht der Problemstellungen (Abschnitt 3.7 auf Seite 43) wird dies in der Spalte *Herkunft* ebenfalls kenntlich gemacht.

PS-01: Ergebnisdarstellung

Der Informationssuchende hat das Ziel, relevante Ergebnisse zu finden. Wenn man ihm bereits auf der Ergebnisseite ermöglicht, dass er erkennen kann, ob ein Ergebnis relevant ist, so erhöht dies die Effizienz der Suche. Je nach Forschungsfrage, könnte sogar der Blick auf die Suchergebnisseite ausreichen (Q0024: 4). Zu viele Ergebnisse können zur Frustration des Anwenders führen, wenn deren Relevanz nicht erkennbar ist (Q0020, Q0021, Q0022, Q0025: 20). Weiterhin kann bereits die Recherche von Metadaten Ziel der Suche sein, beispielsweise „wer mit wem wann worüber korrespondiert hat [oder] wann bestimmte Ereignisse, Personen etc. vom wem erwähnt wurden.“ (Q0009: 53) Welche Metadaten verbreitet sind, kann in Tabelle 3.6 auf Seite 29 eingesehen werden.

Die Problemstellung der Ergebnisdarstellung hat Einfluss auf mehrere Funktionalitäten, darunter die Darstellung der Trefferanzahl (F-02), die Vorschau des Inhalts (F-03), ob die Treffer hervorgehoben werden (F-04) und die Darstellung der Metadaten im Ergebnis (F-09).

Wie kann man die Suchergebnisse darstellen, sodass der Benutzer einen guten Überblick bekommt und die relevanten Ergebnisse identifizieren kann?

PS-02: Kombination aus Entitäts- und Volltextsuche

Mehrere Teilnehmer der Umfrage gaben in der offenen Frage, wie Suchfunktion verbessert werden können an, dass die Suchfunktion einfach und verständlich sein soll. (Q0016: 8, 26, 49, 61) Eine Herausforderung hierbei ist, dass sie „so generisch wie möglich und zugleich so spezifisch wie nötig" (Q0017: 26) ist. „Es sollte auf eine optisch klare und einfache Struktur geachtet werden, die eine intuitive Bedienung ermöglicht - kaum einer liest sich lange Editionsinformationen durch." (Q0017: 59).

Die Problemstellung hat Einfluss auf die Kombination von Volltextsuche (F-01) und die Umsetzung der Filter (F-05).

Wie kann man dem Benutzer eine komplexe Suchanfrage (Kombination aus Entitäts- und Volltextsuche) mit einem einfachen Userinterface ermöglichen?

PS-03: Hilfestellung für den Benutzer

Dem Benutzer Alternativen aufzuzeigen, wie er die Suche verändern kann, könnte für eine geringere Frustration sorgen. Ein Teilnehmer der Umfrage war beispielsweise verwirrt, als er mit einer Suche keine Ergebnisse bekam. (Q0020: 20) Hilfestellungen können Funktionalitäten erläutern oder gewünschte Aktionen vorhersehen und diese vorschlagen.

Wie kann dem Benutzer eine Hilfestellung angezeigt werden, sodass er vor der Suche weiß, was er eingeben kann und bei fehlgeschlagener Suche Hinweise bekommt, um seine Anfrage zu verbessern?

PS-04: Entitätsfilter in unterschiedlichen Rollen

Entitäten können in unterschiedlichen Rollen auftreten, beispielsweise kann eine Person in der Briefedition als Absender, Empfänger oder durch eine Erwähnung in Bezug zu einem Dokument stehen. Die Umfrage hat gezeigt, dass „Absender" und "Empfänger" relevante Filterkriterien sind, aber auch die allgemeine Form „Korrespondent". Umgekehrt hat sich dies beim „Ort" dargestellt, da hier die spezifischeren „Absendeorte" und „Empfangsorte" relevanter waren. Die Problemstellung, die sich hier ergibt, ist, wie man diese unterschiedlichen Rollen abbilden sollte. Beispielsweise könnte man die allgemeine Form, als auch die spezielle Form zugleich anbieten oder es wird pro Entität ausgewählt, in welchen Rollen diese gesucht werden soll.

Die Problemstellungen hat Einfluss auf die Umsetzung der Filter (F-05).

Wie können Entitätsfilter in ihren unterschiedlichen Rollen abgebildet werden?

PS-05: Zeitselektion

Bei der zeitlichen Selektion gibt es mehrere Problemstellungen. Zum einen können zu einem Dokument unterschiedliche Zeitpunkte zugeordnet werden, beispielsweise ein Funddatum, aber nicht das Entstehungsdatum. (Q0014:

8) Außerdem ist zu klären, wie mit einem unvollständigen Datum umgegangen werden kann, wenn ein Teil des Datums nicht bestimmbar oder das Korrespondenzstück nur in einen Zeitraum terminiert werden kann.

Wie kann eine zeitliche Selektion umgesetzt werden?

PS-06: Suche in Kommentaren

Die gesamte Edition einschließlich der Kommentare sollte durchsuchbar sein. Man sollte allerdings den Bereich einschränken können, wenn man nur nach bestimmten Teilen suchen möchte. (Q0016: 53) „Manchmal ist eine differenzierte, separate Suche in Quelltext und Kommentaren nicht sehr komfortabel" (Q0014: 26). Ein weiterer Teilnehmer der Umfrage würde gerne „ausschli[e]ßlich in Kommentaren bestimmter Personen suchen können" (Q0014: 26) Generell könnte die Suche verbessert werden, wenn „mehr Entitäten, Durchsuchbarkeit der Apparate und Kommentare" ermöglicht werden würde. (Q0016: 30)

Wie können Kommentare durchsuchbar gemacht werden?

PS-07: Suche in Metadaten und ihren Beschreibungen

Wenn zu den Metadaten und Entitäten Beschreibungen angelegt wurden, sollten diese auch durchsuchbar sein. (Q0016: 53) In welcher Form diese Ergebnisse dargestellt werden sollen, muss geklärt werden.

Wie können Worte innerhalb der Metadaten und deren Beschreibungen durchsuchbar gemacht werden?

PS-08: Flexible Fehlertoleranz

Die „Fehlertoleranz sollte optional ein- oder ausgeschaltet werden können (manchmal sucht man ein bestimmtes, genau so geschriebenes Wort und möchte keine ähnlichen)" (Q0016: 53) Generell kommt es auf den Schwerpunkt der Edition an, aber eine hohe „Fuzzyness" mit sinnvollen Eingrenzungsmöglichkeiten bei zu vielen Ergebnissen wird geschätzt. (Q0015: 60) Dabei sollten sämtliche Formen von Worten gesucht werden (Q0020, Q0021, Q0022, Q0025: 4)

Diese Problemstellung bezieht sich sowohl auf die Funktionalität der Fehlertoleranz der Suche (F-12) als auch auf die Wortstammsuche (F-14).

Wie kann die Fehlertoleranz flexibel gestaltbar gemacht werden?

PS-09: Speicher-, referenzier- und navigierbare Suchabfragen

Die Suchfunktion könnte durch speicherbare und referenzierbare Suchergebnisse verbessert werden. (Q0016: 55) Man sollte vom Suchergebnis auf die zuletzt durchgeführte Suchanfrage zurückkehren können.

Wie können Suchergebnisse speicher-, referenzier- und navigierbar gestaltet werden?

PS-10: Vorschlag alternativer Suchbegriffe

Die Suchfunktion könnte Anwender besser unterstützen, wenn sie Wortalternativen (Q0025: 66) oder alternative Begriffe zum Suchthema (Q0025: 67) vorschlagen würde.

Wie können alternative Suchbegriffe vorgeschlagen werden?

PS-11: Sammlung von Dokumenten zusammenstellen

Bei der Untersuchung einer Forschungsfrage kommt es vor, dass man eine Menge an Briefen zu einem Thema manuell zusammenstellen musste (Q0020: 51). Dazu wurde die Volltextsuche genutzt. Diese Menge an Briefen musste in eine chronologische Abfolge gebracht werden. Eine Zusammenstellung von Dokumenten in der Suchfunktion könnte ermöglichen, dass weitere Suchen und Operationen auf dieser Sammlung durchgeführt werden könnten.

Wie kann man in der Suchfunktion das Zusammenstellen und auf dieser Sammlung aufbauende Suchen ermöglichen?

PS-12: Erschließung von Hintergrundinformationen

Wenn bei der Suche unbekannte Entitäten entdeckt werden, könnte das Interesse des Informationssuchenden geweckt werden, mehr über die Entität zur erfahren („Browsing"). Einem Teilnehmer der Umfrage halfen „Personen- und Ortsregister [...] für die Erschließung des Hintergrundes" (Q0020: 59) In der Suchfunktion könnten diese zusätzlichen Informationen bereits zugänglich gemacht werden.

Wie kann die Erschließung von Hintergrundinformationen in den Ergebnissen erleichtert werden?

PS-13: Hilfsmaterialen und Dokumentation

„Umfangreiche Hilfematerialien fehlen in vielen Fällen. Die Suchschnittstellen sind in einigen Fällen nicht dokumentiert." (Q0025: 48) Gerade bei Funktionalitäten wie unscharfer Suche oder erweiterten Sucheinstellungen erschließt sich dem Benutzer nicht intuitiv, wie diese funktionieren. Gerade wenn es um die exakte Interpretation der Suchergebnisse geht, sollte nachvollzogen werden können, wie die Ergebnisse Zustande kommen.

Wie kann die Suchfunktion mit Hilfsmaterialen und Dokumentation unterstützt werden?

PS-14: Phrasensuche

Eine Phrase ist definiert, als Worte die aufeinander Folgen. Dabei können Füllwörter zugelassen werden. Ob und wie viele Füllwörter zugelassen werden sollen, könnte konfigurierbar oder durch Platzhalter konfigurierbar gestaltet werden. (Q0015: 20, 23) In dieser Problemstellung soll untersucht wer-

den, wie die Phrasensuche intuitiv abgebildet werden kann. Es soll geprüft werden, wie die Flexibilität zugänglich umgesetzt werden kann.

Wie kann eine Phrasensuche abgebildet werden?

PS-15: XML Export

Nach Franzini, Mahony und Terras sollten gute digitale Editionen den XML-Standard unterstützen und den Export der Daten als XML-Dateien ermöglichen [14, S. 180]. Wenn Anwender eine große Auswahl an Dokumenten selektieren möchten, wäre es benutzerfreundlich, alle Dokumente einer Suchanfrage auf einmal exportieren zu können. Ein Umfrageteilnehmer schätzt an Suchfunktionen in Editionen, wenn er Dokumente zur Weiterverwendung in XML/TEI exportieren kann (Q0015: 49). Einem anderen fehlt häufig die Exportfunktion für Suchergebnisse (Q0014: 61).

Wie können die XML Daten exportierbar gemacht werden?

PS-16: XML Abfrage

Nach Sahle ist XML/TEI der führende Standard der „Digital Humanities" für die Transkription [45, S. 343]. Die Verwendung von Auszeichnungssprachen ermöglicht es, mehr Informationen als im Klartext zu kodieren [45, S. 108ff.]. Diese Informationen abfragbar zu gestalten, könnte mit einem Query Builder umgesetzt werden. Diese sind aber einem Umfrageteilnehmer aber oft zu wenig flexibel. (Q0020, Q0021, Q0022, Q0025: 44) Dennoch könnte eine „stärkere Nutzung der XML-Kodierung der Dokumente" zur Verbesserung von Suchfunktionen beitragen (Q0016: 44).

Wie kann die Suche in den XML Daten ermöglicht werden?

PS-17: Sortieren

Wonach in der Konkurrenzanalyse am häufigsten sortiert werden konnte, ist in Tabelle 3.7 auf Seite 30 dargestellt. Am häufigsten waren dies Sortierungen nach Datum und der Relevanz. Diese Problemstellung bezieht sich auf die Funktionalität Treffer sortieren (F-10).

Wie können die Ergebnisse sortiert werden?

PS-18: Filtern

Diese Problemstellung greift das Thema Filtern der Suchergebnisse allgemeiner auf als die speziellen Problemstellungen PS-02, PS-04 und PS-05. Die Herausforderung hierbei besteht darin, wie eine einheitliche Lösung über die einzelnen Problemstellung abgebildet werden kann.

Wie können Filter umgesetzt werden?

PS-19: Autovervollständigung

Die Autovervollständigung wurde in etwa von der Hälfte aller Befragten der Umfrage als Funktionalität in einer Suchfunktion erwartet (Abbildung 3.5

auf Seite 46), obwohl die Funktionalität (F-13) in der Konkurrenzanalyse nicht weit verbreitet war. Mit dieser Problemstellung soll erarbeitet werden, wie Vorschläge für Autovervollständigung generiert werden können.

Wie kann eine Autovervollständigung umgesetzt werden?

PS-20: Vorschlag letzter Suchanfragen

In der ersten Iteration (Abschnitt 4.3 auf Seite 51) wurde die Idee generiert, neben Vorschlägen basierend auf dem Inhalt auch vergangene Suchanfragen anzuzeigen. Diese Problemstellung wurde so nicht in der Analyse aufgedeckt, soll aber hier ergänzt werden. Der Double Diamond Designprozess erlaubt auch die Problemstellungen Iterativ zu erweitern und anzupassen.

Wie kann man dem Benutzer letzte Suchanfragen vorschlagen?

PS-21: Schlagwörtern oder Themen

An Suchfunktionen wird geschätzt, wenn diese ein „Normdaten basierte[s] Schlagwort-Register" besitzen (Q0015, Q0014: 50). Eine Themensuche könnte die Suche verbessern (Q0025: 8).

Wie kann die Suchfunktion um Schlagwörter oder Themen erweitert werden?

PS-22: Korrespondenzen

Korrespondenzstücke können Teil einer längeren Korrespondenz sein. Diese Problemstellung bezieht sich darauf, wie die Beziehung von Dokumenten berücksichtigt werden könnten? Die Suche könnte darstellen, auf welchen Brief ein Brief antwortet (Q0025: 51). Außerdem wäre eine Suche innerhalb von Korrespondenzen interessant.

Wie können Korrespondenzen in der Suche berücksichtigt werden?

PS-23: Wildcards

Wildcards sind definierte Zeichen, die als Platzhalter für Zeichen in einer Suchanfrage dienen. Mit Wildcards können beispielsweise Wortteile gesucht werden. Die Funktionalität wurde in der Umfrage geschätzt (Q0015: 54) und die Abwesenheit von Suchmöglichkeiten mit Wildcards wurde kritisiert (Q0014: 61).

Wie kann die Volltextsuche mit Wildcards umgesetzt werden?

PS-24: Treffer verlinkt

Diese Problemstellung ist zur Umsetzung von Funktionalität F-15 relevant.

Wie kann beim Öffnen des Dokuments direkt auf den Treffer gesprungen oder die Treffer hervorgehoben werden?

PS-25: Vorschau konfigurierbar

In der Evaluation der vierten Iteration (Abschnitt 4.3 auf Seite 51) wurde der Vorschlag gemacht, dass man die Vorschau der Treffer konfigurierbar gestalten könnte. Hierbei könnte die Länge der Vorschau variabel gestaltet werden.

Wie könnte man die Vorschau des Suchtreffers konfigurierbar gestalten?

PS-26: Darstellung des Suchzustands

Bei einer komplexen Suchanfrage haben mehrere Bereiche Einfluss auf das Suchergebnis. Bereits eine Kombination von Volltextsuche und Filtersuche kann dazu führen, dass dem Benutzer nicht klar ist, welche Einstellungen gerade aktiv sind. Dies wurde in der Evaluation der fünften Iteration (Abschnitt 4.7 auf Seite 77) deutlich. Daher soll für die nachfolgende Problemstellung ebenfalls eine Lösung erarbeitet werden: Wie kann dem Benutzer der Suchzustand verdeutlicht werden?

Die gesammelten Problemstellungen werden in der Tabelle in Tabelle 3.10 auf der nächsten Seite als Übersicht dargestellt.

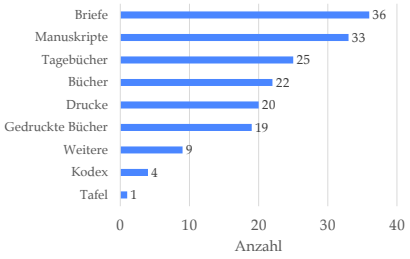
3.7 ÜBERSICHT DER PROBLEMSTELLUNGEN

In diesem Abschnitt werden die Problemstellungen in Tabelle 3.10 auf der nächsten Seite zusammengefasst dargestellt. Sie bilden die Grundlage für die Bearbeitung

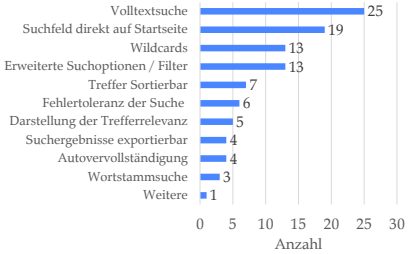
ID	Titel	Herkunft	Problemstellung
PS-01	Ergebnisdarstellung	ANALYSE	Wie kann man die Suchergebnisse darstellen, sodass der Benutzer einen guten Überblick bekommt und die relevanten Ergebnisse identifizieren kann?
PS-02	Kombination aus Entitäts- und Volltextsuche	ANALYSE	Wie kann man dem Benutzer eine komplexe Suchanfrage (Kombination aus Entitäts- und Volltextsuche) mit einem einfachen Userinterface ermöglichen?
PS-03	Hilfestellung für den Benutzer	ANALYSE	Wie kann dem Benutzer eine Hilfestellung angezeigt werden, sodass er vor der Suche weiß was er eingeben kann und bei fehlgeschlagener Suche, Hinweise bekommt, seine Anfrage zu verbessern?
PS-04	Entitätsfilter in unterschiedlichen Rollen	ANALYSE	Wie können Entitätsfilterin ihren unterschiedlichen Rollen abgebildet werden?
PS-05	Zeitselektion	ANALYSE	Wie kann eine zeitliche Selektion umgesetzt werden?
PS-06	Suche in Kommentaren	ANALYSE	Wie können Kommentare durchsuchbar gemacht werden?
PS-07	Suche in Metadaten und ihren Beschreibungen	ANALYSE	Wie können Worte innerhalb der Metadaten und deren Beschreibungen durchsuchbar gemacht werden?
PS-08	Flexible Fehlertoleranz	ANALYSE	Wie kann die Fehlertoleranz flexibel gestaltbar gemacht werden?
PS-09	Speicher-, Referenzier- und Navigierbare Suchabfragen	ANALYSE	Wie können Suchergebnisse speicher-, referenzier- und navigierbare gestaltet werden?
PS-10	Vorschlag alternativer Suchbegriffe	ANALYSE	Wie können alternative Suchbegriffe vorgeschlagen werden?
PS-11	Sammlung von Dokumenten zusammenstellen	ANALYSE	Wie kann man in der Suchfunktion das Zusammenstellen und auf dieser Sammlung aufbauende Suchen ermöglichen?

PS-12	Erschließung von Hintergrundinformationen	ANALYSE	Wie kann die Erschließung von Hintergrundinformationen in den Ergebnissen erleichtert werden?
PS-13	Hilfsmaterialen und Dokumentation	ANALYSE	Wie kann die Suchfunktion mit Hilfsmaterialen und Dokumentation unterstützt werden?
PS-14	Phrasensuche	ANALYSE	Wie kann eine Phrasensuche abgebildet werden?
PS-15	XML Export	ANALYSE	Wie können die XML Daten exportierbar gemacht werden?
PS-16	XML Abfrage	ANALYSE	Wie kann die Suche in den XML Daten ermöglicht werden?
PS-17	Sortieren	ANALYSE	Wie können die Ergebnisse sortiert werden?
PS-18	Filtern	ANALYSE	Wie kann man Filter umsetzen?
PS-19	Autovervollständigung	ANALYSE	Wie kann eine Autovervollständigung umgesetzt werden?
PS-20	Vorschlag letzter Suchanfragen	ITERATION 1	Wie kann man dem Benutzer letzte Suchanfragen vorschlagen?
PS-21	Schlagwörtern oder Themen	ANALYSE	Wie kann die Suchfunktion um Schlagwörter oder Themen erweitert werden?
PS-22	Korrespondenzen	ANALYSE	Wie können Korrespondenzen in der Suche berücksichtigt werden?
PS-23	Wildcards	ANALYSE	Wie kann die Volltextsuche mit Wildcards umgesetzt werden?
PS-24	Treffer verlinkt	ANALYSE	Wie kann man beim Öffnen des Dokuments direkt auf den Treffer springen oder die Treffer hervorheben?
PS-25	Vorschau konfigurierbar	ITERATION 4	Wie könnte man die Vorschau des Suchtreffers konfigurierbar gestalten?
PS-26	Darstellung des Suchzustands	ANALYSE	Wie kann dem Benutzer der Suchzustand verdeutlicht werden?

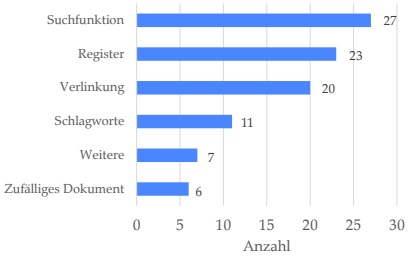
Tabelle 3.10: Übersicht über die Problemstellungen



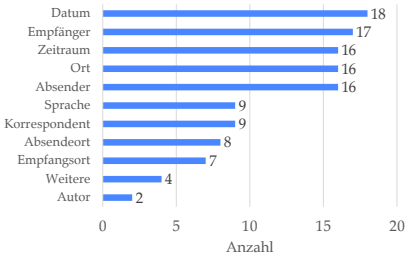
(a) Bitte wählen alle Dokumenttypen mit denen Sie sich in digitalen (historisch-kritischen) Editionen beschäftigen



(b) Besitzt die Suchfunktion Ihrer digitalen (historisch-kritischen) Edition folgende Funktionalität



(c) Einstiegsmöglichkeiten



(d) Welche Filterkriterien bietet die Suchfunktion

Abbildung 3.4: Was die Editionen der Befragten an Funktionalitäten anbieten

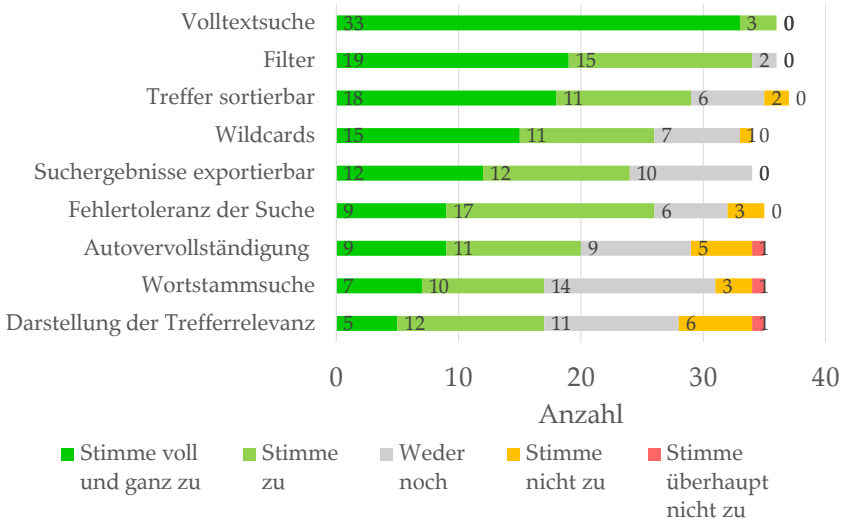


Abbildung 3.5: Erwartung an Funktionalität in Editionen

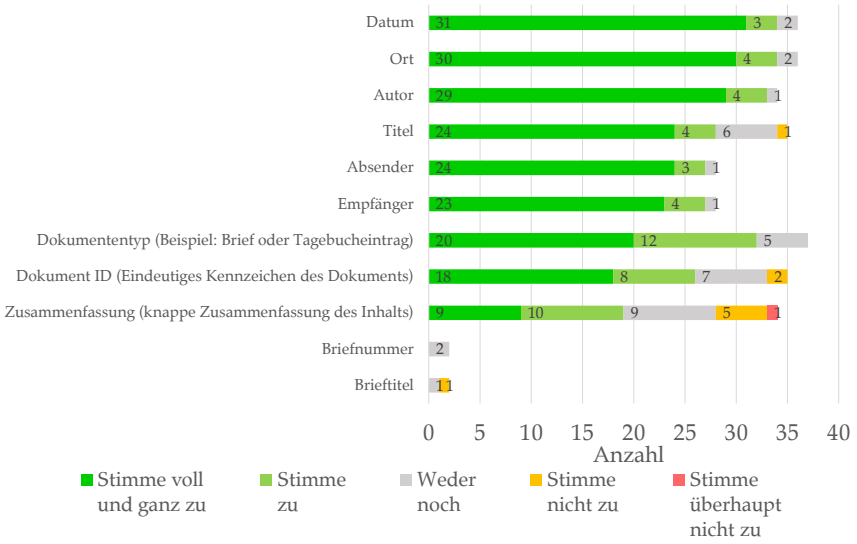


Abbildung 3.6: Relevanz von Metadaten in der Darstellung der Suchergebnisse

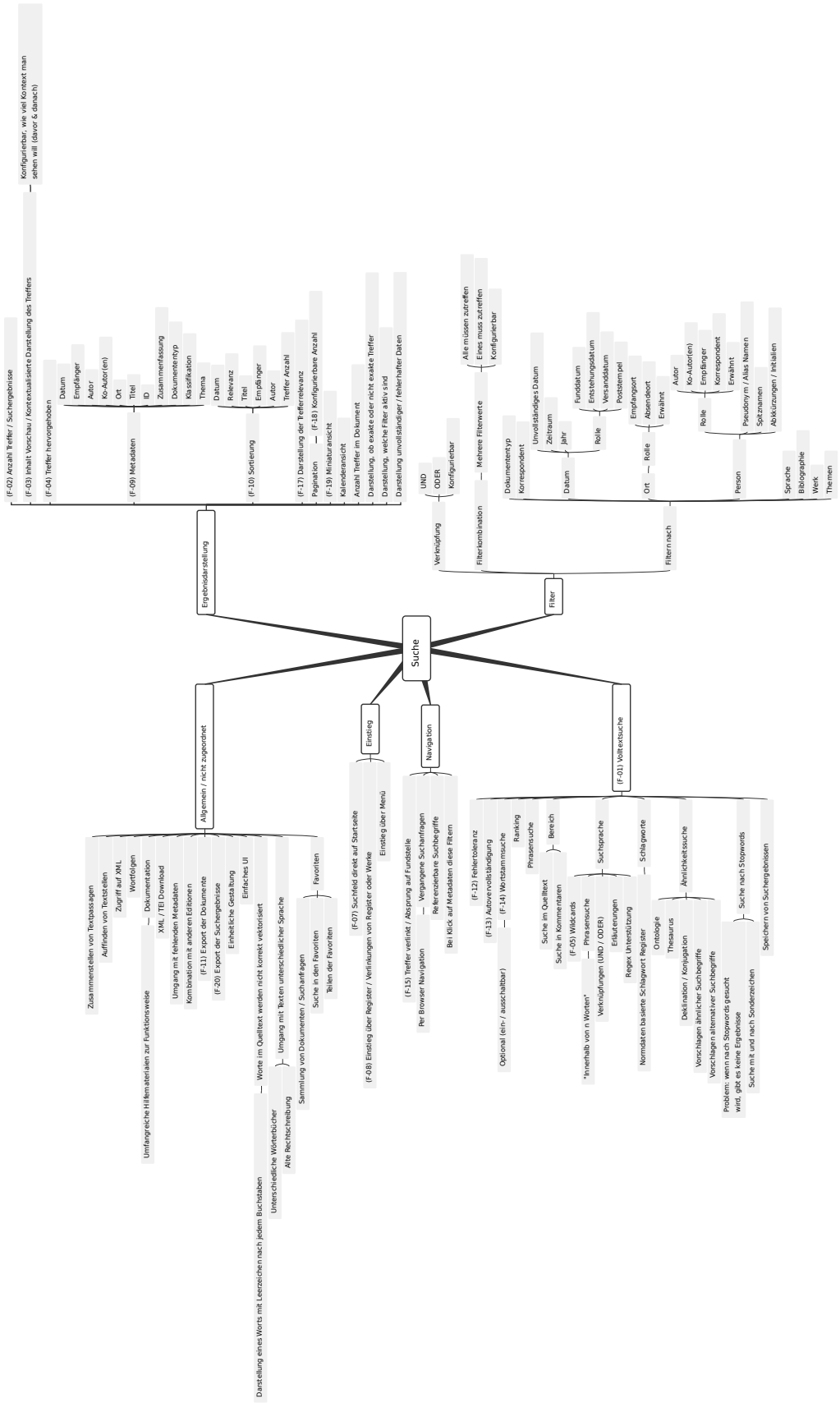


Abbildung 3.7: Mindmap mit Zusammenfassung der aufgenommenen Themen

4 DEVELOP UND DELIVER

Das in dieser Arbeit beschriebene Vorgehen orientiert sich an dem Double-Diamond Prozess. Dieses Kapitel bezieht sich auf die Iterationen des zweiten Diamanten (blau hervorgehoben in Abbildung 4.1). Dabei werden zunächst die jeweiligen Problemstellungen der Iteration dargelegt. Diese können mit den Erkenntnissen der vorhergehenden Iterationen angepasst werden. Basierend auf den Problemstellungen wird ausgeführt, wie mit Sketches, Prototypen und Design-Workshops Lösungen für diese erarbeitet wurden. Abschließend werden pro Iteration die Ergebnisse der Evaluationsphase beschrieben. Um die Nachvollziehbarkeit zu gewährleisten, werden Zwischenschritte des Designs vorgestellt und auf das Feedback der User eingegangen. In den höheren Iterationen und sobald der High-Fidelity-Prototyp ausgereifter ist, wird mittels Remote-Usability-Tests die Evaluation durchgeführt.

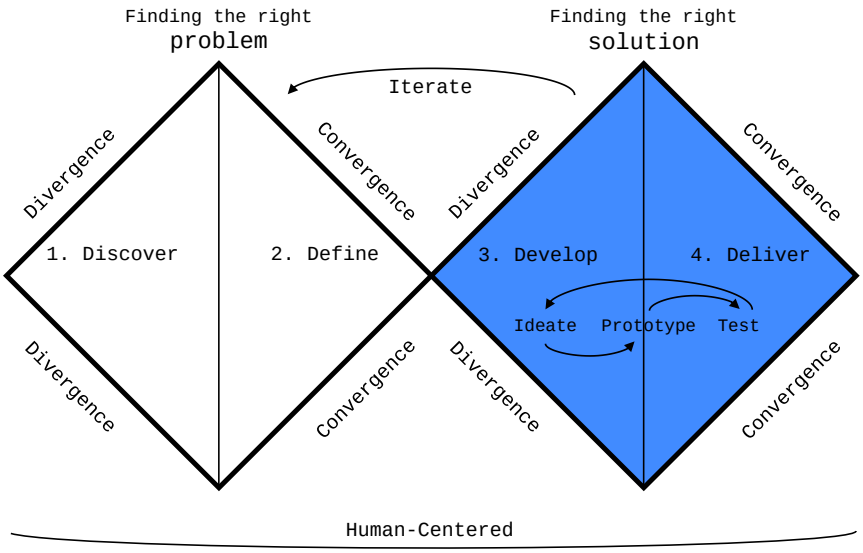


Abbildung 4.1: Hervorhebung des Develop und Deliver Diamanten (Abbildung in Anlehnung an [52, S. 6, 24])

4.1 BESCHREIBUNG DER EVALUATIONS METHODIK

In den Iterationen wurde das Feedback von Nutzern und Experten mit unterschiedlichem Hintergrund eingeholt. Bei den in den jeweiligen Abschnitten beschriebenen Rückmeldungen wird jeweils auf bestimmte, anonymisierte Personen verwiesen. Informationen zu diesen Personen wie Alter, Forschungsfeld und Beteiligung am Forschungsprojekt zum Aufbau der Briefe-

dition sind in Tabelle 4.1 dargestellt. Dabei hatten die befragten Personen unterschiedlich lange Erfahrung mit digitalen (historisch-kritischen) Editionen. Die Personen wurden in Form von Workshops, Guerrilla Tests, Usability-Inspektion und mit einem moderierten Remote-Usability-Test in den Prozess eingebunden.

P-1 und P-2 haben im Verlauf ihres Germanistikstudiums von 2015-2020 und 2013-2018 Erfahrung mit unterschiedlichen digitalen (historisch-kritischen) Editionen[1], die überwiegend für Seminararbeiten und Hausarbeiten in der Literaturwissenschaft genutzt wurden. Zusätzlich wurde regelmäßig das Feedback einer weiteren Person (P-3) eingeholt, die noch gar keine Erfahrung mit Editionen oder Frank Wedekind hat, um noch weitere Informationen zur Usability zu erhalten. Die Personen P-4 bis P-10 sind aktuelle oder ehemalige Projektbeteiligte an der Briefedition. Das interdisziplinäre Projekt wurde mitinitiiert von P-5 und von P-10. Die Personen P-4 und P-6 bis P-9 sind später zum Projekt hinzugestoßen.

Nr.	Geschlecht	Alter	Forschungsfeld	Rolle	Wedekind-projekt
P-1	F	20-30	Germanistik	Nutzer	0 Jahre
P-2	M	20-30	Germanistik	Nutzer	0 Jahre
P-3	F	30-40	Informatik	Nutzer	0 Jahre
P-4	M	30-40	Informatik	Forschung Usability Experte	2 Jahre
P-5	F	40-50	Informatik	Forschung	10 Jahre
P-6	F	40-50	Editionsphilologie	Editor	2 Jahre
P-7	F	50-60	Editionsphilologie	Editor	0 Jahre
P-8	M	40-50	Editionsphilologie	Editor	2 Jahre
P-9	F	50-60	Editionsphilologie	Editor	2 Jahre
P-10	M	70-80	Editionsphilologie	Beratung	10 Jahre

Tabelle 4.1: Übersicht über die an der Evaluation beteiligten Personen

Um den High-Fidelity-Prototyp zu testen, werden Remote-Usability-Tests durchgeführt. Nachfolgend wird das Design des Remote-Usability-Tests beschrieben.

Der Test besteht aus drei Teilen. Zunächst wird ein kurzes semistrukturiertes Interview geführt. Dies hat zum Zweck die Erwartung des Benutzers abzufragen, welchen Bezug sie bereits zu digitalen Editionen und dem Suchen in Editionen hatten und um der Testperson einen angenehmen Einstieg in das Gespräch zu ermöglichen.

Anschließend erhält der Benutzer Aufgaben, die er lösen soll. Erst jetzt wird der Prototyp geöffnet, sodass er beim Interview nicht von der Suchfunktion beeinflusst wird. Bei der Lösung der Aufgaben teilt die Testperson den Bildschirm, sodass die Lösung der Aufgaben beobachtet werden kann.

1 Dies waren die beispielsweise die Editionen Faust Edition http://faustedition.net/ oder das Heinrich Heine Portal http://www.hhp.uni-trier.de/

Zusätzlich wird die Testperson darum gebeten, zu beschreiben, welche Gedanken sie bei der Interaktion mit der Applikation hat. Während der Lösung der Aufgabenstellungen werden durch den Tester keine Hinweise gegeben, um die Testperson nicht zu beeinflussen.

Abgeschlossen wird der Remote-Usability-Test durch eine retrospektive Betrachtung mit einem weiteren semistrukturierte Interview, um zusätzlich zu den Erkenntnissen aus der Beobachtung der Vorgehensweise der Testperson eine Einschätzung der Testperson zur Usability der Anwendung zu erhalten.

Der Remote-Usability-Test ist in Anhang D auf Seite 132 beschrieben.

4.2 ÜBERSICHT ÜBER DIE ITERATIONEN

In Abbildung 4.2 wird der Design Prozess im zeitlichen Ablauf dargestellt. Die Darstellung ist dabei angelehnt an [20, Fig. 1.] von Holstein und Störl, die ebendiesen Designprozess in ihrer Arbeit mit der Digital Humanities Forschungsgruppe nutzen. Zunächst wurde der erste Diamant mit Konkurrenzanalyse, Umfrage und Definition der Problemstellung durchgeführt. Anschließend wurden iterativ mehrere Develop und Deliver Phasen bearbeitet. Die dritte und vierte Iteration liefen hierbei parallel ab, wobei in der dritten neue Problemstellungen betrachtet wurden und in der vierten identifzierte Usability-Probleme bearbeitet wurden. Mit der fünften Iteration wurden diese Stränge wieder zusammengeführt. Zusätzlich dargestellt werden die relevanten Versionen des Prototypen, welche am Ende der jeweiligen Iteration evaluiert wurden. Im Sinne von *Ideate, Prototype, Test* (Abbildung 4.1 auf Seite 48) wurde auch in den Zwischenversionen mit Guerrilla Tests Feedback eingeholt.

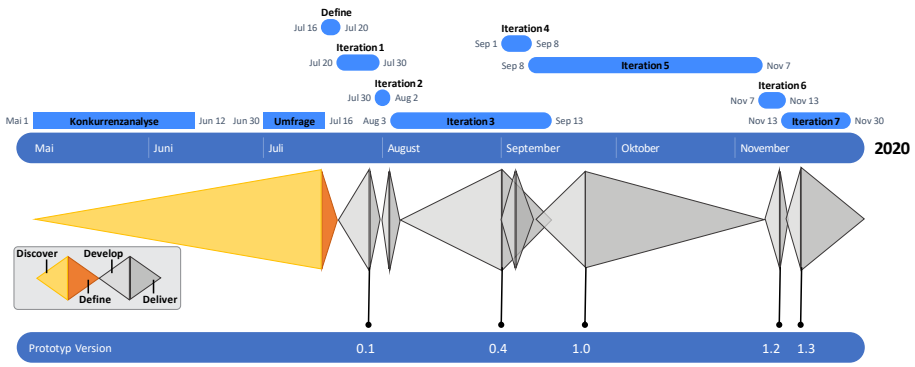


Abbildung 4.2: Die Iterationen im zeitlichen Verlauf

In den weiteren Abschnitten dieses Kapitels werden die durchgeführten Iteration beschrieben. Hierbei wird zunächst die Problemstellung der Iteration beschrieben, gefolgt von den Develop und Deliver Phasen.

4.3 ERSTE ITERATION

4.3.1 *Problemstellung*

Die Ziele der ersten Iteration sind zum einen, eine möglichst große Anzahl von Ideen zu generieren, um die im Kontext der Suchfunktion identifzierten Probleme aus der Analysephase zu lösen, zum anderen, die Evaluation und Auswahl der besten Lösungsvorschläge.

Für die erste Iteration werden zunächst die Problemstellungen PS-01 (Ergebnisdarstellung), PS-02 (Komplexe Suche, einfach gestaltet) und PS-18 (Filter) ausgewählt. Die detaillierte Problembeschreibung ist in Abschnitt 3.6 auf Seite 36 beschrieben.

4.3.2 *Develop*

In der ersten Iteration wird die Annahme getroffen, dass eine Suche mit einer Volltexteingabe beginnt. Lösungsansätze für die Anfragenformulierung im Problem PS-02 (Komplexe Suche, einfach gestaltet) sind in Abbildung 4.3 auf der nächsten Seite abgebildet. Allen gefundenen Optionen ist gemein, dass sie aus einem Suchfeld und einem Button zum Ausführen der Suche bestehen.

Hilfestellungen für den Benutzer

In Abbildung 4.3a auf der nächsten Seite wird dargestellt, wie Hinweise für den Benutzer angezeigt werden könnten. Die Idee hierbei ist es, dem Benutzer einen Vorschlag für die Eingabe zu geben, um bei der Suche zu unterstützen.

Eine Abwandlung davon ist in Abbildung 4.3b auf der nächsten Seite skizziert, wobei hier statt Vorschlägen die in der Vergangenheit durchgeführten Suchanfragen angeboten werden. Bei der Anzeige der letzten Suchanfragen ist es dem Benutzer zusätzlich möglich, mit einem Klick auf das „X“ diese zu entfernen.

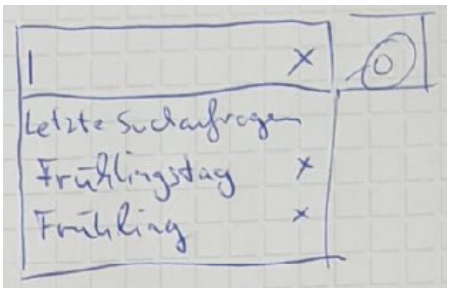
Abbildung 4.3c auf der nächsten Seite zeigt, wie man Vorschläge aus den letzten Suchanfragen und einer Autocompletion kombinieren könnte. In der Autocompletion wird mithilfe eines textuellen Labels ergänzt, um welche Art von Entität es sich handelt.

Möglich wäre auch die Kennzeichnung mit einem Symbol, wie in Abbildung 4.3d auf der nächsten Seite dargestellt. Die Herausforderung bei der Verwendung dieser ist es, dass für die verwendeten Entitäten einfach verständliche Symbole gefunden werden müssen, da sie ansonsten eher zu Verwirrung führen könnten. Eine andere Erweiterung wäre es, dass bei der Anzeige der letzten Suchanfragen nur diejenigen abgelegt werden, für die die Suche Treffer lieferte.

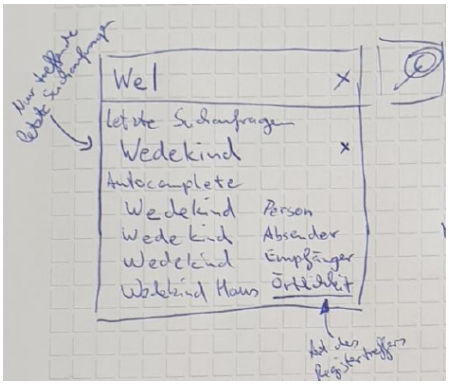
Um die Suche möglichst einfach zu gestalten, aber ebenso umfangreiche Einstellungsmöglichkeiten anzubieten, könnte man die erweiterten Suchoptionen zunächst ausblenden und als einblendbar gestalten. So kann der Be-

nutzer diese erst anzeigen, wenn er sie benötigt. Diese Funktionalität ist in Abbildung 4.3e skizziert.

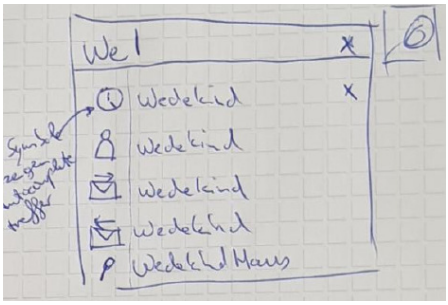
(a) Sucheingabe Vorschlag: Mögliche Anfragen



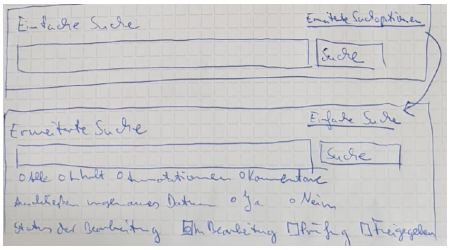
(b) Sucheingabe Vorschlag: Letzte Anfragen



(c) Sucheingabe Vorschlag: Kombination Letzte Suchanfrage und Autocompletion



(d) Kennzeichnung des Vorschlags mit Symbol



(e) Erweiterte Suchoptionen

Abbildung 4.3: Suchfeld - Sketches der Möglichkeiten

Kontextabhängig könnten weiterhin verschiedene Vorschlagsmodi definiert werden. Wenn beispielsweise noch keine vergangenen Anfragen bekannt sind, wird angenommen, dass es sich beim Anwender um einen neuen Benutzer handelt. In dem Fall wäre der Vorschlag möglicher Suchbegriffe eine passende Hilfestellung. Insgesamt wird aus den hierbei bestehenden Optionen ersichtlich, dass dieses Problem in weitere Unterprobleme zerlegt werden kann, die zusätzliche Funktionalität zur Eingabe des Suchbegriffs bedingen.

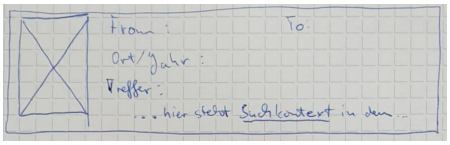
Die Basisfunktionalität zur Suche wird durch eine einfache Texteingabebox erfüllt. Die weiteren Probleme der Autovervollständigung (PS-19), Hilfe-

stellung für den Benutzer (PS-03) und Vorschlag letzter Suchanfragen (PS-20) werden separat betrachtet.

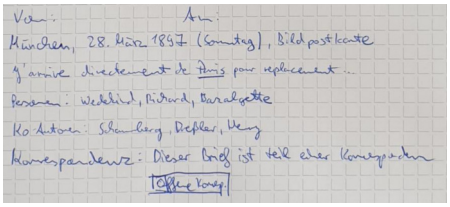
Ergebnisdarstellung

Nach der Ausführung der Suche werden die gefundenen Ergebnisse dargestellt. Das Ziel der Ergebnisdarstellung ist dabei, dass der Benutzer die für ihn relevanten Dokumente identifizieren kann. Die Problemstellung PS-01 (Ergebnisdarstellung) wird in den Skizzen in Abbildung 4.4 behandelt.

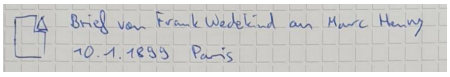
Als erste Option wurde die Darstellung des Ergebnisses als Karte identifiziert. Diese Karte, abgebildet in Abbildung 4.4a dient dabei der logischen Zusammenfassung eines Suchergebnisses. Zu diesem Suchergebnis werden verschiedene Metadaten, eine Abbildung des Faksimiles und die Treffer im Volltext, die zum Fund geführt haben, angezeigt werden.



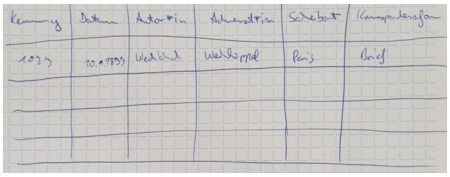
(a) Darstellung als Karte



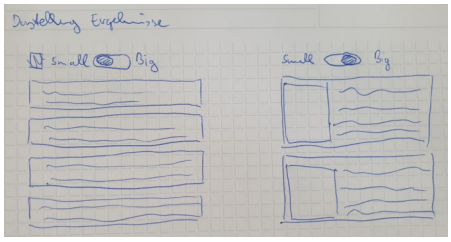
(b) Darstellung als Textzeilen



(c) Kurzdarstellung



(d) Darstellung als Tabelle



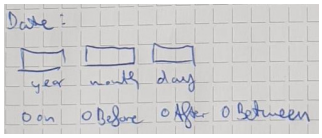
(e) Umschaltung Groß und Klein

Abbildung 4.4: Ergebnis Darstellung - Sketches der Möglichkeiten

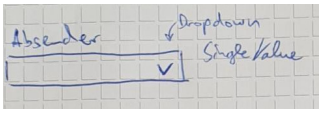
Filter

Für die Umsetzung der Problemstellung PS-18 (Filter) wurden die Sketches in Abbildung 4.5 auf der nächsten Seite entworfen. Für die Auswahl von Entitäten gibt es die Möglichkeit, der Darstellung mit Checkboxen (Abbildung 4.5c auf der nächsten Seite). Hierbei könnte man mehrere Werte auswählen, die Anzahl der Treffer pro Entität darstellen sowie mit *show more* nur dann mehr Elemente anzeigen, wenn die gewünschte Entität nicht in der Liste vorhanden ist. Diese Auswahl ermöglicht eine Mehrfachauswahl. Die zweite Alternative wäre eine Selektion per Dropdown-Box, was in Abbildung 4.5b auf der nächsten Seite dargestellt ist.

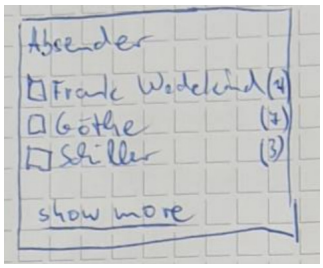
Für die Umsetzung des Zeitfilters wurden drei Möglichkeiten skizziert. Bei Abbildung 4.5a werden Jahr, Monat und Tag separat eingegeben. Somit könnte man auch das Eingeben eines unvollständigen Datums erlauben. Per Radiobutton kann dann ausgewählt werden, ob das Datum exakt getroffen werden soll oder man einen Zeitraum vor oder nach dem Datum selektieren möchte. Ein zweite Möglichkeit besteht in der Selektion des Zeitraums mit zwei Datumsauswahlboxen (Abbildung 4.5d). Dies ermöglicht es, einen Zeitraum zwischen zwei exakten Daten zu selektieren. Die Zeitselektion kann auch so gestaltet werden, dass nur das Jahr auswählbar ist. Diese Option ist in Abbildung 4.5e dargestellt und würde eine Konsistenz der Benutzerschnittstelle gewährleisten, wenn die anderen Entitätsfilter wie in Abbildung 4.5c umgesetzt würden.



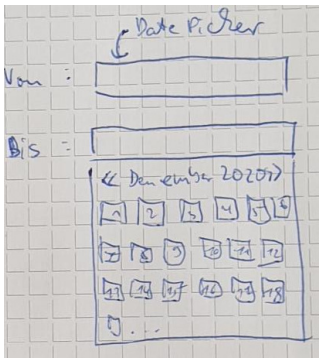
(a) Zeitfilter mit separater Eingabe von Jahr, Monat und Tag



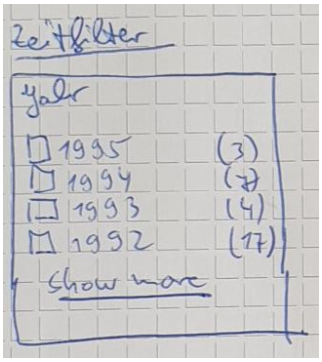
(b) Filter als Dropdown



(c) Filter mit Checkboxen



(d) Zeitfilter mit Von-Datum und Bis-Datum



(e) Zeitfilter mit Checkboxen

Abbildung 4.5: Möglichkeiten Filter umzusetzen

4.3.3 *Deliver*

Die Deliver Phase hat zum Zweck, das Feedback von Benutzern einzuholen, um dieses zur Bewertung der Lösungen zu verwenden. Für die Evaluation

wurden die Sketches aus der Develop-Phase digitalisiert, an zwei Studenten der Germanistik (P-1 und P-2) verschickt und anschließend wurden separate Befragungen durchgeführt. Dazu wurde zunächst durch die Skizzen geführt und anschließend wurden sie um eine Bewertung der jeweiligen Vor- und Nachteile der einzelnen Lösungsansätze gebeten, sowie um allgemeines Feedback. Die Gesprächsnotizen wurden durch die Gesprächspartner für die Veröffentlichung in dieser Arbeit freigegeben und befinden sich in Anhang B auf Seite 123.

Im Bezug auf die Möglichkeiten der Ergebnisdarstellung (Abbildung 4.4 auf Seite 53) konnte eine klare Präferenz für die Ergebnisdarstellung als Karte (Abbildung 4.4a auf Seite 53) festgestellt werden. Diese Möglichkeit wurde als übersichtlich beschrieben, da hier relevante Informationen auf einen Blick dargestellt werden. Die Idee, mehr Details einzublenden (Abbildung 4.4e auf Seite 53) wurde als sinnvoll erachtet und könnte mit der Darstellung der Karte kombiniert werden.

Bei den Filtermöglichkeiten wurde die Auswahl mit Checkboxen in einer Liste (Abbildung 4.5c auf der vorherigen Seite) als übersichtlich eingeschätzt. Bei den Zeitfiltern bestand Unsicherheit darüber, ob die Lösung Abbildung 4.5a auf der vorherigen Seite auch funktionieren kann, wenn man das genaue Datum nicht kennt. Bei Abbildung 4.5d auf der vorherigen Seite wurde der Zeitdialog als problematisch eingeschätzt, wenn man ein Datum in einem vergangenen Jahrhundert sucht. Hier könnte es sein, dass man sehr oft klicken müsste, wenn der Zeitdialog nicht darauf optimiert ist, große Zeitschritte zu ermöglichen. Die Lösung in Abbildung 4.5e auf der vorherigen Seite wurde als unübersichtlich wahrgenommen. Die Ideen zu Filtermöglichkeiten müssen in den nächsten Iterationen noch weiterentwickelt werden.

Zum Suchfeld mit Autovervollständigung (Abbildung 4.3 auf Seite 52) wurde angemerkt, dass es wichtig ist, aussagekräftige Symbole zu verwenden. Die kombinierte Darstellung aus den letzten Suchanfragen und der Autovervollständigung wurde als sinnvoll erachtet.

4.4 ZWEITE ITERATION

4.4.1 *Problemstellung*

Das Ziel der nächsten Iteration ist es, einen lauffähigen Prototyp zu entwickeln, der als High-Fidelity-Prototyp dient und dann in weiteren Iterationen stetig weiterentwickelt wird, um den Benutzern die Funktionalitäten innerhalb einer verwendbaren Anwendung anzubieten und dadurch besseres Feedback zu erhalten. Der Protoyp sollte eine Volltextsuche umfassen, Filterfunktionen bereitstellen und die Suchergebnisse darstellen. Dabei werden die Sketches aus der letzten Iteration sowie das darin gewonnene Anwenderfeedback als Ausgangsbasis genutzt.

Einige Anforderungen wurden für die Auswahl der Technologie und Entwicklung des Prototyps getroffen. Diese sind nicht von den identifizierten

Problemstellungen abhängig, sondern beziehen sich auf das Vorgehensmodell, welches für diese Arbeit gewählt wurde. Damit soll gewährleistet werden, dass innerhalb einer möglichst kurzen Zeitspanne mehrere Iterationen des Prototyps ausgeliefert und evaluiert werden können.

- Das Frontend soll Data-Driven entwickelt werden und die Verwendung von Templates, die mit Daten gefüllt werden, soll möglich sein. Dies sorgt für eine hohe Flexibilität bei der Frontendgestaltung und ermöglicht kleinere Anpassungen auf einfache Weise.
- Das Projekt sollte bei der Entwicklung Hot Reload unterstützen oder zumindest sehr schnell auslieferbar sein. Somit können kleinere Anpassungen am Frontend mit Workshopteilnehmern live vorgenommen werden, um sofortiges Feedback zu erhalten. Dieses Vorgehen ist inspiriert von Holstein und Störl, welche per Livecoding mit der Browserkonsole in Workshops Funktionalitäten und Möglichkeiten aufzeigten [20, S.15].
- Es sollen realistische Testdaten (z. B. Kopie der Produktivdatenbank) verwendet werden.
- Das Backend bzw. die Datenbank soll eine möglichst große Detailtiefe der Daten bereitstellen. Durch Template-Ansatz und Hot Reload im Frontend kann die Erweiterung der Darstellung mit hoher Flexibilität durchgeführt werden.
- Für Remotetests soll der Prototyp über das Internet erreichbar sein.

4.4.2 *Develop*

Komponenten

Die grundlegende Architekur eines Information Retrieval Systems wurde in Abschnitt 2.2 auf Seite 5 bereits beschrieben. Daraus können die verschiedenen Komponenten abgeleitet werden, die für den Prototyp benötigt werden. Eine Webapplikation besitzt typischerweise eine Drei-Schichten-Architektur, die aus Browser, Applikationsserver und Datenbank besteht. In welcher Schicht die Komponenten jeweils umgesetzt werden sollen, wird in Abbildung 4.6 auf der nächsten Seite dargestellt. Grau markiert ist der Webservice, da dieser für ein übliches IR System nicht benötigt wird, aber in einer Webapplikation als Schnittstelle zum Frontend erforderlich ist.

Datenbank und Backend

Für die Datenbank wird auf PostgreSQL gesetzt. Wie Silva in seinem Vergleich der Volltextsuche eines konventionellen IR Systems mit den standardmäßig zur Verfügung stehenden Volltextsuchfunktionalitäten der PostgreSQL aufzeigt, kann die Volltextsuche von PostgreSQL die üblichen Anforderungen eines IR Systems abbilden [49, S. 126]. Wie in den Grundlagen aufgezeigt, bietet PostgreSQL die Funktionalitäten zur Normalisierung von Texten

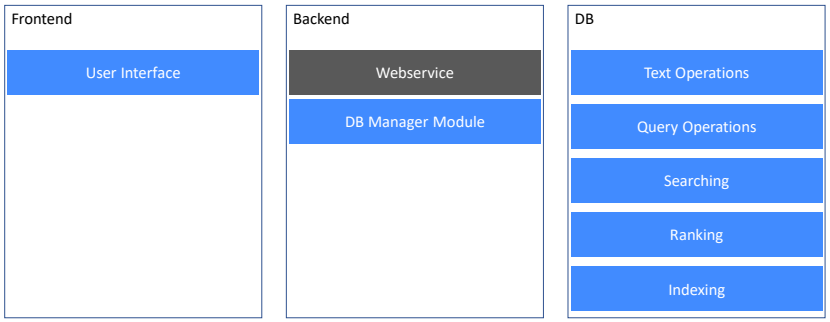


Abbildung 4.6: Komponenten der Webapplikation

(*Text Operations*). Die Normalisierung muss sowohl auf die Dokumente als auch auf die Anfrage des Benutzers ausgeführt werden. Die Ausführung der Volltextsuche, unter Berücksichtigung weiterer Einschränkungen (*Query Operations*), wird mit einer SQL-Anfrage durchgeführt. Das *Ranking* kann ebenfalls durch PostgreSQL abgebildet werden. Zur Verbesserung der Suchperformance unterstützt PostgreSQL auch das *Indexing* der normalisierten Dokumente.

Zur Entwicklung des Prototyps wurde vom Projekt ein Datenbank-Image bereitgestellt. Für den High-Fidelity-Prototyp sind keine Anpassungen am Datenmodell erforderlich. Somit kann das Datenbank-Image einfach importiert werden und die Datenbank muss nicht verändert werden.

In der Webapplikation wird ein Backend benötigt, welches die Anfragen aus dem Frontend entgegennimmt und an die Datenbank zur Verarbeitung weitergibt. Die Schnittstelle zwischen Frontend und Backend bildet der Webserver, die Schnittstelle zur Datenbank das *DB Manager Module*. Als Applikationsservertechnologie wird auf *Node.js* [31] gesetzt, als Web Framework wird *Express* [10] genutzt. Der Applikationsserver soll im Prototyp nur die Anfragen vom Frontend entgegennehmen und an die Datenbank durchreichen, daher wurde hier auf eine dynamische typisierte Programmiersprache gesetzt.

Für die Suche wird vom Backend der API-Endpunkt */search* bereitgestellt. Die Queryparameter werden im *Body* übertragen und bestehen im einfachsten Fall aus der Volltextquery. Die Suche wird ausgeführt, indem ein *POST*-Request durch das Frontend ausgeführt wird. Das Backend erstellt anschließend die SQL-Anfrage mit den gewählten Parametern und übergibt diese zur Ausführung an das *DB Manager Module*.

Die SQL-Anfrage nutzt die entsprechenden Komponenten für *Text Operations*, *Query Operations* und das *Ranking*. Die *Text Operations* und *Query Operations* werden mit der *WHERE*-Bedingung in Listing 4.1 auf der nächsten Seite abgebildet. Zur Erstellung der *ts_query* wird die Funktion *unaccent* sowie *websearch_to_tsquery* eingesetzt. Die letztere Funktion nutzt eine ähnliche Syntax wie Web-Suchmaschinen [38]. Es soll evaluiert werden, ob das

für die Anwender hilfreich ist. Wie man Zeile 2 entnehmen kann, wird im Prototyp zur Laufzeit der Query der *ts_vector* für alle Dokumente erzeugt.

Bei SQL Anfragen, in denen Benutzereingaben direkt übernommen werden, besteht das Risiko einer SQL-Injection [47, S. 238f]. Hier sei darauf hingewiesen, dass die Variable $1 durch die Datenbank-Library *pg-promise* für PostgreSQL sicher übergeben wird [64]. Die Funktion *websearch_to_tsquery* ist für direkte Benutzereingaben vorgesehen und erzeugt, unabhängig von den Eingaben, keine Syntaxfehler [38].

```
1 websearch_to_tsquery('german', unaccent($1)) @@
2 to_tsvector('german', unaccent(documentcontentcleaned))
```

Listing 4.1: Die WHERE Bedingung für Volltextsuche

Für eine größtmögliche Flexibilität bei der Gestaltung des Frontends, sollen alle Informationen der Ergebnisse übertragen werden, nicht nur die aktuell benötigten Metainformationen. Somit kann der Prototyp durch Anpassungen des Templatecodes ohne Anpassung des Backends erweitert werden, um beispielsweise zusätzliche, von Anwendern gewünschte Informationen anzuzeigen. Dazu wird vom Backend ein Array geliefert, welches pro Dokument ein Objekt mit allen potentiell benötigten Informationen für die Darstellung der Ergebnisse enthält.

Da einige Daten im Datenmodell normalisiert abgelegt sind, müssen sie zunächst per Join angereichert werden. Dabei gibt es auch Beziehungen, welche eine Kardinalität vom Dokument zur Entität von *1:n* besitzen. Dies muss beim Join berücksichtigt werden, sodass alle Einzelsätze zu einem Dokument aggregiert werden. Dies ist beispielsweise beim Adressaten der Fall, denn eine Korrespondenz kann an mehrere Personen adressiert sein. Das Modell eines Dokuments enthält alle Informationen, die im Ergebnis angezeigt werden könnten.

Um die Suchtreffer im Kontext zu bestimmen, kann *ts_headline* genutzt werden. Für den ersten Prototyp werden hier feste Konstanten eingesetzt. Eine mögliche Evaluation für spätere Phasen wäre, welche Werte hier sinnvoll wären oder wie man sie durch den Benutzer konfigurierbar gestalten könnte. Diese Problemstellung wird in PS-25 weiterverfolgt. Wie man dem Ausschnitt in Listing 4.2 entnehmen kann, wird der Funktion *ts_headline* direkt das Dokument übergeben. Als *FragmentDelimiter* wird *NFRGMNT* verwendet, da hier eine Zeichenkette genutzt werden sollte, die im Korpus nicht vorkommt. Der Rückgabewert von *ts_headline* ist *text*. Das bedeutet, mehrere Treffer werden durch den *FragmentDelimiter* unterteilt. Für die Anzeige soll aber jeder Treffer einzeln vorliegen, daher wird im Backend für jedes Ergebnis ein Array erstellt, welches anhand der Zeichenkette *NFRGMNT* alle Treffer als eigenen Text zurückgibt.

```
1 ts_headline('german', documentcontentcleaned, websearch_to_tsquery('
    german', ($1)), 'MaxFragments=10, MaxWords=14, MinWords=3, StartSel
    ="<b><u>", StopSel="</b></u>", FragmentDelimiter=NFRGMNT') as
    headlines
```

Listing 4.2: Bestimmung des Treffers im Kontext

Das Ranking der Treffer wird mit *ts_rank* bestimmt. Im ersten Prototyp wird der *ts_vector* hier auch während der Laufzeit der Query für alle Dokumente erzeugt.

```
ts_rank(to_tsvector('german', unaccent(documentcontentcleaned)),
    websearch_to_tsquery('german', unaccent($1))) as rank
```

Listing 4.3: Ranking

Neben den Dokumenten die zu einer Suchanfrage passen, wird noch eine zweite Liste zurückgegeben. Diese enthält alle eindeutigen Werte der referenzierten Entitäten, die in mindestens einem der Treffer vorkommen. Diese Liste wird zur Umsetzung der kaskadierenden Filter benötigt, da im Filter nur die Entitäten vorgeschlagen werden sollen, die auch in der Ergebnismenge vorhanden sind.

Frontend

Das Frontend des Prototyps wird mit *Vue.JS* [58] und den Komponenten aus *Vue Material* [57] umgesetzt.

Vue.JS bietet ein Komponentensystem, welches die Entwicklung eigener Komponenten erlaubt. Mit *two-way data binding* können Daten zwischen dem Model und dem HTML-Template verknüpft werden. Die *Vue.JS* Komponenten setzen auf deklaratives Rendering mit einer Template-Syntax. Das bedeutet, dass deklariert werden kann, wie das Model dargestellt werden soll und die Erzeugung der Elemente erfolgt dann zur Laufzeit im Browser.

Das Frontend setzt sich aus zwei eigenen Komponenten zusammen, dem *SearchView* zur Formulierung der Suchanfrage und der *ResultComponent*, welche die Ergebnisse darstellt.

Die *SearchView* Komponente besitzt dabei ein Eingabefeld und einen Button zum Start der Suche. Die Methode zur Ausführung der Suche wird beim Klick auf den Suchen-Button sowie bei Bestätigung mit *enter* aufgerufen. Hierbei wird der *POST*-Request an das Backend geschickt. Die vom Backend gelieferten Ergebnisse werden im Model der *SearchView* abgelegt und durch das Data Binding an die *ResultComponent* übergeben.

Wie das Data Binding das Prototyping vereinfachen kann, soll anhand des gekürzten Templatecodes aus der *ResultComponent* verdeutlicht werden (siehe Listing 4.4). Das Beispiel zeigt, wie die deklarative Beschreibung funktioniert. In Zeile eins wird mit *v-for* definiert, dass für jedes Objekt im Array *result*, eine HTML Element vom Typ *md-card* erzeugt wird. Auf dieses Suchergebnis (*item*) kann dann innerhalb des Elements zugegriffen werden. Wie man in Zeile vier sieht, kann man die *v-for* Direktive auch schachteln. Somit werden im konkreten Beispiel alle Ko-Autoren als eigener *md-chip* dargestellt.

```
<md-card slot="md-table-row" v-for="item in result">
  <div class="md-helper-text">Autor*in</div>
  <md-chip class="md-primary">{{ item.author.firstname + " " + item.
      author.surname }}</md-chip>
```

```
4   <md-chip v-for="coauthor in item.coauthors" v-bind:key="'coauth_' +
        item.id + '_' + coauthor.id">
5     {{ coauthor.firstname + " " + coauthor.surname }}
6     <md-tooltip>Ko-Autor*in</md-tooltip>
7   </md-chip>
8 </md-card>
```

Listing 4.4: SearchView Template

Der lauffähige Prototyp, wie er in der ersten Iteration entstanden ist, wird in Abbildung 4.7 dargestellt. Hier kann auch die Darstellung des Ko-Autoren nachvollzogen werden, denn nur wenn eine Korrespondenz einen oder mehrere Ko-Autoren besitzt, wird dieser dargestellt.

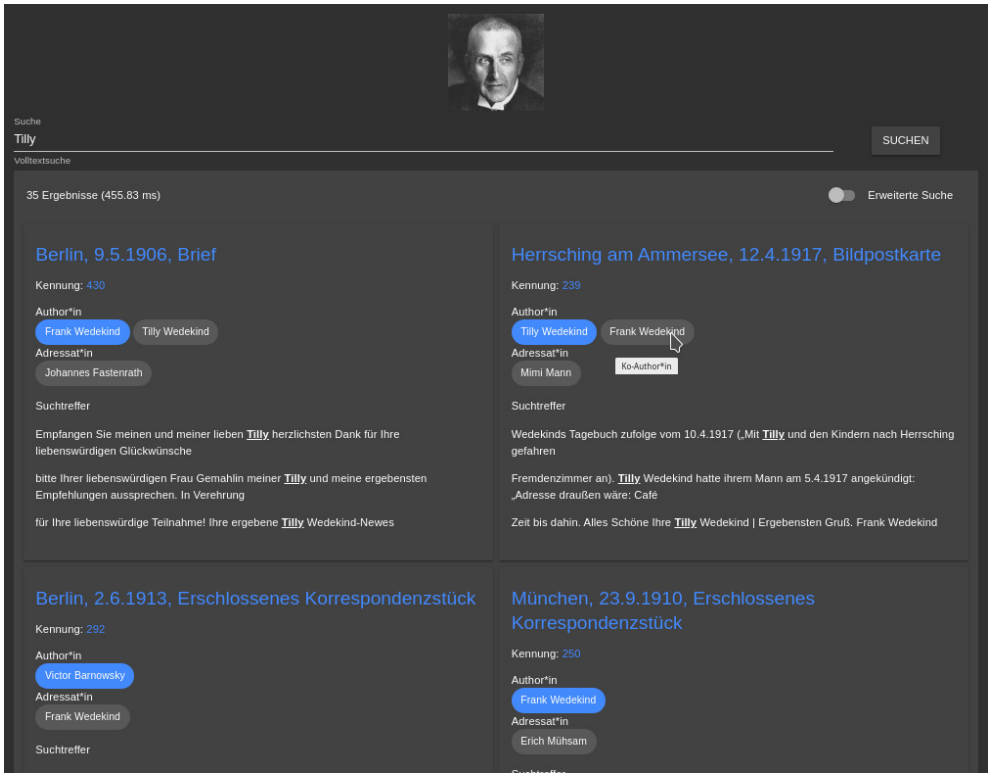


Abbildung 4.7: Screenshot des ersten High-Fidelity-Prototyps (zweite Iteration Versionstag 0.1)

Deployment

Wie in den Anforderungen dieser Iteration beschrieben, soll der Prototyp über das Internet erreichbar sein. Für ein einfaches Deployment kann das Projekt als Docker-Setup gestartet werden.

Im Repository des Projekts ist alles vorhanden, was für die Lauffähigkeit des Prototyps benötigt wird. Es fehlen nur zwei Objekte, die nicht eingecheckt werden können. Zum einen ist dies das *database_binary*, welches geschützte Daten enthält, daher muss es separat bereitgestellt werden. Zum

anderen müssen für die Konfiguration der Datenbank Umgebungsvariablen des Betriebssystems gesetzt werden. Dazu kann das Beispielfile *envexample* nach *.env* kopiert werden.

Die Dockerumgebung besteht aus zwei Containern, die mithilfe von *docker-compose* gestartet werden können. Der PostgreSQL Container stellt die Datenbank zur Verfügung. Wenn die Datei *database_binary* im root des Projekts ablegt wird, wird dieses automatisch als Volume in den Container eingehängt und beim ersten Start im *init-database.sh* Skript importiert.

Der zweite Container enthält den Applikationsserver, der sowohl das Backend als auch das Frontend ausliefert. Der Server ist auf dem lokalen Port *8337* erreichbar.

Um die Applikation aus dem Internet zu erreichen, muss sie über einen Reverse-Proxy öffentlich verfügbar gemacht werden. Dabei wird eine Passwort-Abfrage vorgeschaltet, sodass keine Verwechslung zum eigentlichen Wedekind-Projekt stattfinden kann. Außerdem werden Verbindungen durch den Reverse-Proxy mit einem HTTPS-Zertifikat ausgeliefert.

Neben dem produktiven Deployment des Prototyps, gibt es auch die Möglichkeit eine Entwicklungsumgebung lokal zu betreiben. Details hierzu können der *README.md* des Projekts entnommen werden.

4.4.3 *Deliver*

Der Prototyp wurde wie beschrieben passwortgeschützt online gestellt und mit der Forschungsgruppe geteilt, mit dem Hinweis, dass es sich hierbei um einen ersten Prototyp handelt.

Das Feedback zum Prototyp ergab, dass noch nicht ersichtlich ist, welche Funktionalitäten von der Volltextsuche bereitgestellt werden. Hierbei wurde der Vorschlag gemacht, eine Seite mit Release Notes einzufügen, sodass erkenntlich ist, welcher Funktionsumfang ergänzt wurde. Darüber hinaus wurde festgestellt, dass der Prototyp eine gute Ausgangsbasis darstellt.

Positiv aufgefallen ist die Vorschau der Suchergebnistreffer, welche im Kontext angezeigt werden.

Zuletzt wurde noch ein Problem bei der Datenqualität erkannt, welches zu unerwarteten Ergebnissen führt. Hierbei müssen bestimmte Datensätze ausgefiltert werden, wenn ihre Gültigkeit abgelaufen, oder das Flag *isPublishedInDb* nicht *WAHR* ist. Dieses Feedback wird in der nächsten Iteration berücksichtigt.

4.5 DRITTE ITERATION

4.5.1 *Problemstellung*

Das Ziel der nächsten Iteration ist es, die erkannten allgemeinen Probleme aus der zweiten Iteration zu beheben und den Prototyp um die Filtermöglichkeit (PS-02), die Speicher-, Referenzier- und Navigierbarkeit (PS-09) zu erweitern (Beschreibung siehe Abschnitt 3.6 auf Seite 36). Dazu werden ver-

schiedene Optionen entwickelt, um die Problemstellungen zu lösen und anschließend durch Anwender evaluiert. Außerdem soll geprüft werden, inwiefern die Performance der Suche verbessert werden kann.

4.5.2 *Develop*

Behebung allgemeiner Probleme aus der letzten Iteration

Zunächst sollen allgemeine Probleme behoben werden, auf die in der letzten Iteration hingewiesen wurde. Eines bestand darin, eine Release Seite einzuführen, um zur Verfügung stehende und neue Funktionalitäten darzustellen. Dazu wurde eine weitere Seite hinzugefügt, auf die von der Suchseite aus verlinkt wird. Damit diese nicht zu präsent platziert, aber dennoch einfach zu finden ist, wurde sie im Footer verlinkt. Zusätzlich wurde in den Footer eine Versionsnummer aufgenommen. Die Release-Seite zeigt die Funktionalitäten an, die je Version erweitert wurden und von ihr aus ist die Navigation zurück zur Suche möglich.

Ein weiteres allgemeines Thema ist die falsche Anzeige aufgrund von Problemen mit der Datenqualität. Es führte dazu, dass in den Ergebnissen teilweise Testdaten angezeigt werden. Um dieses Problem zu beheben, wurde die Datenbankabfrage so erweitert, dass gültige Entitäten nicht mehr einbezogen werden, indem die Spalte *valid-to* im Filter berücksichtigt wird. Dies führte allerdings zu einem weiteren Problem: teilweise sind Entitäten als abgelaufen markiert, obwohl diese weiterhin verwendet werden. Dies bewirkt, dass für vereinzelte Entitäten kein Text angezeigt werden kann. Für den Prototyp wird dieses Problem zunächst in Kauf genommen und als noch zu lösende Problemstellung vermerkt. Im Hauptprojekt wurde ein Ticket für dieses Problem mit einer Analyse der betroffenen Entitäten erstellt.

Bereitstellung von Filtern (PS-02)

Eine der bearbeiteten Problemstellungen ist die Entwicklung der Kombination aus Entitäts- und Volltextsuche mit einem möglichst einfachen Userinterface (PS-02). Die Idee zur Lösung ist die Bereitstellung von Filtermöglichkeiten nach der Ausführung einer Volltextsuche, sodass zusätzlich auf Entitäten gefiltert werden kann. Um die Eingabe zu vereinfachen, sollen nur Werte auswählbar sein, die auch in den Suchtreffern vorhanden sind.

Aus dem Feedback der ersten Iteration bestand bei den Befragten eine Tendenz zur Umsetzung der Filter mit der Variante wie in Abbildung 4.5c auf Seite 54 dargestellt. Dazu wurde Filter für mehrere Entitäten nebeneinander unter dem Suchfeld und über der Liste der Ergebnisse implementiert. Bei einer großen Zahl von Ausprägungen in der Filterliste tritt allerdings das Problem auf, dass beim Klick auf *mehr anzeigen*, um die angezeigte Liste zu erweitern, die Suchergebnisse immer weiter nach unten verschoben werden. Dadurch wurde die Seite sehr unübersichtlich und die Filter waren mit vielen Listenelementen sehr schwer zu bedienen. Daher wurde als alternative Lösung eine Variante angelehnt an Abbildung 4.5b auf Seite 54 gewählt, mit

der die Ausprägungen des Filters über ein Dropdown geöffnet werden und die zu filternden Elemente über eine Checkbox ausgewählt werden können. Dadurch wird weiterhin die Auswahl von mehreren Ausprägungen ermöglicht, ohne dass die Darstellung der anderen Seitenelemente beeinflusst wird. Zusätzlich hat diese Lösung den weiteren Vorteil, dass die Anzeige auf mobilen Endgeräten besser ist, da der Filter ein- und ausblendbar ist und dadurch weniger Platz auf der Seite benötigt wird.

Die technischen Anforderungen zur Umsetzung dieser Funktionalität besteht sowohl in der Erweiterung des Frontends als auch des Backends. Das Frontend muss hierbei um die Filterauswahlen erweitert werden, sowie die Übergabe der gewünschten Filter an das Backend muss implementiert werden. Im Applikationsserver muss die Datenbankabfrage um die Einschränkung der Filter erweitert werden.

Speicher-, Referenzier- und Navigierbarkeit von Suchergebnissen (PS-09)

Eine weitere betrachtete Problemstellung besteht in der Verbesserung der Usability in Bezug auf die Speicher-, Referenzier-, und Navigierbarkeit der Suchergebnisse. Ziel ist es, den Anwendern zu ermöglichen, Bookmarks ihrer Suchen zu speichern oder diese mit anderen Personen zu teilen. Zusätzlich soll eine Zurücknavigation möglich sein, sodass Benutzer nachdem sie auf ein Dokument abgesprungen sind, wieder zurück zu ihrer ursprünglichen Suche springen können, ohne dass sie die Suchparameter neu eingeben müssen.

Zur Umsetzung dieser Anforderungen muss nicht nur die Eingabe in die Volltextsuche, sondern auch die Auswahl der Filterwerte in der URL kodiert werden. Dies erfordert es, den Zustand der Suchanfrage an einer zentralen Stelle zu halten und die URL bei jeder erfolgten Änderung (z. B. Filterwert wechseln) anzupassen. Da es sich um eine Single-Page-Webapplikation[2] handelt, muss die Verarbeitung der URL Parameter im Frontend durchgeführt werden. Das bedeutet, wenn Suchparameter in der URL gesetzt sind, muss die Suche beim Öffnen ausgeführt werden. Im Ausblick auf weitere Iterationen ist davon auszugehen, dass noch weitere Parameter hinzukommen werden.

Wenn in einer Applikation mehrere Komponenten von einem gemeinsamen Zustand abhängig sind und in der Lage sind, diesen Zustand zu manipulieren, sollte der Zustand zentral verwaltet werden. Mit einem State-Management-Pattern kann es gewährleistet werden, dass der Zustand zentral verwaltet wird und alle Komponenten über Aktualisierungen informiert werden, auch wenn sie selbst diese nicht ausgelöst haben. *Vuex* folgt dem State-Management-Pattern und bietet eine tiefe Integration mit der Reaktivität von *vue.js* an. Dabei wird der gesamte Zustand in einem einzelnen *State*-Objekt abgelegt, welches als „Single Point of Truth“ den Status der Applikation enthält. [59, 60]

2 Bei einer Single-Page-Webapplikation wird nur eine HTML Seite geladen, die Navigation erfolgt vollständig durch client-seitige Anpassung ohne ein Neuladen der Seite. [7]

Der Zustand des Frontends soll im zentralen *State* abgelegt werden. Dieser wird wie in Listing 4.5 aufgesetzt. Bevor eine Anfrage ausgeführt wird, wird diese in der *prepareQuery* vorbereitet. Somit kann die Anfrage zentral zusammengestellt werden, bevor sie an den Server geschickt wird. Die *prepareQuery* wird dabei von unterschiedlichen Komponenten des Frontends zusammengesetzt. In weiteren Iterationen bleibt somit die Businesslogik an der zentralen Stelle und die Komponenten enthalten lediglich notwendige Logik zur Darstellung und Interaktion mit dem Zustand. Die Komponenten des Frontends können somit flexibel geschachtelt, verschoben oder ersetzt werden. Außerdem können mehrere unterschiedliche Komponenten gleiche Teile der Abfrage manipulieren.

```
1  const state = {
2      // prepare the search. On fetchSearchResults it will be used to
           query the db
3      prepareQuery: {
4        fulltextquery: "",
5        entitiesquery: {}
6      },
7      searchquery: null,  // holds the Query which was used to fetch the
           latest results
8      searchstate: 0, // 0 = initial, 1 = loading, 2 = results fetched
9      searchResults:[],
10     foundEntities:[],
11     runtimeInfo:{}, // performance keyfigures
12     serverError: false
13 };
```

Listing 4.5: *State* des Frontends

Der Zustand wird von den Komponenten nicht direkt manipuliert, sondern kann nur mit *Mutations* verändert werden. Eine *Mutation* besteht aus einer Handlerfunktion, welche den *State* verändert und kann optional einen Payload erhalten. Sie wird weiterhin immer synchron ausgeführt. [61] Für asynchrone Operationen werden *Actions* verwendet, diese verändern den *State* nicht aktiv, sondern müssen eine *Mutation* aufrufen, können dafür aber asynchrone Operationen enthalten. [62]

Mit einer *Action* wird die vorbereitete Suchanfrage ausgeführt und die URL-Parameter werden aktualisiert. Die *Action fetchSearchResults* ist in Listing 4.6 auf der nächsten Seite dargestellt. Da die vorbereitete Anfrage bereits im *State* vorliegt, besitzt sie keinen zusätzlichen Payload. Aus der vorbereiteten Query wird der HTTP-Request an das Backend vorbereitet. Wenn tatsächlich eine Suchanfrage vorliegt, wird diese ausgeführt, andernfalls wird der Zustand der Suche auf initial gesetzt. Bevor der HTTP-Request ausgeführt wird, wird der *searchstate* auf 1 (Ladezustand) gesetzt. Sobald die Antwort des Servers vollständig empfangen wurde, wird der Ladezustand beendet und die Ergebnisse werden im *State* mit den *Mutations* abgelegt. Durch die Reaktivität aktualisieren sich daraufhin alle Komponenten auf den neuen Zustand. Zuletzt wird die Suchanfrage in der URL kodiert, sodass sie

den aktuellen Zustand widerspiegelt und ein Teilen und Navigieren dieser ermöglicht wird.

```
1  async fetchSearchResults({commit, state}) {
2    const {fulltextquery, entitiesquery} = query;
3    const searchQuery = {};
4    if (fulltextquery !== null) {
5      searchQuery.fulltextquery = fulltextquery;
6    }
7    if (entitiesquery !== null) {
8      const selectedKeys = helpers.selectedKeys(entitiesquery);
9      if (selectedKeys.length>0) searchQuery.entitiesquery = selectedKeys;
10   }
11   // no query defined / reset search to initial
12   if (Object.keys(searchQuery).length === 0) {
13     commit('setSearchstate',0);
14     return;
15   }
16   commit('setSearchstate',1); // start loading
17   const httpResponse = await HTTP.post('search', searchQuery);
18   commit('setSearchstate',2); // finished loading
19   commit('setSearchQuery', query);
20   commit('setSearchResults', httpResponse.data.searchResults);
21   commit('setFoundEntities', httpResponse.data.foundEntities);
22   router.push({ path: 'search', query: searchQuery })
23 }
```

Listing 4.6: Ausführen der Suche mit *fetchSearchResults*

Index auf Datenbank

Es wird ein Datenbankindex angelegt, um die Volltextsuche zu beschleunigen. Hierzu muss zunächst eine neue Datenbankspalte erzeugt werden, um die Dokumente nicht mehr zur Laufzeit in die *ts_vector* Darstellung umzuwandeln. Die durchzuführenden Anpassungen am Datenmodell des Prototyps sollen mit der Version eingecheckt werden, da das Datenbankimage nicht mit eingecheckt werden kann. Alle Anpassungen am Datenmodell werden für den Prototyp idempotent definiert. Dadurch kann bei jedem Start des Prototyps die Migration ausgeführt werden. Wenn also eine neue Version eingespielt wird oder die Testdaten erneut importiert wurden, führt der Server beim Start die Migration aus und somit sind alle Anpassungen am Datenmodell direkt verfügbar. Gleichzeit ermöglicht dies, in der Versionshistorie des Prototyps die Anpassungen am Datenmodell (im Vergleich zur Produktivversion des Datenmodells der Edition) nachzuvollziehen.

Für die Erstellung der persistierten, vektorisierten Darstellung des Dokuments kann eine *generated column* verwendet werden. Die *generated column* ist eine Spalte, die immer vom Wert einer anderen Spalte abgeleitet wird. Mit dem Schlüsselwort *stored* kann PostgreSQL angewiesen werden, die Spalte zu persistieren. Das Verhalten ist ähnlich zu einem materialisierten View, nur dass die *generated column* automatisch aktualisiert wird, wenn die abhängi-

ge Spalte aktualisiert wird. Hier kann also eine generierte Spalte auf dem Dokument angelegt werden. Die Ableitungsfunktion der generierten Spalte muss dabei *immutable* sein und darf sich nur auf Spalten der aktuellen Zeile beziehen. [41]

unaccent wird von externen Konfigurationsdateien beeinflusst, deshalb kann es per Definition nicht *immutable* sein [22]. Zu Problemen führt dies aber nur, wenn die Konfiguration abgeändert wird. Für den Prototyp ist daher ein möglicher Workaround, eine eigene Funktion anzulegen und diese als *immutable* zu deklarieren.

Für den Produktivbetrieb birgt das allerdings das Risiko, dass bei Änderung der externen Dateien oder des Suchpfades die generierte Spalte und der Index nicht mehr korrekt wären, da die Ableitungsregeln nicht erneut angewendet werden. Für den Prototyp wird dies vernachlässigt und die Redefinition in Listing 4.7 wird vorgenommen.

```
1 CREATE OR REPLACE FUNCTION public.f_unaccent(text)
2   RETURNS text AS
3 $func$
4 SELECT public.unaccent('public.unaccent', $1)
5 $func$ LANGUAGE sql IMMUTABLE PARALLEL SAFE STRICT;
```

Listing 4.7: Definition einer eigenen immutable *unaccent* Funktion

Diese Funktion kann dann genutzt werden, um eine generierte Spalte anzulegen und den GIN-Index darauf zu definieren. Der GIN-Index (generalized inverted index) ist zur Indexierung der *tsvector* Spalten geeignet, da er mehrere unterschiedliche Ausprägungen pro Feld unterstützt. Pro Lexem werden die Zeilen-IDs im Index gespeichert, in denen das Lexem vorkommt. Dabei werden Zeilen-IDs mehrfach im Index auftauchen, wenn sie mehr als ein Lexem enthalten. [16] In Listing 4.8 wird die Erstellung der generierte Spalte und das Anlegen des Index gezeigt.

```
1 ALTER TABLE "document" ADD COLUMN documentcontentcleaned_tsvector
      tsvector GENERATED ALWAYS AS (to_tsvector('german'::regconfig,
      f_unaccent(documentcontentcleaned))) STORED;
2
3 CREATE INDEX textsearch_idx ON "document" USING GIN (
      documentcontentcleaned_tsvector);
```

Listing 4.8: Erstellung des Indexes

Das Ergebnis dieser Iteration wird in Abbildung 4.8 auf der nächsten Seite dargestellt. Referenzierte Metadaten können als Chips optional eingeblendet werden, hier ist beispielsweise der Ort mit eingeblendet. Außerdem funktioniert der Entitätsfilter und es kann auf in den Dokumenten referenzierte Entitäten gefiltert werden. Im Beispiel ist die Selektion nach Personen geöffnet. Hierbei wird eine Mehrfachauswahl unterstützt. Alle Parameter der Suchanfrage werden in der URL kodiert und werden beim Öffnen des vollständigen Links ausgeführt. Dies ermöglicht die Speicherung als Bookmark, das Teilen des Links sowie die Zurücknavigation.

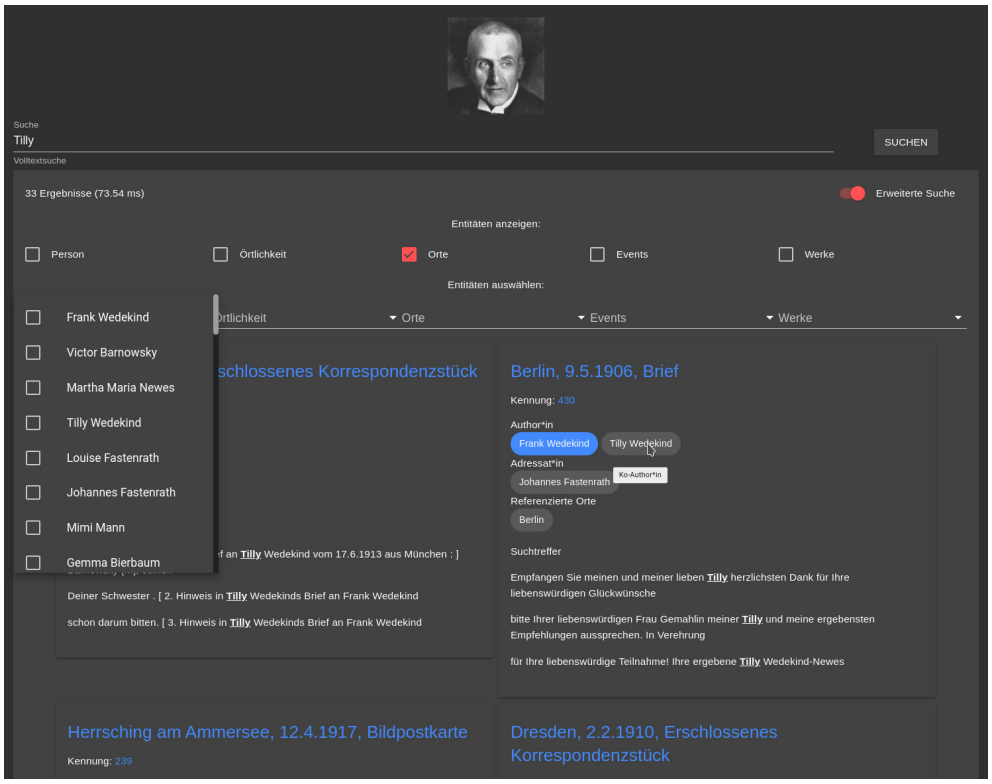


Abbildung 4.8: Screenshot des weiterentwickelten High-Fidelity-Prototyps (dritte Iteration Versionstag 0.4)

4.5.3 *Deliver*

Um Nutzerfeedback einzuholen, wurden Guerrilla Tests mit mehreren Benutzern auf unterschiedlichen Endgeräten durchgeführt. Das Testskript bestand in den nachfolgenden Anweisungen:

- Gehen Sie bitte auf die Seite[3] und geben Sie das Passwort ein
- Bitte führen Sie einige Suchanfragen durch. Achten Sie bitte darauf, was Ihnen positiv oder negativ auffällt, was unklar ist und was man vielleicht noch verbessern könnte
- Notieren Sie bitte, was Ihnen positiv oder negativ aufgefallen ist, was unklar war und was man vielleicht verbessern könnte. Bitte erwähnen Sie auch Unklarheiten, die sich ausgeräumt haben.

Das Feedback nach den Tests war deutlich umfangreicher als das der bisherigen Iterationen und bezog sich sowohl auf allgemeine Themen als auch auf die Verhaltensweise der Funktionalitäten. Im Allgemeinen wurde die Darstellung durch die schlichte Darstellung und farbliche Gestaltung positiv bewertet und als seriös empfunden. Ein mit digitalen (historisch-kritischen) Editionen unerfahrener Benutzer merkte an, dass ihm beim Einstieg die zur

3 Link zum Prototyp

Verfügung stehenden Funktionalitäten unklar waren und nannte als Verbesserungsmöglichkeit Hilfestellungen wie die Anzeige von Suchvorschlägen oder Autocompletion beim Tippen.

Einige Rückmeldungen gab es zum Verhalten der Volltextsuche. Positiv wurde die nicht-exakte Suche bewertet, sodass auch ähnliche Begriffe gefunden wurden. Bei der Suche nach Stoppworten verwirrte das Verhalten der Suche, da diese entfernt wurden und die Suche dadurch keine Treffer lieferte. Der Anwender wurde darüber nicht informiert und daher war das Verhalten für diesen nicht transparent. Zusätzlich wurde darauf hingewiesen, dass es gegebenenfalls notwendig wäre, auch nach solchen entfernten Wörtern wie z. .B Anreden zu suchen.

Zu den angezeigten Informationen zur Suche gab es Verbesserungsvorschläge. Das war zum einen die Anzeige der Trefferanzahl innerhalb eines Dokuments, zum anderen ein Hinweis zum Vorkommen von nicht-exakten Treffer in der Ergebnismenge. Zusätzlich wurde angemerkt, dass die Ergebnisse nach ihrer Relevanz sortiert werden sollten und dabei nicht-exakte Treffer niedriger priorisiert werden sollten.

Viele Anregungen gab es zu der Filterfunktionalität. Dabei wurden zum einen zusätzliche Funktionalitäten als wünschenswert genannt, wie der Einstieg der Suche über einen Filter, das Zurücksetzen einzelner Filter oder die Suche nach der Rolle einer Person. Die Benennung eines Filters führte zu Verwirrung bei den Anwendern, da der Filter für die im Dokument referenzierten Personen mit der Überschrift *Person* benannt wurde. Da Autor und Empfänger ebenso Personen sind, war den Anwendern unklar, worauf sich der Filter bezog und die Filterwerte sowie die Ergebnisse waren nicht intuitiv nachvollziehbar.

Dass die Filter in der Suchanfrage mit einer UND-Verknüpfung abgefragt werden, führte ebenso zu unverständlichen Suchtreffern. Von einigen Anwendern wurde eine ODER-Verknüpfung erwarteten.

Weitere Anmerkungen bezogen sich auf die Anzeige der Ergebnisse und es wurden Vorschläge zur optischen Hervorhebung (z. .B fette Schrift für Überschriften) und Darstellung der Ergebniskarten gemacht. Weiterhin störten die leeren Textboxen, die durch Entitäten ohne Beschreibung erzeugt wurden, was durch das bereits genannte Problem mit der falsch gesetzten Gültigkeit verursacht war.

Die farbliche Markierung fiel einerseits positiv auf, andererseits war einem Anwender die Semantik der Markierung (in dem Fall der (Haupt-)Autor) nicht klar. Zusätzlich erwünschte Funktionen waren die Konfiguration der Länge des angezeigten Textausschnittes des Ergebnisses sowie die Anzeige der Position des Treffers im Text.

Weiterhin wurden von den Anwendern zusätzliche Funktionen zum schnelleren Navigieren und Filtern als potentiell hilfreich empfunden. Beispielsweise war ein Absprung an die Textstelle mit dem Treffer oder ein automatisches Filtern beim Klick auf eine Entitätsausprägung erwünscht.

4.6 VIERTE ITERATION

4.6.1 *Problemstellung*

Da diese Iteration parallel zu der letzten Iteration gestartet wurde, sei hier nochmal auf Abbildung 4.2 auf Seite 50 verwiesen. Die Ergebnisse aus der letzten Iteration lagen hier also noch nicht als Input vor.

Ziel der vierten Iteration ist es, nutzerzentriert Lösungen zu den identifizierten Problemstellungen zu entwickeln. Die Problemstellungen aus Abschnitt 3.6 auf Seite 36 wurden während der Iteration priorisiert. Detailliert betrachtet wurden dabei die Entitätsfilter in unterschiedlichen Rollen (PS-04), die Zeitselektion (PS-05), die Suche in Kommentaren (PS-06), die flexible Fehlertoleranz (PS-08), der Vorschlag alternativer Suchbegriffe (PS-10), die Zusammenstellung von Dokumenten (PS-11), die Erschließung von Hintergrundinformationen (PS-12) sowie die Phrasensuche (PS-14).

4.6.2 *Develop und Deliver*

Für die Entwicklung von Lösungsansätzen wird ein Workshop mit zwei Germanistikstudenten (P-1 und P-2) durchgeführt. Sie haben jeweils einen akademischen Abschluss der Germanistik (Bachelor) und im Rahmen ihres Studiums bereits Erfahrungen mit digitalen (historisch-kritischen) Editionen gesammelt. Zunächst wurden ihnen die Ergebnisse der Analyse erläutert.

In anschließenden Arbeitsphasen werden die Designprobleme aus der Problemstellung abgearbeitet und gemeinsam Lösungen erarbeitet. Jede Arbeitsphase unterteilt sich dabei in zwei Unterphasen. Die erste Phase dient der Elaboration, die zweite der Konvergenz. Die bearbeiteten Probleme werden in den nachfolgenden Unterabschnitten erläutert. Alle bei der Elaboration entstandenen Skizzen befinden sich zur Ansicht im Anhang. In diesem Kapitel werden nur die Ideen vorgestellt, die vom Workshopteam in der Konvergenzdiskussion als besonders sinnvoll oder hilfreich errachtet wurden.

PS-04: Entitätsfilter in unterschiedlichen Rollen

Das erste betrachtete Problem ist, wie Entitätsfilter in unterschiedlichen Rollen abgebildet werden können (PS-04). Eine Entität kann zu Dokumenten in unterschiedlicher Beziehung stehen. Für die Suche kann es relevant sein, dass nur Dokumente gefunden werden, die eine spezifische Beziehung zu dieser Entität besitzen. Gleichzeitig soll es aber auch möglich sein, alle Dokumente abzufragen, unabhängig davon, in welcher Beziehung die abgefragte Entität zum Dokument steht. Beispielsweise kann der Anwender nach einer Person suchen und will nur Dokumente als Ergebnis erhalten, bei denen diese Person Empfänger war oder er möchte nach der Person, unabhängig von ihrer Rolle im Dokument, suchen.

Eine erste Möglichkeit dafür ist, dass Prioritäten für das Ranking der Suchergebnisse vergeben können. Je höher die vergebene Priorität, desto höher

wird die Relevanz eines Treffer gewertet. Im skizzierten Beispiel in Abbildung 4.9b auf der nächsten Seite sollen Korrespondenzen weiter oben in der Ergebnisliste auftauchen, wenn „Constanze“ Empfänger des Dokuments ist. Danach folgen Ergebnisse, in denen „Constanze“ Korrespondent des Dokuments und zuletzt wenn sie in diesem nur erwähnt wurde. Ein Vorteil dieser Variante ist, dass die Person in allen für den Anwender relevanten Rollen gefunden wird und der Anwender flexibel die Reihenfolge der Ergebnisse bestimmen kann.

Eine weitere Variante ist die Erweiterung um eine Auswahl der Rolle wie in Abbildung 4.9a auf der nächsten Seite dargestellt. Für jede Entität (beispielsweise Person oder Ort) gibt es ein eigenes Eingabefeld, in dem zunächst nach einer Ausprägung gesucht werden kann. Initial wird diese mit allen Rollen berücksichtigt, in der die Entität in den Ergebnissen vorkommt. Zusätzlich steht neben dem Suchfeld eine Mehrfachauswahl zur Verfügung, in der man die Rollen filtern kann, die für die Ergebnismenge berücksichtigt werden.

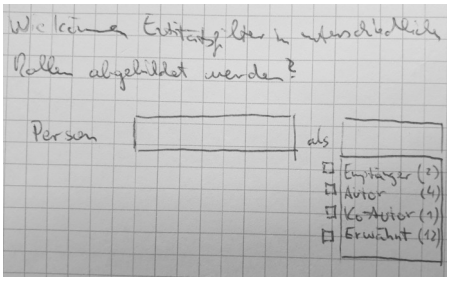
Eine dritte Variante ist in Abbildung 4.9c auf der nächsten Seite zu sehen. Die Idee dabei ist, dass jeweils die Entitäten und ihre Rolle in einer Eingabezeile gesammelt als Chips dargestellt werden. Je Rolle sind die Chips zusätzlich unterschiedlich farblich markiert und in einer Legende ist die Zuordnung erklärt oder alternativ wird im Chip ein Symbol verwendet, um die Rolle darzustellen. Mit einem Klick auf den Chip öffnet sich ein Menü, in dem man die Rolle der Person nachträglich anpassen kann.

PS-05: Zeitselektion

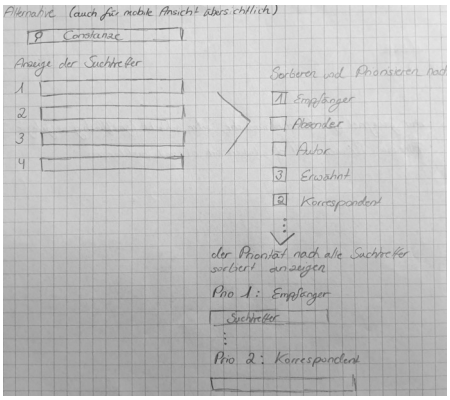
Bei dieser Problemstellung (PS-05) werden Lösungen für die Umsetzung zur Abfrage des Datums beschrieben. Dabei soll die Frage beantwortet werden, wie ein zeitlicher Filter umgesetzt werden kann, wenn ein Datum nicht vollständig bekannt ist. Da dies bereits in Iteration 2 behandelt wurde, konnten die Grafiken (Abbildungen 4.5a bis 4.5e auf Seite 54) mit in den Workshop eingebracht werden. Zusätzlich wurde noch eine weitere Lösung skizziert, die in Abbildung 4.10 auf Seite 72 dargestellt ist.

Die erste Möglichkeit in Abbildung 4.5a auf Seite 54 nutzt mehrere Textboxen, in denen die Werte für Jahr, Monat und Tag separat eingegeben werden können. Zusätzlich sind unter diesen Radiobuttons, mit denen man die Suche steuern kann. Wählt man *Between*, so kann ein zweites Datum ausgewählt werden. Unvollständige Eingaben wären ebenso möglich. Wenn beispielsweise zwei Monate ausgewählt werden, aber Jahr und Tag leer gelassen, so könnte man die Sommermonate aller Jahre selektieren. Bei dieser Variante wäre es wichtig, dass unsinnige Kombinationen abgefangen würden und der User darauf hingewiesen wird.

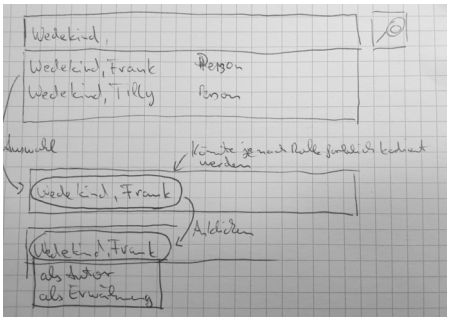
Eine weitere Variante besteht in der Verwendung von Datumsauswahlboxen. Von- und Bis-Datum können, wie in Abbildung 4.5d auf Seite 54 als Beispiel angezeigt, selektiert werden. Dabei muss allerdings auch ein konkretes Datum ausgewählt werden, selbst wenn nur grobe Zeiträume interessant sind.



(a) Sketch1



(b) Sketch2



(c) Sketch9

Abbildung 4.9: Entitätsfilter in unterschiedlichen Rollen PS04

Eine nächste Möglichkeit zur Umsetzung von Zeitfiltern ist die Bereitstellung von Checkboxen mit ausschließlich den vorkommenden Ausprägungen und Anzeige der Anzahl der Treffer. In Abbildung 4.5e auf Seite 54 ist ein Beispiel für die Auswahl des Jahres zu sehen. Der Vorteil darin besteht, dass einfach ersichtlich ist, in welchem Zeitraum mehr Treffer sind. Nachteil ist allerdings, dass der Zeitraum nicht genauer eingeschränkt werden kann.

Eine Lösung zum Umgang mit unvollständig bekannten Daten der Korrespondenzen ist in Abbildung 4.10 auf der nächsten Seite abgebildet. Dabei kann der Anwender entscheiden, ob unklare Daten mit einbezogen werden sollen. In dem Fall soll dem Anwender über Hervorhebung des Treffers klar kenntlich gemacht werden, dass der Treffer ein unvollständig bekanntes Datum hat. Zusätzlich könnte man Treffer mit unvollständigem Datum im Ranking schlechter stellen.

In der Diskussionsrunde wurde eine Kombination aus den Radiobuttons in Abbildung 4.5a auf Seite 54 mit den Datumsauswahlboxen aus Abbildung 4.5d auf Seite 54 empfohlen. Für den Umgang mit unvollständigem Datum kann dies zusätzlich mit der Checkbox aus Abbildung 4.10 auf der nächsten Seite kombiniert werden.

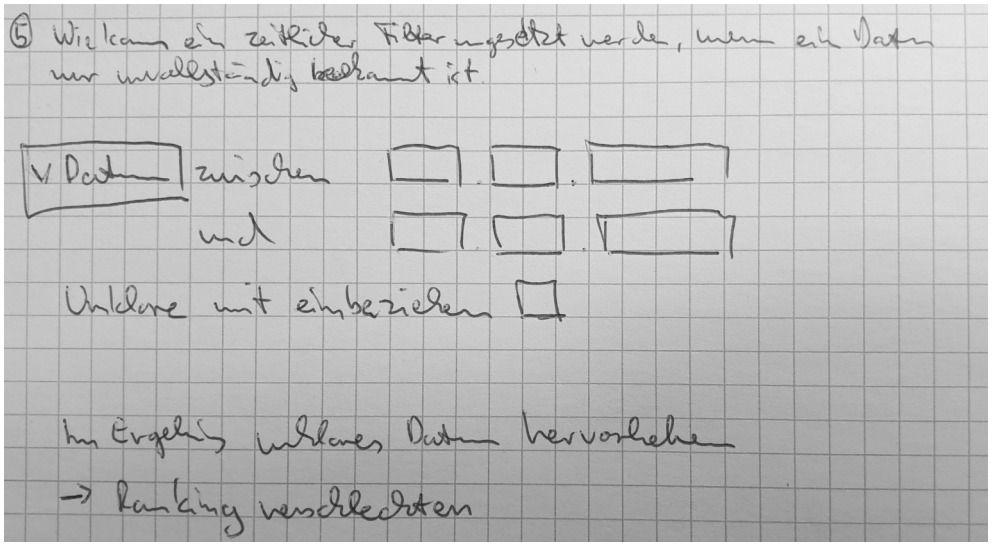


Abbildung 4.10: Sketch1

PS-06: Kommentare durchsuchbar gestalten

Wenn die Suche sich nur auf den Quelltext bezieht, fehlt ein wichtiger Teil der Suchfunktionen einer historisch-kritischen Edition, nämlich die Kommentare. Da es jedoch vom Informationsbedarf abhängt, ob die Kommentare mit durchsucht werden sollen, muss evaluiert werden, wie man Kommentare durchsuchbar gestalten kann, ohne die Suche im Quelltext zu beeinträchtigen. Die Sketches sind in Abbildung 4.11 auf der nächsten Seite dargestellt.

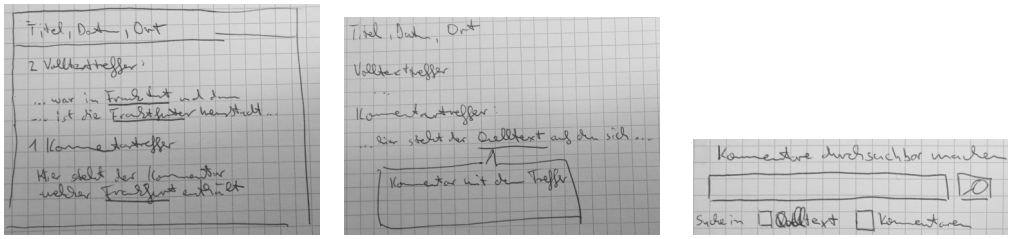
In Abbildung 4.11c auf der nächsten Seite ist eine Lösung dargestellt, in der bei der Suche über Checkbox ausgewählt wird, ob im Quelltext oder in den Kommentaren oder in beidem gesucht werden soll.

Die Abbildung 4.11a auf der nächsten Seite und Abbildung 4.11b auf der nächsten Seite zeigen zwei andere Alternativen, bei der man das Vorkommen des Treffers in den Kommentaren in der Ergebnisdarstellung anzeigen könnte. Die erste Möglichkeit in Abbildung 4.11a auf der nächsten Seite zeigt, dass man die Volltext- und Kommentartreffer separat darstellen könnte. In der zweiten Variante würde der Dokumenttext angezeigt, allerdings ist die Stelle mit dem Kommentartreffer hervorgehoben, der bei Bedarf ebenso angezeigt werden kann.

Für eine mögliche technische Umsetzung müssen die Kommentare geparst und mit anderen Gewichten in den *ts_vector* eingebaut werden. Abhängig davon, worauf die Suche sich auswirken soll, wird das entsprechende Gewicht auf die *ts_query* angewendet.

PS-08: Fehlertoleranz flexibel gestaltbar

Wie bereits in PS-08 ausgeführt, kann es zu Problemen kommen, wenn der Grad der Fehlertoleranz nicht mit der Erwartung des Anwenders übereinstimmt. Es wurde klar gezeigt, dass die Anforderung einer toleranten Suche generell besteht, allerdings kann dies bei spezifischen Suchen hinderlich



(a) Sketch1 (b) Sketch2 (c) Sketch3

Abbildung 4.11: Suche in Kommentaren PS06 (Alle entstandenen Sketches in Abbildung C.2 auf Seite 127)

sein. Es soll evaluiert werden, wie die unterschiedlichen Level der Toleranz für den Benutzer effizient und einfach gestaltet werden können.

Die nachfolgenden Einstellungen wurden als Parameter einer fehlertoleranten Suche identifiziert.

- Groß-, Kleinschreibung
- Reduktion auf Wortstamm
- Ignorieren von Artikel, Konjunktionen, Präpositionen (Stoppworte)
- Synonyme finden

In der Diskussion ist aufgekommen, dass der Begriff „Fehlertoleranz“ als Bezeichnung nicht optimal ist. Daher wurden verschiedene Begriffe diskutiert. Mithilfe einer kleinen Stichprobe von sechs Anwendern wurde die Bezeichnung „Verfeinerung der Suchtreffer“ aus den nachfolgenden Begriffen als am besten passend identifiziert. Ihnen wurde erläutert, welche Funktionalität beschrieben werden soll und sie sollten dann entscheiden, welcher Begriff aus ihrer Sicht am besten passt.

FEHLERTOLERANTE SUCHE Dieser Begriff wurde als zu technisch und teilweise als unverständlich empfunden. Ebenso implizierte das enthaltene Wort „Fehler“ fälschlicherweise eine Wertung.

NORMALISIERUNG DER SUCHANFRAGE Zu diesem Begriff gab es ein unterschiedliches Verständnis zur semantischen Bedeutung und wurde daher deutlich negativ bewertet.

ERWEITERTE SUCHE Mit diesem Begriff erwarteten die Befragten mehr oder andere Optionen zur Suche und daher führte dies zu einer falschen Asssoziation.

SUCHOPTIMIERUNG, TREFFEROPTIMIERUNG, SUCHERGEBNISOPTIMIERUNG Die unterschiedlichen Abwandlungen des Begriffs wurden als negativ bewertet, da eine Veränderung der Sucheinstellung nicht bedeuten würde, dass die Ergebnisse besser werden. „Optimierung“ würde dies aber suggerieren.

VERFEINERUNG DER SUCHTREFFER Diese Bezeichnung wurde bei Anwendern mit verschiedenem Hintergrund als positiv bewertet, da sie alle damit gemeinten Funktionalitäten abdeckt und verständlich klingt.

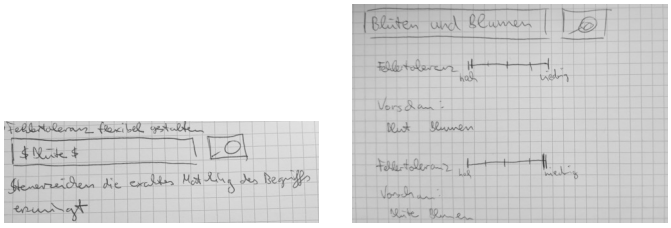
UNSCHARFE SUCHE Der Begriff wurde je nach Hintergrund sehr unterschiedlich bewertet. Für Befragte mit technischer Ausbildung war dies die beste Bezeichnung, für andere Anwender einer der schlechtesten, da Unklarheit darüber bestand, was in diesem Kontext als „Unschärfe" bezeichnet wird.

GENAUIGKEIT, EXAKTHEIT Diese Begriffe wurde durchgehend als passend, aber nicht besonders gut passend, empfunden.

FUZZYNESS Diese Bezeichnung wurde als deutlich zu technisch wahrgenommen und teilweise war überhaupt nicht klar, was damit gemeint sein sollte.

Zur Umsetzung der fehlertolreanten Suche wurden ebenso mehrere Umsetzungsvarianten identifiziert. In der ersten wie in Abbildung 4.12a gezeigt, werden Steuerzeichen verwendet, um die Fehlertoleranz nur für einen speziellen Begriff zu steuern. Die Umfrage hat gezeigt, dass die Anforderungen an eine Phrasensuche über das exakte Matching (wie beispielsweise bei Internetsuchmaschinen) hinausgeht. Daher müssen verschiedene Zeichen eingesetzt werden und dem Anwender sollte eine Hilfestellung dazu geboten werden werden, dass und welche Steuerzeichen es für die Suche gibt. Z. B. kann ein Steuerzeichen zur Verfügung gestellt werden, das alle Begriffe innerhalb dieser exakt matcht.

Eine Alternative wäre die Umsetzung der Steuerung der Suchgenauigkeit durch den Anwender, beispielsweise über einen Regler, wie in Abbildung 4.12b dargestellt. Um dem Anwender die Auswirkungen des Reglers zu vermitteln, könnte man zusätzlich eine Vorschau der Ergebnisse, die Matching Grundlage (die Lexeme nach der Anwendung aller gewählten Normalisierung) oder Beispiele zum Verhalten anzeigen.



(a) Sketch1 (b) Sketch3

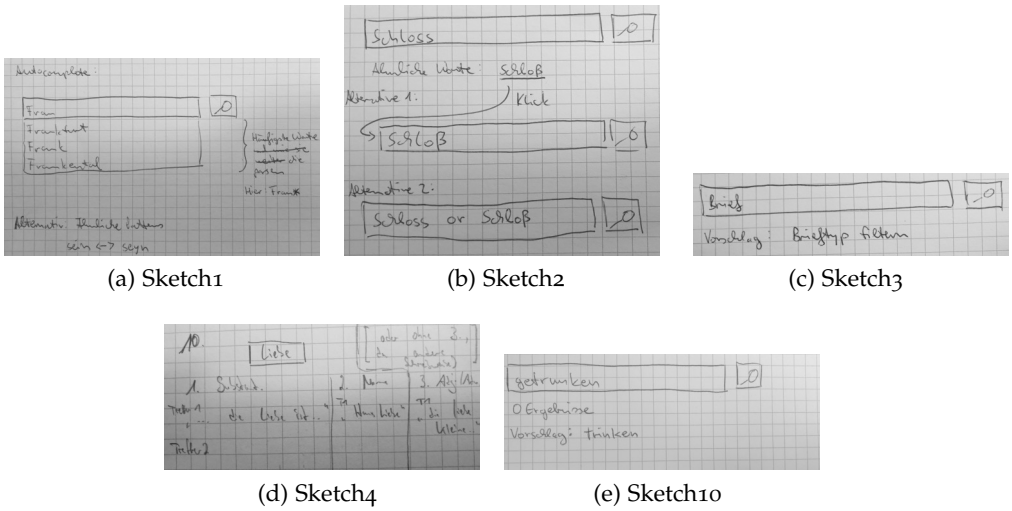
Abbildung 4.12: Flexible Fehlertoleranz PS08 (Alle entstandenen Sketches in Abbildung C.3 auf Seite 128)

PS-10: Alternative Suchbegriffe vorschlagen

Das Vorschlagen alternativer Suchbegriffe zu einem Thema kann den Benutzer bei seiner Suche unterstützen (PS-10). Es sollen Ideen und Ansätze

generiert werden, in welchen Fällen dies sinnvoll sein könnte und was für Vorschläge möglich wären. Die Ideen können nach zwei Dimensionen kategorisiert werden, zum einen wann die Vorschläge gemacht werden und zum anderen, welche Arten von Vorschlägen möglich sind.

Eine erste Möglichkeit dem Benutzer Vorschläge anzuzeigen ist direkt bei der Eingabe. Die Autovervollständigung in Abbildung 4.13a dargestellt und zeigt, dass man dem Benutzer bei der Formulierung seiner Anfrage Worte vorschlagen kann. Die Vorschläge können darin bestehen, dass man häufige Worte vorschlägt, die mit den bereits eingegebenen Buchstaben beginnen. Eine weitere Möglichkeit besteht darin, ähnliche Worte vorzuschlagen, die in der Edition vorkommen. Dies könnte dem Nutzer Hinweise auf alternative Rechtschreibungen oder Rechtschreibfehler geben.



(a) Sketch1 (b) Sketch2 (c) Sketch3

(d) Sketch4 (e) Sketch10

Abbildung 4.13: Vorschlag alternativer Suchbegriffe PS10 (Alle entstandenen Sketches in Abbildung C.5 auf Seite 130)

Eine weitere Möglichkeit besteht darin, die Vorschläge anzuzeigen, nachdem der Benutzer die Suche durchgeführt hat. Hierbei können für alle Worte der Anfrage alternative Vorschläge gemacht werden. Wie in Abbildung 4.13b dargestellt, kann der Benutzer dann diese Begriffe übernehmen. Hier kann entweder der Begriff ersetzt werden oder der Begriff zusätzlich in der Frage ergänzt werden. Es könnten auch alternative Wortarten (Substantive, Verben, Adjektive und Adverben) vorgeschlagen werden (wie in Abbildung 4.13d). Insbesondere wenn es keine Ergebnisse gibt, kann es eine Option sein, dem Benutzer alternative Suchbegriffe vorzuschlagen. Dabei könnte auch das Vorschlagen von Synonymen oder Oberbegriffen interessant sein (Abbildung 4.13e).

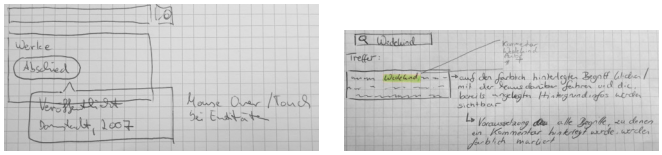
Außerdem können dem Benutzer Vorschläge bereits vor der Suche gemacht werden (Abbildung 4.3a auf Seite 52). Somit können neuen Benutzern Begriffe vorgeschlagen werden, sodass diese eine Idee bekommen, wonach sie suchen könnten.

Eine weitere Option besteht darin, zu versuchen, die Intention des Benutzers vorherzusehen und konkrete Aktionen zu hinterlegen, die ein Benutzer erwarten könnte. Beispielsweise könnte bei der Eingabe eines Dokumententyps wie z. B. *Brief* vorgeschlagen werden, *Brief* direkt in den Korrespondenztypfilter zu übernehmen. In Abbildung 4.13c auf der vorherigen Seite ist dargestellt, wie man dies abbilden könnte. Für eine gute Usability und wenn möglich, sollte man die Aktion direkt mit einem Klick durchführen können.

PS-12: Erschließung von Hintergrundinformationen vereinfachen

Die Variante in Abbildung 4.14a zeigt die Darstellung weiterer Details zu den Entitäten auf der Ergebniskarte bei Mousover oder Touch durch den Anwender. Mit Klick auf die Entität erfolgt beispielsweise eine neue Suche mit einem Filter auf dieser.

Alternativ können, wie in Abbildung 4.14b, annotierte Entitäten im Treffer farblich hervorgehoben werden. Weitere Details zu dieser Entität, wie beispielsweise der vollständige Name oder ein Einzelstellenkommentar könnte bei Mousover dargestellt werden.



(a) Sketch1 (b) Sketch2

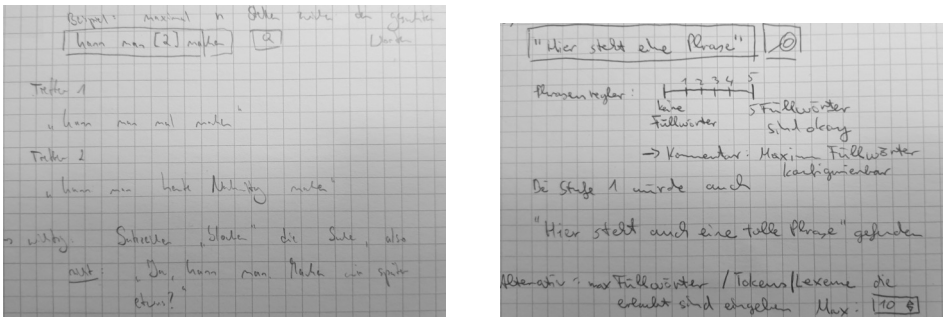
Abbildung 4.14: Erschließung von Hintergrundinformationen PS12 (Alle entstandenen Sketches in Abbildung C.6 auf Seite 130)

PS-14: Phrasensuche konfigurierbar gestalten

In dem Workshop wurden Lösungsansätze dafür gesucht, wie eine Phrasensuche konfigurierbar gestaltet werden könnte (PS-14).

Ein Lösungsansatz ist die Nutzung von Anführungszeichen für Phrasensuche, da diese ein gängiges Steuerzeichen dafür sind. Wenn Anführungszeichen in der Suchanfrage verwendet werden, könnten erweiterte Konfigurationsmöglichkeit eingeblendet werden, die nur für die Phrasensuche relevant sind. In Abbildung 4.15b auf der nächsten Seite wird die Phrase „Hier steht eine Phrase“ gesucht. Unter der Suchbox kann konfiguriert werden, wie tolerant die Phrasensuche ist und wie viele Worte zwischen den Wörtern der Phrase vorkommen dürfen und es könnte ein Hinweis mit Beispiel für den Anwender angezeigt werden. Die Abbildung kann dabei beispielsweise über einen Slider erfolgen.

Eine weitere Variante ist die Definition der innerhalb der Suchanfrage, wieviele Worte zwischen Begriffen vorkommen dürfen. In Abbildung 4.15a auf der nächsten Seite werden für den Suchbegriff „kann man [2] machen“ zwei Wörter zwischen „man“ und „machen“ erlaubt.



(a) Sketch3 (b) Sketch4

Abbildung 4.15: Phrasensuche PS14 (Alle entstandenen Sketches in Abbildung C.7 auf Seite 131)

4.7 FÜNFTE ITERATION

4.7.1 *Problemstellung*

In der dritten Iteration (Abschnitt 4.5 auf Seite 61) wurde der erste Prototyp evaluiert. Hierbei konnten Usability-Probleme identifiziert werden, die durch die vorhergehende Evaluation nicht aufgedeckt wurden. Die fünfte Iteration hat zum Ziel, diese Probleme zu adressieren.

Für die Problemstellung PS-18 (Filtern) wird untersucht, wie eine Filterung bereits von Anfang an (ohne vorherige Volltextsuche) ermöglicht werden kann. Die Filterboxen sollen sortiert werden und es wird geprüft, wie mit langen Listen umgegangen werden kann, da diese als unübersichtlich identifiziert wurden. Weiterhin wird für die Problemstellung PS-04 (Entitätsfilter in unterschiedlichen Rollen) eine Lösung erarbeitet, sodass Entitäten in unterschiedlichen Rollen abgefragt werden können. Es soll eine einfache Suche und Auswahl von Entitäten ermöglicht werden.

Wie die Evaluation der dritten Iteration gezeigt hat, sollte die Fehlertoleranz flexibel gestaltet werden können. Dies wird mit der Problemstellung PS-08 (Flexible Fehlertoleranz) untersucht. In der vierten Iteration (Abschnitt 4.6 auf Seite 69) wurden bereits Lösungsansätze für einzelne Problemstellungen betrachtet und evaluiert. In dieser Iteration wird geprüft, wie die Lösung für die konfigurierbare, fehlertolerante Volltextsuche in den Prototyp integriert werden kann. Dabei muss beachtet werden, dass für PS-09 (Speicher-, Referenzier- und Navigierbare Suchabfragen) sichergestellt wird, dass diese Einstellungen mitberücksichtigt werden.

4.7.2 *Develop*

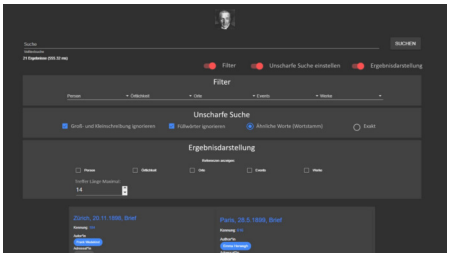
Zur Entwicklung der Wireframes wird erneut ein Workshop mit P-1 und P-2 durchgeführt. Auf Karteikarten werden verschiedene Elemente skizziert und unterschiedliche Anordnungen ausprobiert und diskutiert. Die entwickelten Wireframes werden genutzt, um ein breiteres Benutzerfeedback einzuholen.

Somit kann in der finalen Iteration dieser Arbeit der High-Fidelity-Prototyp genutzt werden, um die Usability der Lösungsansätze zu evaluieren.

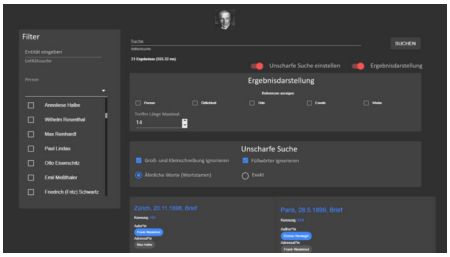
PS-18: Wie kann man Filter umsetzen?

Im aktuellen Zustand erlaubt die Suchfunktion nur eine Volltextsuche und erst anschließend die Filterung. Dies soll in dieser Iteration verbessert werden, sodass von Anfang an auch ohne Suchtext gefiltert werden kann. Dazu müssen Anpassungen am Layout vorgenommen werden, da die Filtermöglichkeiten in der Ergebnisdarstellungskomponente angelegt sind. Hierbei sollen drei Möglichkeiten diskutiert werden. Die erste Möglichkeit (Abbildung 4.16a) besteht darin, die Filterkomponente über den Ergebnissen zu platzieren und von Anfang an einblendbar zu gestalten. Hierbei wurden zwei Probleme festgestellt: Zum einen ist die Filterfunktion versteckt und könnte somit zunächst übersehen werden, zum anderen gibt es ohne vorherige Suche sehr viele Entitäten pro Filter zur Auswahl. Die Notwendigkeit, mehrere Hundert Entitäten manuell zu durchsuchen, bietet eine schlechte Usability.

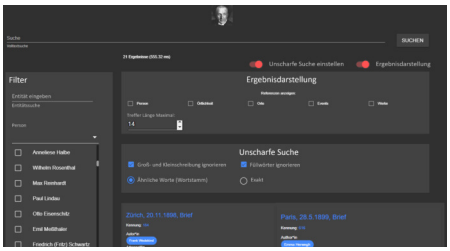
Eine weitere Möglichkeit besteht darin, den Filter dauerhaft anzuzeigen (Abbildungen 4.16b bis 4.16c auf dieser Seite). Der Filter wird dabei am linken Rand platziert. In Abbildung 4.16b fällt der Fokus stark auf den Filter, da der Benutzer oben links mit der Betrachtung der Seite beginnt. Die gewünschte Reihenfolge, erst Volltextsuche, gefolgt von Filterung und anschließend Ergebnisdarstellung wird mit einer Anordnung wie in Abbildung 4.16c abgebildet.



(a) Sketch1



(b) Sketch2



(c) Sketch3

Abbildung 4.16: Mockup des Layouts für Iteration 5

Ein Ansatz für die Selektion von Entitäten in unterschiedlichen Rollen wird im Wireframe Abbildung 4.17 dargestellt. Hierbei besteht die Idee, dass

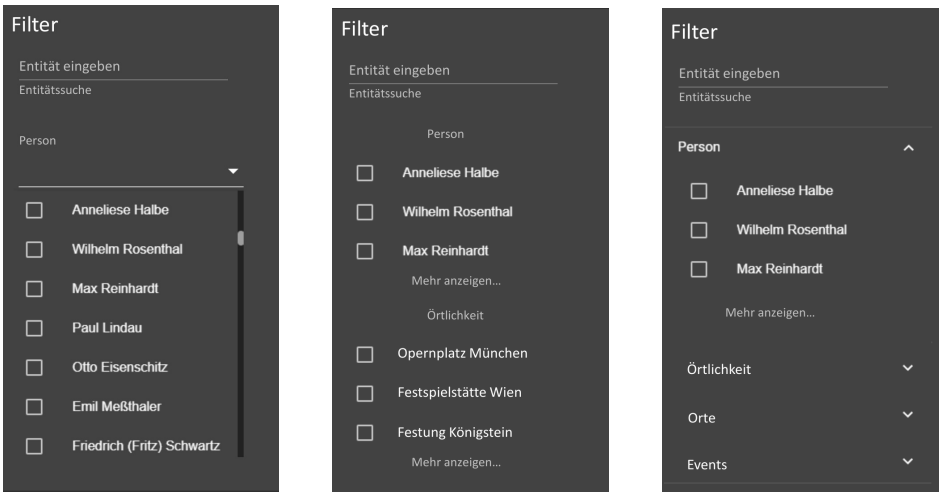
man bei den Entitätsfiltern zunächst die Rolle auswählt, und danach die Entität. In der Diskussion der Grafik wurde das Problem festgestellt, dass ein Wechsel der Entität dazu führen könnte, dass dann keine Ergebnisse mehr gefunden werden, wenn die Entität in dieser Rolle nicht auftritt. Außerdem könnte die Darstellung verwirren, da sie nicht intuitiv wirkt. Ein weiteres Problem besteht in der Kombination von Filtern. Möchte man beispielsweise gleichzeitig Autor und Empfänger filtern, so ist dies mit diesem Ansatz nicht möglich.

Abbildung 4.17: Sketch der Selektion von Entitäten in unterschiedlichen Rollen

Daher wurde ein weiterer Lösungsansatz entwickelt, der darin besteht, pro Rolle eine eigene Selektion zu ermöglichen. Entitäten stehen entweder in den Metadaten zu einem Korrespondenzstück (Absender, Empfänger, Schreibort) oder durch Annotation (erwähnte Person, Örtlichkeit) in Beziehung zum Dokument. Hier liegt die Annahme zugrunde, dass bei der Umsetzung eines eigenen Filters für jede Rolle klar ist, worauf der Filter sich bezieht. Eine Kombination von unterschiedlichen Rollen sollte somit intuitiv möglich sein.

Eine weitere Verbesserung soll für die Auswahl der Filter entworfen werden. Die Entscheidung für ein Dropdown wurde in der dritten Iteration vor allem aus Layoutgründen getroffen. Durch die neue Anordnung ergeben sich hier neue Möglichkeiten. Die Dropdownbox hat außerdem zu Usability-Problemen geführt. In Abbildung 4.18 auf der nächsten Seite werden neben der dieser zwei weitere Ideen dargestellt, wie der Filter gestaltet werden könnte. Die Möglichkeiten in Abbildung 4.18b auf der nächsten Seite und Abbildung 4.18c auf der nächsten Seite verbindet, dass die Entitäten nun als Liste dargestellt werden. Dies soll ermöglichen, dass direkt ersichtlich wird, welche Funktionalität hier ausgeführt werden kann. Zunächst sollen nicht alle Entitäten angezeigt werden, sondern diese sollen bei Bedarf nachgeladen werden können. Die Abbildung mit einer einklappbaren Liste wie in Abbildung 4.18c auf der nächsten Seite hat den Vorteil, dass nur die Entitätsfilter geöffnet werden müssen, die der Anwender auch verwenden möchte.

Von P-1 und P-2 wurde Abbildung 4.18c präferiert, da man hier bei vielen Daten die Listen zuklappen kann und nur die Entitäten aufklappen kann,



(a) Sketch1 (b) Sketch2 (c) Sketch3

Abbildung 4.18: Mockup der Filter für Iteration 5

die man filtern möchte. Es wurde angemerkt, dass die Liste sortiert werden sollte.

Ein weiteres Problem besteht in der Auswahl der Entität, wenn es sehr viele Treffer gibt. Dies ist vor allem zu Beginn der Suche der Fall, wenn noch keine Suche durchgeführt wurde und direkt gefiltert werden soll. Aus diesem Grund wird eine Suche über die Entitäten mit Autovervollständigung entworfen. Sie soll alle möglichen Treffer für die Eingabe anzeigen und die gewählte Entität filtern, sobald der Benutzer sie auswählt. Da Entitäten in unterschiedlichen Rollen vorkommen können, muss zusätzlich zur Entität auch ihre Rolle angezeigt werden, da sonst nicht klar ist, welcher Filter gesetzt werden soll.

Um die Autovervollständigung im Prototyp umzusetzen, kann die Komponente *md-autocomplete* aus dem *vue-material* Framework genutzt werden. Die Komponente sucht in allen Feldern der Objekte eines Arrays. Bei Auswahl wird das Objekt an den Auswahl-Handler übergeben. Diese Information kann dann genutzt werden, um den entsprechenden Filter zu setzen.

Im Prototyp können die Entitäten, die zur Filterung bereitstehen genutzt werden, da sie im Suchergebnis bereits als Objekte in einem Array zurückgegeben werden (siehe Listing 4.9) und als solches im *State* der Applikation abgelegt. Alle benötigten Informationen stehen somit ab der ersten Suchanfrage zur Verfügung.

```
1 {
2   count: 1324,
3   descriptive_name: "Frank Wedekind",
4   descriptive_type: "Person",
5   entityid: "person_7",
6   type: "person"
7 }
```

Listing 4.9: JSON-Repräsentation der Entitäten im Filterarray

Um die Filterung und Entitätssuche von Anfang an zu ermöglichen, müssen die verfügbaren Entitäten beim ersten Öffnen der Applikation bereits geladen werden. Dazu wird das Backend um einen GET-Handler erweitert, um alle verfügbaren Entitäten in der obigen Form zurückzugeben, ohne eine Suche durchführen zu müssen. Somit kann bereits von Anfang an ein Filtern ermöglicht werden. Sobald eine Suchanfrage durchgeführt wird, werden die verfügbaren Entitäten überschrieben mit den tatsächlich vorkommenden. Somit können weiterhin nur valide Entitätskombinationen ausgewählt werden.

PS-08: Wie kann die Fehlertoleranz flexibel gestaltbar gemacht werden?

Der Prototyp soll außerdem um eine konfigurierbare Fehlertoleranz erweitert werden. Wie bereits in der vierten Iteration (Abschnitt 4.6 auf Seite 69) diskutiert, gibt es unterschiedliche Möglichkeiten, um die Fehlertoleranz konfigurierbar zu gestalten. In diesem Abschnitt wird beschrieben, wie eine der Möglichkeiten ausgewählt wird, sodass sie mit dem Prototyp umgesetzt werden kann. Dazu wurde zunächst der Papierprototyp in Abbildung 4.19 auf der nächsten Seite entwickelt, um die Idee eines Schiebereglers zur Konfiguration der Fehlertoleranz umzusetzen. Dies wäre aus Implementierungssicht zu bevorzugen, da so nur bestimmte Kombinationen der Toleranz implementiert und angeboten werden könnten.
Dies ist aber gleichzeitig auch eine Schwäche eines Reglers. Denn die Stufen müssen dem Benutzer erläutert werden, sodass er sie nachvollziehen kann. Dies könnte mit einem Beschreibungstext abgebildet werden, der sich je nach Einstellung verändert. Zusätzlich könnte man darstellen, welche Lexeme tatsächlich für die Suche verwendet werden.
Mithilfe des Wireframes wurde daher weiteres Feedback eingeholt, unter anderem auch von einem Usability Experten (P-4). Der Schieberegler als Konfigurationselement wurde hierbei als wenig transparent und schwer nachvollziehbar angesehen. Die zusätzlichen Hinweise zur Erläuterung der jeweiligen Stufen wurde als schwierig verständlich empfunden.

Aus diesen Gründen soll für die Konfiguration eine alternative Konfigurationsmöglichkeit implementiert werden. Hierbei sollen die Normalisierungen einzeln gesteuert werden können. Ein Mockup hierzu ist in Abbildung 4.18 auf der vorherigen Seite dargestellt. Unter dem Begriff „unscharfe Suche“ sollen die Optionen angeboten werden. Für den Prototyp werden die Optionen „Wortstammsuche“, „Füllwörter ignorieren“ und „Akzente ignorieren“ umgesetzt.

Um die Volltextsuche konfigurierbar zu gestalten, müssen unterschiedliche Normalisierungen auf die Suche angewandt werden. Dies wird in PostgreSQL mit Textsuchkonfigurationen abgebildet. Die Textsuchkonfiguration muss einerseits bei der Erstellung des *ts_vector* angewendet werden und andererseits auf den Suchbegriff bei der Umwandlung in die *ts_query*.

Zunächst müssen in PosgreSQL die möglichen Textsuchkonfigurationen implementiert werden. Dargestellt werden die möglichen Konfigurationen in der Wahrheitstabelle (Tabelle 4.2).

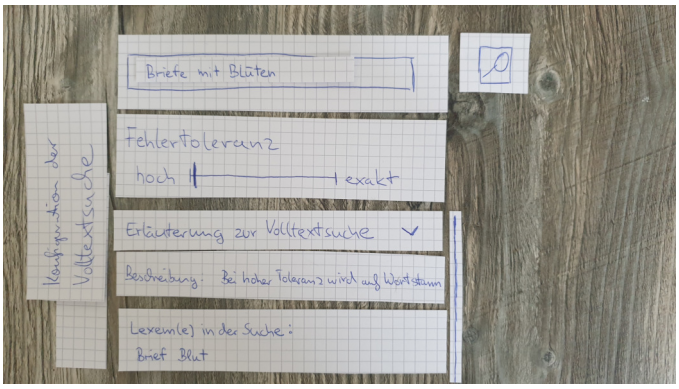


Abbildung 4.19: Sketch der konfigurierbaren Volltextsuche

Wortstamm (*STEM*)	**Füllwörter (*STOP*)**	**Akzente (*ACCENT*)**
WAHR	WAHR	WAHR
WAHR	WAHR	FALSCH
WAHR	FALSCH	WAHR
WAHR	FALSCH	FALSCH
FALSCH	WAHR	WAHR
FALSCH	WAHR	FALSCH
FALSCH	FALSCH	WAHR
FALSCH	FALSCH	FALSCH

Tabelle 4.2: Wahrheitstabelle der Volltextsuchkombinationen

Da alle Kombinationen unterstützt werden sollen, müssen alle Textsuchkonfigurationen angelegt werden. Sie unterscheiden sich darin, welche Normalisierung auf die Dokumente und Suchanfrage bei der Erzeugung der Lexeme angewandt werden. Die Wortstammreduktion (*STEM*) und das Entfernen von Füllwörtern (*STOP*) wird mithilfe von Dictionaries umgesetzt. Eine Beispielkonfiguration ist in der ersten Zeile in Listing 4.10 dargestellt.

```
1 CREATE TEXT SEARCH DICTIONARY german_stem_keep_stop (TEMPLATE = snowball
    , LANGUAGE = german);
2 CREATE TEXT SEARCH DICTIONARY german_stem_rem_stop (TEMPLATE = snowball,
     LANGUAGE = german, STOPWORDS = german);
```

Listing 4.10: Dictionaries mit Reduktion auf deutschen Wortstamm und mit Entfernung von Stoppworten

Hierbei wird der Snowball-Stemmer mit dem Parameter für die deutsche Sprache eingesetzt und es findet keine Entfernung von Stoppworten statt.

Um diese zu entfernen, muss bei der Erstellung des Dictionaries eine Stoppwortliste angegeben werden. In Zeile 2 wird mit „german“ die von PostgreSQL mit ausgeliefert Stoppwortliste konfiguriert. Mithilfe der Wörterbücher können nun die Textsuchkonfigurationen erstellt werden. Hierbei wird die Standardsuchkonfiguration kopiert und dann das Mapping für alle natürlichsprachlichen Lexeme neu definiert (siehe Listing 4.11) Bei der Redefinition des Mappings können mehrere Wörterbücher angegeben werden, die nacheinander angewendet werden. In diesem Schritt wird unaccent zwischengeschaltet, wenn die Konfiguration gemäß der Wahrheitstabelle die Entfernung von Akzenten erfordert. Diese Wörterbücher werden in der Reihenfolge angewendet, in der sie aufgenommen wurden.

```
1 CREATE TEXT SEARCH CONFIGURATION
    german_no_stem_keep_stop_unaccent_config (copy = pg_catalog.german);
2 ALTER TEXT SEARCH CONFIGURATION german_no_stem_keep_stop_unaccent_config
     ALTER MAPPING FOR asciiword, asciihword, hword_asciipart, hword,
    hword_part, word WITH unaccent, german_no_stem_keep_stop;
```

Listing 4.11: Textsuchkonfiguration mit Reduktion auf deutschen Wortstamm

Die Suchanfrage, die vom Frontend gesandt wird, muss um die erweiterten Textsuchoptionen ergänzt werden. In einem zusätzlichen Feld im Body des Requests wird die Konfiguration ergänzt. Abhängig von der gewählten Konfiguration wird dann im Backend die korrekte Textsuchkonfiguration der Datenbank ausgewählt und die Volltextsuchanfrage durchgeführt.
Im Frontend wird eine Komponente implementiert, wie im Mockup (Abbildung 4.18 auf Seite 80) skizziert, welches die Auswahl der Textsuchkonfiguration ermöglicht. Damit die Suche weiterhin navigierbar und die URL teilbar ist, muss diese Konfiguration auch in der URL kodiert werden.
Das Ergebnis dieser Iteration wird in Abbildung 4.20 auf der nächsten Seite dargestellt. Die Filterkomponente wurde an den linken Rand verschoben und eine Autovervollständigungssuche wurde umgesetzt. Die unscharfe Suche ist konfigurierbar, sobald eine Volltextsuche ausgeführt wurde, indem man den Bereich „Unscharfe Suche einstellen“ öffnet.

4.7.3 *Deliver*

Auch diese Version des Prototyps wurde passwortgeschützt online gestellt und mit der Forschungsgruppe geteilt, um die Usability zu bewerten. Dabei wurden keine weiteren Erläuterung zum Prototyp mitgegeben, um die erste Interaktion der Benutzer ohne Hilfestellungen einschätzen zu können.

In einer Usability Inspektion von P-4 wurden einige Usability-Probleme festgestellt. In der Darstellung (Abbildung 4.20 auf der nächsten Seite) ist erkennbar, dass das Layout der Ergebnisdarstellung noch nicht optimal ist. Hier wird links und rechts Platz nicht effizient genutzt. Darüber hinaus sind Überschriften nicht einheitlich zentriert oder linksbündig und sollten linksbündig dargestellt werden.

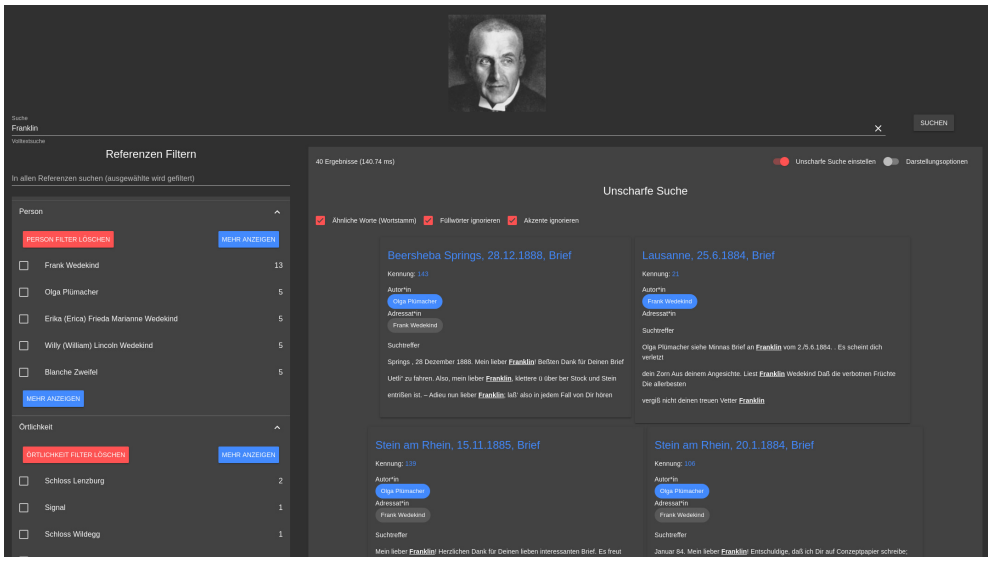


Abbildung 4.20: Screenshot des weiterentwickelten High-Fidelity-Prototyps (fünfte Iteration Versionstag 0.6-1)

Weiterhin wurde festgestellt, dass die Anzahl der Dokumente (in der Filteransicht) keine Einheit besitzt und somit unklar ist, worauf sich diese Zahl bezieht.

Die Löschung eines Entitätsfilter per Button wird vermutlich eher selten vorkommen, unter der Annahme, dass eher selten nach dem Vorkommen mehrerer Entitäten im selben Dokument gefiltert wird, daher spart der Löschenbutton pro Entitätsfilter kaum Klicks und kann entfernt werden. Darüber hinaus ist ein Button ausreichend, um die Liste zu erweitern.

Von P-5 wurde ein unstrukturierter Usability Test durchgeführt. Hierbei wurde die Anwendung erkundet und es wurden Eindrücke niedergeschrieben, die während der Erkundung gemacht wurden. Dadurch konnte ein erster Eindruck der *First-use learnability* gewonnen werden.

Bei diesem Test ist aufgefallen, dass der aktuelle Suchzustand der Applikation schwer zu erfassen ist. Wenn bereits Filter gesetzt sind, so ist dies aufgrund der Anzahl an Filtern nicht ersichtlich, da nicht alle auf den Bildschirm passen. Da dies sich aber auf die Filtermöglichkeiten auswirkt, muss eine Darstellung gefunden werden, um den Zustand der Suche zu verdeutlichen. Eine weitere Problematik besteht darin, dass Benutzer die Eingabe zur Volltextsuche nicht finden, da diese sehr unauffällig gestaltet ist. Die Optik insgesamt wurde als unruhig wahrgenommen, vermutlich aufgrund das gewählten Farbschemas mit den kräftigen Farben.

Um sicherzustellen, dass der Remote-Usability-Test wie konzipiert funktioniert und sowohl die zeitlichen und technischen Rahmenbedingungen eingehalten werden können, wurde der Remote-Usability-Test mit P-3 durchgeführt. Die Ergebnisse des Tests und die Perspektive von P-3 sind insbesondere deshalb interessant, da P-3 als Informatikerin noch keinen Kontakt mit digitalen Editionen hatte. Das Protokoll des Tests befindet sich im Anhang.

Hierbei konnten weitere Usability-Probleme identifiziert werden. Der Testperson war während des Tests nicht transparent, dass bereits ein Filter aktiv war. Die Autovervollständigung zur Suche nach Entitäten war zu unauffällig und wurde nicht auf Anhieb entdeckt. Bei der Markierung eines Volltexttreffers führte der Klick auf das Ergebnis zur Verwirrung, da hierbei das Ergebnis geöffnet wurde. Die Aktion wurde nicht erwartet, da die Volltexttreffer nicht als Link hervorgehoben sind.

4.8 SECHSTE ITERATION

4.8.1 *Problemstellung*

Ziel dieser Iteration ist es, die Usability-Probleme aus der letzten Phase zu beheben. Für das Problem, dass eine komplexe Suchanfrage schwer nachvollziehbar ist, wird eine eigene Problemstellung (PS-26) definiert. Außerdem wurden in dieser Iteration Anpassungen am Layout vorgenommen, da hier mehrere Verbesserungsmöglichkeiten identifiziert wurden. Die Anpassungen können somit im Remote-Usability-Test direkt evaluiert werden.

4.8.2 *Develop*

PS-26: Darstellung des Suchzustands

Da im Feedback des Usability Tests aufgezeigt wurde, dass den Benutzern oft nicht klar ist, wie die Ergebnisse Zustande gekommen sind, soll dies besser visualisiert werden. Hierzu soll der Suchzustand an einer zentralen Stelle erläutert werden. Der Suchzustand besteht aus einer durchgeführten Volltextsuche sowie den gesetzten Filtern. Diese Informationen sollen gemeinsam mit dem Ergebnis dargestellt werden. Ein Mockup befindet sich in Abbildung 4.21. Hierbei wird die ausgeführte Suche erneut angezeigt und es wird dargestellt, wonach gefiltert wurde. Außerdem wird dem Benutzer die Möglichkeit gegeben, Filter zu löschen.

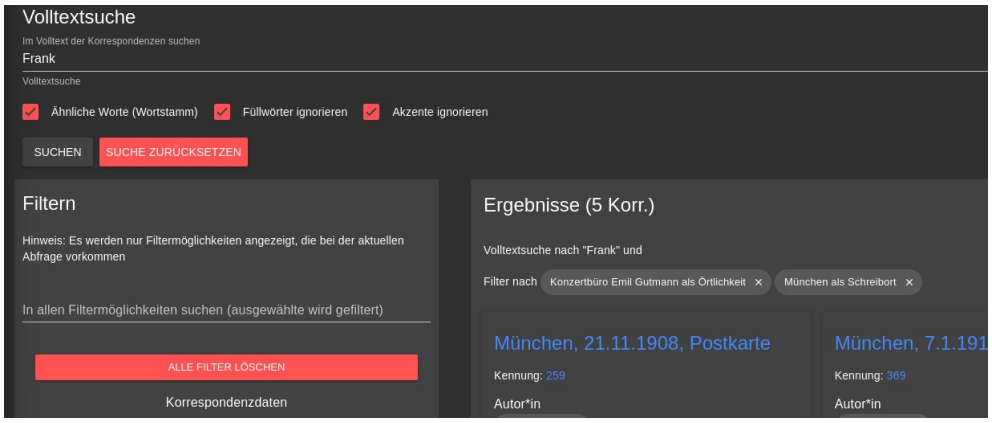


Abbildung 4.21: Mockup der Explain Component

Zur Umsetzung der Komponente sind noch nicht alle benötigten Informationen im Frontend vorhanden. Dies liegt darin begründet, dass zu den selektierten Entitäten nur der Schlüssel bekannt ist. Es muss eine Möglichkeit implementiert werden, zu diesem Schlüssel die Beschreibung zu bestimmen. Hierbei wurden drei Umsetzungsvarianten identifiziert und verglichen.

Die erste Möglichkeit besteht darin, zusätzlich zum Schlüssel auch die Beschreibung beim Filtern abzulegen. Diese Idee wurde verworfen, da der Beschreibungstext dann zusätzlich in der URL kodiert werden müsste, da sonst bei der Navigation oder dem Öffnen eines bestimmten Suchzustandes die Beschreibungen fehlen würden.

Eine weitere Möglichkeit besteht darin, zur Abfrage der Filtertexte eine Anfrage an den Server zu stellen und die gefilterten Entitäten zu übergeben. Der Server würde die Beschreibungstexte zurückgeben. Da jeder Suchzustand aus einer Suchanfrage besteht und die Filterwerte dabei übergeben werden, können die Texte in der Antwort mitgesendet werden.

Die dritte Möglichkeit besteht darin, alle Beschreibungstexte pro Entität einmal zu laden und dann im Frontend vorzuhalten. Dies ermöglicht das Nachlesen der Beschreibungen direkt im Frontend und die Entitäten müssen nur einmal, dafür aber vollständig übertragen werden. Diese wurde gewählt, da dadurch eine höhere Flexibilität bei weiteren Features im Frontend ermöglicht wird. Der Zugriff auf alle Entitäten im Frontend, auch wenn diese nicht gefiltert sind, hilft perspektivisch, beispielsweise Tooltips in den Ergebnissen anzuzeigen. Welche Details der Entitäten übertragen werden, kann einfach ergänzt werden und somit stehen weitere Informationen dann flexibel im Frontend zur Verfügung.

Um beim Start der Applikation die Entitätsinformationen abzurufen, wird sowohl die Suchanfrage *POST /search* als auch der *GET /entitites* erweitert, sodass sie ein weiteres Objekt *lookup* zurückgeben. Dieses Objekt besitzt als Schlüssel die Entitätsschlüssel und als Wert ein Objekt mit allen Details zur Entität, die benötigt werden. In diesem Fall ist dies zunächst nur der Beschreibungstext (siehe Listing 4.12). Damit das Objekt nicht bei jeder Suchanfrage übertragen wird, wird ein Header-Feld eingeführt, mit dem gesteuert wird, ob das Lookup-Objekt mitgesendet werden soll.

```
1  {
2    lookup:
3      {
4        event_1: {descriptive_name: "Wedekinds 50. Geburtstag"},
5        ...
6        person_1: {descriptive_name: "Frank Wedekind"},
7        ...
8        city_1: {descriptive_name: "Darmstadt"},
9        ...
10     }
11 }
```

Listing 4.12: JSON-Repräsentation der Entitäten im Lookup Objekt

Die Komponente zeigt über das Databinding an den *State* der Applikation die Volltextsuche an. Der Filterzustand wird über ein *computed property* angereichert. Dies ist in Listing 4.13 dargestellt. Für alle gefilterten Entitäten wird ein Objekt zusammengesetzt, welches die Informationen enthält, in welcher Rolle eine Entität gefiltert wurde (*type*), welche Schlüssel die Entität hat (*entityid*), welcher menschenlesbare Beschreibungstext diese repräsentiert (*descriptive_name*) und zuletzt, einen beschreibenden Text für die Rolle, in der die Entität gefiltert wurde (*descriptive_type*). Für das Entfernen einer Entität aus dem Filter, wird eine *Action* implementiert. Die relevanten Informationen hierfür sind der Entitätsschlüssel und die Rolle der Entität.

```
1  if (this.$store.getters.searchquery === null) return [];
2  // holds the Query which was used to fetch the latest results:
3  const selectedEntities = this.$store.getters.searchquery.entitiesquery
       ;
4  const allSelected = [];
5  Object.keys(
6    selectedEntities
7  ).map(key => { // for all entity roles (person, author, etc.)
8      const entities = selectedEntities[key];
9      return entities.map(entityid => {
10         const {descriptive_name} = this.$store.getters.lookupEntity(
               entityid);
11         const descriptive_type = TYPE_DESCRIPTION[key];
12         allSelected.push(
13           {
14             type: key,
15             entityid: entityid,
16             descriptive_name,
17             descriptive_type
18           });
19     });
20 })
21 return allSelected;
```

Listing 4.13: Anreicherung der Daten für die Filterdarstellung

Überarbeitung des Layouts

Das dunkle Farbschema des Prototyps wurde von mehreren Testern als unangenehm empfunden. Daher wird das Farbschema der Applikation umgestellt auf ein helles Design. Da im Usability Test die Volltextsuchoptionen nicht aufgefallen sind, sollen diese nun prominenter in der Nähe der Volltextsuche platziert werden. Da sie sich auf die Volltextsuche beziehen, werden sie unter der Volltextsuche permanent angezeigt, statt sie in der Ergebnisdarstellung in einem aufklappbaren Menü zu verstecken.

Die Eingabefelder zur Volltextsuche und zur Filterung wurden beim letzten Usability Test nicht sofort als Eingabefelder erkannt, daher soll der Ansatz ausprobiert werden, ob eine Lupe vor der Eingabe zur Verdeutlichung genutzt werden kann. Da die Lupe vor beiden Eingabefeldern dargestellt

wird, soll dem Benutzer auffallen, dass er zwei Suchmöglichkeiten hat. Um die Aufmerksamkeit des Benutzers noch mehr auf die Filtersuche zu lenken, werden zusätzlich nur noch die häufigsten fünf Elemente angezeigt. Die Funktionalität „Mehr anzeigen“ wird entfernt und es wird ein Hinweis eingeblendet, dass man die Filtersuche nutzen soll.

Das Layout in Abbildung 4.22 wurde erneut einer Usability Inspection unterzogen. Hierbei wurde festgestellt, dass das Layout durch das Lupensymbol durcheinander wirkt. Außerdem wurde der Vorschlag gemacht, am Desktop die Darstellung weiterer Details in den Suchergebnissen intuitiver zu gestalten, indem auf ein Dropdown mit Checkboxen umgestellt wird.

Diese Vorschläge wurden noch umgesetzt bevor der Prototyp einem Remote-Usability-Test unterzogen wurde.

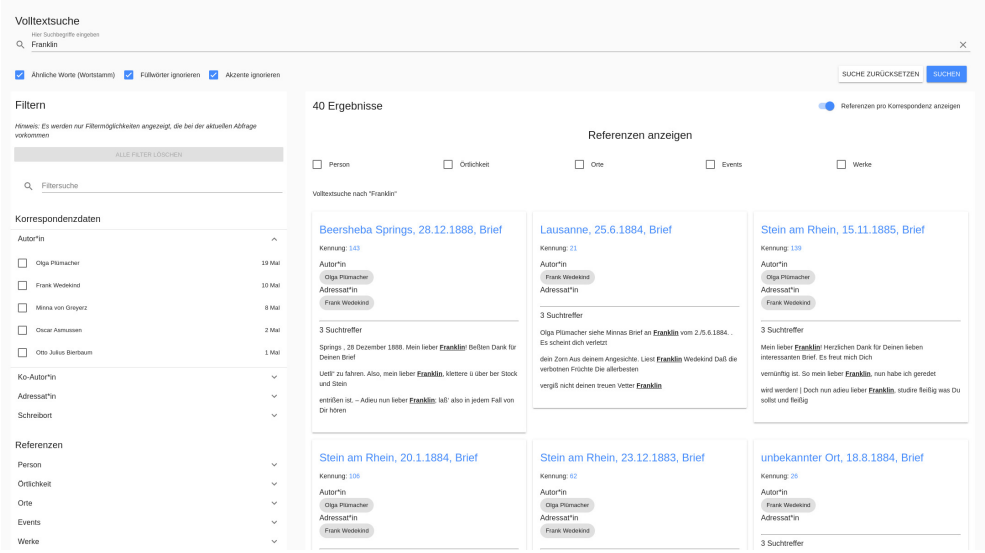


Abbildung 4.22: Screenshot des Layout Entwurfs

4.8.3 *Deliver*

Remote-Usability-Test

Zur Evaluation des Prototyps wird der bereits in Iteration 5 vorgestellte Remote-Usability-Test mit P-6 durchgeführt. Das vollständige Testprotokoll ist im Anhang zu finden. In diesem Kapitel wird vor allem dargestellt, welche Aspekte der Applikation gut funktioniert haben und welche Usability-Probleme festgestellt werden konnten. Evaluiert wurde der Prototyp, wie er in Abbildung 4.23 auf der nächsten Seite dargestellt ist.
Ein Problem, welches sich durch die gesamte Lösung der Aufgaben zog, war, dass die Testperson nicht auf die Volltextsuche aufmerksam wurde. Daraus entstanden einige Folgeprobleme.

Da über die Filtersuche eine Volltextssuche vorgeschlagen wird, wenn keine Entitäten gefunden wurden, konnte sie in einigen Fällen dennoch Volltextsuchen ausführen.

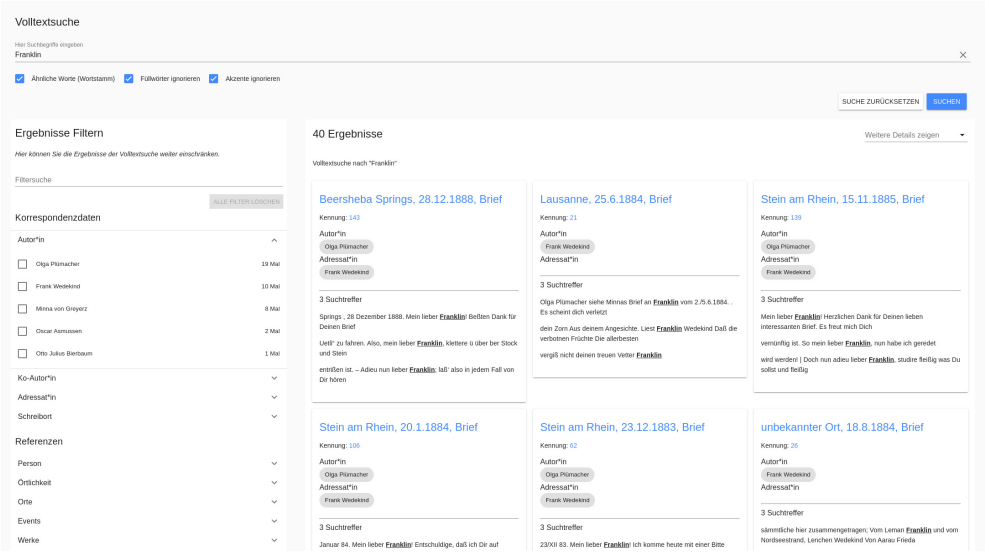


Abbildung 4.23: Version des Prototyps beim Usability Test mit P-6

Das Zurücksetzen der Suche wurde bei der Bearbeitung der ersten Fragen durch die Testperson nicht entdeckt, daher trat unter anderem das Problem in Abbildung 4.24 auf. Beim Versuch „harry graf kessler“ zu suchen, gab es keine Ergebnisse, da noch eine Volltextsuche aktiv war. Den Hinweis, dass die Volltextsuche nach „flügel“ noch aktiv ist, hat die Testperson ebenso nicht wahrgenommen.

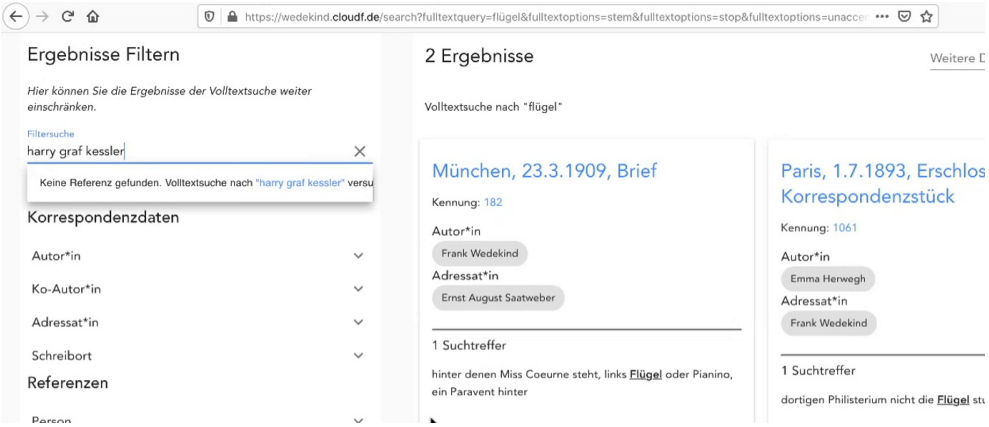


Abbildung 4.24: Aktive Volltextsuche behindert Entitätssuche Usability Test mit P-6

Ein weiteres Problem trat in der Verwendung der Filtersuche auf. Wenn auf der gesamten Menge gefiltert werden sollte, war die Entitätssuche sehr träge und Firefox zeigte sogar die Warnung an, dass der Browser verlangsamt wird.

Im Feedbackgespräch nach den Aufgabenstellungen wurde festgestellt, dass die Filtersuche und die Übernahme eines Wertes in die Volltextsuche sehr langsam waren. Als positiv wurde gesehen, dass man in der Suchhilfe die Rolle der Entität (beispielsweise ist die Person als Autor oder Empfänger) direkt auswählen konnte. Problematisch wurde gesehen, dass die Volltextsu-

che nur dann funktioniert, wenn keine Entität zum Begriff gefunden wird. Für die Testperson war keine Volltextsuche nach Wein möglich. Dies lag daran, dass die Testperson das Volltextsuchfeld nicht gefunden hat und somit immer den Umweg um die Empfehlung der Verwendung der Volltextsuche gehen musste.

Das Zurücksetzen der Suche war aus ähnlichen Gründen zunächst nicht aufgefunden worden. Das Layout mit Filter am Rand wurde als übersichtlich angesehen, die Ergebnisdarstellung mit der Vorschau der Treffer ebenfalls. Vermisst wurden durch die Testperson Möglichkeiten der zeitlichen Selektion, sowie die Möglichkeit die Ergebnisse zu sortieren. Um die Suchfunktion einzusetzen, sollte die Volltextsuche ergänzt und die Performance der Filterung verbessert werden.

Im Abschluss des Gesprächs wurde kurz vorgestellt, wie einzelne Funktionalitäten vorgesehen waren. Daraufhin hat die Testperson reflektiert, warum sie den Button für das Zurücksetzen nicht gefunden hatte und dabei festgestellt, dass sie sich auf den linken Bildschirmbereich fokussiert hatte. Die Überschrift „Volltextsuche“ wurde als Überschrift für die gesamte Seite wahrgenommen und nicht als Titel für die Funktion der Volltextsuche. Das Eingabefeld wurde aufgrund der Darstellung als lange Linie nicht als ein solches erkannt.

Die erkannten Usability-Probleme sollen in einer weiteren Iteration angegangen und, wenn möglich, bereits behoben werden.

Applikationsperformance

Um zu messen wie performant der Prototyp ist und wie viele Anwender parallel unterstützt werden, wird ein Performancetest durchgeführt. Nach Molyneaux ist eine Anwendung performant, wenn „sie es dem Benutzer erlaubt, eine Aufgabe durchzuführen, ohne eine unangenemessene Verzögerung oder Irritation zu erleben.“ [28, S. 1] Die Performanceanforderung hängt dabei also von der Erwartung des Benutzers ab. Wie Molyneaux weiter ausführt, gibt es keinen Industrie Standard für angemessene Antwortzeiten. [28, S. 1 ff.]

Die Performance kann anhand von Indikatoren bewertet werden. Diese werden in der nachfolgenden Liste aufgezählt und erläutert [28, S. 37 ff.]:

AVAILABILITY Ist das System für den Nutzer erreichbar?

CONCURRENCY Wie viele gleichzeitig aktive Benutzer kann das System bedienen?

THROUGHPUT Wie viele Aktionen können pro Zeiteinheit geleistet werden, beispielsweise Suchanfragen pro Sekunde?

RESPONSE TIME Zeitliche Dauer, die der Benutzer auf eine Antwort des Systems warten muss

NETWORK AND SERVER Auslastung von Netzwerk und Server

Es gibt verschiedene Arten von Performancetests. Um die maximale Kapazität einer Applikation für hohe Last zu bestimmen, eignet sich der Stresstest. Um einen Stresstest durchzuführen wird die Last auf die Applikation solange erhöht, bis die Infrastruktur versagt oder ein Schwellwert für angemessene Antwortzeiten überschritten wird. Somit kann bestimmt werden, wie viel Last eine Applikation auf der gewählten Infrastruktur unterstützt. [28, S. 50]

Für den Prototypen soll ermittelt werden, wie viele Benutzer parallel unterstützt werden. Dazu werden im nächsten Abschnitt die Testumgebung und die Use-Cases beschrieben.

DEFINITION DER TESTUMGEBUNG

Zur Evalution der Performance wird eine Testumgebung benötigt [28, S. 70].

Für die Evaluation sollen zwei Testumgebungen definiert werden. Zum einen soll ein günstiger Testserver (S-1) verwendet werden. Hierbei wurde der kleinste Virtual Private Server bei Hetzner [18, 19] ausgewählt. Er stellt somit die günstigste Möglichkeit dar, die Suchfunktion gehostet zu betreiben. Zum anderen soll ein großer dedicated Server (S-2) getestet werden, um die maximale Kapazität einer einzelnen Instanz zu bestimmen. Aus Kostengründen wird hierzu ein bereits vorhandener, jedoch ähnlich ausgestatteter Rechner in der lokalen Umgebung verwendet, anstatt einen Server für den Performancetest zu provisionieren. Die Ausstattung wird in Tabelle 4.3 dargestellt.

Komponente	**Virtual Private Server (S-1)**	**Dedicated Server Äquivalent (S-2)**
CPU	AMD EPYC 2nd Gen VCPU (1 Core)	AMD Ryzen 9 3900 (12 Core)
RAM	2 GB	32 GB
DISK	20 GB (Network Disk)	880 GB NVMe SSD
Preis/Monat	etwa 3 EUR / Monat [18]	etwa 100 EUR / Monat [19]

Tabelle 4.3: Übersicht über die Testumgebungen

USECASES SCRIPTING UND TEST SZENARIO

Im nächsten Schritt müssen Usecases definiert und das Test Szenario definiert und implementiert werden [28, S. 71 ff.]. Für die Durchführung des Tests wird Locust genutzt, ein „open source load testing tool.“[26].

Als Use-Cases werden das Durchführen einer Suchanfrage (Use-Case 1) sowie das Öffnen der Startseite (Use-Case 2) definiert. Zur Ermittlung von möglichen Suchanfragen wurde das Logging lokal aktiviert und es wurden einige Anfragen durchgeführt. Der erste Use-Case wählt von diesen Anfragen eine zufällige aus und führt diese dann durch. Für die Simulation des Öffnens der Startseite, wird ein *get* Request ohne weiteren Pfad in der URL durchgeführt.

Aus diesen Use-Cases wird ein Benutzer modelliert. Es wird die Annahme getroffen, dass ein durchschnittlicher Benutzer die Seite öffnet und etwa

10 Suchen durchführt. Zwischen diesen Suchen wartet er durchschnittlich zwischen 2 und 5 Sekunden.

Der gesamte Code befindet sich im digitalen Anhang (Anhang E auf Seite 144) im Ordner *locust*. Eine Kurzform zur Erläuterung wird in Listing 4.14 dargestellt. Mit der *task*-Annotation wird das Verhältnis der Tasks zueinander gesteuert. Durchschnittlich wird für alle zehn Suchanfragen einmal die Startseite aufgerufen. Die Suchanfrage wird dabei zufällig aus der Liste *queries* ausgewählt.

```
class UserBehavior(TaskSet): # Verhalten Suchbenutzer
    @task(10)  # 10 Suchen
    def test_query(self):  # Use-Case 1
        queries = [
            {"entitiesquery":["work_107"],"fulltextoptions":["stem","
                stop","unaccent"]},
            {"fulltextquery":"Flugel","fulltextoptions":["stem","stop","
                unaccent"]},
            # hier folgen noch deutlich mehr Abfragen
        ]
        self.client.post("/search", json=random.choice(queries))

    @task(1) # Startseite oeffnen, alle 10 Suchanfragen
    def call_home(self):  # Use-Case 2
        self.client.get("/", auth=("frank", "volltext"))

class WebsiteUser(HttpUser): # Benutzer
    host="https://wedekind.cloudf.de" # Standard Server
    wait_time = between(2, 5) # Warte 2 bis 5 Sekunden
    tasks = {UserBehavior: 1}
```

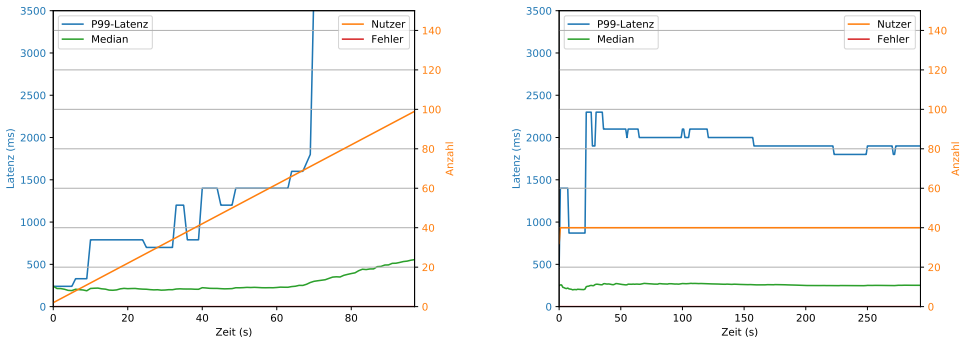
Listing 4.14: Use-Case Script

DURCHFÜHRUNG DES TESTS

Zur Durchführung des Tests wird die Locust Umgebung gestartet. Im Webinterface von Locust definiert man die obere Schranke zu simulierender Benutzer und die „Spawn Rate“, also wie viele Benutzer pro Sekunde hinzukommen sollen.

Mit dem Virtual Private Server (S-1) wurden zunächst der Stresstest durchgeführt. Dazu wurde pro Sekunde ein Nutzer hinzugefügt. In Abbildung 4.25a auf der nächsten Seite ist die Median-Antwortzeit und der 99. Perzentil der Antwortzeit in Millisekunden aufgetragen. Auf der zweiten Y-Achse sind die fehlgeschlagenen Requests und die Anzahl Nutzer dargestellt. Man sieht, dass die Median-Antwortzeit bis um die 60 Benutzer stabil ist und hier dann langsam ansteigt. Gleichzeit sieht man bei der Betrachtung des 99. Perzentils der Antwortzeit, dass hier ebenfalls ein Anstieg festzustellen ist. Ab 60 Benutzern steigt die Latenz des 99. Perzentils über Sprunghaft über 3500 ms. Die Fehlerrate bleibt im betrachteten Horizont auf null.

Basierend darauf wurde angenommen, dass etwa 40 Benutzer unterstützt werden. Um das zu verifizieren, wird ein zweiter Test durchgeführt, indem die Last von 40 Benutzern simuliert wird. Die Ergebnisse dieses Tests sind in Abbildung 4.25b dargestellt. Die Latenz des 99. Perzentils und der Median sind in diesem Test über einen längeren Zeitraum stabil.



(a) Stresstest VPC (S-1)

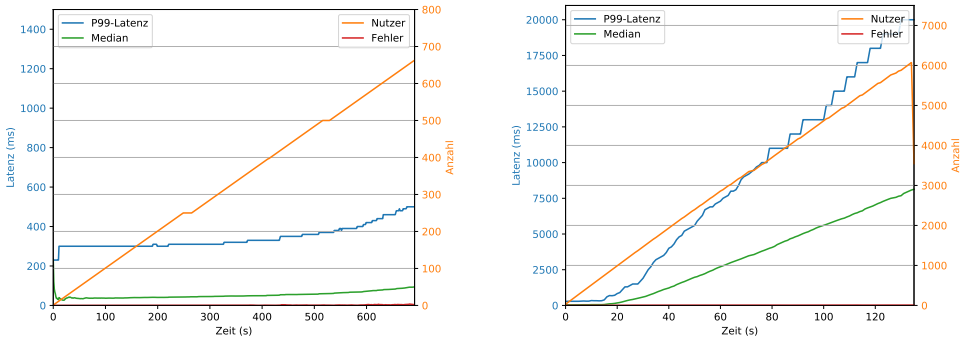
(b) Dauertest mit konstant 40 Nutzern VPC (S-1)

Abbildung 4.25: Performancetest Virtual Private Server (S-1)

Anschließend wurde der Test mit dem Dedicated Server Äquivalent (S-2) durchgeführt. In Abbildung 4.26a auf der nächsten Seite wird ein Benutzer pro Sekunde hinzugefügt. Es ist zu beachten, dass die Latenz hier auf der Y-Achse anders skaliert ist, da die Median Ausführungszeit und das 99. Perzentil deutlich kleiner waren. Außerdem wurde der Test deutlich länger durchgeführt. Zunächst wurde bis 250 Nutzer getestet, nachdem der Server das ohne Probleme leisten konnte, wurde ein neuer Zielwert mit 500 gesetzt. Nachdem dieser ebenfalls problemlos erreicht werden konnte, wurde ein noch höheres Ziel gesetzt und der Test dann bei etwas über 650 Nutzern abgebrochen.

Um den dedicated Server an seine Grenzen zu bringen und dann zu ermitteln, welche Komponente in diesem Fall der Flaschenhals ist, wurde wie in Abbildung 4.26b auf der nächsten Seite dargestellt, die Last deutlich erhöht. Der Test wurde mit einem Zielwert von 6000 Nutzern und einem Wachstum von 50 Nutzern pro Sekunde erneut ausgeführt. Bei etwa 800 Nutzern sieht man zunächst einen Anstieg im 99. Perzentil, kurz darauf ebenfalls beim Median. Bei etwa 1500 Nutzern ist die Latenz im 99. Perzentil vergleichbar, mit der des Virtual Private Servers (S-1) bei 20 Nutzern.

Bei der Last von 3000 Benutzern wurde mittels Prozessmonitor geprüft, ob identifiziert werden kann, welche Komponente den Flaschenhals bildet. Dabei konnte festgestellt werden, dass der Server noch nicht unter Volllast läuft. Der einzelne Node.js Prozess lief allerdings mit einer Auslastung von 100 Prozent, was der vollen Auslastung eines Rechenkerns entspricht. Somit sollte zur Erhöhung der Kapazität bei einem Multicore System mit einem Loadbalancer und mehreren Nodeprozessen die Applikationsperformance verbessert werden.



(a) Stresstest Dedicated Server (S-2)

(b) Steilerer Stresstest mit Dedicated Server (S-2)

Abbildung 4.26: Performancetest Dedicated Server (S-2)

Zusammengefasst kann festgestellt werden, dass bereits mit der kleinsten und kostengünstigsten Variante etwa 20 Benutzer unterstützt werden können. Mit dem Dedicated Server Äquivalent (S-2) konnten ohne weitere Anpassungen 800 Benutzer mit deutlicher besserer Antwortzeit bedient werden. Sollen mehr Benutzer unterstützt werden, müssen Anpassungen am Deployment oder der Software vorgenommen werden.

4.9 SIEBTE ITERATION

4.9.1 *Problemstellung*

Ziel dieser Iteration ist es, die Usability-Probleme aus der letzten Phase zu beheben. Da das Filtern nun von Anfang an möglich ist, enthält die Filtersuche (PS-18) sehr viele Elemente und ist auf schwächeren Endgeräten träge. In dieser Iteration wird daher die Filtersuche in Hinblick auf die Performance verbessert. Außerdem wird geprüft, wie mit einer Hilfestellung (PS-03) hervorgehoben werden kann, dass bereits eine Suche aktiv ist, die das Vorhandensein von Entitäten in der Filtersuche beeinflusst.

Anschließend wird der High-Fidelity Protoyp mit dem bereits in der vorherigen Iteration verwendeten Remote-Usability-Test und vier weiteren Editionsphilologen (P-7 bis P-10) evaluiert.

4.9.2 *Develop*

PS-18: Verbesserung der Autovervollständigung für den Filter

Die Performance der Entitätssuche soll verbessert werden. Beim Öffnen der Suchhilfe werden alle Entitäten gerendert. Dies ist nicht notwendig, da nicht zu erwarten ist, dass Anwender manuell eine große Liste unsortierter Entitäten durchsuchen. Daher soll die Liste so angepasst werden, dass die zwanzig Entitäten mit den meisten Treffern angezeigt werden.

Um dieses Problem zu lösen, kann eine asynchrone Variante von *md-autocomplete* genutzt werden. Dazu wird anstatt die gesamte Liste zu rendern, eine Methode aufgerufen, welche die anzuzeigenden Treffer bestimmt. Hierbei können die Ergebnisse dann auf 20 Treffer begrenzt werden.

Dazu werden die zwei Funktionen in Listing 4.15 implementiert. Die asynchrone Variante funktioniert dabei so, dass sie einen *Promise* erwartet. Somit bleibt das User-Interface sogar dann aktiv, wenn die Bestimmung der 20 Treffer länger andauert.

```
1    topTwentyEntities(searchTerm) {
2      this.entities = this.findMatchingEntities(searchTerm, 20)
3    }
4
5    async findMatchingEntities(term, count) {
6      if (term===undefined || term === null || term?.length === 0) {
7        return this.foundEntities.slice(0,count);
8      }
9      if (typeof term === "object" ) return term; // term is the entity
           if it is found, then we can just return
10     const lowerterm = term.toLowerCase();
11     return this.foundEntities.filter(
12       entity => entity.descriptive_name.toLowerCase().includes(
             lowerterm)
13     ).slice(0,count)
14   }
```

Listing 4.15: Optimierung Autovervollständigung Performance

PS-03: Hilfestellung für den Benutzer

Ein weiteres Problem bestand darin, dass die Entitätssuche sich nur auf die Ergebnisse bezieht, die im aktuellen Filterzustand vorkommen. Dies hat zwar den Vorteil, dass nur gültige Kombinationen gefiltert werden können, allerdings hat dies im Usability Test zu Verwirrung geführt.

Um diesen Problem zu beheben, soll ein Hinweis bei einer fehlgeschlagenen Entitätssuche ergänzt werden. Wie man in der Abbildung 4.27 auf der nächsten Seite sieht, wird bei vorhandener Volltextsuche auf diese hingewiesen und sogar die Aktion, diese zurückzusetzen, angeboten. Der Hinweis auf die Volltextsuche soll nur angezeigt werden, wenn im aktuellen Zustand eine Suche aktiv ist, die sich auf die Auswahl der Entitäten auswirkt.

Das HTML Template in Listing 4.16 wird gerendert, wenn die Autovervollständigung kein Ergebnis liefert. Die ersten drei *p*-Elemente werden nur angezeigt, wenn eine Volltextsuche (*searchquery.fulltextquery*) existiert. Im dritten Paragraph wird die Funktion zum Zurücksetzen der Volltextsuche bei Klick zugeordnet. Ein Klick auf den Link im vierten Paragraphen führt eine neue Volltextsuche nach dem eingegeben Suchbegriff aus.

```
1 <template slot="md-autocomplete-empty" slot-scope="{ term }">
2   <p v-if="searchquery && searchquery.fulltextquery !== null">
```

```
3     Hinweis Volltextsuche nach "{{ searchquery.fulltextquery }}" ist
          aktiv.
4   </p>
5   <p v-if="searchquery && searchquery.fulltextquery !== null">
6     Es werden nur Filtervorschlaege fuer vorkommende Entitaeten gezeigt
7   </p>
8   <p v-if="searchquery && searchquery.fulltextquery !== null">
9     <a @click="resetFulltext()">Volltextsuche zuruecksetzen </a>
10  </p>
11  <p>
12  Keine Referenz gefunden. Volltextsuche nach
13    <a @click="moveToFulltext()">"{{ term }}"</a> versuchen?
14  </p>
15 </template>
```

Listing 4.16: Leeres Ergebnis Template

Abbildung 4.27: Hinweis auf aktive Volltextsuche

Layout

Wie der Usability Test mit P-6 gezeigt hat, muss die Volltextsuche noch deutlicher hervorgehoben werden, damit sie erkannt wird. Dazu soll eine große Überschrift eingeführt werden, sodass klar wird, dass die drei Überschriften über Volltextsuche, Filter und Ergebnisdarstellung der gleichen Hierarchiestufe angehören. Die Volltextsuche liegt somit nicht mehr am oberen Bildschirmrand, wo sie leichter übersehen werden kann. Außerdem wird die Volltextsuche, wie die Bereiche Filter und Ergebnisdarstellung umrahmt. Um die Volltextsuche weiter herauszustellen wird sie mit einem Schlagschatten hinterlegt (*Elevation*).

Das überarbeitete Layout wird in Abbildung 4.28 auf der nächsten Seite dargestellt.

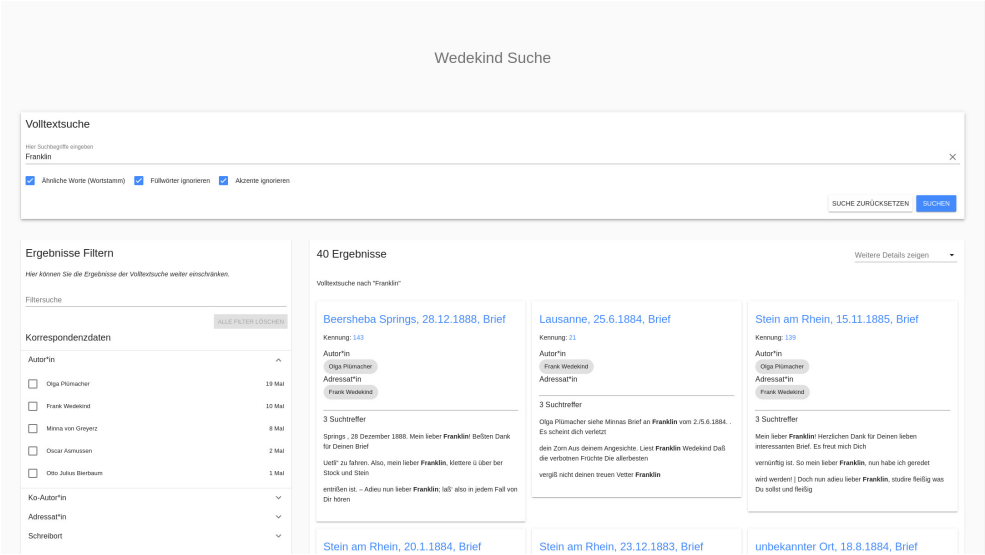


Abbildung 4.28: Das finale Layout der Suchfunktion

4.9.3 *Deliver*

Um den Prototyp zu evaluieren, wurde der Remote-Usability-Tests mit P-7, P-8, P-9 und P-10 durchgeführt. Damit die Ergebnisse vergleichbar sind, wurden keine weiteren Iterationen mehr zwischen den Tests durchgeführt. Die Erkenntnisse werden in diesem Kapitel zusammengefasst. Dazu werden die Beobachtungen aus den einzelnen Tests je Problemstellung beschrieben. Abschließend wird auf die Erwartungen und deren Erfüllung eingegangen.

Ergebnisdarstellung

Die Ergebnisdarstellung (PS-01) wurde von allen Testpersonen als übersichtlich empfunden. Nach P-7 ist die Ergebnisdarstellung „wie bei google“ und wurde von ihr als Standard bezeichnet. Es werden die wichtigsten Metadaten und Briefinformationen dargestellt (P-7, P-8). In der Vorschau ist erkennbar, ob ein Brief relevant ist und man kann entscheiden, ob man abspringen soll (P-7, P-8, P-9). Die Vorschau der Ergebnistreffer im Kontext wurde von den Testpersonen (P-7, P-8, P-9) intuitiv genutzt. Bei der Ergebnisdarstellung wird man auf die wesentlichen Dinge gelenkt, wobei das Design schlicht und ohne Ablenkungen ist (P-9). Die Ergebnisdarstellung wird als übersichtlich empfunden (P-10).

Die Möglichkeit sich referenzierte Entitäten zusätzlich anzeigen zu lassen, hat keine der Testpersonen im Usability Test verwendet. Im Feedback Gespräch wurde aber festgestellt, dass die Funktionalität praktisch ist, aber leider zu versteckt (P-8).

Kombination aus Entitäts- und Volltextsuche

Den Testpersonen wurde vor den Aufgabenstellungen nicht erläutert, wie die Suchfunktion funktioniert. Somit kann beobachtet werden, wie sie mit der Kombination aus Entitäts- und Volltextsuche (PS-02) umgehen.

Obwohl die erste Fragestellung einen Einstieg über den Entitätsfilter nahelegt (Wer hat über das Werk „Die Fürstin Russalka" geschrieben?), wurde von allen Testpersonen die Volltextsuche genutzt. Die Testperson P-8 hat den Filter auf dem Werk „Die Fürstin Russalka" bei aktiver Volltextsuche über die Checkbox ausgewählt. Alle anderen haben die Frage ausschließlich über die Volltextsuche beantwortet.

Dass die Filter von der bisherigen Suche abhängig sind (kaskadierende Filter), hat anfangs bei den Testpersonen (P-7, P-8, P-10) für Verwirrung gesorgt. P-7 und P-10 setzten zwar die Filtersuche zurück, hatten aber aufgrund einer aktiven Volltextsuche weiterhin nur eingeschränkte Filtermöglichkeiten. Dies beeinträchtigte die Funktion des Filters, da erwartet wurde, nun wieder alle Entitäten filtern zu können. Die Beobachtung der Testpersonen P-7, P-8 und P-10 legt die Vermutung nahe, dass das Problem möglicherweise an den unterschiedlichen Möglichkeiten zum Zurücksetzen (Filter zurücksetzen, Volltextsuche zurücksetzen und Suche zurücksetzten) liegt, die vom Benutzer nicht unterschieden werden konnten und mit der gleichen Erwartung („alles auf Anfang") verwendet wurden. P-7 und P-10 entdeckten den Button zum Zurücksetzen der gesamten Suche erst nach mehreren Fragestellungen.

Zur Kombination mehrerer Filter wurde betont, dass dies eine wichtige Recherchemöglichkeit bei der Forschung darstellt. Beispielsweise können dadurch Bezüge hergestellt werden wie beispielsweise: wann kommt *Max Harden* mit dem Werk *In allen Sätteln gerecht* vor (P-9).

Filtern

Hier werden die Beobachtungen zum Filter (PS-18) beschrieben, die keinen Bezug zur Kombination von Filter und Volltextsuche haben.

Die Filterboxen wurden als informativ und interessant wahrgenommen (P-8). Ein Problem stellt allerdings die Limitierung auf nur fünf Entitäten pro Filter dar (P-7, P-8). Der Hinweis auf die Filtersuche hilft hier nur bedingt (P-7 und P-9 haben die Komponente übersehen, P-8 und P-10 haben sie gefunden), da er zu unauffällig ist (P-7). Ein weiteres Problem der Beschränkung auf die fünf Entitäten mit den meisten Treffern ist, dass man die Information verliert, welche Entitäten weniger häufig im Ergebnis vorhanden sind (P-10). Das Entfernen der Funktionalität „Mehr anzeigen" und „Weniger anzeigen" hat also nicht zum gewünschten Ergebnis geführt und sollte auf Basis des Tests wieder aufgenommen werden. Dass mehr als die fünf häufigsten Entitäten relevant sein könnten, wurde im Test von P-10 angemerkt.

Für die Filtersuche müssen weitere Optionen geprüft werden. Das Layout mit Lupe (Abbildung 4.22 auf Seite 88) wurde nicht mit einem Usability Test

evaluiert und könnte eine Lösung darstellen. Mit der Erkennung der Eingabefelder hatten P-7, P-9 und P-10 Schwierigkeiten, hier könnte eine andere Darstellung des Eingabefelds oder der Hinweis auf ein Suchfeld durch die Lupe gegebenenfalls helfen.

Wenn in der Filtersuche keiner der Vorschläge ausgewählt wird, wird keine Entität gefiltert (P-10). Wenn man die Suche mit Enter bestätigt, sollte entweder der beste Treffer ausgewählt werden oder alle Treffer als Dialog angezeigt werden, um den Anwender auf die Auswahl eines Filters hinzuweisen.

Wenn es in einer Abfrage keine Auswahlmöglichkeiten gibt, weil keine Entität dieser Art vorkommt (beispielsweise keine Örtlichkeiten im Ergebnis), dann wird hierfür kein Hinweis angezeigt. Beim Aufklappen ist zwar zu sehen, dass sich etwas verändert (etwas mehr Platz wird eingenommen und der Pfeil wird umgekehrt), allerdings ist nicht eindeutig erkennbar, warum nichts angezeigt wird (P-8).

Hilfestellung für den Benutzer

Um die Usability der Suchfunktion zu erhöhen, wurden Hilfestellungen (PS-03) umgesetzt. Die Tooltips zur Erläuterung der Volltextsuchfunktionalitäten wurden von P-7 gelesen. Die Testperson P-8 hat den Hinweis auf die aktive Volltextsuche zwar gelesen, hat daraus aber einen falschen Schluss gezogen und den Filter statt der Volltextsuche zurückgesetzt. Die Annahme, dass dies ein Folgefehler der mehreren Zurücksetzen-Funktionalitäten ist, ist naheliegend. P-10 hat die im Hinweis auf die aktive Volltextsuche vorgeschlagene Aktion *Volltextsuche zurücksetzen* per Klick auf den Link intuitiv verwendet. Die Komponente zur Erläuterung der Ergebnisse und die Funktionalität von dort Filter zu entfernen, wurden von P-9 als praktisch angesehen.

Im abschließenden Feedbackgespräch mit P-7 wurde eine Möglichkeit der Hilfestellung diskutiert. Hierbei wäre eine Erläuterung der Funktionalitäten wünschenswert, die sich nicht auf Anhieb erschließen oder in unterschiedlich Editionen anders umgesetzt sind. Beispielsweise kann der Operator *OR* verwendet werden, dies erschließt sich aber nur durch ausprobieren. Hier könnte ein Popup Fenster abhilfe schaffen (P-7), indem die Volltextsuche kurz erklärt wird.

Entitätsfilter in unterschiedlichen Rollen

Um die Entitätsfilter in unterschiedlichen Rollen (PS-04) zu testen, wurden Entitäten in unterschiedlichen Rollen abgefragt. Es ist nicht sofort aufgefallen, dass Personen in unterschiedlichen Rollen auftreten (P-7). Die Filterung mit der Filtersuche nach Adressatin hat nicht intuitiv funktioniert (P-8).

Zeitselektion

Für zeitliche Filter (PS-05) wurden bereits Papierprototypen und Lösungsideen generiert. Es konnte auch im Remote-Usability-Test bestätigt werden, dass die Selektion des Zeitraums relevant ist (P-6). Bei der Eingabe eines

Briefes, würde es helfen im Umfeld einer Person in Kombination mit einer Selektion des Zeitraums zu suchen (P-6).

In der Aufgabe zur Selektion eines spezifischen Korrespondenzstücks wurde von mehreren Testpersonen (P-7, P-9) das Datum in die Volltextsuche eingegeben. Bei der weiteren Bearbeitung der Problemstellung kann dies berücksichtigt werden.

Suche in Kommentaren

Im Interview wurde von P-8 angemerkt, dass eine Suche in weiteren Teilen der Edition wünschenswert wäre. Insbesondere der Kommentar und die Annotationen. Außerdem eine Suche im Apparat, indem zusätzliche Informationen wie die Länge und Breite oder das Postkartenmotiv erfasst werden. Eine Suche im Kommentar wurde auch von P-9 als praktisch und erwünscht angegeben.

Bei der Umsetzung dieser Problemstellung (PS-06) sollte berücksichtigt werden, dass man sieht ob ein Treffer aus dem Kommentar stammt oder aus dem Text (P-8).

Flexible Fehlertoleranz

Zur Fehlertoleranz der Suche (PS-08) konnte beobachtet werden, dass die unscharfe Suche teilweise nur auf Nachfrage aufgefallen ist (P-7, P-10). Die Nachfrage hat bei P-7 dazu geführt, dass sie die Konfiguration zur Einstellung der Fehlertoleranz gefunden hat. P-8 stellte fest, dass nicht auf Anhieb klar wird, was "Füllwörter ignorieren"bedeutet. Die unscharfe Suche wurde mit dem gewählten Beispiel (Suche nach Blüten, Wortstamm Blut) von P-10 als unnötig wahrgenommen. Von P-7 wurde angemerkt, dass die Erwartung einer Wortstammsuche der linguistische Wortstamm sein sollte und nicht die aktuell umgesetzte Variante (Snowball-Stemmer).

Speicher-, referenzier- und navigierbare Suchabfragen

Diese Problemstellung wurde im Usability Test nur im Sinne der Navigierbarkeit getestet. Alle Testpersonen (P-8, P-9 und P-10) die auf Dokumente abgesprungen sind, haben intuitiv zur Suche zurückgefunden.

Hilfsmaterialen und Dokumentation

Im Usability Test kam die Anmerkung, dass Hilfsmaterialien und Dokumentationen (PS-13) sinnvoll sein können, da einige Funktionalitäten versteckt sind (P-9) und nicht ersichtlich ist, welche Funktionen die Suche bietet. Der Fokus liegt auf der Volltextsuche, die weiteren Funktionalitäten entdeckt man dagegen ohne Dokumentation vielleicht nicht.

Phrasensuche

Die Phrasensuche (PS-14) wurde nur von P-9 angewandt und hat hierbei intuitiv funktioniert. Wenn eine Hilfestellung PS-03 umgesetzt wird, sollte sie mit beschrieben werden.

Sortieren

Die Möglichkeit, die Ergebnisse zu sortieren (PS-17), wurde im Usability Test von zwei Personen (P-6, P-10) erwartet. Beide betonten, dass das Sortieren nach der zeitlichen Komponente relevant ist (P-6, P-10). Sortieren kann als Ersatz für einen zeitlichen Bereichsfilter genutzt werden (P-6).

Unerwartetes Verhalten

Der *Suche zurücksetzen* Button ist zu unauffällig (P-7). Die Testperson P-8 reflektiert im Feedback, dass sie die Beziehung mit der Filterkomponente noch nicht verstanden hat und sich noch länger mit ihr beschäftigen müsste, um das Prinzip zu verstehen.

Beim Lösen der Aufgabenstellungen konnte beobachtet werden, dass den Testpersonen der Zustand ihrer Abfrage zeitweise unklar war.

Weitere Anpassungswünsche

Im Usability Test wurden die Testpersonen dazu befragt, ob man die Suchfunktion so nutzen wollen würde, oder ob noch Anpassungen gemacht werden sollten. Hierbei wurde von P-8 und P-9 angemerkt, dass bei einem Absprung in das Korrespondenzstück der Treffer im Brief hervorgehoben werden könnte.

Ebenso wurde angemerkt, dass es besser hervorgehoben werden sollte, dass weitere Details zum Suchergebnis angezeigt werden können (P-7).

Die Filter könnten noch weiter ausgebaut werden, beispielsweise wäre Korrespondenzart wünschenswert (P-7).

Außerdem sollten die Texte in der Suchfunktion noch angepasst werden: Ergebnisse filtern impliziert, dass man erst eine Volltextsuche durchführen muss und dass man von Anfang an filtern kann, erschließt sich somit nicht (P-8).

Die Bezeichnung *Wedekind Suche* ist aus philologischer Sicht nicht gut gewählt, hier sollte *Wedekinds Korrespondenzsuche* als Begriff verwendet werden (P-9).

Mit der neuen Funktion zur Parallelansicht von Briefen in der Edition wäre die Auswahl von zwei Ergebnissen zum direkten Vergleich in der Parallelansicht wünschenswert (P-9).

Die Filterkomponente sollte ermöglichen, alle vorkommenden Entitäten anzuzeigen. Auch wenn dies die Auswahl nicht so komfortabel wie mit der Suche gestaltet, hätte man ansonsten keine Möglichkeiten, alle Entitäten anzuzeigen (P-10).

Es wurde weiterhin gewünscht, dass man sich in der Ergebnisdarstellung mehr Informationen zu einer Entität anzeigen lassen kann (P-10). Ansonsten muss der Weg über das Register gewählt werden. Manche Benutzer verwenden sogar ausschließlich die Volltextsuche (P-10).

Bugs

Beim Usability Test sind manche Frontend-Probleme aufgefallen, die in vorherigen Iterationen nicht entdeckt wurden. Wenn in der Filtersuche ein Begriff eingegeben wurde, aber keiner der Vorschläge ausgewählt wird und dann erneut in das Feld geklickt wird, so wird nicht nach dem bereits eingegebenen Text gefiltert, sondern erst wieder, nachdem ein Zeichen verändert wurde (P-8).

Ein weiteres Problem wurde im Test mit P-10 beim Responsive Design festgestellt, da das Endgerät der Testperson eine geringe Auflösung hatte. Hierbei nahm die Überschrift sehr viel Platz ein und somit sah sie beim Öffnen der Seite nur die Volltextsuche. Dies sollte angepasst werden, da der Filterbereich und die Ergebnisdarstellung nicht auf Anhieb gesehen werden. Problematisch ist auch, dass nicht einmal der Ladeindikator dargestellt wurde. Somit schien die Seite nicht zu reagieren, wenn eine Volltextsuche ausgeführt wurde und der Anwender nicht nach unten scrollt (P-10).

Die *Weitere Details zeigen* Komponente behält den Fokus, wenn man eine Auswahl getätigt hat. Wenn die Tastatur zum Scrollen verwendet wird, öffnet sich die Komponente erneut, anstatt dass die gewünschte Scrollaktion ausgeführt wird (P-8).

Layout

Außerdem sollen die Anmerkungen zum Layout zusammengefasst werden. In den Tests ist immer wieder aufgefallen, dass das Zurücksetzen der Suche nicht gefunden wurde. Hierbei könnte man die Position oder farbliche Hervorhebung des Buttons ändern. Die Tests haben deutlich gemacht, dass bei der ersten Anwendung mehrere *Zurücksetzen* Buttons verwirren, daher sollte der Filter zurücksetzen Button entfernt werden. Die Filter können in den Filtersuchkomponenten oder mit der Erläuterungskomponente weiterhin entfernt werden.

Wenn es für eine Entität keine Treffer gibt, sollte eine Information angezeigt werden, dass mit der aktuellen Suche keine Entitäten dieser Art vorhanden sind.

Der Hinweis *Hier können Sie die Ergebnisse der Volltextsuche weiter einschränken* und die Filtersuchkomponente sind untergegangen (P-7) und sollten daher besser hervorgehoben werden. Die Bezeichnung wurde von P-9 als klar formuliert bewertet.

Gesamteindruck

Im Test wurde auch der Gesamteindruck der Testpersonen abgefragt. Mit der Darstellung des Gesamteindrucks der Tester wird die Zusammenfassung des Remote-Usability-Tests abgeschlossen.

Den Testern erschienen weitaus mehr Möglichkeiten zu bestehen als sie sich intuitiv erschließen würden (P-7, P-8), die grundsätzliche Idee der Suchfunktion wurde aber als sehr gut bewertet (P-7). Die Volltextsuche wurde als schnell und intuitiv empfunden (P-8). Außerdem erschien die Volltextsuche nach Aussage von P-9 sehr transparent.

Zum Filter am Rand gibt es gegensätzliche Eindrücke. Einerseits wird dieser als transparent und verständlich wahrgenommen (P-9), andererseits als eine Funktion, die man sich erst erschließen muss (P-8).

Zusammenfassung

Im Remote-Usability-Test konnten einige Funktionalitäten als intuitiv identifiziert werden (Volltextsuche, Navigation, Ergebnisdarstellung). Andere Funktionalitäten waren in der ersten Anwendung durch die Testpersonen dagegen schwierig zugänglich (Filtersuche, Zurücksetzen der Suchfunktion).

Zu einigen Problemstellungen konnten neue Einblicke gewonnen werden, die in der Weiterentwicklung der Lösung Anwendung finden können.

Der Usability Test war so ausgelegt, dass er die Erlernbarkeit der Suchfunktion getestet hat. Um die Ergebnisse des Tests nicht zu beeinflussen wurden durch den Tester während der gesamten Aufgaben keine Hinweise gegeben. Dies hat zum Erkenntnisgewinn beigetragen, beispielsweise wurde so transparent, wie lange der „Zurücksetzen“ Button nicht entdeckt wurde. Rückblickend wäre es möglicherweise sinnvoll gewesen, kleinere Anpassungen (Ausblenden des „Filter zurücksetzen“) oder Änderungen des Layouts (farbliche Hervorhebung eines Buttons) zwischen den einzelnen Usability Tests vorzunehmen. Somit wären die Ergebnisse zwar schlechter vergleichbar gewesen, aber es hätte evaluiert werden können, ob es sich bei einigen der festgestellten Probleme um Folgefehler handelt.

Der enstandene Prototyp steht unter MIT Lizenz, das Repository ist in Anhang E auf Seite 144 beschrieben und kann unter [51] abgerufen werden.

5 FAZIT

In dieser Masterarbeit wurde mit einer Konkurrenzanalyse erhoben, welche Suchfunktionalitäten digitale (historisch-kritische) Editionen besitzen. Außerdem wurde mit einer Nutzerrecherche in Form einer Online-Umfrage untersucht, welche Funktionalitäten die Benutzer von der Suchfunktion erwarten. Hierbei konnte die Diskrepanz festgestellt werden, dass einige Funktionalitäten von den Befragten erwartet werden, aber in der Konkurrenzanalyse nicht besonders weit verbreitet waren. Dies zeigte sich beispielsweise beim Sortieren von Suchergebnissen, Fehlertoleranz in der Suche, Autovervollständigung und Wortstammsuche.

Basierend auf den konsolidierten Ergebnissen aus der Analysephase wurden Problemstellungen abgeleitet. Diese können als Einstiegspunkt für das Design entsprechender Funktionalitäten dienen, um für die identifizierten Probleme in dem jeweiligen Kontext Lösungen zu erarbeiten.

In iterativer Vorgehensweise wurde eine Suchfunktion für die Briefedition Wedekind entwickelt. Es wurden Lösungsansätze für Problemstellungen in Workshops erarbeitet und verglichen. Die Evaluation im Remote-Usability-Test dieser Lösungsansätze hat gezeigt, dass Usability-Probleme häufig erst im High-Fidelity-Prototyp identifziert werden konnten, in dem die unterschiedlichen Funktionalitäten im Zusammenspiel getestet werden konnten. Teilweise lässt sich dies durch die höhere Komplexität erklären, wenn mehrere Lösungsansätze ineinandergreifen müssen. In den Sketches wurde sich stets ausschließlich auf eine Problemstellung fokussiert. Durch die Durchführung mehrerer Iterationen konnten für die entdeckten Usability-Probleme weitere Lösungsansätze generiert und im Prototyp evaluiert werden. Die gewählte Software Architektur für den High-Fidelity-Prototyp hat sich dabei als flexibel herausgestellt und ermöglichte die schnelle Auslieferung neuer Lösungsansätze.

Es wurde gezeigt, dass die Suchfunktion für eine große Anzahl von modellierten, typischen Benutzern performant ist.

Im Remote-Usability-Test erwies sich die Volltextsuche als intuitiv nutzbar. In Kombination mit der Ergebnisdarstellung, in der eine Vorschau der Treffer dargestellt wird, konnten die Testpersonen die Relevanz der Korrespondenzstücke für ihre Suche einfach erkennen. Darüber hinaus wurde festgestellt, dass die Ergebnisdarstellung alle vom Benutzer erwarteten Informationen darstellt.

Die Kombination aus Filter- und Volltextsuche führte anfänglich zu Verwirrung und wurde teilweise erst nach Erläuterung verstanden. Auffällig war hierbei, dass die Filtersuche teilweise sogar übersehen wurden. Dass im Filter mehrere Entitäten kombiniert werden können und die Volltextsuche

weiter eingeschränkt werden kann, wurde hierbei als wichtige Funktionalität von den Testpersonen betont.

Die Darstellung von Hilfestellungen konnte dazu beitragen, dass die Transparenz zum aktuellen Suchzustand verbessert werden konnte. Dies reichte aber nicht in allen Szenarien aus, denn einige Hilfestellungen wurden nicht deutlich genug angezeigt und daher übersehen.

Für das Zurücksetzen des Suchzustandes wurden drei individuelle Funktionalitäten geschaffen, je eine zum Zurücksetzen der Volltextsuche, der Filtersuche sowie der gesamten Suche. Hierbei hat sich herausgestellt, dass die Testpersonen die Suche nicht in diesen Dimensionen trennen konnten und jeweils erwarteten, dass der gesamte Suchzustand zurückgesetzt wird. Dies führte zu Folgefehlern in der Anwendung der Suchfunktion.

Die Navigation vom Suchergebnis auf Korrespondenzstücke und zurück zur Suche über die Browsernavigation konnte von allen Testpersonen intuitiv verwendet werden. Beim Absprung in das Zieldokument wäre zukünftig eine Hervorhebung des Suchbegriffs wünschenswert.

Der Gesamteindruck der Testpersonen zur Volltextsuche wurde als schnell und intuitiv beschrieben. Der kaskadierende Filter wurde nur teilweise intuitiv entdeckt und muss durch Probieren oder Zuhilfenahme der Dokumentation erlernt werden. Dann stellt es aber eine von den Anwendern als praktisch empfundene Funktionalität dar, da verschiedene Entitäten kombiniert gefiltert werden können.

Es wurden bereits für weitere Problemstellungen Lösungsansätze entwickelt, welche noch nicht im High-Fidelity Protoypen umgesetzt wurde. Im Remote-Usability-Test wurden manche davon als wünschenswert angesprochen, darunter die Möglichkeit die Ergebnisse zu sortieren, eine zusätzliche Suche des Kommentars und die Umsetzung eines Zeitfilters.

Im Usability Test wurde vor allem die Erlernbarkeit überprüft, welche gerade bei Nutzern digitaler (historisch-kritischer) Editionen besonders relevant ist. Die Prüfung zusätzlicher Dimensionen der Usability könnten interessant sein, insbesondere wenn die Edition im Rahmen eines Forschungsprojektes genutzt wird und somit über einen längeren Zeitraum verwendet wird. Beispielsweise könnte die Effizienz durch das Erlernen komplexerer Funktionalitäten gesteigert werden.

Um mehrere Lösungsansätzen zu vergleichen, wäre auch die Durchführung eines A/B-Tests in Frage gekommen. Dies könnte mit einer größeren Anzahl an Testern bei der Weiterentwicklung der Suchfunktion in Betracht gezogen werden.

Zusammenfassend wurden die für die Nutzer relevantesten Problemstellungen und unterschiedliche Lösungsansätze dafür identifiziert und einige davon in einem Prototyp bereits kombiniert umgesetzt. Das nutzerzentrierte Vorgehen hat sich als hilfreich herausgestellt und sollte bei der Weiterentwicklung beibehalten werden, um die intuitive Verwendbarkeit sicherzustellen. Dabei muss darauf geachtet werden, dass neue Funktionalitäten im Gesamtkontext des Prototyps und nicht separat davon evaluiert werden. Weiterhin sollten die bestehenden Funktionalitäten optimiert werden, sodass

diese einfacher zu nutzen sind. Die Suchfunktion sollte von den Anwendern produktiv genutzt werden, sodass sie in realistischen Szenarien mit der Anwendung interagieren. Dadurch kann die Anwendung mit ihrem kontinuierlichen Feedback und ihren Erfahrungen über einen längeren Zeitraum zusätzlich verbessert werden.

Teil II

ANHANG

ANHANG A

A.1 ANSCHREIBEN

Das nachfolgende Anschreiben wurde für die Bewerbung der Umfrage Abschnitt 3.4 auf Seite 32 verwendet:

> «Anrede»,
> mein Name ist Matthias Fabinski. Ich studiere Informatik an der der Hochschule Darmstadt und schreibe derzeit meine Masterarbeit bei Prof. Dr. Uta Störl und Tobias Holstein.
> In diesem Rahmen führen wir eine Umfrage durch. Ihre Antworten sollen helfen, die Suchfunktionalitäten in digitalen Editionen zu verbessern. Ich schreibe Sie an, wegen Ihrer Beteiligung an «Grund». Wir würden uns sehr freuen, wenn Sie sich ca. 15 Minuten Zeit nehmen, um den nachfolgenden Fragebogen online auszufüllen:
> https://briefdb3.fbi.h-da.de/?lang=de
> Die Umfrage dient ausschließlich zu Forschungszwecken und alle Antworten werden absolut anonym und vertraulich behandelt. Nach Beendigung der Umfrage werden wir die Ergebnisse als OpenData veröffentlichen.
> Leiten Sie diese Umfrage gerne an interessierte Kolleginnen und Kollegen aus Ihrem Team weiter.
> Sollten Sie Fragen zur Umfrage haben, können Sie sich jederzeit per Mail an mich (matthias.fabinski@stud.h-da.de) wenden.
> Vielen Dank für Ihre Unterstützung
> Matthias Fabinski

Die verwendete URL sollte nicht verwendet werden, da es sich um eine interne URL handelt. Bei künftigen Umfragen sollte `https://test.briefedition.wedekind.h-da.de/` als Einstieg genutzt werden.

A.2 FRAGEBOGEN DEUTSCH

Umfrage zu Suchfunktionen in digitalen (historisch-kritischen) Editionen

Zwischengespeicherte Umfrage laden Sprache: Deutsch

0%

Sprache: Deutsch Sprache ändern

Umfrage zu Suchfunktionen in digitalen (historisch-kritischen) Editionen

Diese Umfrage wird im Rahmen der Verbesserung und Erweiterung von Suchfunktionen digitaler (und historisch-kritischer) Editionen durchgeführt. Ihre Antworten helfen die Prioritäten bestimmter Suchfunktionalitäten und Filterkriterien festzulegen, die wir anhand von Literaturrecherchen und einem Vergleich von 42 digitalen Editionen* erhoben haben. Des Weiteren untersuchen wir welche Suchfunktionalitäten in digitalen Editionen fehlen bzw. brauchbar wären.

Die Umfrage wird im Rahmen der Masterarbeit von Matthias Fabinski durchgeführt. Die Ergebnisse werden nach Abschluss der Masterarbeit zur Verfügung gestellt, so dass diese in der Forschung und Weiterentwicklung weiterer digitaler Editionen verwendet werden können. Die ungefähre Bearbeitungszeit dieser Umfrage beträgt 15 Minuten. Wenn Sie möchten, können Sie am Ende der Umfrage eine E-Mail Adresse hinterlassen um über die Veröffentlichung der Umfrageergebnisse und weitere Umfragen informiert zu werden.

Wir interessieren uns sowohl für digitale (historisch-kritische) Editionen mit denen sie Erfahrung gesammelt haben als auch für Editionen an denen Sie mitgewirkt haben, beispielsweise als Editor. Der Fragebogen ist dynamisch aufgebaut und basierend auf Ihren Antworten können Folgefragen oder zusätzliche Antwortoptionen angezeigt werden. In den offenen Fragen können Sie gerne sehr detailliert mit Beispielen, Anekdoten oder weiteren Eindrücken antworten. Die qualitativ erfassten Einblicke helfen uns, ein besseres Verständnis der typischen Suchprobleme in digitalen (historisch-kritischen) Editionen zu bekommen.

*Für den Aufbau dieser Umfrage wurde bereits eine umfangreiche Voranalyse durchgeführt. Briefeditionen und DFG-geförderte Editionen aus dem ***Catalogue of Digital Editions von*** Franzini, G. (10.5281/zenodo.1161425) wurden auf Ihre Funktionalitäten im Bezug auf die Suchfunktionen untersucht.

In dieser Umfrage sind 33 Fragen enthalten.

Weiter

Allgemeine Angaben

*Was ist Ihr Geschlecht?

Bitte wählen Sie eine der folgenden Antworten:

- männlich
- weiblich
- anderes
- Keine Antwort

*Wie alt sind Sie?

Bitte wählen Sie eine der folgenden Antworten:

- 0-30 Jahre
- 31-40 Jahre
- 41-50 Jahre
- 51-60 Jahre
- 61+ Jahre
- Keine Antwort

Welche berufliche Tätigkeit üben Sie derzeit aus?

Was ist Ihr Forschungsfeld?

Bitte wählen Sie eine der folgenden Antworten:

- Geschichte
- Literatur
- Archeologie
- Linguistik
- Kunstgeschichte
- Theologie
- Informationswissenschaft
- Philosophie
- Informatik
- Germanistik
- Kulturwissenschaft
- Sonstiges:

*Was ist Ihr höchster akademischer Grad/Titel?

Bitte wählen Sie eine der folgenden Antworten:

- Bachelor
- Master
- Doktor
- Professor
- Keiner
- Keine Antwort
- Sonstiges:

Wie viele Jahre Erfahrung haben Sie mit der Arbeit an digitalen (historisch-kritischen) Editionen?

In dieses Feld dürfen nur Zahlen eingegeben werden.

Weiter

Ihr Bezug zu digitaler (historisch-kritischen) Edition

Bitte wählen Sie, welche der nachfolgenden Aussagen auf Sie zutreffen.

Bitte wählen Sie die zutreffenden Antworten aus:

- [ ] Sie sind Editor einer oder mehrerer digitaler (historisch-kritischen) Editionen
- [ ] Sie sind Benutzer einer oder mehrerer digitaler (historisch-kritischen) Editionen
- [ ] Sie nutzen digitale (historisch-kritische) Editionen im Rahmen einer wissenschaftlichen Arbeit bzw. in der Forschung
- [ ] Sonstiges:

Bitte wählen alle Dokumenttypen mit denen Sie sich in digitalen (historisch-kritischen) Editionen beschäftigen.

Bitte wählen Sie die zutreffenden Antworten aus:

- [ ] Manuskripte
- [ ] Briefe
- [ ] Drucke
- [ ] Bücher
- [ ] Kodex
- [ ] Tafel
- [ ] Gedruckte Bücher
- [ ] Tagebücher
- [ ] Weitere

Fragen zu Ihrer digitalen (historisch-kritischen) Edition

Arbeiten Sie mit einer bestimmten digitalen (historisch-kritischen) Edition besonders intensiv, beantworten Sie bitte die nachfolgenden Fragen. Andernfalls können Sie diese Fragengruppe überspringen und (am Ende der Seite) mit der nächsten fortfahren.

Welche Einstiegsmöglichkeiten bietet die digitale (historisch-kritische) Edition um ein Dokument auszuwählen bzw. zu finden?

Bitte wählen Sie die zutreffenden Antworten aus:

- [ ] Register
- [ ] Verlinkung
- [ ] Suchfunktion
- [ ] Zufälliges Dokument
- [ ] Schlagworte
- [ ] Weitere

Welche Filterkriterien bietet die Suchfunktion Ihrer digitalen (historisch-kritischen) Edition?

Bitte wählen Sie die zutreffenden Antworten aus:

- [ ] Absender
- [ ] Absendeort
- [ ] Empfänger
- [ ] Empfangsort
- [ ] Korrespondent
- [ ] Ort
- [ ] Datum
- [ ] Zeitraum
- [ ] Sprache
- [ ] Weitere

Besitzt die Suchfunktion Ihrer digitalen (historisch-kritischen) Edition folgende Funktionalität:

Bitte wählen Sie die zutreffenden Antworten aus:

- [ ] Suchfeld direkt auf Startseite
- [ ] Volltextsuche
- [ ] Erweiterte Suchoptionen / Filter
- [ ] Treffer Sortierbar
- [ ] Wortstammsuche
- [ ] Autovervollständigung im Suchfeld
- [ ] Fehlertoleranz der Suche (Ähnliche Suchbegriffe werden gefunden)
- [ ] Suchergebnisse exportierbar
- [ ] Darstellung der Trefferrelevanz
- [ ] Wildcards (Platzhalter, häufig * oder ?)
- [ ] Weitere

Wozu nutzen Sie bei Ihrer Forschung Ihre digitale (historisch-kritische) Edition?

Welche Rolle spielt dabei die Suchfunktion?

Aus welchem Zeitraum sind die Dokumente in Ihrer digitalen (historisch-kritischen) Edition?

Bitte nennen Sie den Zeitraum in dem sich die Entstehung des Korpus Ihrer digitalen (historisch-kritischen) Edition befindet. Beispiel: ***1864-1918*** oder ***16 Jhd.***)

Weiter

Fragen zu Suchfunktionalität in digitalen (historisch-kritischen) Editionen

Bitte beantworten Sie die nachfolgenden Aussagen. Bei einer digitalen (historisch-kritischen) Edition würde ich die folgenden Funktionalität erwarten:

	Stimme voll und ganz zu	Stimme zu	Weder noch	Stimme nicht zu	Stimme über-haupt nicht zu
Volltextsuche	○	○	○	○	○
Filter	○	○	○	○	○
Treffer sortierbar	○	○	○	○	○
Wortstammsuche	○	○	○	○	○
Autovervollständigung im Suchfeld	○	○	○	○	○
Fehlertoleranz der Suche (Ähnliche Suchbegriffe werden gefunden)	○	○	○	○	○
Suchergebnisse exportierbar	○	○	○	○	○
Darstellung der Trefferrelevanz	○	○	○	○	○
Wildcards (Platzhalter, häufig * oder ?)	○	○	○	○	○

Bitte beantworten Sie die nachfolgenden Aussagen. Bei der Darstellung der Suchergebnisse sind die nachfolgenden Metadaten relevant:

	Stimme voll und ganz zu	Stimme zu	Weder noch	Stimme nicht zu	Stimme überhaupt nicht zu
Empfänger	○	○	○	○	○
Datum	○	○	○	○	○
Autor	○	○	○	○	○
Ort	○	○	○	○	○
Titel	○	○	○	○	○
Absender	○	○	○	○	○
Zusammenfassung (knappe Zusammenfassung des Inhalts)	○	○	○	○	○
Dokumententyp (Beispiel: Brief oder Tagebucheintrag)	○	○	○	○	○
Dokument ID (Eindeutiges Kennzeichen des Dokuments)	○	○	○	○	○

Welche Suchfunktionalitäten schätzen Sie an digitalen (historisch-kritischen) Editionen?

Welche Suchfunktionalitäten vermissen Sie bei digitalen (historisch-kritischen) Editionen?

Wie könnte die Suchfunktion digitaler (historisch-kritischer) Edition aus Ihrer Sicht verbessert werden?

Weiter

Fragen zur Forschung mit digitalen (historisch-kritischen) Editionen

Beschreiben Sie eine Forschungsfrage die Sie mit einer digitalen (historisch-kritischen) Edition untersucht haben.

Welche Rolle hat die Suchfunktion bei der Untersuchung Ihrer Forschungsfrage gespielt?

Welche Herausforderungen hatten Sie mit der Suchfunktion und wie sind Sie damit umgegangen?

Wie hätte die Suchfunktion Sie besser unterstützen können?

Weiter

Abschluss

Haben Sie weitere Anmerkungen zu Suchfunktionen in digitalen (historisch-kritischen) Editionen?

Haben Sie Kommentare oder Anregungen zur durchgeführten Umfrage?

Wenn Sie möchten können Sie uns gerne die URLs zu Ihren digitalen (historisch-kritischen) Editionen nennen.

Weiter

Mitteilung der Ergebnisse und Möglichkeit für erneute Befragung

Vielen Dank für Ihr Interesse und Ihre Beteiligung an dieser Umfrage!

Wenn Sie möchten, können Sie eine E-Mail Adresse hinterlassen um über die Veröffentlichung der Umfrageergebnisse und weitere Umfragen informiert zu werden.

Die Angabe der E-Mail ist freiwillig

Absenden

A.3 FRAGEBOGEN ENGLISCH

Survey on search functions in digital (scholarly) editions

Load unfinished survey Language: English

0%

Language: English Change the language

Survey on search functions in digital (scholarly) editions

This survey is being conducted as part of the improvement and expansion of search functions for digital (scholarly) editions editions. Your answers will help us to prioritise certain search functions and filter criteria, which we have established based on literature research and a comparison of 42 digital (scholarly) editions*. Furthermore, we investigate which search functionalities are missing or could be useful in digital (scholarly) editions.

The survey is conducted in the context of the master thesis of Matthias Fabinski. The results will be made available after completion of the master thesis, so that they can be used in research and further development of digital (scholarly) editions. The approximate processing time for this survey is 15 minutes. If you wish, you can leave an email address at the end of the survey to be informed about the publication of the survey results and other surveys.

We are interested in digital (scholarly) editions you have used and gained experience with, as well as editions that you created, e.g., as part of the editing team, or as a reviewer. The questionnaire is dynamic, based on your answers follow up questions and/or additional answer options may be shown. Please feel free to provide detailed answers, examples, anecdotes or other insights in open-questions. The qualitative insight can help us to gain a better understanding of everyday search problems in digital (scholarly) editions.

*An extensive preliminary analysis has been carried out beforehand to build this survey. The basis for the preliminary analysis were editions from the ***Catalogue of Digital Editions*** by Franzini, G. (10.5281/zenodo.1161425). They were selected with a focus on editions presenting letters and DFG-funded editions to extract functionalities and criteria of search functions.

There are 33 questions in this survey.

Next

General information

*What is your gender?

Choose one of the following answers

- male
- female
- other
- No response

*How old are you?

Choose one of the following answers

- 0-30 years
- 31-40 years
- 41-50 years
- 51-60 years
- 61+ years
- No response

What kind of work are you currently doing?

What is your field of research?

Choose one of the following answers

- History
- Literature
- Archeology
- Linguistics
- Art History
- Theology
- Information Science
- Philosophy
- Computer Science
- German Studies
- Cultural Studies
- Other:

*What is your highest academic degree?

Choose one of the following answers

- Bachelor
- Master
- Doctor
- Professor
- None
- No response
- Other:

How many years of experience do you have in working on digital (scholarly) editions?

Only numbers may be entered in this field.

Next

Your releation to digital (scholarly) editions

Please select which of the following statements apply to you.

Check all that apply

- [ ] You are editor of one or more digital (scholarly) editions
- [ ] You are user of one or more digital (scholarly) editions
- [ ] You use digital (scholarly) editions in the context of a scientific work or research
- [ ] Other:

Please select all document types you are working with in digital (scholarly) editions.

Check all that apply

- [ ] Manuscript
- [ ] Letter
- [ ] Print
- [ ] Book
- [ ] Codex
- [ ] Tablet
- [ ] Printed book
- [ ] Diary
- [ ] Additional

Questions about your digital (scholarly) edition

If you work intensively with a particular digital (scholarly) edition, please answer the following questions. Otherwise you can skip this group of questions and continue with the next one (go to next at the and of this page).

Which access possibilities does the digital (scholarly) edition offer to select or find a document?

Check all that apply

- [ ] Register
- [ ] Link
- [ ] Search function
- [ ] Random document
- [ ] Keywords
- [ ] Additional

What filter criteria does the search function of your digital (scholarly) edition offer?

Check all that apply

- [ ] Sender
- [ ] Dispatch location
- [ ] Recipient
- [ ] Place of receipt
- [ ] Correspondent
- [ ] Place
- [ ] Date
- [ ] Period
- [ ] Language
- [ ] Additional

Does the search function of your digital (scholarly) edition have the following functionality:

Check all that apply

- [ ] Search field directly on start page
- [ ] Full text search
- [ ] Advanced search options / filters
- [ ] Hits Sortable
- [ ] Word root search
- [ ] Auto-complete in the search field
- [ ] Approximate string matching (match the search pattern approximately)
- [ ] Search results exportable
- [ ] Representation of the hit relevance
- [ ] Wildcards (placeholders, often * or ?)
- [ ] Additional

What do you use your digital (scholarly) edition for in your research?

What is the role of the search function for your research?

From which period are the documents in your digital (scholarly) edition?

Please indicate the period of time in which the corpus of your digital (scholarly) edition was created. Example: ***1864-1918*** or ***16th century***).

Next

Questions about search functionality in digital (scholarly) edition

Please answer the following statements. With a digital (scholarly) edition I would expect the following functionality:

	Strongly Agree	Agree	Neither Agree Nor Disagree	Disagree	Strongly Disagree
Full text search					
Filter					
Results sortable					
Word stem search					
Auto-complete in the search field					
Approximate string matching (match the search pattern approximately)					
Search results exportable					
Display of the results relevance					
Wildcards (placeholders, often * or ?)					

Please answer the following statements. The following metadata are relevant for the presentation of the search results:

	Strongly Agree	Agree	Neither Agree Nor Disagree	Disagree	Strongly Disagree
Receiver					
Date					
Author					
Place					
Title					
Sender					
Summary (short summary of the documents contents)					
Document type (Example: letter or diary entry)					
Document ID (unique identifier for this document)					

Which search functionalities do you appreciate in digital (scholarly) editions?

Which search functionalities are missing in digital (scholarly) editions?

How could the search function of digital (scholarly) editions be improved from your perspective?

Next

Questions about research with digital (scholarly) editions

Please describe a research question you have investigated with a digital (scholarly) edition.

What role did the search function play in the investigation of your research question?

What challenges did you have with the search function and how did you deal with them?

How could the search function have better supported you?

Next

Closing

Do you have further comments on search functions in digital (scholarly) editions?

If you like you can leave us the URLs to your digital (scholarly) editions.

Next

Do you have any comments or suggestions regarding the survey?

Communication of the results and possibility for more surveys

Thank you for your interest and your contribution to this survey!

If you wish, you can leave an e-mail address to be informed about the publication of the survey results and other surveys.

The submission of your e-mail is completely voluntary

Submit

B

ANHANG B

B.1 NOTIZEN FEEDBACK PERSÖNLICHES GESPRÄCH ZU ITERATION 1

Das Gespräche wurde mit P-1, einer Germanistikstudentin mit Bachelorabschluss, geführt. Ziel des Gesprächs war es Feedback einzuholen. Die nachfolgenden Gesprächsnotizen wurden aufgezeichnet. Sie beziehen sich auf die Sketches in Abbildung 4.4 auf Seite 53:

- Der zweite Vorschlag: Teil einer Korrespondenz anzeigen ist praktisch; Problem bei der Suche nach einer Korrespondenz von Theodor Storm; Briefe waren da nicht in der richtigen Reihenfolge
- Beim vierten: Man kann nicht erkennen wonach man gesucht hat (Gemeint war: man sieht nicht den Treffer im Kontext)
- Genauso beim dritten
- Präferenz: Die Kartendarstellung, mit Umschaltung (Kombination aus Abbildung 4.4a auf Seite 53 und Abbildung 4.4e auf Seite 53)

Zu den Sketches aus Abbildung 4.5 auf Seite 54:

- Der erste Vorschlag scheint übersichtlich und schnell
- Der dritte Vorschlag ist unkompliziert und einfach, aber funktioniert das auch, wenn man kein genaues Datum kennt?
- Die fünfte Vorschlag sieht unübersichtlich aus
- Das vierte funktioniert nicht ohne genaues Datum und eine Zeitfilterbox, in der man die Monate durchklickt ist vielleicht nicht ganz so zielführend und relativ zeitintensiv, wenn man nach Dokumenten aus einem anderen Jahrhundert sucht

Zu den Sketches aus Abbildung 4.3 auf Seite 52:

- Das dritte ist sinnvoll und hilfreich, weil es eine gute Kombi aus dem ersten und dem zweiten ist
- Die Symbole könnten verwirrend sein, wenn keinen guten gewählt werden

B.2 NOTIZEN FEEDBACK TELEFONAT ZU ITERATION 1

Das Gespräche wurde mit P-2, einem Germanistikstudenten mit Bachelorabschluss, geführt. Ziel des Termins war es Feedback einzuholen. Die nachfolgenden Gesprächsnotizen wurden aufgezeichnet. Sie beziehen sich auf die Sketches in Abbildung 4.4 auf Seite 53:

- Beim vierten: Damit könnte man besser sortieren
- Beim ersten: Man sieht alles relevante auf einen Blick
- Beim ersten: Die Vorschau (das Faksimile) könnte unnötig sein, Vielleicht lieber ein Symbol anzeigen
- Beim fünften: Umschalten könnte Sinn machen, dass man mehr Details anzeigen kann wenn nötig
- Die Kartendarstellung scheint die sinnvollste Darstellung zu sein

B.3 FEEDBACK ZU ITERATION 3

Die Guerillatests beziehen sich auf Rückmeldung zur Verwendung des Prototyps.

- Beim Start sind mir die möglichen Funktionalitäten unklar.
- Man kann nur durch Volltextsuche die Suche starten und nicht direkt über die Suche über einen Filter.
- Vorschläge für die Suche, z. B. beim Tippen wären hilfreich, wenn man noch nicht weiß, wonach man genau suchen möchte.
- Die optische Darstellung der Ergebnisse ist suboptimal, da sich die Größe der Karten unterscheidet.
- Rechtschreibfehler Autor statt Author
- Wieso ist der Autor blau markiert?
- Optisch ansprechender wäre es, wenn Autor und Adressat nebeneinander dargestellt wären und die Richtung der Korrespondenz z.B. mit Pfeil dazwischen (Autor -> Adressat) dargestellt wird
- Die Überschrift „Suchtreffer“ wäre in fettem Stil besser, da es derzeit zu wenig präsent auf der Ergebniskarte erscheint.
- Generelle wäre es besser, wenn die Überschriften fett oder unterstrichen wären.
- Textausschnitte wären besser in einer Form, bei der erkennbar ist, dass es sich um einen Ausschnitt handelt („[...] Text [...]“)

- Die Position des Treffers im Text wäre eine interessante Information (Konkrete Zeile oder etwa Anfang Mitte oder Ende des Gesamttextes)
- Der direkte Absprung an die Textstelle des Funds wäre hilfreich
- Intuitiver wäre ein Absprung durch Klick auf den Text anstatt durch Klick auf die Überschrift
- Klick auf Informationen wie Autor oder Adressat sollte zum Filter auf diesen führen
- Die Anzahl der Treffer im jeweiligen Text wäre interessant, also wie oft wurde ein Wort in diesem Text gefunden.
- Ein Hinweis auf nicht-exakte Treffer in den Ergebnissen wäre gut (Blut statt Blüten!)
- Insgesamt wäre es hilfreich, wenn die Länge des Textausschnittes konfigurierbar wäre, d.h. wieviel vom Text vor und nach dem Treffer sieht man auf der Ergebniskarte
- Wieso werden „Referenzierte Orte“ o.a. angezeigt, wenn keine Info hinterlegt ist? Die Überschrifft verwirrt so eher.
- Wieso kann ich nicht nach Rolle (Adressat oder Autor) filtern sondern nur insgesamt nach Person? Was wird da gesucht?
- Donald Wedekind fehlt unter „Person“ -> Wieso? (siehe Screenshots)
- Wonach bestimmen sich die Werte der Suchhilfe?
- Sucht Filter auf Person nur nach Autor? Wenn ich nach Frank Wedekind suche, ergibt das Ergebnis keinen Sinn
- Ich habe mittlerweile verstanden, dass „Person“ nur auf die „referenzierten Personen“ im Text sucht. Das verwirrt trotzdem, weil Autor und Adressat ja grundsätzlich auch Personen sind. Eine detailliertere Abgrenzung wäre hilfreich, um Verwirrung zu vermeiden.
- Wieso kann man „keine Angabe“ nicht auch filtern, um beispielsweise die Ergebnissmenge weiter einzuschränken?
- Es ist unpraktisch, wenn man mehrere Optionen gefiltert hat und dann merkt, dass man nur noch einen Filter anwenden will. Dann müssen zuerst alle Filter wieder gelöscht werden und der Filter neu gesetzt werden.
- Es wäre praktisch, wenn ich die Sachen, die ich gefiltert hab, auch automatisch in der Karte sehe: Filter auf Events -> Anzeige der Events.
- Die Ergebnisse auf Filter Örtlichkeit verstehe ich nicht - wenn auf mehrere gefiltert ist, wird kein Ergebnis geliefert.

- Es wäre interessant, wenn man sieht, welche Wörter insgesamt gefunden wurden und man danach filtern könnte (bloß, bloßes, Blöße)
- Außerdem wäre es besser, wenn zum Suchergebnis ähnlichere Wörter weiter oben stünden. Also Begriffe mit anderem Worstamm sollten niedriger priorisiert werden.

Mobiles Endgerät:

- Rechtschreibung: Autor/in (ohne h)
- Gut: statt „Haus“ findet er auch Hause, Häuser etc
- Gut: Filter funktionieren, Anzeige etwas seltsam bzw. noch nicht fertig? (Siehe Bild) –> Bild zeigt leere „Bobble“ –> Problem im Datenmodell, ist bekannt
- Verweise zu referenzierten Personen als Klick gewünscht
- Sehr allgemeine Begriffe finden kein Ergebnis (wie mein, dein, ich). Vermutlich weil es zu viele Treffer gibt? Kann das ggf. irgendwie kenntlich gemacht werden? Es könnte ja mitunter auch relevant sein, wenn man z.B. die verwendeten Anreden untersucht
- Inwiefern werden immer alle Ergebnisse gelistet? Z.B. ist „i“ (für „Ich“) auch im Abschnitt zu finden wenn man nach „schlecht“ sucht, aber wird nicht angezeigt wenn man nach „i“ sucht. Dafür kommt dort eine Ortsangabe.
- Dass / daß liefern beide ein seltsamtes Ergebnis, aber scheinen zumindest den gleichen Filter zu haben.
- Generell ist die Darstellung visuell schön: schlichte und seriöse Gestaltung mit dem Kopf an der Kopfseite und die Farbwahl, wichtige Dinge sind farblich abgesetzt.

C

ANHANG C

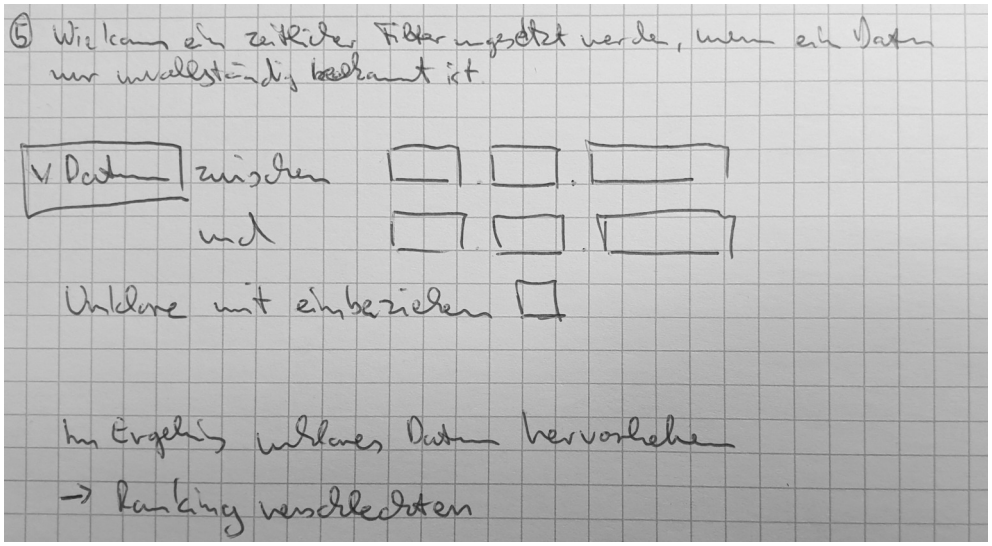
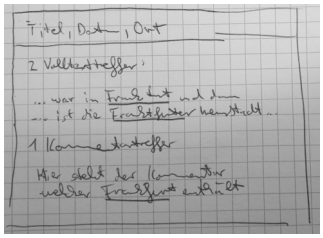
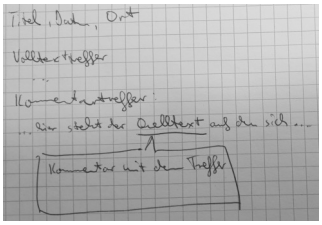


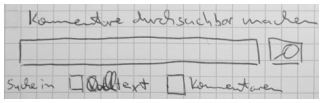
Abbildung C.1: Zeitselektion PS-05



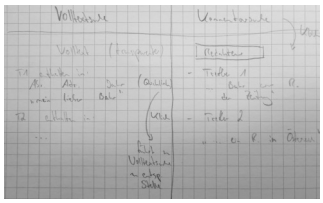
(a) Sketch1



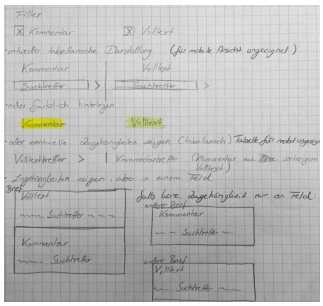
(b) Sketch2



(c) Sketch3

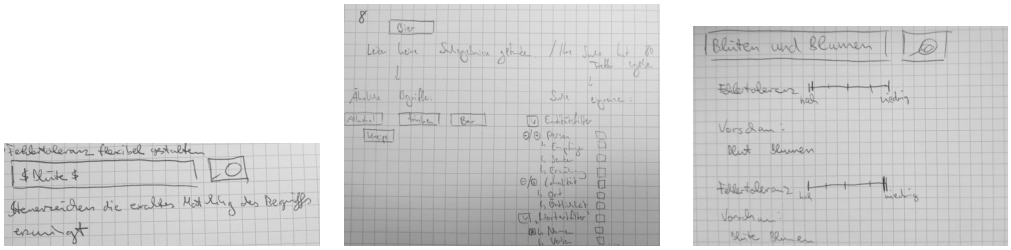


(d) Sketch4



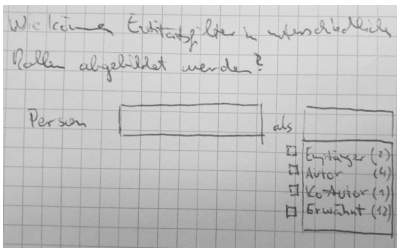
(e) Sketch5

Abbildung C.2: Suche in Kommentaren PS-06

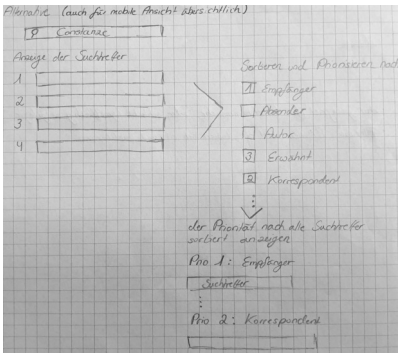


(a) Sketch1 (b) Sketch2 (c) Sketch3

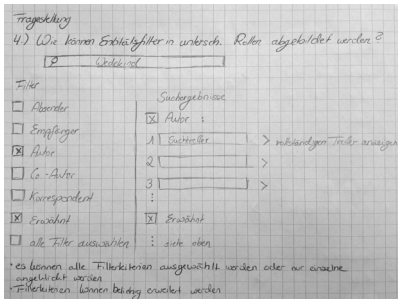
Abbildung C.3: Flexible Fehlertoleranz PS-08



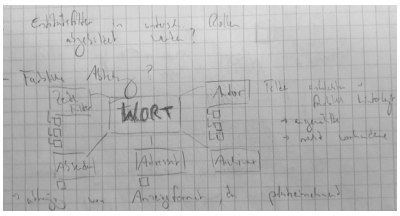
(a) Sketch1



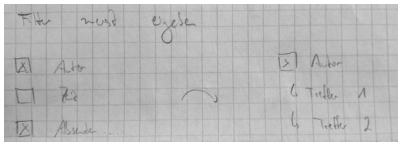
(b) Sketch2



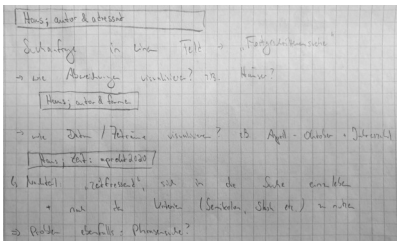
(c) Sketch3



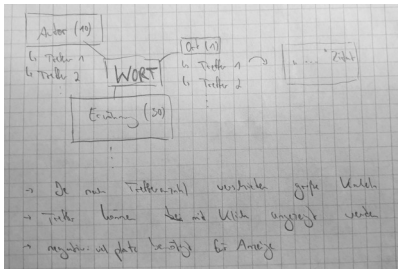
(d) Sketch4



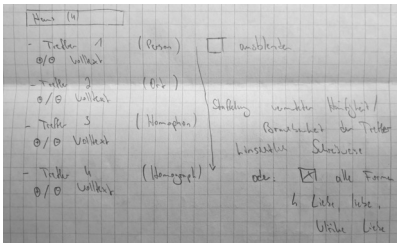
(e) Sketch5



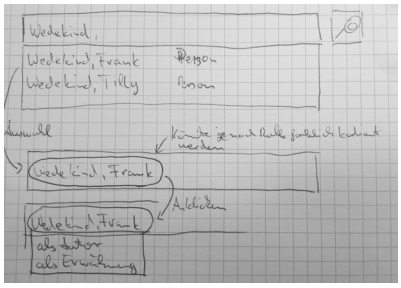
(f) Sketch6



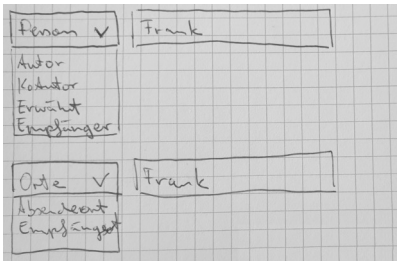
(g) Sketch7



(h) Sketch8

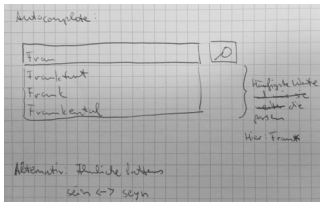


(i) Sketch9

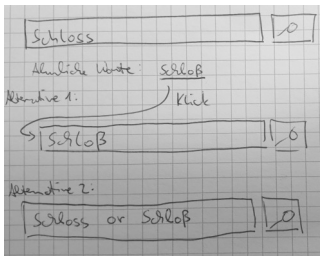


(j) Sketch10

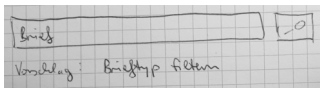
Abbildung C.4: Entitätsfilter in unterschiedlichen Rollen PS-04



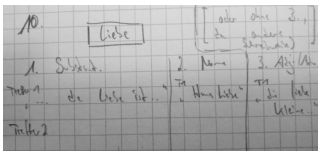
(a) Sketch1



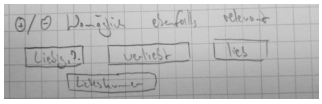
(b) Sketch2



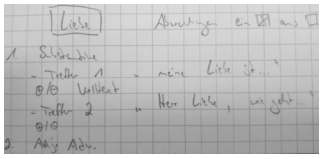
(c) Sketch3



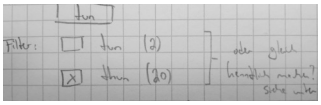
(d) Sketch4



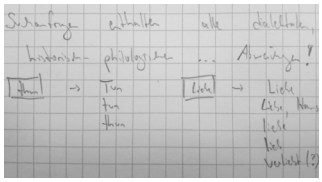
(e) Sketch5



(f) Sketch6

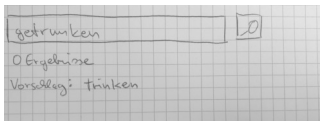


(g) Sketch7



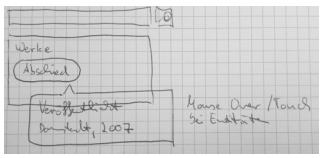
(h) Sketch8

(i) Sketch9

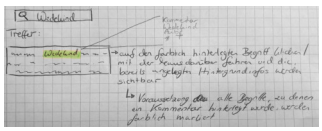


(j) Sketch10

Abbildung C.5: Vorschlag alternativer Suchbegriffe PS-10

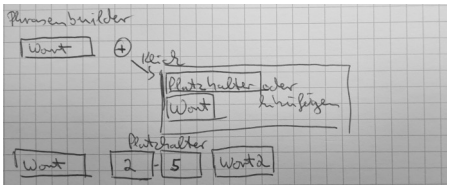


(a) Sketch1

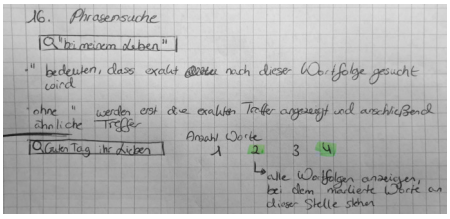


(b) Sketch2

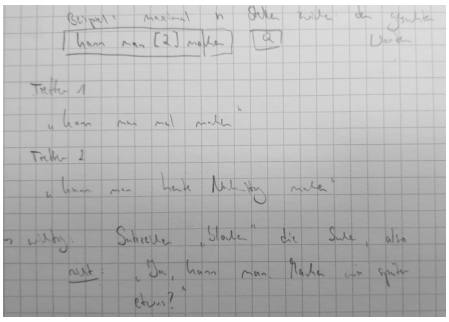
Abbildung C.6: Erschließung von Hintergrundinformationen PS-12



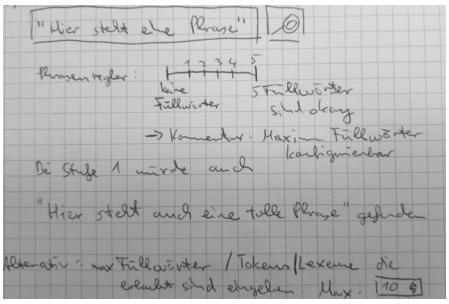
(a) Sketch1



(b) Sketch2



(c) Sketch3



(d) Sketch4

Abbildung C.7: Phrasensuche PS-14

D

ANHANG D

D.1 INTERVIEW UND AUFGABENSTELLUNGEN

Usability Test Termin in 3 Parts. Part 1 Interview, Part 2 Aufgaben, Part 3 Feedback-Interview je max. 15 Minuten. Ziel: 45 Minuten

Interview

- Welchen Bezug haben Sie zur Briefedition Wedekind?
- Wie lange arbeiten sie bereits mit der Edition?
- Welche Rolle haben sie bei der Arbeit mit der Edition?
- Wozu nutzen Sie die Suchfunktion der Edition?
- Welche Funktionalitäten erwarten sie bei der Suchfunktion?
- Welche Funktionalitäten haben sie dabei in der Vergangenheit vermisst?

Aufgaben

Erst jetzt wird die Seite des Prototypen geöffnet, damit die Funktionalitäten das Interview nicht beeinflussen.

Nachfolgend die Zugangsdaten:

- `https://wedekind.cloudf.de/`
- User: frank
- Passwort: volltext
- Wer hat über das Werk „Die Fürstin Russalka“ geschrieben?
- In welchen Korrespondenzen wird „Flügel“ als Instrument, in welchen als Körperteil eines Vogels verwendet?
- Finden sie die Korrespondenz:
  „Harry Graf Kessler an Frank Wedekind, 23.10.1906. Frank Wedekinds Korrespondenz digital. `http://briefedition.wedekind.h-da.de` (12.11.2020).“
- Von wem erhielt Tilly Wedekind Korrespondenzen? Welche Korrespondenzformen waren das?

- An welchen drei Personen hat Frank Wedekind am häufigstens Briefe geschrieben?
- Im Zusammenhang mit welchen Örtlichkeiten wird Wein erwähnt?
- In welchen Korrespondenzen werden „Blüten“ erwähnt?
- Wird die Redensart „Reinen Wein“ einschenken genutzt?

Optional (wenn noch Zeit ist) Hatten Sie weitere Fragestellungen in letzter Zeit, die sie mit der Suche untersuchen könnten?

Feedback

- Was ist ihr Gesamteindruck wenn Sie die Suchfunktion verwenden?
- Enthält die Suchfunktion die erwartenden und benötigten Funktionen?
- Entspricht die Bedienung ihrer Erwartung?
- Welche Funktionen waren ihnen unklar?

  Optional: Wie hätte man es klarer Ausdrücken können?
- Passt die Darstellung der Suchergebnisse?
- Haben ihnen Informationen gefehlt, bzw. könnte man mehr anzeigen?
- Würden Sie die Suchfunktion so verwenden wollen, oder was müsste noch angepasst oder ergänzt werden?
- Haben Sie Vorschläge für Fragestellungen in weiteren Tests?

D.2 USABILITYTEST MIT P-7

Es folgen die Notizen zum Usability Test, das vollständige Transkript befindet sich im digitalen Anhang E auf Seite 144.

Interview

- Welchen Bezug haben Sie zur Briefedition Wedekind?
  Ganz neu eingestiegen, seit 2 Wochen dabei
- Wie lange arbeiten sie bereits mit der Edition?
  Seit 2 Wochen dabei
- Welche Rolle haben sie bei der Arbeit mit der Edition?
  Wissenschaftliche Mitarbeiterin, Editorin
- Wozu nutzen Sie die Suchfunktion der Edition?
  Kennt die Daten bereits ein wenig

- Welche Funktionalitäten erwarten sie bei der Suchfunktion?
  - Gezielt suchen nach Autor / Empfänger allg. nach Entitäten
  - Komfortable Volltextsuche
  - Suche in Metadatenfeldern
  - Filter nach angegebener Literatur
  - In einem Brief auf einen Verweis springen
- Welche Funktionalitäten haben sie dabei in der Vergangenheit vermisst? Ereignisse beim Register –> In einer Suchliste

Aufgabenstellungen

Beobachtungen beim Lösen der Aufgabenstellungen

- Entitätssuche zuerst in der Volltextsuche eingegeben
- Hat den Filter aber noch bemerkt
- Fehlerhafte Schreibweise von Fürstin Russalka
- Verwendung eines booleschen Operators
- Verwendung der Tooltips für Erläuterung der erweiterten Suchfunktionen
- Eingabe eines Datums in die Volltextsuche
- Wenn es keine Ergebnisse gibt, kann man nicht Filtern, aber die Filterkomponente wird trotzdem angezeigt, mit keiner Auswahlmöglichkeit
- Bei Frage 4 hat die Testperson verstanden, wie der Filter funktioniert (Kaskadierendes Filtern)
- Das Personen in unterschiedlichen Rollen auftreten, ist nicht direkt aufgefallen
- Korrespondenzart im Filter vermisst
- Filtern von Entität war nicht möglich, da die Filtersuchkomponente (zweite Eingabe) nicht gefunden wurde
- Wieder Verwirrung, da noch eine Suche aktiv war
- Suche zurücksetzen wurde erst nicht gefunden
- Blut und Blüten Problem nicht erkannt
- Auf Rückfrage hat sie Ähnlich Worte verwendet
- Anmerkung: Wortstamm Erwartung wäre eher der Linguistisch korrekte Worstamm

Feedback

- Was ist ihr Gesamteindruck wenn Sie die Suchfunktion verwenden?
  - Weitaus mehr Möglichkeiten als sie sich selbst erschließen würde
  - Man kommt nicht intuitiv darauf, könnte an festen Vorstellung liegen
  - Die Idee wie man hier Suchen kann findet sie sehr gut
  - Ausbaufähig: Suche nach Metadaten und weiteren Details wie die Korrespondenzart (Postkarte) oder Ereignisse wäre schön
  - Vom Register kann man nicht auf die Briefe springen
- Entspricht die Bedienung ihrer Erwartung?
  - Suche zurücksetzen besser hervorheben (vielleicht ROT?)
  - Volltextsuche kann weitere eingegrenz werden, das ist positiv
- Passt die Darstellung der Suchergebnisse?
  - Standard heutzutage, wie bei google
  - alles zu sehen, man hat die Metadaten
  - Wenn es einen interessiert kann Brief angeklickt werden
- Würden Sie die Suchfunktion so verwenden wollen, oder was müsste noch angepasst oder ergänzt werden?
  Weitere Details zeigen ist zu unauffällig
- Suche zurücksetzen wurde lange nicht gefunden
  Der „Suche zurücksetzen“ muss genauso hervorgehoben werden. Vielleicht Rot für zurücksetzen
- Warum wurde die Auswahl einer Entität mit Filtersuche nicht gefunden
  - Irritation weil sie in der Auswahl der Filter die Entität nicht direkt gefunden hat (nur 5 Treffer)
  - Der Text *Hier können Sie die Ergebnisse der Volltextsuche weiter einschränken* ist farblich untergegangen
  - Das Filtersuche Eingabefeld wurde nicht als Eingabefeld erkannt
- Hinweis auf weitere Ergebnisse wurde nicht gelesen
  Man müsste den Hinweis präsenter darstellen
- Weitere Details
  Schön dass man sich weitere Details

- Gebrauchsanleitung wäre gut
  - Beschreibung der Funkionalitäten der Suche
  - Welche Füllwörter werden ignoriert
  - Hilfestellung auf der Seite

D.3 USABILITYTEST MIT P-8

Es folgen die Notizen zum Usability Test, das vollständige Transkript befindet sich im digitalen Anhang E auf Seite 144.

Interview

- Welchen Bezug haben Sie zur Briefedition Wedekind?
  In der ersten Projektphase von 2018-2020 wissenschaftlicher Mitarbeiter. Im ersten Team das Projekt aufgesetzt und zahlreiche Briefe ediert. Einige Workshops zur Briefedition Wedekind.
- Wie lange arbeiten sie bereits mit der Edition?
  2 Jahre
- Welche Rolle haben sie bei der Arbeit mit der Edition?
  Wissenschaftlicher Mitarbeiter
- Wozu nutzen Sie die Suchfunktion der Edition?
  zur praktischen Arbeit Am Anfang 180 Briefe
- Welche Funktionalitäten erwarten sie bei der Suchfunktion?
  Text zuverlässig finden in den Briefen Entitätssuche (Namen, Werke und Orte) Gerne: Andere Teile der Edition durchsuchbar gestalten (Kommentare und Dokumentation)
- Welche Funktionalitäten haben sie dabei in der Vergangenheit vermisst?
  Andere Teile der Edition durchsuchbar gestalten (Kommentare und Dokumentation)

Aufgabenstellungen

Beobachtungen beim Lösen der Aufgabenstellungen

- Der Benutzer scrollt mit der Tastatur
- Die Filter sind bereits informativ und interessant
- Navigation in einen Brief und zurück über den Browser hat intuitiv funktioniert

- Filtersuche wurde entdeckt und die Warnung vor der aktiven Suche konnte erklären, warum man Tilly Wedekind nicht findet. Allerdings hat das zurücksetzen der Suche mit „Alle Filter Löschen“ nicht funktioniert, da hierdurch die Volltextsuche nicht zurückgesetzt wurde
- Die Suche nach Entitäten nur als Adressatin hat intuitiv nicht funktioniert
- Eine Verlinkung vom Suchtreffer auf die Trefferstelle im Brief wäre wünschenswert
- Die Vorschau der Ergebnistreffer wurde intuitiv genutzt

Feedback

- Was ist ihr Gesamteindruck wenn Sie die Suchfunktion verwenden?
  - Scheint gut zu funktionieren. Die Volltextsuche ist schnell und intuitiv.
  - Mit den weiteren Funktionalitäten ist das intuitive umgehen schwierig
  - Filterung müsste die Testperson sich nochmal anschauen um sie zu verstehen.
- Enthält die Suchfunktion die erwartenden und benötigten Funktionen?
  - Ja die Suchfunktionalität enthält soweit man sehen kann, die benötigten Funktionen.
  - Es wird allerdigns nur der Text durchsucht, nicht die Kommentare.
- Entspricht die Bedienung ihrer Erwartung?
  - Volltextsuche intuitiv
  - Nachdenken erforderlich, z. B. was bedeutet *Füllwörter ignorieren*
  - Ergebnisse Filtern erschließt sich nicht. Man müsste sich damit noch beschäftigen um das zu verstehen
- Passt die Darstellung der Suchergebnisse?
  - Die Darstellung passt.
  - Man kann die Stellen finden.
  - Man kann zurücknavigieren.
  - Die wichtigsten Briefdaten werden dargestellt

  - Man wird auf die wesentlichen Dinge gelenkt, das Design ist schlicht ohne Ablenkungen

- Haben Ihnen Informationen gefehlt, bzw. könnte man mehr anzeigen?

  - Ausreichend Informationen in dem Rahmen, indem mit der Edition gesucht wird
  - Man kann in der Vorschau erkennen, ob ein Brief relevant ist und ob man abspringen soll

- Würden Sie die Suchfunktion so verwenden wollen, oder was müsste noch angepasst oder ergänzt werden?

  - Mehr der Annotierten Informationen sollte durchsuchbar sein (Hierbei hervorheben, von wo es kommt)
  - Würde sie benutzen und sich gerne damit Vertraut machen
  - Wenn man länger damit arbeitet, kann man die Frage detaillierter beantworten

- Haben Sie Vorschläge für Fragestellungen in weiteren Tests?
  Die Fragen waren eigentlich ganz gut. Worte, Phrasen und so weiter

- Versuch nach Tilly Wedekind suchen
  - Hier ist der Benutzer geprägt von der aktuellen Edition.
  - Ergebnisse Filtern impliziert, dass man erst eine Volltextsuche durchführen muss, dass man sofort filtern kann erschließt sich nicht
  - Wenn man aus der Filtersuche klickt und noch Text eingegeben ist, und man klickt erneut rein, wird der eingegebene Begriff nicht direkt gefiltert (Bug)
  - Der kaskadierende Filter ist etwas gewöhnungsbedürftig
  - Wenn man sich in den Filter einarbeitet, dann findet man sich auch zurecht

- Weitere Details anzeigen
  - Weitere Details wurde zuerst nicht gesehen
  - Wenn man am Desktop mit der Tastatur scrollt, hat das Element den Fokus und öffnet sich, man kann also erst scrollen, wenn man woanders hingeklickt hat (Bug)
  - Dass man sich die Referenzen in den Ergebnissen zusätzlich einblenden kann, ist sehr gut
  - Ist leider etwas versteckt

D.4 USABILITYTEST MIT P-9

Es folgen die Notizen zum Usability Test, das vollständige Transkript befindet sich im digitalen Anhang E auf Seite 144.

Interview

- Welchen Bezug haben Sie zur Briefedition Wedekind?
  Leiterin im Editionsteam
- Wie lange arbeiten sie bereits mit der Edition?
  Seit April 2018
- Welche Rolle haben sie bei der Arbeit mit der Edition?
  Editorin
- Wozu nutzen Sie die Suchfunktion der Edition?
  - Register
  - Volltextsuche
- Welche Funktionalitäten erwarten sie bei der Suchfunktion?
  - Kommentar auch im Volltext
  - Flexionen (Dativ, Genitiv)
  - Teile von Worten suchen (Beispiel Spiegel –> zu Spiegelkabinett)

Aufgabenstellungen

Aufgrund von technischen Problemen, konnte bei der Lösung der Aufgabenstellungen keine Bildschirmübertragung durchgeführt werden. Denoch konnten Beobachtungen gemacht werden, durch die Erläuterungen der Aktionen, durch die Testperson.

- Einstieg über die Volltextsuche, keine Verwendung der Filtersuche
- Zurücknavigation war schwierig in der alten Suchfunktion, die Zurücknavigation per Browser wurde gefunden
- Vorschau der Suchtreffer wird von der Testperson gelesen
- Testperson gibt das Datum in die Volltextsuche ein
- Versuch das Datum in Anführungszeichen zu setzen
- Das Auffinden des Briefes wurde über die Volltextsuche gelöst
- Tilly Wedekind kann nicht ausgewählt werden als Adressatin. Die Filtersuchkomponente wurde nicht gefunden

- Erweiterte Darstellungsoptionen nach Örtlichkeiten wurden nicht genutzt
- Beim Absprung in den Brief wäre die Hervorhebung des Treffers wünschenswert
- Phrasensuche funktioniert intuitiv

Feedback

- Was ist ihr Gesamteindruck wenn Sie die Suchfunktion verwenden?
  - Volltextsuche und was dort einstellbar ist sehr transparent, auch mit den Erläuterungen „Suchbegriff hier eingeben“
  - Die Gesamtüberschrift sollte umformuliert werden (aus philologischer Sicht), von Wedekind Suche –> Wedekinds Korrespondenzsuche
  - Die Auswirkungen der Volltextsuche auf die Filter am Rand ist ebenfalls transparent. Das kann man gut verstehen
- Enthält die Suchfunktion die erwartenden und benötigten Funktionen?
  Ja
- Entspricht die Bedienung ihrer Erwartung?
  Rückblickend gesehen konnte die Testperson keine Probleme bei der Bedienung feststellen. Alles hat sich wie erwartet verhalten
- Passt die Darstellung der Suchergebnisse?
  Kontext erkennbar ist super. Man kann schnell entscheiden, ob es für einen relevant ist und es anklicken
- Haben ihnen Informationen gefehlt, bzw. könnte man mehr anzeigen?
  In der Suchfunktion passt die Darstellung, wenn man auf das Dokument springt, muss man den Begriff erneut suchen mit der Browsersuche, da er nicht hervorgehoben wird. Der Umweg über die Browsersuche ist zumutbar, aber eine Hervorhebung wäre eine nette Ergänzung
- Würden Sie die Suchfunktion so verwenden wollen, oder was müsste noch angepasst oder ergänzt werden?
  Im angeklickten Text –> Fundstelle hervorheben
- Haben Sie Vorschläge für Fragestellungen in weiteren Tests?
  Die Fragen waren schon gut
- Weitere Ideen?
  Markieren von zwei Ergebnissen zum gemeinsamen ansehen in einer Parallelansicht
- Entitätssuche

  - Was man eigentlich braucht, ist die Kombination von mehreren Entitäten, also beispielsweise Bezüge herzustellen zwischen zwei Personen. Für die Arbeit ist das sehr wichtig, dass man diese Kombinationen filtern kann. Beispielsweise wann kommt *Max Harden* mit dem Werk *In allen Sätteln gerecht* vor.
  - Es ist toll, dass das jetzt gemacht

- Weitere Details Anzeigen (Annotationen)
- Erläuterungskomponente
  Die Darstellung was zum Ergebnis geführt hat ist praktisch. Auch das man dort Filter entfernen kann.
- Erläuterungen notwendig

  - Funktionalitäten sind versteckt. Was man machen kann muss man wissen und man weiß nicht wie man herausfindet, was die Suche kann.
  - Der Fokus liegt auf der Volltextsuche, die weiteren Funktionalitäten entdeckt man vielleicht garnicht

D.5 USABILITYTEST MIT P-10

Es folgen die Notizen zum Usability Test, das vollständige Transkript befindet sich im digitalen Anhang E auf Seite 144.

Interview

- Welchen Bezug haben Sie zur Briefedition Wedekind?
  Bezug beginnt mit dem Vorschlag die Briefdatenbank zu initieren mit Kolleginen von der Hochschule, vor fast 10 Jahren. Begleitung der Entwicklung der ersten Briefedition mit dem Fachbereich Informatik. Seit dem bewilligten DFG Antrag mit zuarbeit und beratend im Hintergrund aktiv.
- Welche Rolle haben sie bei der Arbeit mit der Edition?
  Beratende Rolle im Bezug zur Edition. Im Bezug zu Frank Werdekind: Forschungsnebenstelle in Darmstadt, welche sich um Briefe kümmert, zwischen Eltern und Kindern (exkl. Frank Wedekind), welche nicht zur Eingabe in der Edition bestimmt sind
- Wozu nutzen Sie die Suchfunktion der Edition?
  Verwendung der Suchfunktion zum Auffinden von Informationen für den Kommentar. Beispielsweise Personen finden oder Anspielung auf anderen Briefpartner
- Welche Funktionalitäten erwarten sie bei der Suchfunktion?
  Erwartungen sind identisch zur bestehenden Suchfunktion Sortierung ist merkwürdig, da man die Seite neuladen muss.

Aufgabenstellungen

Beobachtungen beim Lösen der Aufgabenstellungen

- Zunächst fällt auf, dass bei der geringen Auflösung des Gerätes die Überschrift sehr viel Platz einnimmt. Somit sieht man beim Öffnen der Seite nur die Volltextsuche.
- Zum Einstieg wird die Volltextsuche verwendet
- Absprung aus der Suche in Personendetails oder weitere Details zu Entitäten anzeigen
- Off-Topic Idee: Im Register könnte man ein Alphabet anzeigen, und beim Klick auf einen Buchstaben des Alphabets direkt an diese Stelle der Registereinträge springen
- Filtersuche wurde durch den Hinweis unter den Entitätslisten gefunden.
- Irritation, da der angewendete Filter nur noch die Möglichkeiten anbietet, die im Ergebnis vorkamen
- Suche zurücksetzen zu unauffällig
- Autovervollständigung mit Auswahl der Entität nach Rolle führte zu Verwirrung
- Zurücknavigation hat intuitiv funktioniert
- Volltextsuche zurücksetzen über die Filtersuche hat intuitiv funktioniert
- Bei geringer Auflösung sieht man nach der Volltextsuche weder den Ladebalken noch die Ergebnisse
- Bei Suche nach Blüten, tauchen auch Treffer für Blut auf. Dies wurde als unnötig wahrgenommen

Feedback

- Enthält die Suchfunktion die erwartenden und benötigten Funktionen?
  Zunächst auf jedenfall
- Entspricht die Bedienung ihrer Erwartung?
  Da bereits Erfahrung mit der Edition vorliegt, hat es den Erwartungen entsprochen
- Passt die Darstellung der Suchergebnisse?
  Die Darstellung ist Übersichtlich

- Haben ihnen Informationen gefehlt, bzw. könnte man mehr anzeigen? Man muss die Darstellung auch so eingrenzen dass nur das benötigte angezeigt werden muss
- Würden Sie die Suchfunktion so verwenden wollen, oder was müsste noch angepasst oder ergänzt werden?
  Sieht nicht was noch fehlt
- Autovervollständigung in Filtersuche Die Testperson hat keinen Vorschlag angeklickt sondern nebendran geklickt. Dadurch wurde keine Entität ausgewählt. Eine Entität wird nur ausgewählt, wenn man einen Vorschlag selektiert
- Filterliste Es wäre schön, wenn man doch alle Entitäten anzeigen könnte
- Chronologische Suche Eine Einschränkung mit Zeitraum wäre noch wünschenswert

E

ANHANG E

Der digitale Anhang dieser Arbeit kann im gitlab des Fachbereichs Informatik eingesehen werden [51]. Die Ordnerstruktur des Repositories ist wie nachfolgend beschrieben:

.ASSETS Enthält Abbildungen für die *README.md* Datei

.VSCODE Enthält (optionale) Konfigurationen für die Entwicklungsumgebung Visual Studio Code

FRONTEND Enthält die Frontend-Applikation und eine kurze *README.md* mit Entwicklungsinformationen

LOCUST Enthält den Applikationsperformancetest

SEARCH Enthält die Backend-Applikation

RESEARCH Enthält Dateianhänge mit den erhobenen Daten aus der Analyse und Design Phase des Projekts

Im Ordner *research* befinden sich die nachfolgenden Dateien:

KONKURRENZANALYSE.XLSX Enthält das Konkurrenzanalyse Dokument und die Auswertung

QUANTITATIVE_AUSWERTUNG_UMFRAGE.XLSX Enthält die Quantitative Auswertung der Umfrage und die Diagramme

QUALITATIVE_AUSWERTUNG_UMFRAGE.XLSX Enthält die Antworten der Befragten auf die offenen Fragen

UMFRAGE.LSS Enthält den Fragebogen im Limesurvey Format zum einfachen Import

USABILITYTEST P-6.DOCX Enthält das Transkript des Usabilitytests mit P-6

USABILITYTEST P-7.DOCX Enthält das Transkript des Usabilitytests mit P-7

USABILITYTEST P-8.DOCX Enthält das Transkript des Usabilitytests mit P-8

USABILITYTEST P-9.DOCX Enthält das Transkript des Usabilitytests mit P-9

USABILITYTEST P-10.DOCX Enthält das Transkript des Usabilitytests mit P-10

LITERATUR

[1] R. Baeza-Yates und B. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press books. ACM Press, 1999. ISBN: 9780201398298.

[2] Bill Buxton. *Sketching User Experiences: Getting the Design Right and the Right Design*. First. Morgan Kaufmann, 2007.

[3] William Buxton. *Sketching user experiences: getting the design right and the right design*. Englisch. 443 p. Amsterdam ; 2007.

[4] Federico Caria und Brigitte Mathiak. "A Hybrid Focus Group for the Evaluation of Digital Scholarly Editions of Literary Authors". In: *Scholarly Digital Editions as Interfaces* 12 (2018), S. 267.

[5] Stefano Ceri, Alessandro Bozzon, Marco Brambilla, Emanuele Della Valle, Piero Fraternali und Silvia Quarteroni. *Web Information Retrieval*. 2013. URL: https://doi.org/10.1007/978-3-642-39314-3.

[6] Deutsche Forschungsgemeinschaft (DFG). *Edition der Korrespondenz Frank Wedekinds als Online-Volltextdatenbank*. https://gepris.dfg.de/gepris/projekt/389236467?context=projekt&task=showDetail&id=389236467. [Online; Abruf: 18.10.2020].

[7] Andreas Domin. *Was ist eigentlich eine Single-Page-Webanwendung?* https://t3n.de/news/single-page-webanwendung-1023623/. [Online; Abruf: 28.11.2020]. 2018.

[8] Carole Eissa. *Nutzerzentriertes Designen. Der Double Diamond Prozess*. https://www.testingtime.com/blog/double-diamond-design-prozess/. [Online; Abruf: 17.10.2020]. 2017.

[9] Thomas Erben. *Einführung in Unix/Linux für Naturwissenschaftler : Effizientes wissenschaftliches Arbeiten mit der Unix-Kommandozeile*. 2017. URL: https://doi.org/10.1007/978-3-662-50301-0.

[10] *Express - Node.js Web Framework*. https://expressjs.com/de/. [Online; Abruf: 28.10.2020].

[11] *Frank Wedekinds Korrespondenz digital. Editorial*. https://briefedition.wedekind.h-da.de/view/editorial.xhtml. [Online; Abruf: 12.10.2020].

[12] *Frank Wedekinds Korrespondenz digital*. http:// briefedition.wedekind.h-da.de. [Online; Abruf: 01.03.2020].

[13] Greta Franzini. *gfranzini/digEds_cat: First release*. Version v1.0. Jan. 2012. DOI: 10.5281/zenodo.1161425. URL: https://doi.org/10.5281/zenodo.1161425.

[14] Greta Franzini, S Mahony und M Terras. "A catalogue of digital editions". In: Open Book Publishers, 2015.

[15] Norbert Fuhr und Claus-Peter Klas. *Fachgruppe Information Retrieval in der Gesellschaft für Informatik (GI)*. https://fg-retrieval.gi.de/fachgruppe/ueber-die-fachgruppe. [Online; Abruf: 03.08.2020].

[16] *GIN Indexes. Introduction*. https://www.postgresql.org/docs/13/gin-intro.html. [Online; Abruf: 04.11.2020].

[17] Saul N. Greenberg. *Sketching user experiences*. Heidelberg [u.a.], 2014. URL: http://d-nb.info/1038363187/04.

[18] *Hetzner Cloud Hosting*. https://www.hetzner.com/de/cloud. [Online; Abruf: 30.11.2020].

[19] *Hetzner Rootserver Dedicated Hosting*. https://www.hetzner.com/de/dedicated-rootserver. [Online; Abruf: 30.11.2020].

[20] Tobias Holstein und Uta Störl. "Insights into the Work of Editors of Digital Scholarly Editions for Correspondences". In: *Human-Computer Interaction. Design and User Experience*. Hrsg. von Masaaki Kurosu. Cham: Springer International Publishing, 2020, S. 273–292. ISBN: 978-3-030-49059-1.

[21] Ankur Joshi, Saket Kale, Satish Chandel und D Kumar Pal. "Likert scale: Explored and explained". In: *Current Journal of Applied Science and Technology* (2015), S. 396–403.

[22] Tom Lane. *unaccent() function should be marked IMMUTABLE*. https://www.postgresql .org/message-id/16472.1291351806%40sss.pgh.pa.us. [PostgreSQL Bugs Mailingliste; Abruf: 30.10.2020]. 2010.

[23] *LimeSurvey - Einfache Online-Umfragen*. https://www.limesurvey.org/de/. [Online; Abruf: 01.04.2020].

[24] Jasper Liu. *Visualizing the 4 Essentials of Design Thinking*. https://medium.com/good-design/visualizing-the-4-essentials-of-design-thinking-17fe5c191c22. [Online; Abruf: 27.11.2020]. 2016.

[25] Yandong Liu, Jiang Bian und Eugene Agichtein. "Predicting Information Seeker Satisfaction in Community Question Answering". In: *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR 08. Singapore, Singapore: Association for Computing Machinery, 2008, S. 483–490. ISBN: 9781605581644. DOI: 10.1145/1390334.1390417. URL: https://doi.org/10.1145/1390334.1390417.

[26] *Locust*. https://locust.io/. [Online; Abruf: 20.11.2020].

[27] *Mirror of the official PostgreSQL GIT repository. german.stop*. https://github.com/postgres/postgres/blob/REL_12_4/src/backend/snowball/stopwords/german.stop. [Online; Abruf: 22.11.2020].

[28] Ian Molyneaux. *The art of application performance testing: from strategy to tools*. O'Reilly Media, Inc., 2014. ISBN: 978-1-4919-0052-9.

[29] Kate Moran und Kara Pernice. *Remote Moderated Usability Tests: Why to Do Them*. https://www.nngroup.com/articles/moderated-remote-usability-test-why/. [Online; Abruf: 01.10.2020]. 2020.

[30] Dan Nessler. *How to apply a design thinking, HCD, UX or any creative process from scratch — Revised & New Version*. https://uxdesign.cc/how-to-solve-problems-applying-a-uxdesign-designthinking-hcd-or-any-design-process-from-scratch-v2-aa16e2dd550b. 2018.

[31] *Node.js*. https://nodejs.org/. [Online; Abruf: 28.10.2020].

[32] Klaus Pohl und Chris Rupp. *Basiswissen Requirements Engineering - Aus- und Weiterbildung zum "Certified Professional for Requirements Engineering"; Foundation Level nach IREB-Standard (3., korrigierte Auflage)*. dpunkt.verlag, 2011, S. I–XV, 1–176. ISBN: 978-3-89864-771-7.

[33] Martin F Porter. *Snowball: A language for stemming algorithms*. 2001.

[34] *PostgreSQL*. https://www.postgresql.org/. [Online; Abruf: 12.10.2020].

[35] *PostgreSQL. Full Text Search - Introduction*. https://www.postgresql.org/docs/13/textsearch-intro.html. [Online; Abruf: 15.10.2020].

[36] *PostgreSQL. Text Search Types - Data Types*. https://www.postgresql.org/docs/13/datatype-textsearch.html. [Online; Abruf: 15.10.2020].

[37] *PostgreSQL. Text Search Functions and Operators - Functions and Operators*. https://www.postgresql.org/ docs/13/functions-textsearch.html. [Online; Abruf: 17.10.2020].

[38] *PostgreSQL. Controlling Text Search - Full Text Search*. https://www.postgresql.org/docs/13/textsearch-controls.html. [Online; Abruf: 28.10.2020].

[39] *PostgreSQL. Full Text Search - Parsers*. `https://www.postgresql.org/docs/current/textsearch-parsers.html`. [Online; Abruf: 04.11.2020].

[40] *PostgreSQL. Full Text Search - Dictionaries*. `https://www.postgresql.org/docs/13/textsearch-dictionaries.html`. [Online; Abruf: 04.11.2020].

[41] *PostgreSQL. Data Definition - Generated Columns*. `https://www.postgresql.org/docs/12/ddl-generated-columns.html`. [Online; Abruf: 30.10.2020].

[42] Bernhard Preim und Raimund Dachselt. *Interaktive Systeme - Band 2: User Interface Engineering, 3D-Interaktion, Natural User Interfaces*. Apr. 2015. ISBN: 978-3-642-45246-8. DOI: `10.1007/978-3-642-45247-5`.

[43] Michael Rohrschneider. *Tutorium Quelleneditionen analog und digital*. `https://www.historicum-estudies.net/etutorials/tutorium-quelleneditionen/definition/editionstypen/`. [Online; Abruf: 02.05.2020]. URL: `https://dx.doi.org/10.18716/estudies/00011`.

[44] Patrick Sahle. *Digitale Editionsformen. Zum Umgang mit der Überlieferung unter den Bedingungen des Medienwandels. Teil 2: Befunde, Theorie und Methodik*. 2013. URL: `http://nbn-resolving.de/urn:nbn:de:hbz:38-53523`.

[45] Patrick Sahle. "Digitale Editionsformen. Zum Umgang mit der Überlieferung unter den Bedingungen des Medienwandels. Teil 3: Textbegriffe und Recodierung. [Preprint-Fassung]". Köln, Universität zu Köln, Diss., 2009. Diss. Köln, 2013. URL: `http://nbn-resolving.de/urn:nbn:de:hbz:38-50130`.

[46] Patrick Sahle. *Kriterienkatalog für die Besprechung digitaler Editionen*. https:// www.i-d-e.de/publikationen/weitereschriften/kriterien-version-1-1/. [Online; Abruf: 30.03.2020]. 2014.

[47] Joel Scambray und Mike Shema. *Hacking Exposed Web App*. 3rd ed. Tata McGraw-Hill Education, 2011.

[48] Ahmed Seffah, Mohammad Donyaee, Rex B. Kline und Harkirat K. Padda. "Usability measurement and metrics: A consolidated model". In: *Software Quality Journal* 14.2 (Juni 2006), S. 159–178. ISSN: 1573-1367. DOI: 10.1007/s11219-006-7600-8. URL: https://doi.org/10.1007/s11219-006-7600-8.

[49] Edson Marchetti da Silva. "Comparing the use of full text search between a conventional IR System and a DBMS". In: *for Knowledge Organization in the Digital Age* (2018), S. 120.

[50] International Organization for Standardization. *ISO 9241-11: 2018, Ergonomics of human-system interaction-Part 11: Usability: Definitions and concepts*. 2018.

[51] *Suchmaschine für digitale (historisch-kritische) Editionen*. https://code.fbi.h-da.de/wedekind/search/searchtechnologyperformance.

[52] The British Design Council. *Design methods for developing services*. https://www.designcouncil.org.uk. [Online; Abruf: 02.04.2020].

[53] The British Design Council. *Eleven lessons. A study of the design process*. https://www.designcouncil.org.uk/sites/default/files/asset/document/ElevenLessons_Design_Council%20(2).pdf. [Online; Abruf: 01.06.2020]. 2005.

[54] Tuba Ugras, Sevinç Gülseçen, Ceren Çubukçu, İpek İli Erdoğmuş, Vala Gashi und Merve Bedir. "Research Trends in Web Site Usability: A Systematic Review". In: *Design, User Experience, and Usability: Design Thinking and Methods*. Hrsg. von Aaron Marcus. Cham: Springer International Publishing, 2016, S. 517–528. ISBN: 978-3-319-40409-7.

[55] Cornelis J Van Rijsbergen. *Information Retrieval*. http://www.dcs.gla.ac.uk/Keith/Preface.html. [Online; Abruf: 05.04.2020]. Butterworths, London, 1979.

[56] Ana Villar. *Agreement answer scale design for multilingual surveys: Effects of translation-related changes in verbal labels on response styles and response distributions*. https://digitalcommons.unl.edu/cgi/viewcontent.cgi?article=1002&context=sramdiss. [Online; Abruf: 20.05.2020]. 2009.

[57] *Vue Material*. https://vuematerial.io/. [Online; Abruf: 28.10.2020].

[58] *Vue.js*. https://vuejs.org/v2/guide/. [Online; Abruf: 28.10.2020].

[59] *Vuex. What is Vuex?* https://vuex.vuejs.org/. [Online; Abruf: 16.11.2020].

[60] *Vuex. State*. https://vuex.vuejs.org/guide/state.html. [Online; Abruf: 16.11.2020].

[61] *Vuex. Mutations*. https://vuex.vuejs.org/guide/mutations.html. [Online; Abruf: 16.11.2020].

[62] *Vuex. Actions*. https://vuex.vuejs.org/guide/actions.html. [Online; Abruf: 16.11.2020].

[63] Kathryn Whitenton. *Tools for Unmoderated Usability Testing*. https://www.nngroup.com/articles/unmoderated-user-testing-tools/. [Online; Abruf: 01.10.2020]. 2019.

[64] *pg-promise*. https://github.com/vitaly-t/pg-promise. [Online; Abruf: 28.10.2020].