Bachelor of Science

Mobile Annotator

Eine App für den TextAnnotator

Pascal Adeberg

Abgabedatum: 29.07.2020

Goethe-Universität Frankfurt am Main
Prof. Dr. A. Mehler

Erklärung

Hiermit bestätige ich, dass ich die vorliegende Arbeit selbstständig verfasst habe und keine anderen Quellen oder Hilfsmittel als die in dieser Arbeit angegebenen verwendet habe.

Karben, 29.07.2020

Ort, Datum

Unterschrift

Zusammenfassung

Der Inhalt dieser Arbeit ist die Entwicklung und Evaluation einer mobilen Webanwendung für die Annotation von Texten. Dem Benutzer ist es durch diese Webanwendung, im folgenden auch MobileAnnotator genannt, möglich Wörter und Textausschnitte zu kategorisieren oder auch mit Wissensquellen, zum Beispiel Wikipedia, zu verknüpfen. Der MobileAnnotator ist dabei für mobile Endgeräte ausgelegt und insbesondere für Smartphones optimiert worden.

Für die Funktionalität verwendet der MobileAnnotator die Architektur des bereits existierenden und etablierten TextAnnotators. Dieser stellt bereits eine Vielzahl von Annotations Werkzeugen bereit, von denen zwei auf den MobileAnnotator übertragen wurden. Da der TextAnnotator vollständig für einen Desktopbetrieb ausgelegt wurde, ist es jedoch nicht möglich diese Werkzeuge ohne Anpassungen für ein mobiles Gerät umzubauen. Der MobileAnnotator beschränkt sich somit auf ein Mindestmaß an Funktionen dieser Werkzeuge um sie dem Benutzer in geeigneter Art und Weise verfügbar zu machen.

Für die Evaluation der Benutzerfreundlichkeit des MobileAnnotator und dessen Werkzeuge wurde anschließend eine Studie durchgeführt. Den Probanten war es innerhalb der Studie möglich Aussagen über die Bedienbarkeit des MobileAnnotators zu treffen und einen Vergleich zwischen dem Mobile- und TextAnnotator zu ziehen.

Im folgenden stellt diese Arbeit nun den MobileAnnotator vor. Zunächst werden dafür die theoretischen und konzeptionellen Vorarbeiten betrachtet bevor näher auf die tatsächliche Realisierung eingegangen wird.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einleitung

1.1. Motivation

Die meisten modernen Werkzeuge für *Natural Language Processing* (NLP) sowie dem Kategorisieren und Durchsuchen von Texten benötigen für ihr Training und ihre Evaluation vollständig und korrekt annotierte Daten. Diese Daten zu erzeugen ist allerdings ein sowohl Zeit als auch finanziell sehr kostspieliger Prozess, da die Korrektheit der Annotationen manuell von Menschenhand sichergestellt werden muss. Dabei sind solche Werkzeuge in Anbetracht der gewaltigen Datenmengen, welche sich meist ungenutzt in den Tiefen des World-Wide-Web verbergen, wichtiger denn je. Die meisten Annotationswerkzeuge sind jedoch sehr technisch versiert und werten die Funktionalität deutlich höher als eine erleichterte Benutzeroberfläche.

Für eine zeitliche Optimierung und geringere Belastung der Anwender hat sich dafür ein Human-in-the-loop Prozess etabliert. Dabei werden Daten zunächst automatisiert annotiert und im Anschluss von einem Anwender mit Hilfe eines Annotationswerkzeuges korrigiert. Der Zeitaufwand für den Anwender wird dadurch, im Gegensatz zu einer rein manuellen Annotation, bereits deutlich reduziert, stellt aber dennoch eine Belastung für den Anwender dar, gerade in Bezug auf eine stetig gleichbleibend hohe Qualität.

Die Überlegung, die zu dieser Arbeit führte, war letztlich, mit Hilfe eines Annotationswerkzeuges für Mobilgeräte, das zeitliche Potential der Anwender besser auszuschöpfen. Analog zu den Prinzipien der games-with-a-purpose und dem Crowdsourcing, kann über eine mobile Anwendung, mit entsprechend einfach und intuitiv gehaltener Benutzeroberfläche, ein Mehrwert aus einer größeren Menge erreichbarer Endbenutzer generiert werden. Fehler eines Einzelnen fallen demnach bei entsprechend großer Anzahl Vergleichsdaten anderer Benutzer nicht mehr so stark ins Gewicht, was zu einer geringeren Belastung für den einzelnen Anwender führt. Während die meisten Annotatoren als komplexe Anwendungen nur für einen Desktop Betrieb ausgelegt sind, hätte eine mobile Anwendung somit die Möglichkeit Benutzer auch außerhalb ihres stationären Arbeitsplatzes zu erreichen. Die Schwierigkeit besteht darin, die Komplexität einer Desktop Anwendung auf ein, für ein Mobilgerät geeignetes, Niveau herunter zu brechen, ohne dabei zu viele der Hauptfunktionen zu verlieren.

1.2. Anwendungsfälle

Anwendungsfälle einer mobilen Anwendung sind all jene Situationen, in denen eine Arbeit an einem Desktop Computer unpraktikabel ist und Zeit nicht anderweitig sinnvoll verwendet werden kann.

Ein naheliegendes Beispiel für einen solchen Anwendungsfall sind Berufspendler, welche täglich mit dem öffentlichen Personen-Nahverkehr ihren Weg zur Arbeit zurücklegen. Nicht selten bleibt die Zeit in Bus und Bahn ungenutzt oder wird durch die Nutzung von Unterhaltungsmedien wenig gewinnbringend vertan. Eine mobile Anwendung kann dort eine sinnvolle Alternative bieten. Rund 78,8%[1] der deutschen Pendler benötigen täglich über 20 Minuten für ihren Hin- und Rückweg von ihrem Arbeitsplatz, 30,2%[1] sogar über eine Stunde. Auf ein Jahr betrachtet baut sich schnell ein großes Potential an nutzbarer Zeit für eine solche Anwendung auf.

Dem gleichen Prinzip folgend bieten unter anderem auch Termine beim Arzt eine exzellente Möglichkeit einen Mehrwert durch eine mobile Anwendung zu generieren. Anstatt die Zeit im Wartezimmer durch lesen mehr oder minder aktueller Zeitschriften zu vertreiben könnte eine mobile Anwendung auch hier sinnvolle Verwendung finden. Im Durchschnitt verbringen 59%[2] aller gesetzlich Versicherten und 48%[2] der privat Versicherten länger als 15 Minuten beim Aufsuchen eines Arztes im Wartezimmer.

Analog zu diesen Beispielen lassen sich leicht weitere Anwendungsfälle finden wie Behördengänge, Reisezeit im Flugzeug oder Fernreisen mit dem Zug.

[1] http://www.destatis.de/DE/Themen/Arbeit/Arbeitsmarkt/Erwerbstaetigkeit/Tabellen/pendler1.html aufgerufen am 23.03.2020

[2] http://www.fernarzt.com/wissen/studien/arztpraxisreport/ aufgerufen am 23.03.2020

2. Ähnliche Arbeiten

Es existiert bereits eine Vielzahl von Anwendungen für die manuelle Annotation von Dokumenten und Textkorpora. Die meisten dieser Anwendungen wurden allerdings für einen speziellen Zweck, beziehungsweise Anwendungsfall, entwickelt und haben daher oftmals einen Schwerpunkt auf der technischen Funktionalität liegen statt der Bedienbarkeit. Für diese Arbeit wurden aus diesem Grund hauptsächlich Anwendungen betrachtet, die mindestens einem der folgenden Kriterien entsprechen:

- gute Konfigurierbarkeit
- intuitive Benutzeroberfläche
- Nutzbarkeit auf mobilen Endgeräten.

Aus diesen Anforderungen resultierend wurden für diese Arbeit die Anwendungen BRAT (Stenetorp u. a. 2012), Protégé (Musen 2015), WordFreak (Morton und LaCivita 2003), GATE Teamware (Bontcheva u. a. 2013), WebAnno (Yimam u. a. 2013), BOEMIE (Fragkou u. a. 2008) und der TextAnnotator (Abrami, Stoeckel und Mehler 2020) ausgewählt und anhand der Kriterien genauer betrachtet.

BRAT wurde bereits mit dem Ziel einer intuitiven Benutzeroberfläche entwickelt. Dazu bedient sich BRAT bereits etablierter Funktionen von klassischen Textbearbeitungs Programmen. Aus diesem Grund ist BRAT allerdings auch unpraktisch für eine mobile Nutzung, da die exakte Auswahl einer bestimmten Textstelle per Mauszeiger sich auf mobilen Endgeräten als schwierig erweist. Außerdem bietet BRAT nur die Möglichkeit Textabschnitte mit Labeln zu versehen und Beziehungen dazwischen zu erzeugen. Möchte man weitere, tiefer gehende Annotationen tätigen müsste man demnach zu einem anderen Werkzeug wechseln. Die aktuelle Version 1.3 von BRAT wurde 2012 veröffentlicht.

Protégé ist ein seit den 1980er Jahren stetig weiterentwickeltes Annotations Werkzeug. Die letzte veröffentlichte Version 5.5.0 erschien 2019. Einen großen Anteil an der Langlebigkeit von Protégé trägt die Unterstützung von Plugins wie zum Beispiel Knowtator (Ogren 2006). Die Funktionalität dieser Plugins reicht dabei von Versionierung der Dokumente, über erweiterte Annotations Möglichkeiten bis zu einer veränderten Visualisierung. Die starke Konfigurierbarkeit von Protégé bringt allerdings Anforderungen an ihren Benutzer mit sich. Der sehr technisch betonte Aufbau von Protégé setzt beim Benutzer ein gewisses Maß an Vorwissen über das Werkzeug und seinen Einsatzbereich voraus.

Die Schwerpunkte für Wordfreak sind zwar eher eine leichte Erweiterbarkeit und die Einbindung von automatisierten Annotatoren, dennoch verfügt es über eine benutzerfreundliche Oberfläche. WordFreak bietet dem Benutzer dabei unter anderem die Möglichkeit verschiedene Visiualisierungen eines Dokumentes gleichzeitig zu betrachten. Es ermöglicht au-

ßerdem andere Werkzeuge wie zum Beispiel OpenNLP (OpenNLP 2010) als Plugins zu integrieren und über seine Benutzeroberfläche nutzbar zu machen. Die Benutzeroberfläche ist allerdings stark für einen Desktop Betrieb ausgelegt und nicht für eine mobile Nutzung geeignet. Außerdem kann man die Anwendung mittlerweile als veraltet betrachten, da das letzte veröffentlichte Update 2.2 aus dem Jahr 2004 stammt.

GATE Teamware bietet ein umfangreiches Repertoire an Funktionen für Annotations Projekte. Dies beinhaltet unter anderem auch eine Rechte- und Rollenverteilung der Benutzer um die Komplexität solcher Projekte für den einzelnen Anwender herunter zu brechen. Die Benutzeroberfläche wurde in Blick auf Benutzer ohne informationstechnischen Hintergrund angelegt. Dennoch ist die Benutzeroberfläche nicht vollends intuitiv und bedarf einer vorherigen Einarbeitung bevor die Anwendung effektiv genutzt werden kann. Die letzte Version 8.4.1 wurde 2017 veröffentlicht.

WebAnno ist von Umfang und Funktionalität sehr ähnlich der GATE Teamware. Jedoch grenzt sich WebAnno unter anderem dadurch ab, dass es als Web Anwendung über einen leichteren Zugang ohne vorherige Installation und über eine Anbindung an Crowdsourcing Plattformen verfügt. Außerdem setzt sich WebAnno selbst die Anforderung auch für Annotations unerfahrene Anwender eine verständliche Oberfläche zu bieten. WebAnno ist somit das Werkzeug, das am nächsten an die in dieser Arbeit vorgestellte Anwendung herankommt. Allerdings ist WebAnno ebenfalls für einen Desktopbetrieb ausgelegt und so fehlt eine Oberfläche für eine mobile Nutzung. Die aktuelle Version 3.6.5 von WebAnno wurde Ende 2019 veröffentlicht.

Das BOEMIE Annotations Werkzeug wurde im Rahmen eines gleichnamigen Projektes entwickelt, nachdem kein anderes Werkzeug die nötigen Funktionen innerhalb einer Anwendung bereitstellen konnte. BOEMIE verfügt zwar nur über eine eingeschränkte Funktionalität, welche die Anforderungen des Projekts abdeckt, aber es ist eine robuste und benutzerfreundliche Anwendung. Leider war es jedoch nicht mehr möglich eine Version der Anwendung oder Hinweise über deren Verbleiben zu finden.

Der TextAnnotator wurde mit dem Gedanken einer einheitlichen Basis für die vielzahligen Annotations Werkzeuge entwickelt. Ähnlich der GATE Teamware bietet der TextAnnotator ebenfalls ein Portfolio an Funktionen. Unter anderem umfassen diese Funktionen ein Anwender bezogenes Rechtemanagement (Gleim, Mehler und Ernst 2012) sowie eine automatisierte Analyse und Vorverarbeitung von Dokumenten. Zum Zeitpunkt dieser Arbeit befindet sich der TextAnnotator in der Beta Phase und soll nach Veröffentlichung als OpenSource Projekt weitergeführt werden.

Nichtsdestoweniger besticht allerdings die Tatsache, dass alle diese Anwendungen einzig für einen Desktopbetrieb ausgelegt sind. Auch nach längerer Recherche konnte zum Zeitpunkt dieser Ausarbeitung keine relevante mobile Anwendung für den Bereich der Annotation für NLP gefunden werden, weshalb für eine mobile Anwendung eine neue Entwicklung nötig ist. Wie bereits dargelegt, existiert bereits eine Vielzahl von Anwendungen, die über ausreichende Funktionalitäten verfügen, und denen nur eine mobil fähige Oberfläche fehlt. Eine vollständige Neuentwicklung ist somit nicht notwendig, sondern einzig die Erstellung einer geeigneten Benutzeroberfläche für eine der bereits vorhandenen Anwendungen.

Für diese Arbeit wurde daher auf dem TextAnnotator aufgebaut, welcher im folgenden noch genauer beschrieben wird. Die Entscheidung, den TextAnnotator zu verwenden, ist zum großen Teil der Veröffentlichung als OpenSource Projekt geschuldet. Durch eine solche Veröffentlichung ist eine längerfristige Fortentwicklung durch eine Entwicklergemeinschaft gewährleistet als im Vergleich zu geschlossenen Projekten. Zudem ist es dadurch zukünftig möglich die mobilfähige Oberfläche gleichzeitig mit dem dahinter liegenden Backend zu entwickeln. Zudem bietet der TextAnnotator als noch junges Projekt bereits viele Funktionen der anderen Anwendungen, wie zum Beispiel ein Rechte Management der Benutzer.

3. Verwendete Ressourcen

Die in dieser Arbeit vorgestellte Anwendung baut auf der bereits existierenden TextAnnotator Implementierung auf und ist als Ergänzung dieser zu verstehen. Aufgrund dieser Abhängigkeit werden im folgenden zunächst der TextAnnotator und seine darunterliegenden Elemente beschrieben.

3.1. TextImager

Der TextImager (Hemati, Uslu und Mehler 2016) vereint Funktionalität und Visualisierung bereits etablierter Frameworks wie GATE (Cunningham u. a. 2013), WebNLP (Burghardt u. a. 2014) und OpenNLP (OpenNLP 2010) mit einem vereinheitlichten I/O-Format und einer vereinfachten grafischen Benutzeroberfläche. Durch letztere ist es auch ohne nennenswerte Fähigkeiten im Bereich der Programmierung möglich Prozessketten (Pipelines) für automatisierte Text Analysen zu definieren und auszuführen. Jede Ausführung einer solchen Pipeline wird dabei entsprechend ihren Abhängigkeiten über eine beliebige Anzahl von Server hinweg parallelisiert. Somit bietet der TextImager eine Zeit effiziente Möglichkeit Texte durch NLP Werkzeuge automatisiert zu verarbeiten. Die einzelnen Komponenten des TextImager verwenden dabei das *Unstructured Information Management Applications* (UIMA) Format, weshalb die Einbindung neuer UIMA genügenden Komponenten ein Leichtes ist. Dabei reichen die Werkzeuge von Tokenisierung, Lemmatisierung, über POS-Tagging bis hin zu Text Klassifizierungen und vielem mehr. Die so angereicherten Dokumente können anschließend für eine erneute Prozesskette verwendet werden oder von anderen Anwendungen, wie zum Beispiel dem TextAnnotator, herangezogen werden.

3.2. TextAnnotator

Der TextAnnotator fand 2018 mit seinem ersten Werkzeug, dem TreeAnnotator (Helfrich u. a. 2018), Erwähnung. Seitdem wurde er stetig weiterentwickelt und zum Zeitpunkt dieser Arbeit wurden bereits 7 verschiedene Werkzeuge implementiert. Unter diesen Werkzeugen befinden sich auch der sogenannte QuickAnnotator (Abrami, Stoeckel und Mehler 2020) und KnowledgeBaseLinker (Abrami, Mehler, Lücking u. a. 2019), welche für diese Arbeit besondere Relevanz besitzen und im folgenden noch ausführlicher beschrieben werden. Als Grundlage dienen dem TextAnnotator dabei Dokumente, welche vom TextImager bereits analysiert und entsprechend aufbereitet wurden. Diese Dokumente werden vom TextAnnotator versioniert und den Benutzern in sogenannten Sichten zur Verfügung gestellt. Eine Sicht kann als eine Art Kopie des Dokuments verstanden werden auf der Benutzer arbeiten können ohne direkten Einfluss auf andere Sichten oder das ursprüngliche Dokument zu haben. Somit können beispielsweise mehrere Benutzer parallel in verschiedenen Sichten des gleichen Dokuments arbeiten ohne sich gegenseitig zu behindern.

3.2.1. QuickAnnotator

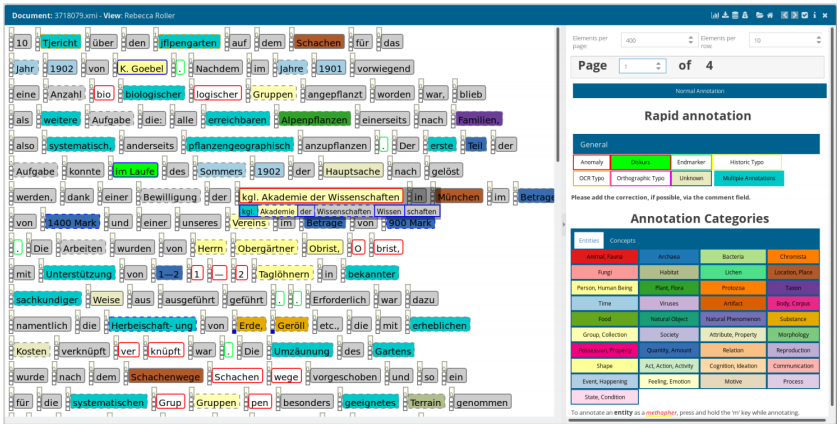


Abbildung 3.1.: Ansicht des QuickAnnotator Werkzeuges des TextAnnotator (Abrami, Stoeckel und Mehler 2020)

Der QuickAnnotator bietet eine schnelle Möglichkeit Eigennamen, Taxa und übliche Nomen sowie Satzgrenzen zu annotieren. Das Dokument wird dazu als Reihung von Wort Token dargestellt, welche wiederum zu einem sogenannten Multitoken vereint werden können. Token können mit dem QuickAnnotator einer oder mehreren Kategorien zugewiesen werden, die sich optisch durch eine farbliche Kodierung hervorheben. Außerdem existiert die Option ein Kommentar für ein Token zu verfassen und etwaige Kommentare anderer Benutzer zu lesen.

3.2.2. KnowledgeBaseLinker

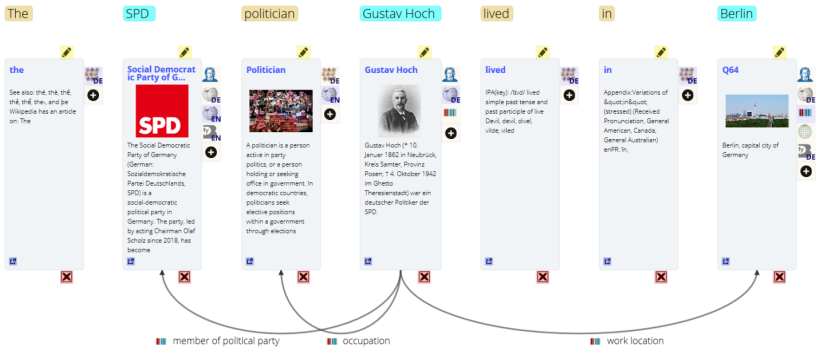


Abbildung 3.2.: Ansicht des KnowledgeBaseLinker Werkzeuges des TextAnnotator (Abrami, Mehler, Lücking u. a. 2019)

Der KnowledgeBaseLinker ist ein Werkzeug für die Annotation von Entitäten mit Wissensressourcen. Mit einer einfachen benutzerfreundlichen Oberfläche ermöglicht der KnowledgeBaseLinker die Integration und Verbindung zu Informationsdatenbanken. Für jedes To-

ken des Dokumentes werden dabei Informationen zu bereits verbundenen Ontologien angezeigt, wie zum Beispiel Bilder, Beschreibungen und Hyperlinks. Die derzeit so unterstützten Ontologiequellen sind Wikidata, Wikipedia, Wiktionary, Geonames, die Deutsche National Bibliothek, GermaNet und Babelfy. Die auf diese Weise abgebildeten Verbindungen von Entitäten zu Elementen von Onotolgie Quellen können unter anderem für das Training von sogenannten Taggern benutzt werden. Ein Anwendungsfall dafür ist das BIOfid[1] (Driller u. a. 2018) Projekt an dem das TextTechnologyLab beteiligt ist.

3.3. Architektur

Wie in Abbildung 3.3 zu sehen ist ermöglicht die TextImager Architektur Text Eingaben aus verschiedensten Quellen, welche anhand einer, im sogenannten Orchestrator zusammengestellten, Pipeline von UIMA AS Services automatisiert analysiert und angereichert werden. Die so resultierenden annotierten Dokumente werden vom ResourceManager (Gleim, Mehler und Ernst 2012) über eine *UIMA Database Interface* (UIMA DI) (Abrami und Mehler 2018) Schnittstelle in einer Datenbank abgelegt und können vom Benutzer erneut für eine weitere Verarbeitung herangezogen werden. Verschiedene Anwendungen, wie zum Beispiel der TextImager und TextAnnotator, können ebenfalls auf die so abgelegten Dokumente zugreifen und diese bearbeiten. Der in dieser Arbeit vorgestellte MobileAnnotator nutzt dem entsprechend, ebenso wie der VAnnotatoR (Abrami, Mehler, Spiekermann u. a. 2020), den TextAnnotator und verwendet dessen Backend Schnittstellen für eine Annotation der Dokumente. Im Gegensatz zu den eher klassischen Bildschirm Ansätzen des TextAnnotators und MobileAnnotators ermöglicht der VAnnotatoR Annotationen in 3D *Virtual Reality* (VR). Der VAnnotatoR macht dabei einen Vorstoß in Gesten gestützte *Human Computer Interaction*.

[1] http://www.biofid.de/de/

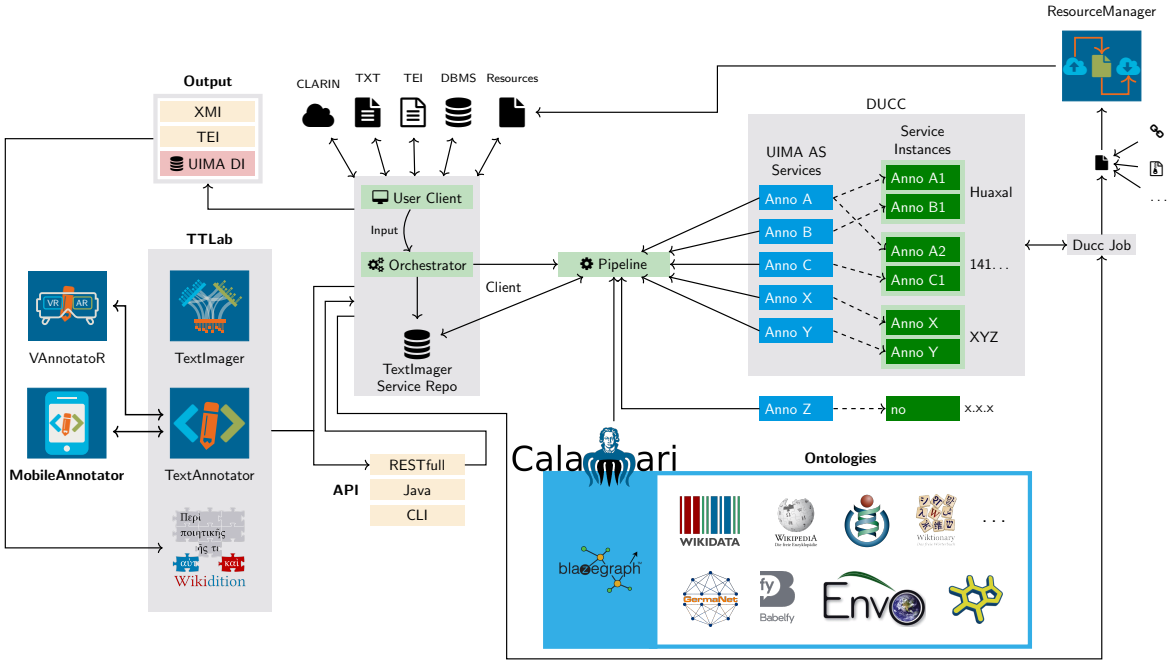


Abbildung 3.3.: TextImager Systemkomponenten und Relationen (Hemati, Mehler u. a. 2019)

4. Konzeptionelle Vorarbeit

4.1. Art der Anwendung

Die grundlegendste Fragestellung ist, ob es eine nativ für ein bestimmtes Betriebssystem laufende Anwendung oder Browser basierte Web Anwendung werden soll. Die Vorteile einer nativ entwickelten Anwendung sind neben einer einfachen lokalen Datenspeicherung auf dem Endgerät, vor allem eine Optimierung für das entsprechende Betriebssystem im Bezug auf Rechenleistung und Performanz. Dagegen bietet eine Webanwendung durch Betriebssystem Unabhängigkeit und den Vorteil auch ohne vorherige Installation verwendbar zu sein.

Auch wenn die einfache Datenspeicherung, insbesondere in Blick auf Offline Nutzbarkeit, durchaus Mehrwert bringen würde, so besticht eine Webanwendung durch ihre Betriebssystem übergreifende Verwendbarkeit. Eine Webanwendung bietet dadurch einfach das größere Potential um möglichst viele Benutzer auf einmal zu erreichen. Auch die Rechenleistung einer nativen Anwendung ist kein entscheidendes Kriterium, da viele Aktionen bereits vom Backend verarbeitet werden und die eigentliche Anwendung somit keine großen Lasten an Rechenarbeit zu bewältigen hat.

4.2. Framework

Für die Auswahl des passenden Frameworks wurden die drei meist verwendeten[1] Frameworks Angular, React und Vue geprüft.

4.2.1. React

React[2] wurde als Framwork von Facebook entwickelt um eine stabile Unterstützung des Facebook Newsfeeds zu gewährleisten. 2013 wurde React zunächst unter der Apache-Lizenz veröffentlicht, aber läuft seit 2017 mit Version 16.0.0 unter der MIT-Lizenz. React wird exzessiv von Facebook für seine Webauftritte (Facebook, WhatsApp, Instagram) verwendet und erfährt regelmäßige Updates. Die aktuelle Version 16.13.1 ist am 19.03.2020 veröffentlicht worden. React ist bei Firmen ein beliebtes[3] Framework und bietet neben einer umfänglichen Dokumentation auch eine große Gemeinschaft[4] an Entwicklern, welche bei Fragen und Problemen Hilfestellung geben können.

[1] http://insights.stackoverflow.com/trends?tags=jquery%2Cangularjs%2Cangular%2Creactjs%2Cvue.js aufgerufen am 23.03.2020

[2] http://reactjs.org

[3] http://medium.com/zerotomastery/tech-trends-showdown-react-vs-angular-vs-vue-61ffaf1d8706 aufgerufen am 26.03.2020

[4] http://insights.stackoverflow.com/trends?tags=angular%2Creactjs%2Cvue.js aufgerufen am 26.03.2020

4.2.2. Angular

Angular[5] erschien im Jahr 2010, und ist somit das älteste der drei Frameworks. Von Google ursprünglich als Javascript Framework entwickelt, erfolgte 2016 mit Version 2.0 eine Umstellung auf Typescript. Die Javascript basierte Version 1 wird dennoch weitergepflegt und läuft unter dem Namen AngularJS. Derzeit läuft Angular auf der am 25.03.2020 veröffentlichten Version 9.1. Ebenso wie React ist auch Angular bei Firmen[6] und Entwicklern[7] beliebt. Neben umfangreicher Dokumentation verfügt Angular im Vergleich zu React auch über eine ähnlich große Entwicklergemeinschaft[7].

4.2.3. Vue

Als Entwicklung eines ehemaligen Google Mitarbeiters erschien Vue[8], oder Vue.js, 2014 und ist somit das jüngste Framework der drei. Vue versucht die positiven Eigenschaften der bisher etablierten Frameworks zu kombinieren. Die aktuell noch in Arbeit befindliche Version 3 von Vue wird, ganz ähnlich zu Angular, auch eine Umstellung auf Typescript beinhalten. Obwohl sich Vue einer schnell wachsenden Beliebtheit[9] erfreut ist die Gemeinschaft an Entwicklern dafür relativ klein[7] im Vergleich zu React und Angluar. Dies kann dazu führen, dass die Hilfe bei Problemen während der Entwicklung ausbleibt oder die Lösung mehr Zeit in Anspruch nimmt.

4.2.4. Vergleich der Frameworks

Während sowohl React als auch Vue mit Javascript arbeiten, ist bei Angular Typescript die vorrangige Sprache. Typescript bringt für Angular eine verbesserte native Typsicherheit für seine Projekte. Durch die statische Typisierung von Typescript können Typfehler bereits frühzeitig bei der Übersetzung festgestellt werden und treten nicht erst zur Laufzeit auf. Jedoch wird Typescript bei Diskussionen um Angular auch häufig als negativ bewertet, weil viele Entwickler sich zunächst Typescript aneignen müssen bevor sie mit der eigentlichen Programmierung beginnen können.

	React	Angular	Vue
Veröffentlicht	2013	2010	2014
Sprache	Javascript	Typescript	Javascript
Model	Virtual DOM	MVC / Real DOM	Virtual DOM
Verwendet von u.a.	Facebook, Netflix	Google, Microsoft	Gitlab, Nintendo

Tabelle 4.1.: Frameworks im Vergleich

[5] http://angular.io

[6] http://medium.com/zerotomastery/tech-trends-showdown-react-vs-angular-vs-vue-61ffaf1d8706 aufgerufen am 26.03.2020

[7] http://insights.stackoverflow.com/trends?tags=angular%2Creactjs%2Cvue.js aufgerufen am 26.03.2020

[8] http://vuejs.org

[9] http://star-history.t9t.io/#facebook/react&vuejs/vue&angular/angular aufgerufen am 26.03.2020

React ist auf Bibliotheken von Drittanbietern angewiesen, was für den Entwickler allerdings eine höhere Flexibilität ermöglicht. Im Gegensatz dazu bietet Angular eine starrere Art der Entwicklung, aber damit verbunden auch eine Vielzahl von mitgelieferten Ressourcen. Vue ist wie React ebenfalls äußerst flexibel gehalten um den Entwicklern viele Möglichkeiten für individuelle Anpassungen zu ermöglichen.

4.2.5. Fazit

Insbesondere in Bezug auf eine stetige Weiterentwicklung als OpenSource Projekt sind die Vorteile einer nativen Typsicherheit nicht von der Hand zu weisen, auch wenn Typescript gleichzeitig eine Hürde für Entwickler darstellen kann, da es nicht so stark verbreitet ist wie Javascript. Auch die bereits in Angular enthaltenen Ressourcen und die starre Projektstruktur dienen der Sicherstellung einer einheitlicheren Implementierung, sollten mehrere Entwickler daran weiter arbeiten. Die Flexibilität von React und Vue durch die Verwendung von Modulen von Drittanbietern kann dahingehend nachteilig werden. Durch unterschiedliche Preferenzen der Entwickler kann es schnell dazu führen, dass verschiedene Module für die selbe Funktionalität herangezogen werden. Dadurch würde die Anwendung nicht nur unnötig an Größe, sondern auch an Komplexität gewinnen, was die Wartung und Erweiterung der Anwendung erschwert. Letzten Endes überwiegen somit die Vorteile für Angular im Bezug auf dieses Projekt.

4.3. Browser Unterstützung

Nachdem geklärt ist, dass es sich um eine Browser basierte Webanwendung handelt, muss eine Spezifikation der unterstützten Browser geklärt werden. Es existiert eine Vielzahl an Browsern für die verschiedenen Endgeräte. Teilweise sind diese Browser nur für bestimmte Endgeräte verfügbar, wie zum Beispiel Apple's Safari Browser. Einige Browser, oder Browserversionen, können auch Einschränkunge durch fehlende Unterstützung aktuellerer Web Standards aufweisen. Eine Anwendung mit gewährleisteter Unterstützung sämtlicher Browser zu entwickeln ist folglich nur mit einem sehr großen Aufwand und entsprechenden Einschränkungen möglich. Um das Verhältnis von Aufwand und Nutzen möglichst ausgeglichen zu halten wird somit im Folgenden genauer spezifiziert für welche Browser eine Unterstützung der Anwendung gewährleistet wird.

Die erste Spezifikation ist eine Einschränkung auf bestimmte Betriebssysteme und deren unterstützte Browser. Wie in Tabelle 4.2 zu sehen ist eine diesbezügliche Einschränkung auf Android und iOS sinnvoll. Android mit 73,5%[10] und iOS mit 25,89%[10] Marktanteil decken somit über 99% der mobilen Endgeräte ab.

	Android	iOS	KaiOS	Samsung
Marktanteile[10] [%]	73,3	25,89	0,23	0,18

Tabelle 4.2.: Marktanteile von mobilen Betriebssystemen

[10]http://gs.statcounter.com/os-market-share/mobile/worldwide aufgerufen am 26.03.2020

Sowohl Android als auch iOS besitzen jeweils einen Standard Browser, welcher für die Endgeräte bereits vorinstalliert mit ausgeliefert wird. Bei Android ist dies Google's Chrome Browser und bei iOS entsprechend der bereits erwähnte Safari Browser von Apple.

	Chrome	Safari	Samsung Internet	UC Browser	Opera
Marktanteile[10] [%]	61,96	23,7	6,33	4,05	1,74

Tabelle 4.3.: Marktanteile von Browsern auf Mobilgeräten

Allein diese beiden Browser decken bereits über 85% des Marktanteils ab (siehe Tabelle 4.3). Eine explizite Gewährleistung der Funktionalität für die danach kommenden Browser Samsung Internet (6,33%[11]) und UC Browser (4,05%[11]) ist im Zuge dieser Arbeit zu aufwendig für den geringen Mehrwert, der dadurch erzeugt werden würde. Es wurde demnach bei der Entwicklung primär auf die Funktionalität innerhalb der Browser Chrome und Safari geachtet, was allerdings nicht ausschließt, dass die Anwendung auch in anderen Browsern problemlos verwendbar ist.

[11]http://gs.statcounter.com/os-market-share/mobile/worldwide aufgerufen am 26.03.2020

5. Realisierung

Nachdem in den vorherigen Kapiteln die Grundlagen geklärt wurden, welche für die Anwendung notwendig waren und letztlich zu dieser führten, folgt nun die Vorstellung der tatsächlich realisierten Anwendung namens MobileAnnotator. Der MobileAnnotator darf dabei als eine Abwandlung des bisher existierenden TextAnnotator verstanden werden. Das Augenmerk für den MobileAnnotator war dabei nicht nur eine mobile Verwendung zu ermöglichen, sondern auch möglichst viele der im TextAnnotator implementierten Funktionalitäten zu übernehmen und gleichzeitig eine intuitive Nutzung durch ein einfach gehaltenes Design zu erlauben.

5.1. Anforderungen

Bevor im Detail auf das realisierte Design eingegangen wird werden zunächst die Anforderungen an die Anwendung genauer betrachtet. Diese Anforderung definieren ein Mindestmaß der zu implementierenden Funktionalitäten.

5.1.1. Use case

Abbildung 5.1 zeigt ein einfaches Anwendungsfalldiagramm (use case diagram) der Anwendung, welches bereits die wesentlichen Anforderungen beinhaltet. So ist eine grundlegende Anforderung die Möglichkeit der Anmeldung und Abmeldung des Benutzers. Etwas weniger trivial hingegen ist die Auswahl eines Werkzeuges. Um ein Werkzeug für die Bearbeitung wählen zu können muss sowohl ein Dokument als auch eine Sicht auf das Dokument gewählt werden. Diese bedingte Anforderung von Dokument, Sicht und Werkzeug entspringt dabei dem TextAnnotator. Letztlich muss es dem Benutzer durch die Anwendung ermöglicht werden Änderungen an dem Dokument vorzunehmen. Diese Änderungen sollen dabei unter anderem in Form einer Erstellung, Bearbeitung oder Löschung einer Annotation möglich sein.

Aus der Anforderung, das Dokument bearbeiten zu können, leiten sich wiederum weitere Anforderungen an die Anwendung ab. So ist für eine Bearbeitung zunächst eine geeignete Darstellung des Dokuments erforderlich. Diese Darstellung muss eine für den Zweck der Bearbeitung passende Auswahl der zu annotierenden Elemente ermöglichen. Ohne eine solche Darstellung wäre es dem Anwender schlicht unmöglich eine genaue Annotation vorzunehmen. Ebenso muss dem Benutzer eine Auswahl an Annotationen zur Verfügung gestellt werden. Dem Anwender muss durch diese Auswahlmöglichkeit klar sein welche Annotationen ihm zur Verfügung stehen und welche Annotation er gerade gewählt hat.

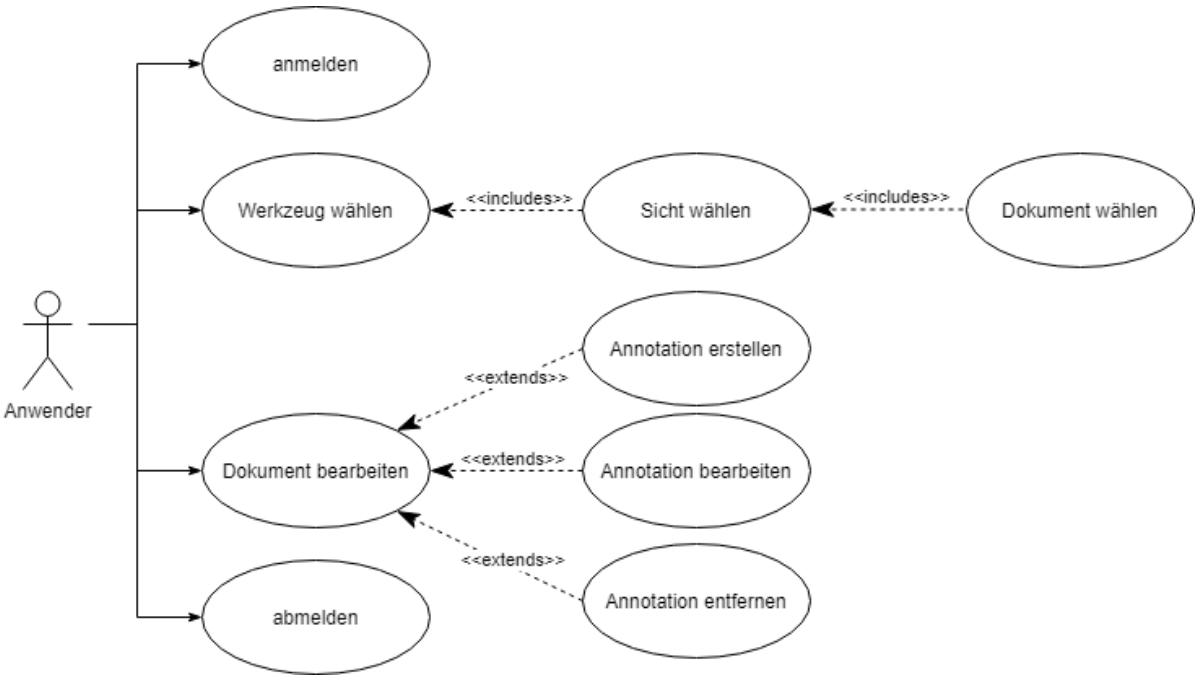


Abbildung 5.1.: Rudimentäres Anwendungsfalldiagramm der Anwendung

Im folgenden Kapitel 5.2 Design wird nun ausführlicher die tatsächliche Realisierung der Anwendung vorgestellt, die sich aus den hier präsentierten Anforderungen ergibt.

5.2. Design

“a simple interface is an important precondition for a positive interactivity and usability experience.” (Lee u. a. 2015 S. 301)

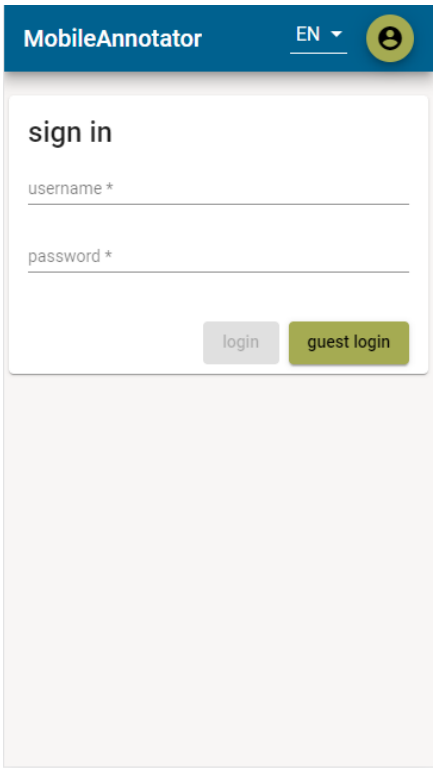
Wie Lee (Lee u. a. 2015) darlegt hat die Einfachheit einer Anwendung einen positiven Effekt auf die Nutzbarkeit und somit auch auf die Zufriedenheit des Benutzers. Insbesondere in Hinblick auf die bisher zumeist eher komplexen Anwendungen (Kapitel 2) kann eine einfach und gut strukturierte Anwendung sich abgrenzen, oder wie Maeda sagt, je mehr Komplexität auf dem Markt ist, desto mehr sticht etwas simples heraus. (Madea 2006 S. 45).

Simplizität kann sich dabei auf verschiedene Art und Weise äußern und nicht nur durch Reduktion erzeugt werden. Maeda hat dazu 10 Gesetze der Einfachheit in seinem Buch (Madea 2006) formuliert, an denen sich das Design der in dieser Arbeit vorgestellten Anwendung orientiert. Die Gesetze “Reduzieren”, “Organisieren”, “Lernen” und “Fehlschläge” nehmen dabei für diese Arbeit eine besondere Rolle ein, um dem Benutzer ein Gefühl von Einfachheit zu vermitteln. Als mobile Abwandlung des TextAnnotators bedeutet “Reduzieren” dabei, Funktionen der Anwendung zu streichen, die nicht für die allgemeine Funktionalität der Anwendung relevant sind. Mit “Organisieren” werden Funktionen verschachtelt und zum Beispiel in Menüs verborgen um die Anzahl an dargestellten Schaltflächen zu minimieren. “Lernen” bezieht sich auf das Wissen eines Benutzers. In dem man bereits etablierte Gesten und Funk-

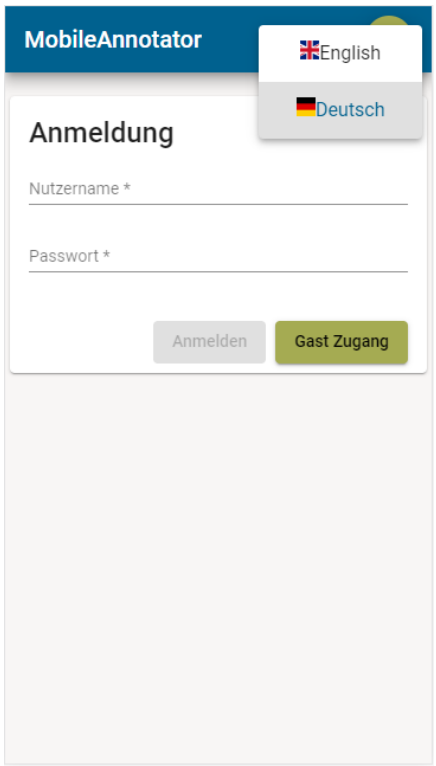
tionen für Anwendungen nutzt, die der Benutzer bereits kennt und mit denen er vertraut ist, kann man ein Gefühl von Einfachheit für den Benutzer erzeugen. Letztlich besagt das Gesetz “Fehlschläge”, dass nicht alles vereinfacht werden kann. Es werden neue Funktionen und auch Gesten für die Bedienung implementiert werden müssen, welche der Benutzer erst lernen muss. Aber auch hier wird versucht die anderen Gesetze zu beachten um die Einfachheit dieser neuen Elemente für den Benutzer zu wahren.

5.2.1. Login

Bei Aufruf der Anwendung wird der Benutzer zur Login Ansicht (siehe Abbildung 5.2) geleitet. Dort bekommt er die Möglichkeit, sich mit seinem Benutzernamen und Passwort anzumelden oder über die in Abbildung 5.2 grün dargestellte Schaltfläche “guest login” sich mit dem öffentlichen Demo Account in die Anwendung einzuwählen. Außerdem erhält der Benutzer hier bereits eine Möglichkeit, wie in Abbildung 5.2b dargestellt, über die sich in der Menüleiste befindliche Schaltfläche die Sprache der Anwendung zu ändern. Nach erfolgreicher Anmeldung gelangt der Anwender zur Startseite/Dashboard (siehe Kapitel 5.2.2).



(a) Login Ansicht auf englisch



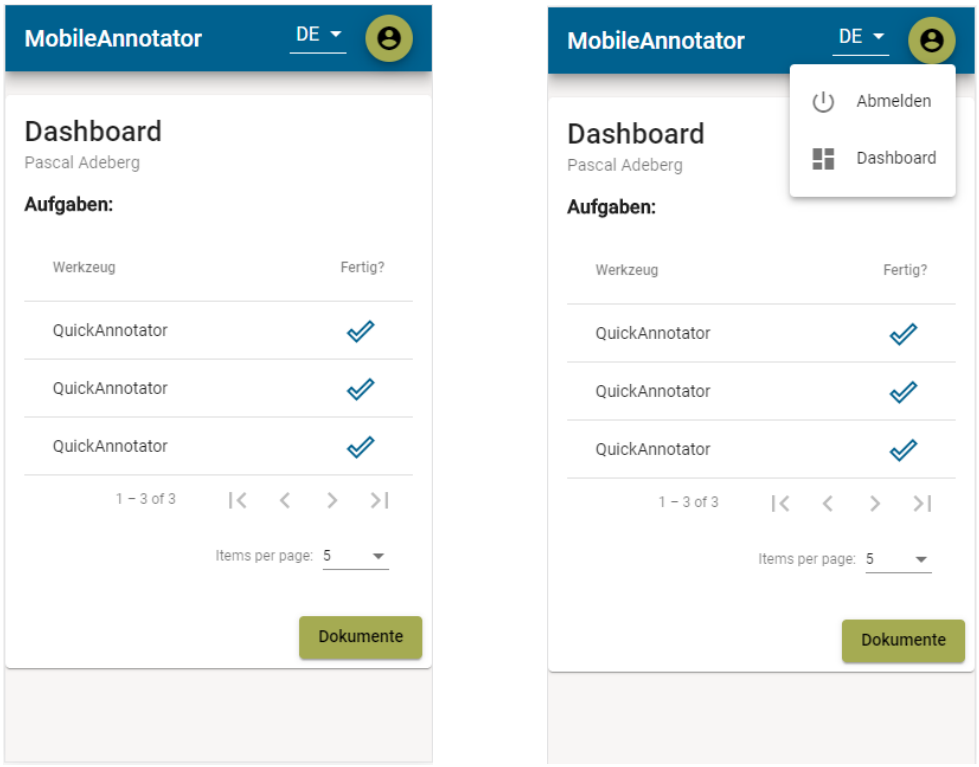
(b) Login Ansicht mit Sprachauswahl

Abbildung 5.2.: Login Ansicht

5.2.2. Dashboard

Das Dashboard ist die persönliche Startseite des Benutzers. Der Anwender findet hier eine Auflistung seiner aktuell zugewiesenen Aufgaben und kann diese, über Auswahl des entsprechenden Eintrags, starten oder als erledigt markieren. Jede Aufgabe ist ein kleiner Auftrag an den Benutzer, mit einem spezifischen Werkzeug die Annotation eines Dokumentes durchzuführen. Die Funktionalität des Werkzeugs kann dabei von der Aufgabe eingeschränkt werden, zum Beispiel kann eine Aufgabe den QuickAnnotator auf eine Teilmenge der eigentlich vorhandenen Kategorien beschränken. Die entsprechende Schnittstelle im TextAnnotator Backend konnte zum Zeitpunkt dieser Arbeit leider nicht fertiggestellt werden. Aus diesem Grund ist das Aufgaben Management in der Anwendung zwar bereits vorgesehen, jedoch bislang nicht vollständig umgesetzt.

Die Menüleiste und deren in Abbildung 5.3b zu sehendes Kontextmenü, welches sich über die grüne Schaltfläche mit dem Personen Symbol aktivieren lässt, bietet dem Benutzer die Möglichkeit zu jedem Zeitpunkt, an dem die Menüleiste sichtbar und anwählbar ist, zu dieser Dashboard Ansicht zurückzukehren. Dies kann dabei auf zweierlei Wege geschehen, entweder über die Auswahl des in Abbildung 5.3b gezeigten Kontextmenü Eintrages “Dashboard” oder über “klicken” auf den Schriftzug “MobileAnnotator”. Die Menüleiste gibt dem Benutzer außerdem zusätzlich die Möglichkeit sich über den Eintrag “Abmelden” im Kontextmenü aus der Anwendung abzumelden und zur Login Ansicht zurückzukehren.



(a) Dashboard mit Liste der Aufgaben

(b) Kontextmenü der Menüleiste

Abbildung 5.3.: Startseite (Dashboard)

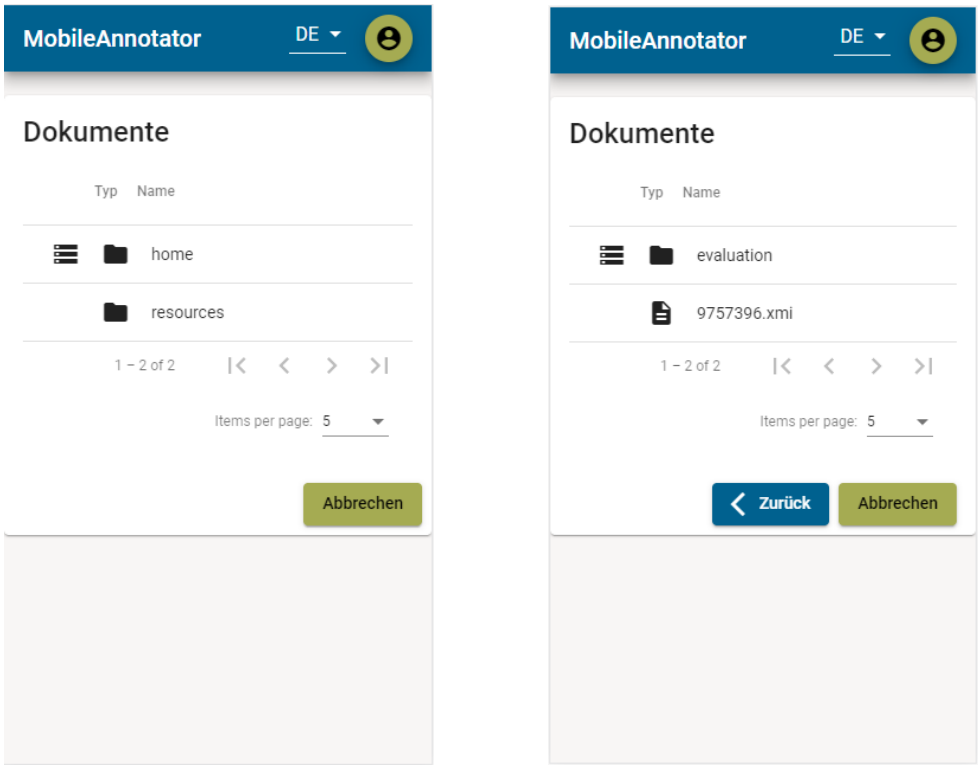
Über eine Schaltfläche unterhalb der Aufgabenliste (in Abbildung 5.3a mit dem Schriftzug “Dokumente” grün dargestellt) gelangt der Benutzer zur Dokumenten Ansicht.

5.2.3. Dokument Selektion

Über die Dokument Selektion Ansicht kann der Benutzer die ihm zugewiesenen Dokumente sehen und für eine Bearbeitung auswählen. Die Ansicht der Dokument Selektion orientiert sich dabei stark an der bereits etablierten Listenansicht für Ordner Strukturen, wie sie zum Beispiel vom Windows Explorer verwendet wird. Dem Benutzer soll durch die bereits bekannte Art der Ordnernavigation die Bedienung erleichtert werden.

Der Inhalt des gerade betrachteten Ordners wird dem Benutzer als drei spaltige Tabelle präsentiert (siehe Abbildung 5.4a). Die linke Spalte dient einer schnelleren Übersicht für die Navigation und zeigt bei Ordner an, ob diese Ordner über weitere Elemente verfügen. Ist ein Ordner ohne Inhalt so bleibt die linke Spalte für diesen leer. Die mittlere Spalte zeigt den Typ des Elementes, zum Beispiel ob es ein weiterer Ordner ist oder ein Dokument (siehe Abbildung 5.4b), während die rechte Spalte den Namen des Elementes anzeigt.

Um bei Ordnern mit vielen Elementen die kleine Benutzeroberfläche nicht zu überladen wurde zusätzlich ein sogenannter Paginator eingesetzt. Dieser verteilt zu große Listen automatisch auf mehrere Seiten zwischen denen man navigieren kann.



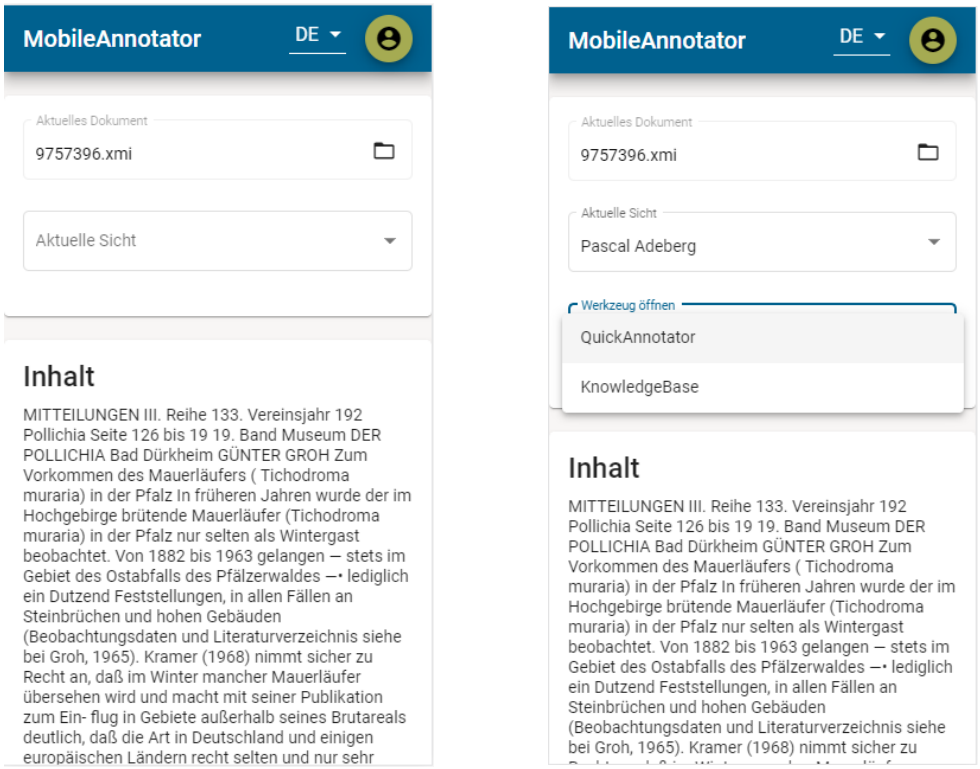
(a) Anzeige der Baumartigen Ordner Struktur (b) Auswählbares Dokument

Abbildung 5.4.: Dokument Selektion Ansicht

5.2.4. Dokument Ansicht

Die Dokument Ansicht verschafft dem Benutzer einen schnellen Überblick über das aktuell geladene Dokument. Sollte kein Dokument geladen sein, so ist die Dokument Ansicht fast leer. In einem solchen Fall wird dem Benutzer einzig eine Auswahlmöglichkeit für das Laden eines Dokumentes angeboten. Mit einem Klick, auf die in Abbildung 5.5a in Form eines Ordners dargestellte Schaltfläche neben dem Dokument Namen, öffnet sich die Ansicht für die Dokumenten Selektion. Das Ordner Symbol soll als visuell verstärkende Komponente dienen um dem Benutzer zu suggerieren dass hier eine Auswahl getroffen werden kann.

Nach dem Laden eines Dokuments wird dem Benutzer nun zusätzlich eine Auswahl für eine Sicht des Dokuments angezeigt und der textuelle Inhalt des Dokuments wird unterhalb der Auswahl Schaltflächen eingeblendet (siehe Abbildung 5.5a). Die Auswahl eines Werkzeuges (siehe Abbildung 5.5b) ist erst nach der Auswahl einer Sicht für den Benutzer verfügbar. Die Wahl eines Werkzeuges leitet den Benutzer zur entsprechenden Werkzeug Ansicht weiter.



(a) Ansicht eines geladenen Dokuments

(b) Auswahl eines Werkzeuges

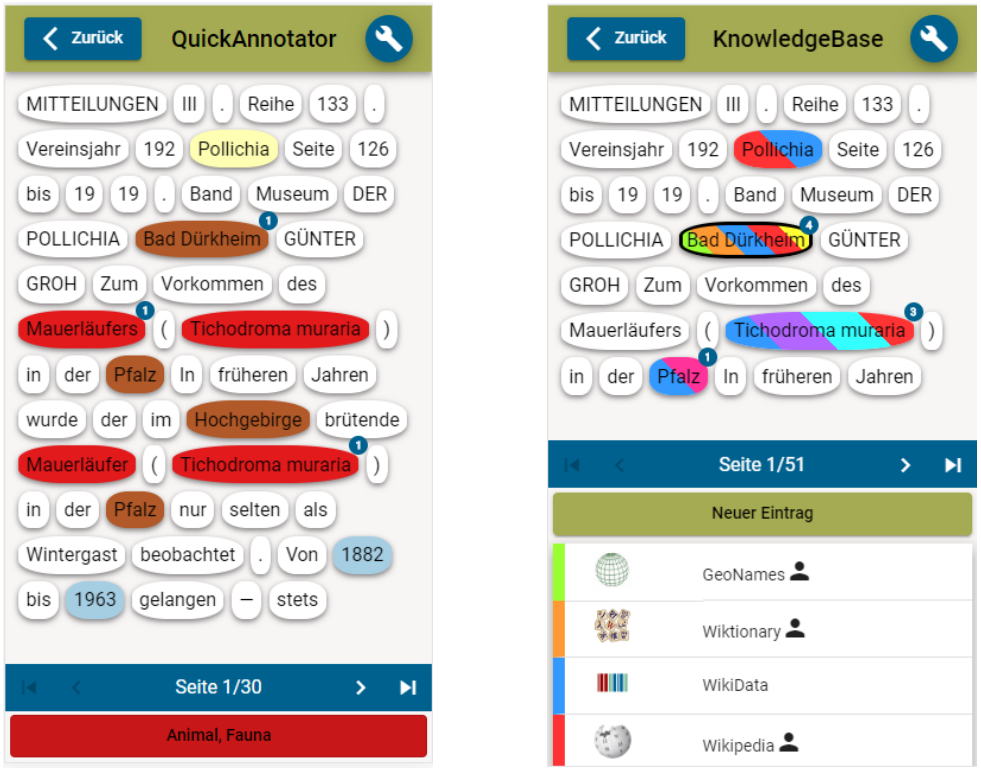
Abbildung 5.5.: Dokument Ansicht

5.2.5. Allgemeines Werkzeug Design

Um den Aufwand für die Einarbeitung in die Werkzeuge des MobileAnnotator zu verringern wurden einige allgemeine Design Elemente und Funktionen definiert und umgesetzt. Der

Benutzer muss somit nur den Umgang mit einem Werkzeug von Grund auf erlernen, während er sich bei jedem weiteren Werkzeug durch die bereits bekannten Funktionen schneller einarbeiten kann. Die gleichbleibenden Funktionen beziehen sich hierbei auf zwei Komponenten, welche im folgenden genauer beschrieben werden.

Dokument Darstellung



(a) Tokenweise Darstellung im Quickannotator

(b) Tokenweise Darstellung im KnowledgeBaseLinker

Abbildung 5.6.: Dokument Darstellung der Werkzeuge

Ungeachtet der Funktionalität der einzelnen Werkzeuge teilen sie dennoch eine große Gemeinsamkeit: Der Inhalt des Dokumentes muss dargestellt werden und auswählbar sein. Eine über die verschiedenen Werkzeuge hinweg gleichbleibende Visualisierung und Bedienung einer solchen Dokumentendarstellung kann dem Anwender dabei eine vertraute Umgebung bietenn in der er sich leichter zurechtfindet. Die vertraute Art und Weise, sich innerhalb eines Werkzeugs durch das Dokument zu navigieren und die für die Annotation relevanten Abschnitte zu wählen, ermöglicht somit die oben beschriebene Verringerung des Lernaufwands für neue Werkzeuge.

Die in dieser Arbeit vorgestellten Werkzeuge QuickAnnotator und KnowledgeBaseLinker verwenden eine identische tokenweise Darstellung für das Dokument (siehe Abbildung 5.6). Das ist dabei ein Bruch mit dem TextAnnotator, welcher für beide Werkzeuge jeweils eine eigene Darstellung realisiert hat.

Für die Darstellung wurde jedes Token in einer ovalen Blase platziert. Im Gegensatz zu einer einfachen Text Darstellung sollen die vom Hintergrund “abgehobenen Blasen” suggerieren, dass der Anwender damit mehr machen kann als nur ihren Inhalt zu lesen. Um den Platz auf dem Bildschirm nicht zu überladen werden die Tokenblasen dabei automatisch auf mehrere Seiten aufgeteilt zwischen denen der Benutzer wechseln kann.

Unterhalb der Tokenblasen befindet sich für die Navigation der Seiten eine eigene Fusszeile. Innerhalb dieser Fusszeile befindet sich dazu eine Anzeige der aktuellen Seitenzahl sowie vier Schaltflächen für das Blättern zwischen den Seiten. Die Schaltflächen bieten dem Benutzer die Möglichkeit eine Seite vor/zurück zu blättern oder zur ersten/letzten Seite zu springen. Alternativ für das Umblättern zu nächsten/vorherigen Seite kann der Benutzer auch eine Wisch-Geste verwenden, wie sie zum Beispiel schon bei Anwendungen für das Lesen von eBooks Verwendung findet.

Die Farbe einer Tokenblase gibt dabei Auskunft über Art und Anzahl der Annotationen für den darin enthaltenen Text. Bei mehr als einer Annotation erhält eine Blase ein Streifen Muster, welches die Farben der verschiedenen Annotationen zu je gleichen Anteilen enthält (siehe Abbildung 5.6b). Für eine leichtere Identifizierung von manuell getätigten Annotationen dient eine Marke am rechten oberen Rand der Tokenblasen. Die in der Marke angezeigte Zahl gibt die Anzahl der manuell erstellten Annotationen wieder.

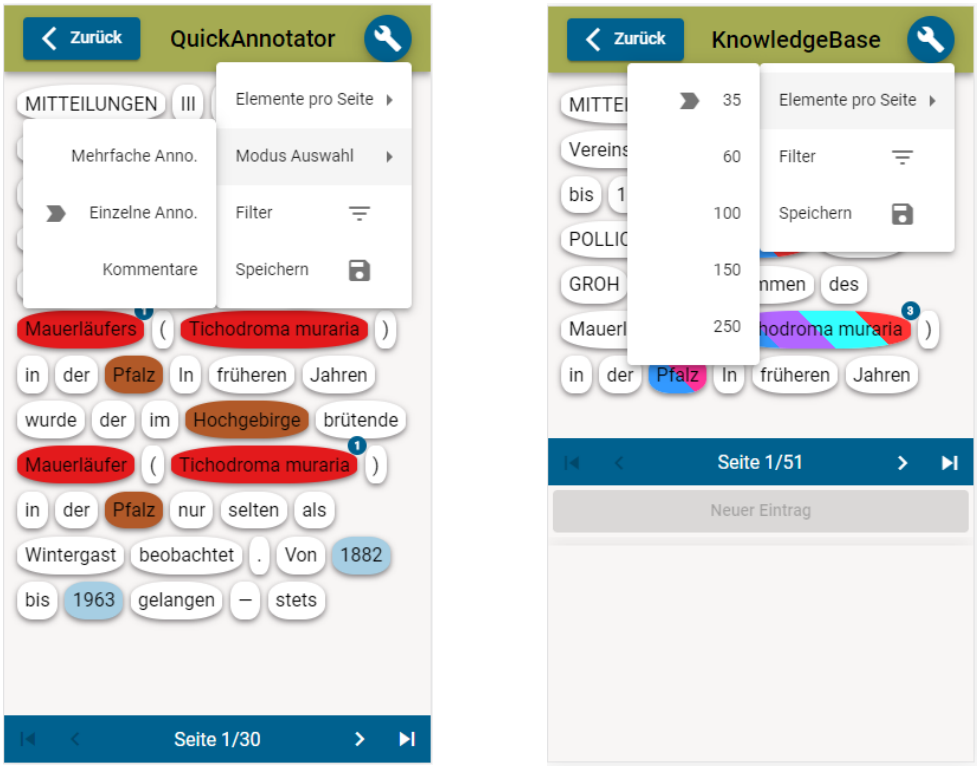
Eine weitere wichtige Funktion für die Darstellung ist die Möglichkeit mehrere Token zu einem sogenannten Multitoken zusammenzufassen und wieder aufzulösen. Ein solches Multitoken vereint alle dafür ausgewählten Token in einer einzigen “Blase”. Um ein Multitoken zu erzeugen muss der Anwender auf eine Tokenblase klicken und länger gedrückt halten. Nach einer kurzen Weile wird die Tokenblase schwarz umrandet und beginnt leicht zu wackeln um die Auswahl des Token visuell zu bestätigen. Anschließend reicht ein einfacher Klick auf ein weiteres Token, um beide Token, und alle sich dazwischen befindlichen, zu einem Multitoken zusammenzufassen. Die Auswahl eines Tokens lässt sich durch klicken auf das selbe Token wieder aufheben. Für das Auflösen eines Multitoken wählt man es wieder durch längeres gedrückt halten aus. Bei der Auswahl eines Multitoken erscheint zusätzlich ein Menü, über welches man das Multitoken wieder in seine ursprünglichen Token aufteilen kann.

Menüleiste

Die zweite allgemeine Werkzeugkomponente, die für ein einheitliches und wiedererkennbares Design sorgen soll, ist erst einmal ein Umbruch zur bisherigen Anwendung. Für eine visuelle Abgrenzung zum Rest der Anwendung wurde für die Werkzeuge eine neue Menüleiste eingefügt (siehe Abbildung 5.7). Somit wird dem Benutzer umgehend suggeriert, dass er sich in einer Werkzeug Ansicht befindet, egal welche Form ein Werkzeug dabei annimmt.

Eine Überlegung, die neue Menüleiste unterhalb der bisherigen einzufügen, wurde während der Implementierung verworfen. Die ursprüngliche (blaue) Menüleiste hatte keinen Mehrwert in einer Werkzeug Ansicht. Für eine Maximierung der nutzbaren Bildschirmfläche wurde sie deshalb für die Werkzeuge entfernt.

Die Werkzeug Menüleiste bietet dem Anwender zwei Schaltflächen sowie eine Anzeige des Werkzeugnamens. Über die Schaltfläche linkerhand des Werkzeugnamens gelangt der



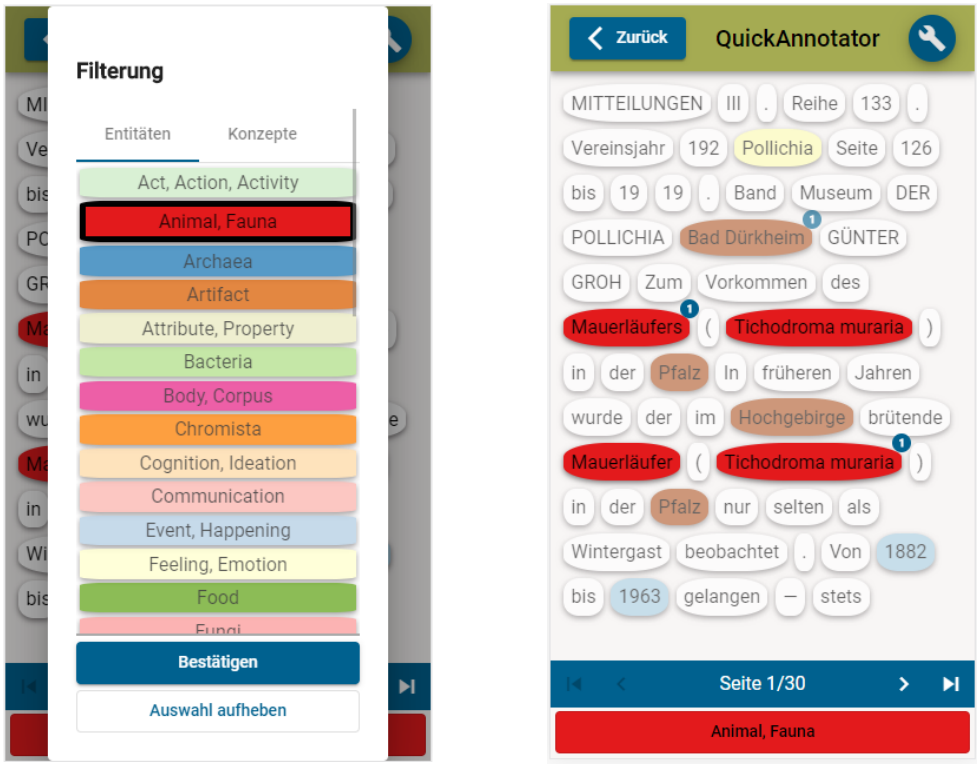
(a) Menüleiste des Quickannotator (b) Menüleiste des KnowledgeBaseLinker

Abbildung 5.7.: Menüleiste der Werkzeuge

Benutzer wieder zurück zur Dokumentenansicht. Die Schaltfläche rechts des Namens öffnet ein Kontextmenü, welches allgemeine und werkzeugspezifische Funktionen enthält. Zu den allgemeinen Funktionen zählen dabei das Speichern des Arbeitsstandes, die Filterung der Tokenanzeige sowie eine Auswahl der Anzahl darzustellendender Tokens pro Seite. Während die Funktionen für Speicherung und Anzahl der Elemente pro Seite selbsterklärend sind, bedarf die Filterung einer näheren Beschreibung.

Bei Auswahl des Kontextmenüeintrags für die Filterung öffnet sich zunächst ein Dialog (siehe Abbildung 5.8a). Innerhalb des Dialogs kann der Anwender eine Auswahl an Kategorien für die Filterung treffen. Die Anzahl und Art der Kategorien kann dabei von Werkzeug zu Werkzeug abweichen, allerdings beziehen sich die Kategorien stets auf die mit dem Werkzeug erstellbaren Annotationen. Nach Bestätigung der Auswahl werden die Token entsprechend ihrer Annotationen und der gewählten Kategorien gefiltert. Die Filterung geschieht durch eine Hervorhebung der relevanten Token indem die restlichen Token leicht verblasst dargestellt werden (siehe Abbildung 5.8b). Dadurch soll dem Benutzer die Orientierung erleichtert und eine schnellere Arbeit ermöglicht werden.

Ein vollständiges Ausblenden der gefilterten Token wurde schnell verworfen, auch wenn der Name Filterung eventuell etwas anderes suggeriert. Zum einen kann der Kontext wichtig für die Bedeutung eines Tokens sein und zum anderen müsste für die Annotation herausgefilterter Token die Filterung wieder aufgehoben werden. Letzteres wäre ein erhöhter



(a) Dialog für Filterung der Token

(b) Quickannotator nach Filterung

Abbildung 5.8.: Filterung der Dokument Darstellung

Zeitaufwand, was einen Widerspruch zum oben beschriebenen Zweck der Filterung darstellt.

5.2.6. QuickAnnotator

Die Hauptkomponente des QuickAnnotators ist die bereits vorgestellte tokenweise Darstellung des Dokumentes.

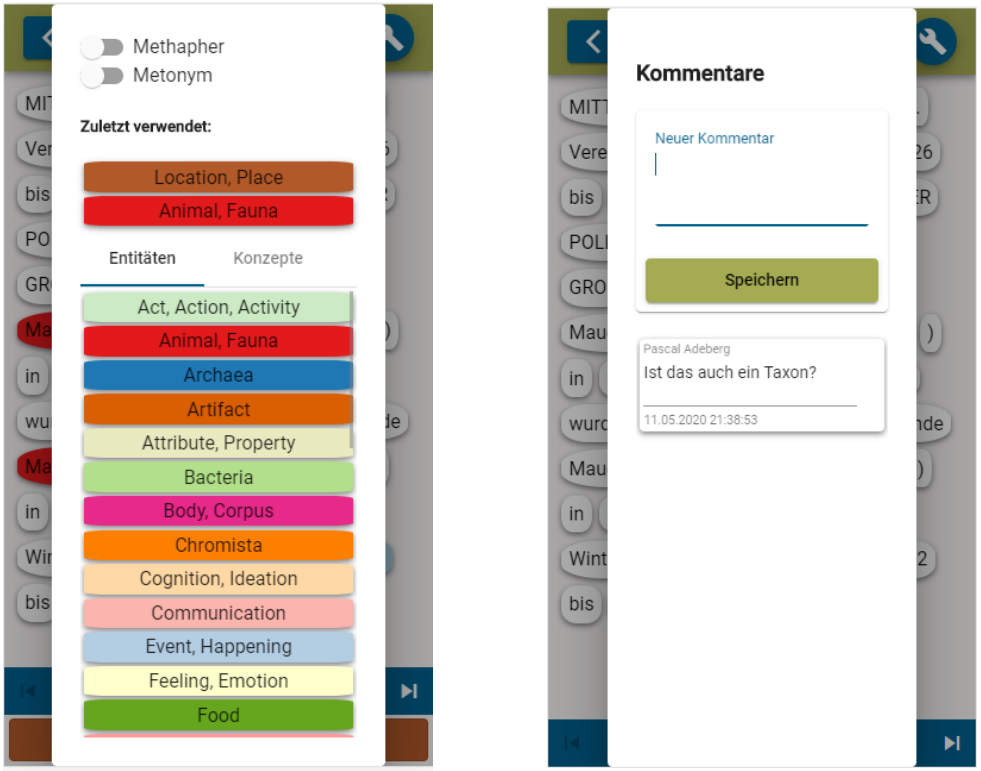
Der QuickAnnotator verfügt über drei verschiedene Modi, welche über das Kontextmenü der Menüleiste ausgewählt werden können.

Im Modus 1 “Mehrfach Anno.” (eng.: repeated anno) lässt sich über die am unteren Rand des Bildschirms befindliche Schaltfläche (siehe Abbildung 5.8b) ein Dialog für eine Kategorieauswahl öffnen (siehe Abbildung 5.9a). Neben einer Auswahlliste sämtlicher verfügbaren Kategorien werden dem Benutzer noch zusätzlich die bis zu drei zuletzt verwendeten Kategorien separat angezeigt. Nach Auswahl einer Kategorie schließt sich der Dialog und die gewählte Kategorie ist nun innerhalb der Schaltfläche am unteren Rand des Bildschirms zu sehen. Mit einem einfachen Klick auf ein Token fügt man ihm eine Annotation mit der gewählten Kategorie hinzu. Auf diese Weise lassen sich schnell mehrere Token mit der selben Kategorie annotieren.

Modus 2 “Einzelne Anno.” (eng.: single anno) ist auf der anderen Seite für eine häufig wechselnde Kategorie ausgelegt und kann zudem genutzt werden um genauere Informationen über die Annotationen eines Tokens anzuzeigen. Im Modus 2 verschwindet die Schalt-

fläche am unteren Rand des Bildschirms und bei Klick auf eine Tokenblase öffnet sich nun ein Dialog. Der Dialog ist dabei fast identisch zum Modus 1, nur werden in Modus 2 die bereits annotierten Kategorien des Token hervorgehoben. Ein Symbol einer Person erscheint außerdem hinter Kategorien, welche manuell von einem Benutzer annotiert wurden.

Bei Auswahl des Modus 3: “Kommentare” (eng: comments) ändert sich die Darstellung der Token. Statt einer farblichen Betonung der Annotationen erfahren nun die Token mit Kommentaren eine Hervorhebung. Die Marke am rechten oberen Rand der Token Blase gibt nun die Anzahl der Kommentare wieder, welche auf dem Token liegen. Per Klick auf ein Token öffnet sich in Modus 3 ein Dialog (siehe Abbildung 5.9b), der erlaubt die Kommentare des Token zu lesen oder neue Kommentare zu verfassen. Somit ist den Benutzern innerhalb des Werkzeuges eine einfache Art der Kommunikation gegeben über die sie sich untereinander austauschen und abstimmen können.



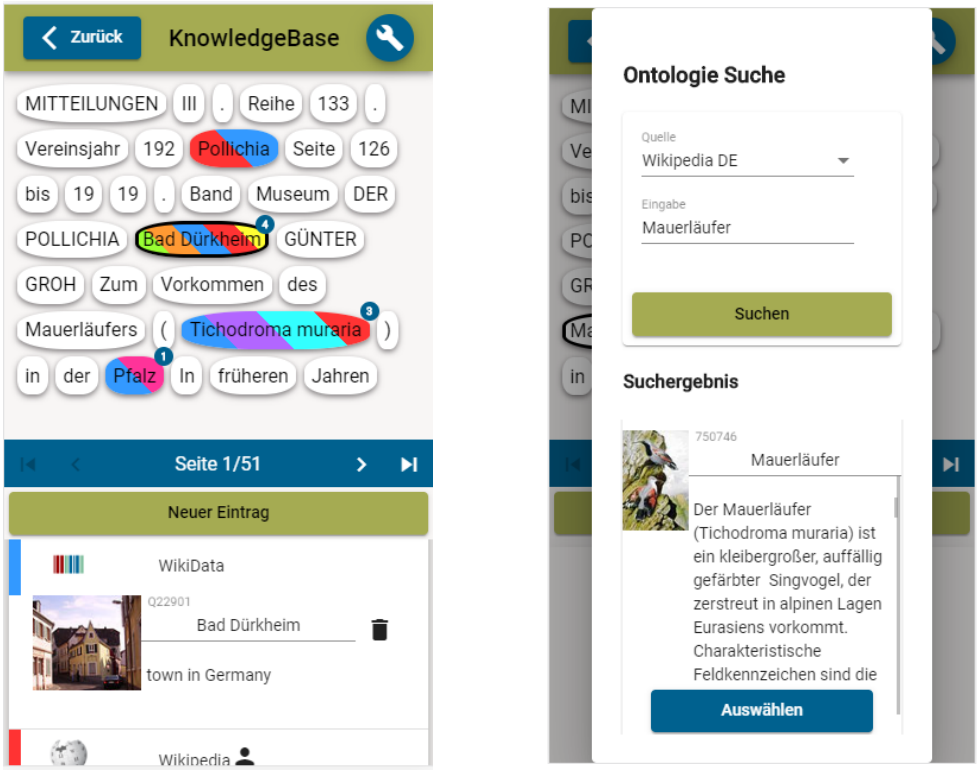
(a) Dialog für die Auswahl einer Kategorie im QuickAnnotator

(b) Dialog für das Lesen und Schreiben von Kommentaren

Abbildung 5.9.: QuickAnnotator

5.2.7. KnowledgeBaseLinker

Ein deutlicher Unterschied zur Ansicht des QuickAnnotator ist die Verkleinerung der Fläche, die für die Darstellung des Dokumentes verwendet wird. Die so freigewordene Fläche wird genutzt um Informationen über die Annotationen eines Tokens anzuzeigen (siehe Abbildung



(a) Details eines Eintrages

(b) KnowledgeBaseLinker Auswahl Dialog

Abbildung 5.10.: KnowledgeBaseLinker

5.6b). Mit einem einfachen Klick auf ein Token kann dieses ausgewählt werden, was durch eine schwarze Umrandung der Token Blase visuell verdeutlicht wird. Für das so gewählte Token werden im unteren Bereich des Bildschirms nun sämtliche verknüpften Ontologien in einer Liste dargestellt. Neben dem Namen und Logo der Ontologiequelle wird hier analog zum QuickAnnotator auch wieder ein Personen Symbol dargestellt um manuell erzeugte Annotationen hervorzuheben. Mit einem Klick auf einen Eintrag kann dieser aufgeklappt werden, was detailliertere Informationen sichtbar macht. Zu diesen Informationen können neben der ID und einem Titel auch Bilder und eine Beschreibung gehören (siehe Abbildung 5.10a).

Oberhalb der Liste befindet sich eine Schaltfläche über die eine weitere Ontologie verknüpft werden kann. Dazu öffnet sich bei Auswahl der Schaltfläche ein Dialog (siehe Abbildung 5.10b) für die Durchsuchung der verschiedenen Ontologiequellen. Innerhalb des Dialoges kann eine der Ontologiequellen gewählt werden und über ein Textfeld die Suche spezifiziert werden. Bei Öffnen des Dialogs wird das Textfeld mit dem Token, bzw. Multitoken, vorbefüllt und kann manuell angepasst werden. Das Suchergebnis erscheint unterhalb der eben beschriebenen Bedienelemente als Liste. Analog zur bereits beschriebenen Listenansicht bereits verknüpfter Ontologien kann auch hier jeder Eintrag per Klick aufegklappt werden um weitere Informationen sichtbar zu machen.

5.3. Backend

Das Backend der Anwendung besteht aus einem Authentifizierungs Server und einem Daten Server. Bevor eine Verbindung mit dem Daten Server hergestellt werden kann muss zuerst eine Anmeldung über den Authentifizierungs Server geschehen. Dabei wird vom Authentifizierungs Server ein Token generiert, mit dem sich der Client gegenüber dem Daten Server authentifizieren kann.

5.3.1. AuthorityManager

Der AuthorityManager (Gleim, Mehler und Ernst 2012) dient als Authentifizierungs Server für den TextAnnotator und seine Applikationen. Der AuthorityManager benutzt dafür eine REST API, die vom Client angesprochen werden kann. Die wichtigsten Funktionen, die unter anderem darüber bereit gestellt werden, sind die Folgenden:

- Anmeldung eines Benutzers
- Abmeldung eines Benutzers
- Verifizierung ob ein Benutzer noch angemeldet ist.

Bei Anmeldung eines Benutzers über seinen Benutzernamen und Passwort wird dabei vom AuthorityManager ein Token generiert und an den Client zurückgegeben. Dieses Token repräsentiert dabei eine aktive Verbindung (Session) mit dem AuthorityManager und kann genutzt werden um sich gegenüber dem Daten Server auszuweisen. Das Token verliert nach der Abmeldung des Benutzers oder nach Ablauf eines von AuthorityManager festgelegten Zeitraumes seine Gültigkeit. Ob ein dem Client vorliegendes Token noch Gültigkeit besitzt kann dabei jederzeit vom AuthorityManager erfragt werden.

5.3.2. ResourceManager

Die für die Arbeit des Benutzers notwendigen Daten werden vom ResourceManager (Gleim, Mehler und Ernst 2012) verwaltet. Der ResourceManager kann beliebige Daten ablegen und dient als Schnittstelle für den Zugriff auf die verschiedenen Datenquellen, wie zum Beispiel die Dokumenten Datenbank. Der Zugriff auf die Datensätze kann dabei für die Benutzer eingeschränkt werden, indem einem Benutzer Rechte für den entsprechenden Datensatz zugewiesen werden können.

Die Kommunikation mit dem ResourceManager läuft ebenfalls über eine REST API und ermöglicht neben dem Auslesen auch eine Auflistung der für den Benutzer freigegebenen Daten.

5.3.3. TextAnnotator

Die hauptsächliche Kommunikation vom MobileAnnotator zum Backend läuft über den TextAnnotator, der dabei sämtliche, für die Annotation relevanten, Funktionen über eine WebSocket (Fette und Melnikov 2011) Schnittstelle bereit stellt.

Das WebSocket Protokoll unterscheidet sich gegenüber HTTP durch eine bidirektionale Verbindung zwischen Client und Server. Während bei HTTP für eine Antwort des Servers stets eine vorherige Anfrage des Client benötigt wird, kann der Server bei einer offenen WebSocket Verbindung jederzeit Nachrichten an den Client senden. Bei einer Verbindungsanfrage wickeln Client und Server einen sogenannten Handshake (übersetzt: Handschlag) ab um sich gegenseitig zu authentifizieren. Nach dem Handshake werden für die Kommunikation zwischen Client und Server keine weiteren zusätzlichen Daten für Anfragen benötigt, was im Gegensatz zu HTTP, das bei jeder Anfrage einen entsprechenden Header benötigt, die übertragene Datenmenge reduziert.

Das WebSocket Protokoll bietet somit durch die bidirektionale Verbindung und reduzierte Datenübertragung zwei deutliche Vorteile für den MobileAnnotator. Die bidirektionale Verbindung ermöglicht es, Änderungen an Dokumenten dem Benutzer in Echtzeit anzuzeigen. Solche Aktualisierungen sind dabei Plattform unabhängig. Damit ist es möglich Änderungen über den MobileAnnotator zu tätigen welche einem anderen Benutzer im TextAnnotator angezeigt werden und vice versa. Dieser Vorteil zeigt sich vor allem bei der Arbeit in Gruppen, wenn mehrere Benutzer am selben Dokument tätig sind. Jeder Anwender kann somit sofort erkennen, wenn ein anderer Nutzer eine neue Annotation tätigt oder eine Annotation bearbeitet.

Die geringere Datenübertragung spielt insbesondere für eine mobile Anwendung eine wesentliche Rolle. Da mobile Datenübertragung von Providern limitiert werden können und das mobile Datennetz nicht homogen verteilt ist, kann es leicht zu Engpässen bei der Datenübertragung kommen. Eine Reduzierung der zu übertragenden Daten kann einem solchen Engpass also bereits vorbeugen.

Die über die WebSocket Schnittstelle des TextAnnotators übertragenen Daten werden grundsätzlich in eingehende Nachrichten (im folgenden Anfragen genannt) und ausgehende Nachrichten (im folgenden Antworten genannt) klassifiziert. Dabei besteht eine Nachricht stehts aus zwei Komponenten. Zum einen ist dies ein textueller Code, welcher die Art der Nachricht beschreibt, und zum anderen die von der Art abhängigen Daten. Über Anfragen kann der Client Anweisungen an den TextAnnotator schicken, deren Ergebnisse als Antworten zurück zum Client laufen.

Anfragen

Die für den MobileAnnotator relevantesten Anfragen sind dabei die folgenden.

- **open_cas**: Öffnet ein Dokument für die weitere Verarbeitung. Die mitgelieferten Daten beinhalten die Dokument ID.
- **open_view**: Öffnet eine Sicht auf ein Dokument auf der gearbeitet werden soll. Die mitgelieferten Daten beinhalten die Dokument und die Sicht ID.
- **open_tool**: Stellt sicher dass das aktuelle Dokument über alle für ein Werkzeug vorausgesetzten Annotationen verfügt. Die mitgelieferten Daten beinhalten die Dokument, Sicht und Werkzeug ID.

- **create**: Erstellt für das aktuell geöffnete Dokument eine neue Annotation. Die mitgelieferten Daten sind Informationen der Annotation, wie zum Beispiel Typ der Annotation und Position im Dokument.
- **remove**: Entfernt eine Annotation aus dem geöffneten Dokument. Die mitgelieferten Daten beinhalten die ID der Annotation.
- **edit**: Ändert eine Annotation des aktuell geöffneten Dokumentes. Die mitgelieferten Daten beinhalten die ID der Annotation sowie die geänderten Eigenschaften.
- **save_cas**: Speichert sämtliche Änderungen am aktuell geöffneten Dokument. Die mitgelieferten Daten beinhalten die Dokument ID.

Der TextAnnotator bietet die Möglichkeit auf weitere Anfragearten, welche an dieser Stelle jedoch übersprungen werden, da diese für den MobileAnnotator nicht weiter relevant sind.

Antworten

Jede Anfrage an TextAnnotator hat eine Antwort zur Folge, welche dem Client eine Aussage über Erfolg oder Misserfolg der Anfrage gibt. Für die oben benannten Anfragen sind dies die Folgenden:

- **open_cas**: Antwort auf eine erfolgreiche *open_cas* Anfrage. Die mitgelieferten Daten beinhalten Informationen über das Dokument, wie zum Beispiel den Klartext und die möglichen Sichten auf das Dokument.
- **open_view**: Antwort auf eine erfolgreiche *open_view* Anfrage. Die mitgelieferten Daten beinhalten die Dokument und die Sicht ID.
- **open_tool**: Antwort auf eine erfolgreiche *open_tool* Anfrage. Die mitgelieferten Daten beinhalten die Dokument, Sicht und Werkzeug ID sowie sämtliche Annotationen.
- **change_cas**: Antwort auf eine erfolgreiche *create*, *remove* oder *edit* Anfrage. Die mitgelieferten Daten beinhalten die geänderten Annotationsdaten des Dokumentes und die Dokument ID. Im Gegensatz zu den restlichen Antworten wird diese an sämtliche Clients geschickt, die gerade auf der entsprechenden Sicht des Dokumentes agieren.
- **msg**: Liefert eine Text Mitteilung an den Client. Hierüber werden unter anderem Fehlermeldungen über gescheiterte Anfragen übertragen. Die mitgelieferten Daten beinhalten die Mitteilung.

Der TextAnnotator bedient sich für seine Funktionen ebenfalls den bereits beschriebenen ResourceManager und AuthorityManager.

Das Laden der Dokumente sowie deren Speicherung werden vom ResourceManager übernommen. Somit ist ein umfassender Zugriff auch auf Dokumente des TextImager gegeben. Für die dafür nötige Authentifizierung wird vom TextAnnotator der AuthorityManager herangezogen. Dadurch wird sichergestellt, dass keine unberechtigten Zugriffe eines Anwenders über den TextAnnotator laufen können.

5.4. Technischer Aufbau

Nachdem bereits sowohl das Design als auch das Backend vorgestellt wurden, werden im folgenden technische Aspekte beschrieben die sich hinter dem Design befinden und für den Anwender größtenteils verborgen bleiben.

5.4.1. Lokale Datenspeicherung

Die Anwendung nutzt für die lokale Datenspeicherung einzig Cookies. Cookies können von Servern und Anwendungen verwendet werden um textuelle Informationen im Browser abzulegen. Die von der Anwendung verwendeten Cookies und die darin abgelegten Informationen beinhalten zum einen das Token (Session Token) für die Authentifizierung gegenüber den Backend Servern und zum anderen die aktuell vom Benutzer gewählte Sprache. Dadurch ist nach einem Neuladen der Seite sowohl die Anmeldung beim Backend als auch die Anzeige der vom Benutzer gewählten Sprache gewährleistet.

5.4.2. Sicherheit

Um eine widerrechtliche Nutzung der Anwendung durch nicht authentifizierte Benutzer zu verhindern sind sogenannte Wächter (Guards) eingesetzt. Die Guards prüfen dabei vor dem Aufruf einer bestimmten Ansicht der Anwendung ob der Benutzer berechtigt ist diese zu sehen und verweigert ihm gegebenenfalls den Aufruf. Die Anwendung bedient sich dafür aktuell drei verschiedener Guards.

Authentifizierungs Guard

Der Authentifizierungs Guard ist der allgemeinste Guard und verwehrt einem nicht angemeldeten Benutzer den Aufruf sämtlicher Ansicht, mit Ausnahme der Login Ansicht. Im Falle eines Neuladens oder Aufrufs einer bestimmten Ansicht verhindert der Authentifizierungs Guard zunächst den Aufruf der Ansicht und leitet den Benutzer zur Login Ansicht weiter. Parallel dazu wird auf einen Cookie mit einem Session Token geprüft und dieses Token gegebenenfalls durch den Authentifizierungs Server validiert. Kann der Benutzer auf diese Weise mit dem Session Token verifiziert werden so erfolgt umgehend eine Weiterleitung von der Login Ansicht zu der eigentlich aufgerufenen, bzw. neu geladenen Ansicht.

Login Guard

Der Login Guard ist in gewisser Weise das Gegenstück zum Authentifizierungs Guard. Statt die Anwendung gegen nicht authentifizierte Benutzer zu schützen, ist das Ziel des Login Guard einzig die Login Ansicht für einen bereits angemeldeten Benutzer zu blockieren. Sollte ein bereits angemeldeter Benutzer versuchen zur Login Ansicht zu gelangen so greift der Login Guard ein und leitet den Benutzer automatisch zum Dashboard weiter.

Werkzeug Guard

Der Werkzeug Guard schützt den Aufruf aller Werkzeuge, die in der Anwendung verwendet werden können. Seine Aufgabe ist dabei sicherzustellen, dass alle für das Werkzeug nötigen Daten bereits vom Daten Server angefragt wurden und der Anwendung zur Verfügung stehen. Sollte bei der Abfrage der Daten vom Server ein Fehler auftreten, so verhindert der Werkzeug Guard den Aufruf. Somit ist abgesichert, dass keine Fehler aufgrund von fehlenden oder falsch geladenen Daten auftreten können. Während der Guard die Daten vom Server abfragt bekommt der Benutzer dabei einen Dialog mit dem aktuellen Stand der Abfrage angezeigt. Innerhalb des Dialoges hat der Benutzer auch jederzeit die Möglichkeit über eine Schaltfläche die Abfrage manuell abzubrechen. In dem Fall wird eine Weiterleitung zum entsprechenden Werkzeug ebenfalls verhindert.

5.4.3. Routing und Pfade

Für das Routing der Anwendung wurde jeder Ansicht explizit ein eigener Pfad zugewiesen. Der Hauptgrund für diesen Ansatz ist eine für den Benutzer einfacher zu verstehende und auch zu verwendende Anwendung. Dem Anwender ist es somit möglich, durch Lesezeichen des Browsers, bestimmte Ansichten zu speichern und aufzurufen ohne sich dafür jedes mal durch die Anwendung arbeiten zu müssen. Dabei dienen Anfrage Parameter für die Pfade als Speicher für das geladene Dokument und die gewählte Sicht. Das ermöglicht das sofortige Öffnen und Verwenden eines Werkzeuges durch ein Lesezeichen oder die Historie des Browsers und bringt dem Anwender ein Zeitersparnis bei die Arbeit mit der Anwendung.

5.4.4. Services

Der MobileAnnotator verwendet mehrere Services für die Kommunikation mit dem Backend und der Datenspeicherung. Abbildung 5.11 zeigt die Serviceschicht des MobileAnnotator mit den einzelnen Services und deren Abhängigkeiten untereinander.

Auf der untersten Ebene befinden sich der Websocket Service und der Menü Service. Diese beiden Services ermöglichen den Zugriff auf grundlegende Funktionalitäten, die unter anderem auch von den restlichen Services genutzt werden.

WebSocket Service

Der WebSocket Service stellt die Kommunikation zum ResourceManager dar. Jede ausgehende Anfrage an den ResourceManger läuft über den WebSocket Service, genauso wie sämtliche eingehenden Daten über ihn angefragt werden können.

Menü Service

Der Menü Service bietet Funktionen für die generelle Darstellung der Anwendung. Seine wichtigste Funktion ist allerdings die Anzeige von Push-Nachrichten für den Anwender. Andere Services können dem Benutzer über den Menü Service Fehlermeldungen oder Informationen anzeigen.

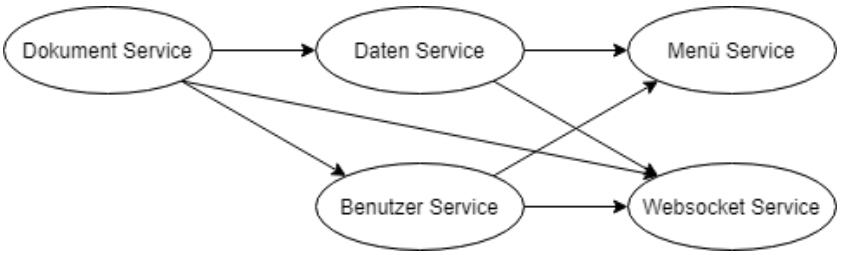


Abbildung 5.11.: Services und deren Abhängigkeiten im MobileAnnotator

Auf dem WebSocket und Menü Service aufbauend befinden sich auf der mittleren Ebene der Benutzer Service und der Daten Service.

Benutzer Service

Der Benutzer Service ist für die Verwaltung des Benutzers zuständig, sowie die Kommunikation mit dem AuthorityManager. Zu den Funktionen des Benutzers Services zählen unter anderem die Anmeldung und Abmeldung des Benutzers. Der WebSocket Service wird vom Benutzer Service zur Authentifizierung gegenüber dem ResourceManager verwendet, während der Menü Service für die Anzeige von etwaigen Fehlern dient.

Daten Service

Der Daten Service verarbeitet die eingehenden Nachrichten des ResourceManager. Dadurch erlaubt der Daten Service einen erleichterten Zugriff auf die entsprechenden Daten. Dies ist notwendig, da die Antwort auf eine Anfrage an den ResourcenManager nicht zwangsläufig in der zeitlich darauf folgenden Nachricht des ResourcenManager enthalten ist. Der Daten Service bietet somit die Funktion explizit auf bestimmte Antworten zu warten. Da die Kommunikation zum ResourceManager über den WebSocket Service läuft, muss der Daten Service diesen zwangsläufig integrieren. Der Menü Service wird vom Daten Service hingegen für die Anzeige von Fehlern oder Informationen verwendet, die vom ResourceManager geschickt werden.

Mit einer Abhängigkeit auf den Benutzer, Daten und Websocket Service steht der Dokument Service an oberster Stelle.

Dokument Service

Im Dokument Service werden die dokumentspezifischen Daten verwaltet. Neben dem aktuell geladenen Dokument beinhaltet dies außerdem die aktuelle Sicht auf das Dokument, sowie das gerade ausgewählte Werkzeug. Zudem bietet der Dokument Service Funktionen für das Laden dieser Daten, indem er die Anfragen über den WebSocket Service an den ResourceManager stellt und über den Daten Service auf die entsprechenden Antworten wartet. Der Benutzer Service dient dem Dokument Service für die Authentifizierung.

6. Evaluation

Im Zuge der Evaluation soll geprüft werden wie gut die Bedienbarkeit des MobileAnnotators ist. Dazu wird zum einen auf die Benutzerfreundlichkeit und intuitive Nutzbarkeit getestet und zum anderen soll für diese Punkte ein Vergleich zum TextAnnotator gezogen werden. Letzteres ist insbesondere von Bedeutung, da der MobileAnnotator eine mobile Variante des TextAnnotator darstellt.

6.1. Evaluations Aufgabe

Für die Evaluation werden Testpersonen gebeten zwei ähnliche, aber nicht identische, Texte mit dem MobileAnnotator und TextAnnotator zu annotieren. Die Texte umfassen dabei jeweils etwa 95 Wörter und handeln von der geographischen Verbreitung bestimmter Tierarten (siehe Anhang B.1.1). Jeweils einer der beiden Texte ist mit einer der Anwendungen zu annotieren, während die Aufgabe innerhalb der beiden Anwendungen stets die gleiche bleibt.

Zunächst sollen die Tiere und Orte des Textes als solche kategorisiert werden. Dazu ist die Testperson aufgerufen den QuickAnnotator zu verwenden und mit dessen Hilfe den entsprechenden Textelementen ihre korrekte Kategorie zuzuweisen. Anschließend sollen die Tiere und Orte eine Verbindung zu ihren entsprechenden Wiktionary Einträgen bekommen. Dies soll mit dem KnowledgeBaseLinker geschehen, welcher die dafür nötige Funktionalität bietet.

Da sich der TextAnnotator und MobileAnnotator zwangsläufig von ihrer Funktionalität und Bedienung ähneln, könnte eine statische Reihenfolge der Anwendungen zu einer Verfälschung der Ergebnisse führen. Wenn zum Beispiel jede Testperson zuerst einen Text mit dem TextAnnotator annotieren muss, bevor sie den MobileAnnotator verwendet, könnten die durch den TextAnnotator gewonnenen Erfahrungen den Anschein einer intuitiven Bedienung des MobileAnnotator erwecken. Um dem vorzubeugen werden die Testpersonen in zwei Gruppen unterteilt. Während die erste Gruppe zuerst Text 1 mit dem TextAnnoator bearbeitet, muss die zweite Gruppe im Gegenzug Text 1 mit dem MobileAnnotator annotieren.

6.2. Evaluations Durchführung

Die Evaluation fand über eine eigens dafür implementierte Webanwendung statt. Testpersonen konnten sich somit unkompliziert und anonym für die Teilnahme an der Studie anmelden und die Evaluation durchführen. Diese Evaluationsanwendung hat dabei die Aufteilung der Testpersonen in die oben genannten Gruppen vorgenommen und die einzelnen Testpersonen durch die Evaluation geführt.

Am Ende wurde den Testpersonen ein *Usability Metric for User Experience* (UMUX, Finstad 2010) Fragebogen vorgelegt (siehe Anhang B.1.2). Innerhalb des Fragebogens werden

den Testpersonen 5 Fragen zur Bedienbarkeit des MobileAnnotator und dessen Vergleich zum TextAnnotator gestellt. Die so gewonnenen Aussagen werden für die Auswertung der Benutzerfreundlichkeit des MobileAnnotator herangezogen. Im folgenden werden nun die einzelnen Ergebnisse der Evaluations Studie dargelegt.

6.3. Ergebnisse

An der Studie zur Bedienbarkeit des MobileAnnotators haben 9 Personen teilgenommen. Zwar lässt diese Anzahl keine statistisch signifikanten Schlüsse zu, nichtsdestoweniger lassen sich dadurch bereits Tendenzen erkennen.

Die Ergebnisse der Studie werden dabei aus zweierlei Quellen gezogen. Zum einen aus dem Fragebogen, welcher jedem Teilnehmer vorgelegt wurde, und zum anderen eine technische Auswertung der von den Teilnehmern getätigten Annotationen innerhalb der entsprechenden Anwendungen. Während der Fragebogen das subjektive Empfinden des Studien Teilnehmers abfragt, wird durch die technische Auswertung ein objektiver Vergleich der beiden Anwendungen möglich.

6.3.1. Fragebogen

Zunächst sind in Tabelle 6.1 die Ergebnisse der Fragebögen zu sehen. Jede Spalte repräsentiert dabei eine Frage des Fragebogens, während jede Zeile die dazu gehörigen Antworten eines Teilnehmers darstellen.

Frage 1	Frage 2	Frage 3	Frage 4	Frage 5
3	6	6	7	6
1	7	6	7	5
1	7	7	7	7
1	6	7	6	4
1	6	7	7	6
1	7	6	7	7
2	6	7	7	7
1	7	7	7	7
1	6	7	6	7

Tabelle 6.1.: Ergebnis der Fragebögen (siehe B.1.2)

6.3.2. Technische Auswertung

Neben den Ergebnissen aus den Fragebögen fand zusätzlich noch eine technische Auswertung der Evaluationen statt, indem die von den Teilnehmern getätigten Annotationen betrachtet wurden. Jede Annotation ist dabei mit einem Zeitstempel versehen, wodurch sich die jeweilige Arbeitszeit eines Teilnehmers ermitteln lässt.

Die Resultate dieser Auswertung sind in Tabelle 6.2 zu sehen. Jede Zeile entspricht der Bearbeitung eines Textes durch einen Studien Teilnehmer. Es wird dabei aufgelistet welcher

Text mit welcher Anwendung bearbeitet wurde. Außerdem wird der Zeitraum aufgelistet, den der Teilnehmer für die Annotation mit dem jeweiligen Werkzeug benötigt hat, ebenso wie die Anzahl der getätigten Annotationen innerhalb dieses Werkzeuges.

		QuickAnnotator		KnowledgeBaseLinker	
Text	Anwendung	Zeit [s]	Annotationen	Zeit [s]	Annotationen
Wolf	TextAnnotator	108,353	17	229,485	16
Luchs	MobileAnnotator	78,926	17	155,943	16
Luchs	MobileAnnotator	146,297	18	179,719	17
Wolf	MobileAnnotator	108,888	14	126,936	12
Luchs	TextAnnotator	173,072	18	295,408	13
Wolf	MobileAnnotator	58,335	14	125,219	16
Luchs	TextAnnotator	77,106	18	359,542	18
Wolf	TextAnnotator	69,269	17	272,490	18
Luchs	MobileAnnotator	53,549	17	140,642	18
Wolf	MobileAnnotator	85,832	13	176,439	13
Luchs	TextAnnotator	428,124	15	937,363	13
Wolf	TextAnnotator	133,612	16	387,248	16
Luchs	MobileAnnotator	227,942	16	373,351	18
Wolf	MobileAnnotator	30,842	14	81,135	14
Luchs	TextAnnotator	146,734	16	295,765	16

Tabelle 6.2.: Ergebnis der technische Auswertung der Evaluationen

Hält man die Teilnehmer Anzahl gegen die in Tabelle 6.2 aufgeführten Auswertungen offenbart sich eine Diskrepanz, dass dort weniger technische Auswertungen gelistet sind als die Teilnehmerzahl erwarten lässt. Dies ist dem Umstand geschuldet, dass bedauerlicher Weise einige wenige Teilnehmer ihre Annotationen nicht gespeichert haben. Dadurch ist es im Nachgang nicht mehr möglich die für die technische Auswertung nötigen Informationen zu beziehen.

6.4. Auswertung

Im folgenden werden nun die Ergebnisse der Evaluations Studie betrachtet und ausgewertet. Anhand der Auswertung soll letztlich eine Aussage möglich sein wie nah das Design des MobileAnnotator tatsächlich an sein Ziel der Intuitivität und Benutzerfreundlichkeit heran kommt.

6.4.1. Fragebogen

In Abbildung 6.1 sind die Mittelwerte für die Antworten des Fragebogens dargestellt. Die einzelnen Antworten der Teilnehmer liegen dabei oftmals sehr dicht beieinander und unterliegen kaum Schwankungen.

Bereits die erste Frage, ob die Verwendung des MobileAnnotators eine frustrierende Erfahrung sei, wird von den Teilnehmern mit einem Mittelwert von 1,33 stark verneint. Die Studien Teilnehmer haben den MobileAnnotator somit durchaus als eine benutzerfreundliche Anwendung wahrgenommen, wie es vom Design her vorgesehen war.

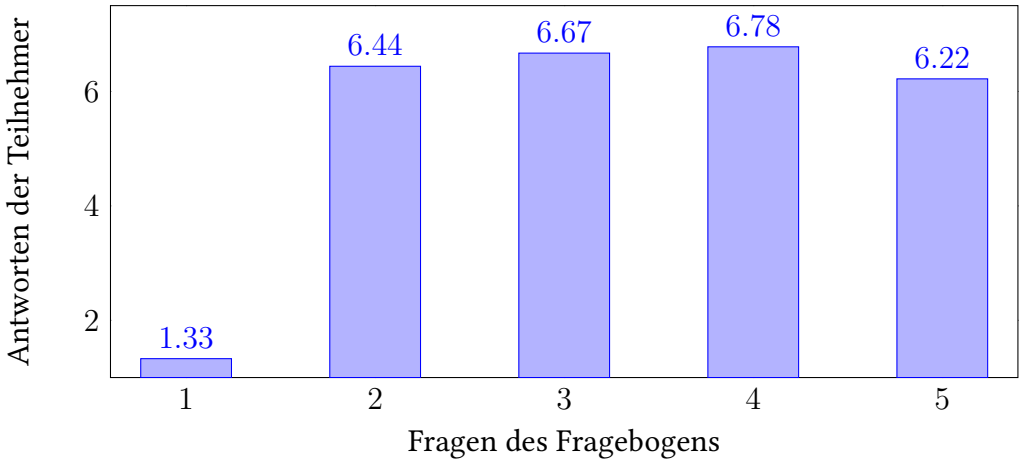


Abbildung 6.1.: Mittelwerte der Fragebögen

Mit durchschnittlich 6,44 von 7 Punkten wurden in Frage 2 die Anforderungen an die Werkzeuge der Anwendung als ausreichend bestätigt. Dies ist eine wichtige Erkenntnis, da es den Werkzeugen des MobileAnnotator gegenüber denen des TextAnnotator an einigen Funktionen fehlt. Als mobile Anwendung mussten einige Funktionalitäten gestrichen oder umgeschrieben werden. Durch die Studie wurde somit bestätigt, dass dabei die relevantesten Funktionalitäten erhalten blieben.

Mit Frage 3 und 4, welche jeweils mit 6,67 bzw. 6,78 von 7 bewertet wurden, haben die Testpersonen eine deutliche Bestätigung der Intuitivität und auch Benutzerfreundlichkeit des MobileAnnotator zum Ausdruck gebracht. Anhand der Frage Nummer 3 wurde die generelle Verwendbarkeit des MobileAnnotator als einfach bekräftigt, während die Antworten auf Frage 4 die einfache Arbeit mit den Werkzeugen untermauern.

Letztlich zeigt Frage 5, dass der MobileAnnotator die Testpersonen gegenüber dem TextAnnotator überzeugt hat. Mit einem anschaulichen Durchschnitt von 6,22 von 7 Punkten bestätigten die Studien Teilnehmer, dass der MobileAnnotator einfacher zu verwenden ist als der TextAnnotator.

Insgesamt bestätigen die Ergebnisse des Fragebogens die ursprünglichen Ziele des Designs einer einfachen und intuitiven Anwendung. Das subjektive Empfinden der Testpersonen gegenüber dem MobileAnnotator ist somit äußerst positiv. Dem gegenüber folgt nun eine objektive Auswertung der Annotationen, welche von den Teilnehmern während der Studie getätigt wurden.

6.4.2. Technische Auswertung

Um einen objektiven Vergleich zwischen dem MobileAnnotator und dem TextAnnotator ziehen zu können werden nun die Ergebnisse der technischen Auswertung (siehe Tabelle 6.2) verarbeitet. Dabei wird insbesondere auf die insgesamt benötigte Zeit und die Geschwindigkeit der Annotationen geachtet. Dazu werden die durchschnittlichen Werte getrennt nach Anwendung, Werkzeug und Text betrachtet. Tabelle 6.3 zeigt die Werte für den QuickAnnotator, während in Tabelle 6.4 die Werte für den KnowledgeBaseLinker zu sehen sind.

	MobileAnnotator		TextAnnotator	
Text	Zeit [s]	$\frac{Annotationen}{Minute}$	Zeit [s]	$\frac{Annotationen}{Minute}$
Luchs	186,585	5,60	206,259	4,87
Wolf	70,974	11,62	103,745	9,64

Tabelle 6.3.: Durchschnittswerte der QuickAnnotator Annotationen

Betrachtet man die beiden Tabellen, bemerkt man einen deutlichen Unterschied im Zeitaufwand zwischen den beiden Texten, welcher unabhängig von Anwendung und Werkzeug auftritt. Dies ist damit zu erklären, dass der Luchs-Text stets zuerst annotiert wurde. Die Befürchtung aus Kapitel 6.1 hat sich somit bestätigt, dass eine statische Reihenfolge der Anwendungen zu einer Verfälschung der Ergebnisse geführt hätte.

	MobileAnnotator		TextAnnotator	
Text	Zeit [s]	$\frac{Annotationen}{Minute}$	Zeit [s]	$\frac{Annotationen}{Minute}$
Luchs	246,406	4,19	472,019	1,91
Wolf	127,432	6,47	296,407	3,37

Tabelle 6.4.: Durchschnittswerte der KnowledgeBaseLinker Annotationen

Vergleicht man die beiden Anwendungen miteinander, so wird ein Geschwindigkeitsvorteil des MobileAnnotator deutlich. Betrachtet man einzig die Auswertung des QuickAnnotator in Tabelle 6.3 so beträgt die Annotationsgeschwindigkeit des TextAnnotator gegenüber dem MobileAnnotator nur 87% für den Luchs-Text und 83% beim Wolf-Text. Ein relativ geringer Unterschied von 13-17 Prozent. Da sich der QuickAnnotator in beiden Anwendungen sehr stark ähnelt, war kein großer Unterschied zu erwarten. Anders ist dies jedoch bei der Auswertung des KnowledgeBaseLinkers, bei dem der MobileAnnotator bewusst mit der Implementierung des TextAnnotators gebrochen hat. Mit gerade einmal 46%, bzw. 52%, der Annotationen pro Minute ist der TextAnnotator hier im Vergleich mit dem MobileAnnotator spürbar langsamer.

6.5. Fazit

Die Auswertung des Fragebogen hat gezeigt, dass der MobileAnnotator durchaus positiv von seinen Anwendern wahrgenommen wird. Sowohl eine Benutzerfreundlichkeit als auch Intuitivität der Anwendung scheinen gegeben und das ohne auf wichtige Funktionen zu verzich-

ten. Außerdem wurde der MobileAnnotator insgesamt von den Testpersonen als einfacher zu verwendende Anwendung gegenüber dem TextAnnotator bestätigt.

Der einfachere Umgang mit dem MobileAnnotator trägt vermutlich Anteil an seiner allgemein höheren Zahl von Annotationen pro Minute, welche durch die technische Auswertung offenbart wurde. Für den KnowledgeBaseLinker ist dieser Unterschied von Faktor 2 durchaus beachtlich. Insgesamt zeigen die Ergebnisse der technischen Auswertung, dass der MobileAnnotator dem TextAnnotator im Punkt Annotations Geschwindigkeit nicht unterlegen ist.

Aufgrund der niedrigen Anzahl an Studien Teilnehmern sind diese Ergebnisse jedoch mit Vorsicht zu betrachten. Um die bisherigen Ergebnisse zu bestätigten, sollte die Studie mit einem größeren Publikum wiederholt werden. Nur so ist eine statistische Relevanz der Ergebnisse möglich. Dabei sollten die Testpersonen im besten Fall weder den MobileAnnotator noch den TextAnnotator bereits kennen. Das ist notwendig, da die bisherigen Auswertungen vermuten lassen, dass die erhobenen Daten dadurch verfälscht werden könnten.

7. Ausblick

7.1. Zukunft des Projekts

Nach Beendigung dieser Arbeit wird der Code des MobileAnnotator veröffentlicht und als OpenSource Projekt voraussichtlich unter der Apache Lizenz weitergeführt. Dadurch soll eine schnellere und stabilere Weiterentwicklung des MobileAnnotator zustande kommen. Zusätzlich werden Erweiterungen und Verbesserungen von Studenten im Rahmen von Praktika an der Goethe Universtität Frankfurt am Main vorgenommen.

7.2. Mögliche Erweiterungen

7.2.1. Weitere Werkzeuge

Im Umfang dieser Arbeit wurden lediglich der QuickAnnotator und der KnowledgeBase-Linker implementiert. Der TextAnnotor selbst verfügt bisher allerdings bereits über 5 weitere Werkzeuge. Diese Werkzeuge können nach Veröffentlichung des MobileAnnotators als OpenSource Projekt von interessierten Entwicklern und/oder Studenten implementiert und eingebunden werden.

7.2.2. Aufgaben Management

Sobald die Erweiterung des TextAnnotators bezüglich des Aufgaben Managements fertiggestellt ist, kann dies auch im MobileAnnotator eingebunden werden.

7.2.3. Offline Daten

Bislang ist der MobileAnnotator auf eine stetige Verbindung zum Internet angewiesen. Gerade auf Reisen sind allerdings Unterbrechungen oftmals nicht zu vermeiden, wie es zum Beispiel bei Bus und Bahn durch Funklöcher oder Tunnel der Fall sein kann.

Eine Alternative wäre es sich Dokumente und Werkzeuge vorher lokal zu speichern und mit diesen Daten zu arbeiten ohne eine Verbindung zum Backend aufrecht erhalten zu müssen. Gleichzeitig müsste es dann eine Möglichkeit geben die lokalen Daten mit dem Backend zu synchronisieren, sobald wieder eine Verbindung möglich ist. Dabei gibt es zwei grundlegende Problematiken, die man für eine solche Offline Daten Nutzung bewältigen müsste:

Lokale Datenspeicherung

Zunächst muss eine Möglichkeit gefunden werden die benötigten Daten in ausreichender Menge auf dem Endgerät zu speichern. Eine mögliche Art der lokalen Datenspeicherung sind Cookies, wie sie bereits vom MobileAnnotator verwendet werden. Die Speichergröße

der Cookies ist allerdings stark limitiert auf 4093 Bytes für sämtliche Cookies einer Domain. Das ist bei weitem nicht ausreichend um die lokale Speicherung eines Dokumentes zu gewährleisten. Sinnvollere Möglichkeiten wären das lokale Dateisystem des Endgerätes oder der Web Storage des Browsers. Ein Zugriff auf das Dateisystem hätte das größte potentielle Volumen für eine lokale Datenspeicherung, aber für die Navigation darin sind weitere manuelle Eingaben des Benutzers erforderlich. Der Web Storage hingegen ist für eine Webanwendung leichter zugreifbar, bringt aber eine Limitierung des Speichervolumens mit sich.

Synchronisation

Die zweite Problematik ist die Synchronisation mit dem Backend. Da es mehreren Benutzern gleichzeitig möglich ist auf ein und dem selben Dokument zu arbeiten, kann es leicht zu Konflikten bei der Synchronisation kommen. Für ein einfaches Beispiel sollen die in dieser Arbeit bereits erwähnten Multitoken dienen. Konflikte können für Multitoken unter anderem entstehen, wenn ein Benutzer auf seinen lokalen Daten ein Multitoken annotiert hat, ein zweiter Anwender jedoch bereits eine Auflösung des Multitoken an das Backend übermittelt hat. Es bedarf also einer vorher ausgearbeiteten Synchronisationsvorschrift, was in diesem und ähnlichen Fällen geschehen soll.

7.2.4. Weitere Sprachen

Bisher kann der MobileAnnotator nur auf deutsch und englisch verwendet werden. Bei der Implementation wurde allerdings auf eine leichte Erweiterung der möglichen Sprachen geachtet, sodass wenig am eigentlichen Code verändert werden muss. Die einzelnen Sprachen werden über Ressource Dateien verwaltet, weshalb der hauptsächliche Aufwand darin bestehen würde eine solche Ressourcen Datei für die hinzu zu fügende Sprache zu schreiben.

7.2.5. Eingabe neuer Texte

Eine weitere Funktion, die bisher noch nicht aus dem TextAnnotator übernommen wurde, ist die Eingabe neuer Texte. Bisher kann der Benutzer über den MobileAnnotator nur auf bereits vorhandene Dokumente zugreifen, während es im TextAnnotator ebenfalls möglich ist Texte einzugeben und verarbeiten zu lassen, damit diese anschließend dem Benutzer zur Verfügung stehen. Eine ähnliche Funktionalität wäre für den MobileAnnotator auch denkbar.

7.2.6. Konfigurierbarkeit

Die aktuelle Implementation ist starr an die Server des TextTechnologyLabs gekoppelt. Insbesondere in Anbetracht der Veröffentlichung als OpenSource Projekt wäre eine stärkere Konfigurierbarkeit eine wünschenswerte Erweiterung. Dadurch wäre es einzelnen Personen und Arbeitskreisen leichter die Anwendung an ihr eigenes Backend zu koppeln.

8. Resümee

Die Aufgabe dieser Arbeit bestand darin die Vorteile einer mobilen Anwendung für die Annotation von Texten zu evaluieren und eine solche Anwendung zu implementieren. Für letzteres konnte auf dem Backend des bereits existierenden TextAnnotators des TextTechnologyLabs aufgebaut werden, welches bereits umfangreiche Funktionalitäten bereitstellt. Dadurch konnte auf eine vollständige Neuentwicklung einer Anwendung verzichtet und statt dessen ein größeres Augenmerk auf eine geeignete Benutzeroberfläche gelegt werden. Die Anforderungen an die mobile Anwendung waren dabei eine möglichst breite Unterstützung für mobile Endgeräte sowie eine intuitive und benutzerfreundliche Oberfläche.

Um die Unterstützung auf möglichst vielen Mobilgeräten zu ermöglichen wurde ein Browser basierter Ansatz gewählt, welcher unter Verwendung des Angular Frameworks realisiert werden konnte. Die so unterstützten Browser Chrome und Safari decken dabei über 85% des Marktanteils an Browsern auf Mobilgeräten (siehe Kapitel 4.3).

Die Intuitivität und Benutzerfreundlichkeit wurden ihrerseits versucht durch eine einfach gehaltene Oberfläche zu erzeugen. Da diese Punkte allerdings subjektiv vom Empfinden des jeweiligen Benutzers abhängt wurde für dessen Quantifizierung eigens eine Studie durchgeführt. Auch wenn für die Studie keine statistisch signifikante Anzahl an Probanten gefunden werden konnte, lässt die Auswertung der Ergebnisse dennoch Tendenzen erkennen. So haben die Testpersonen die Anwendung insgesamt als sehr intuitiv und benutzerfreundlich empfunden, was auch durch technische Auswertungen der Bearbeitungszeiten bestätigt werden konnte. Zudem wurde von den Teilnehmern der Studie angegeben, dass die mobile Anwendung spürbar leichter zu bedienen sei als ihr Desktop Äquivalent der TextAnnotator.

Im Hinblick auf den stetig wachsenden Markt von mobilen Geräten bietet der MobileAnnotator eine große Chance im Bereich der Text Annotation und NLP. Insbesondere die Tatsache, dass mobile Anwendungen für diesen Bereich zur Zeit dieser Arbeit nur spärlich vorhanden sind, lässt ein großes Potential für den MobileAnnotator erkennen.

Dem Prinzip des Crowdsourcing folgend kann mit Hilfe des MobileAnnotator ein großes Publikum erreicht und nutzbar gemacht werden. Wie in dieser Arbeit dargelegt wurde ist unter anderem für eben diesen Zweck auf eine Intuitivität der Anwendung Wert gelegt worden. Dadurch wird auch unerfahrenen und technisch nicht affinen Personen eine leichte Bedienung ermöglicht.

Nichts desto weniger bietet der MobileAnnotator noch weitreichende Möglichkeiten für Verbesserungen. Sei es eine erweiterte Funktionalität durch die Einbindung neuer Werkzeuge oder eine verbesserte Bedienbarkeit durch die Nutzung von offline Daten. Mit Blick auf die Zukunft ist dadurch eine wachsende Bedeutung des MobileAnnotators im Bereich der mobilen Text Annotation nicht auszuschließen.

A. Glossar

Annotation Unter Annotation, vom lateinischen annotatio (Anmerkung/Vermerk) [DUDEN], versteht sich im allgemeinen eine Anreicherung der ursprünglichen Daten um erweiterte Informationen, ohne dabei den Inhalt oder die Struktur dieser Daten zu verändern. Zum Beispiel eine Notiz oder ein Kommentar.

Annotator Als Annotator wird ein Werkzeug bezeichnet, welches ermöglicht Datenquellen entweder automatisiert oder manuell mit Annotationen anzureichern.

API (Application programming interface) Bezeichnet die Schnittstelle eines Programms oder Services, welche anderen Programmen eine Anbindungsmöglichkeit bietet.

Client Ein Programm das auf einem Endgerät läuft und mit einem Server kommuniziert.

Cookie Textuelle Information die zu einer Website im Browser gespeichert werden kann.

Crowdsourcing Verteilung von kleinen Teilaufgaben auf eine Vielzahl von Benutzer. Dadurch wird die gesamte Aufgabe ohne große Aufwände der einzelnen Benutzer gelöst.

HTTP (Hypertext transfer protocol) Ist ein Protokoll für die Übertragung von Daten und wird zumeist für die Übermittelung von Daten über das Internet verwendet.

Ontologie Menge von Informationen und deren gegenseitige Relation.

REST (Representational state transfer) Ist eine Richtlinie für Webservice APIs.

Routing (engl. “Streckenführung”, “Weiterleitung”) bezeichnet im allgemeinen die Festlegung von Wegen für Datenströme.

Service Ein autarkes Element, das spezifische Funktionen beinhaltet und bereitstellt.

Tagger Ein Algorithmus oder Programm um Elemente eines Korpus automatisiert zu kategorisieren.

B. Anhang

B.1. Evaluation

B.1.1. Verwendete Texte

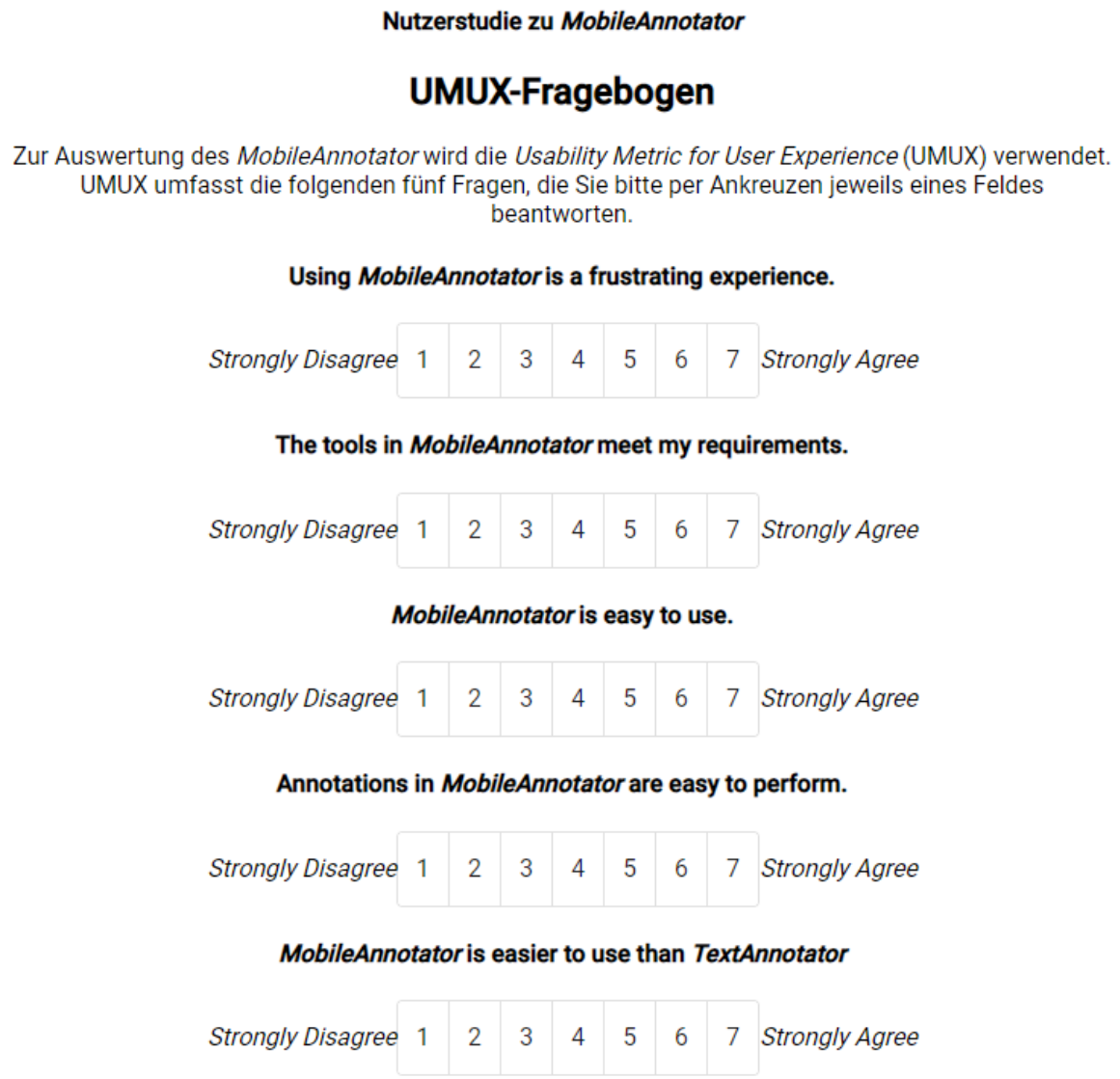
Text 1

Der Wolf war ursprünglich (vor der Ausbreitung des Homo sapiens und der Entwicklung von Land- und Weidewirtschaft) das am weitesten verbreitete Landsäugetier der Erde. Er war in ganz Europa und Asien sowie in Nordamerika beheimatet. In weiten Teilen dieses einst riesigen Verbreitungsgebietes, besonders in großen Teilen Westeuropas und Nordamerikas, wurde die Art durch menschliche Verfolgung ausgerottet. In Osteuropa, auf dem Balkan, in Kanada, Sibirien, der Mongolei und zu einem geringeren Grade in Iran gibt es noch größere zusammenhängende Populationen. Ansonsten ist der Wolf heute nur in isolierten Beständen (manche umfassen weniger als 100 Tiere) anzutreffen.

Text 2

Der Eurasische Luchs besiedelte ursprünglich ein weitgehend geschlossenes Verbreitungsgebiet von den Pyrenäen im Westen bis zum Pazifik im Osten sowie vom Polarkreis bis China; in weiten Teilen dieses Gebiets wurde er ausgerottet, in einigen europäischen Regionen jedoch wieder eingebürgert. Der Pardelluchs war ursprünglich wahrscheinlich über die gesamte Iberische Halbinsel verbreitet, heute ist sein Vorkommen auf kleine, voneinander isolierte Bestände in Südspanien beschränkt. Der Kanadaluchs bewohnt die boreale Zone in Alaska und Kanada, im Süden reicht sein Verbreitungsgebiet bis in einige der Continental United States. Der Rotluchs ist vom südlichen Kanada über die USA bis Mexiko verbreitet.

B.1.2. Fragebogen



Nutzerstudie zu *MobileAnnotator*

UMUX-Fragebogen

Zur Auswertung des *MobileAnnotator* wird die *Usability Metric for User Experience* (UMUX) verwendet. UMUX umfasst die folgenden fünf Fragen, die Sie bitte per Ankreuzen jeweils eines Feldes beantworten.

Using *MobileAnnotator* is a frustrating experience.

Strongly Disagree 1 2 3 4 5 6 7 *Strongly Agree*

The tools in *MobileAnnotator* meet my requirements.

Strongly Disagree 1 2 3 4 5 6 7 *Strongly Agree*

***MobileAnnotator* is easy to use.**

Strongly Disagree 1 2 3 4 5 6 7 *Strongly Agree*

Annotations in *MobileAnnotator* are easy to perform.

Strongly Disagree 1 2 3 4 5 6 7 *Strongly Agree*

MobileAnnotator* is easier to use than *TextAnnotator

Strongly Disagree 1 2 3 4 5 6 7 *Strongly Agree*

Abbildung B.1.: UMUX Fragebogen für die Evaluation

B.2. Hinweise für Entwickler

B.2.1. Package Management Tool

Für die Entwicklung des MobileAnnotator wurde das Package Management Tool Yarn gewählt. Durch die yarn.lock Datei wird somit sichergestellt, dass bei allen Entwicklern die selben Versionen der verwendeten Pakete installiert sind. Zu dem besticht Yarn im Gegensatz zu npm durch eine bessere Performance und einen offline Cache der Packages.

Nach der Installation von Yarn muss die Angular CLI durch folgenden Befehl noch darauf konfiguriert werden.

ng config -g cli.packageManager yarn

B.2.2. Deployment

Bei einem Deployment auf einen Server ist besonders darauf zu achten, dass das Routing des Servers stets die Index.html der Anwendung wiederfindet. Das genutzte Anwendungs interne Routing verändert den URL Pfad der Seite, wodurch es sonst bei einem Neuladen der Seite zu einem HTTP Status Code 404 ”Not Found”kommen kann.

Von einer kompletten Umlenkung sämtlicher Pfade auf die Index.html ist allerdings Abstand zu nehmen, da Ressourcen wie Bilder nicht gefunden werden können.

Literatur

Abrami, Giuseppe und Alexander Mehler (2018). „A UIMA Database Interface for Managing NLP-related Text Annotations“. In: *Proceedings of the 11th edition of the Language Resources and Evaluation Conference, May 7 - 12*. LREC 2018. Miyazaki, Japan.

Abrami, Giuseppe, Alexander Mehler, Andy Lücking, Elias Rieb und Philipp Helfrich (Mai 2019). „TextAnnotator: A flexible framework for semantic annotations“. In: *Proceedings of the Fifteenth Joint ACL - ISO Workshop on Interoperable Semantic Annotation, (ISA-15)*. ISA-15. Gothenburg, Sweden.

Abrami, Giuseppe, Alexander Mehler, Christian Spiekermann u. a. (2020). „Educational Technologies in the area of ubiquitous historical computing in virtual reality“. In: *New Perspectives on Virtual and Augmented Reality: Finding New Ways to Teach in a Transformed Learning Environment*. Hrsg. von Linda Daniela. in press. Taylor & Francis. ISBN: 978-0-367-43211-9.

Abrami, Giuseppe, Manuel Stoeckel und Alexander Mehler (2020). „TextAnnotator: A UIMA based tool for simultaneous and collaborative annotation of texts“. In: *12th International Conference on Language Resources and Evaluation (LREC 2020)*. LREC 2020. accepted. Marseille, France.

Bontcheva, Kalina u. a. (Dez. 2013). „GATE Teamware: a web-based, collaborative text annotation framework“. In: *Language Resources and Evaluation* 47.4, S. 1007–1029. ISSN: 1574-0218. DOI: 10.1007/s10579-013-9215-6. URL: https://doi.org/10.1007/s10579-013-9215-6.

Burghardt, Manuel, Julian Pörsch, Bianca Tirlea und Christian Wolff (2014). „WebNLP: An Integrated Web-Interface for Python NLTK and Voyant“. In: *Proceedings of the 12th edition of the KONVENS conference : Hildesheim, Germany, October 8-10,2014*. Hrsg. von Josef Ruppenhofer und Gertrud Faaß. Hildesheim: Universitätsbibliothek Hildesheim, S. 235–240. URL: https://epub.uni-regensburg.de/35712/.

Cunningham, Hamish, Valentin Tablan, Angus Roberts und Kalina Bontcheva (Feb. 2013). „Getting More Out of Biomedical Documents with GATE's Full Lifecycle Open Source Text Analytics“. In: *PLOS Computational Biology* 9.2, S. 1–16. DOI: 10.1371/journal.pcbi.1002854. URL: https://doi.org/10.1371/journal.pcbi.1002854.

Driller, C. u. a. (2018). „Workflow and Current Achievements of BIOfid, an Information Service Mobilizing Biodiversity Data from Literature Sources“. In: *Biodiversity Information Science and Standards* 2. DOI: 10.3897/biss.2.25876. URL: https://doi.org/10.3897/biss.2.25876.

Fette, Ian und Alexey Melnikov (2011). *The websocket protocol*.

Finstad, Kraig (Sep. 2010). „The Usability Metric for User Experience“. In: *Interact. Comput.* 22.5, S. 323–327. ISSN: 0953-5438. DOI: 10.1016/j.intcom.2010.04.004. URL: https://doi.org/10.1016/j.intcom.2010.04.004.

Fragkou, Pavlina, Georgios Petasis, Aris Theodorakos, Vangelis Karkaletsis und Constantine Spyropoulos (Mai 2008). „BOEMIE Ontology-Based Text Annotation Tool“. In: *Proceedings*

of the Sixth International Conference on Language Resources and Evaluation (LREC'08). Marrakech, Morocco: European Language Resources Association (ELRA). URL: http://www.lrec-conf.org/proceedings/lrec2008/pdf/324_paper.pdf.

Gleim, Rüdiger, Alexander Mehler und Alexandra Ernst (2012). „SOA implementation of the eHumanities Desktop". In: *Proceedings of the Workshop on Service-oriented Architectures (SOAs) for the Humanities: Solutions and Impacts, Digital Humanities 2012, Hamburg, Germany*.

Helfrich, Philipp, Elias Rieb, Giuseppe Abrami, Andy Lücking und Alexander Mehler (2018). „TreeAnnotator: Versatile Visual Annotation of Hierarchical Text Relations". In: *Proceedings of the 11th edition of the Language Resources and Evaluation Conference, May 7 - 12*. LREC 2018. Miyazaki, Japan.

Hemati, Wahed, Alexander Mehler, Tolga Uslu und Giuseppe Abrami (2019). „Der TextImager als Front- und Backend für das verteilte NLP von Big Digital Humanities Data". In: *Proceedings of the 6th Digital Humanities Conference in the German-speaking Countries, DHd 2019*. DHd 2019. Frankfurt, Germany.

Hemati, Wahed, Tolga Uslu und Alexander Mehler (2016). „TextImager: a Distributed UIMA-based System for NLP". In: *Proceedings of the COLING 2016 System Demonstrations*. Federated Conference on Computer Science und Information Systems. Osaka, Japan.

Lee, Dongwon, Junghoon Moon, Yong Jin Kim und Mun Y. Yi (2015). „Antecedents and consequences of mobile phone usability: Linking simplicity and interactivity to satisfaction, trust, and brand loyalty". In: *Information Management* 52.3, S. 295–304. ISSN: 0378-7206. DOI: 10.1016/j.im.2014.12.001. URL: https://doi.org/10.1016/j.im.2014.12.001.

Madea, J. (2006). *The Laws of Simplicity*. MIT Press.

Morton, Thomas und Jeremy LaCivita (2003). „WordFreak: An Open Tool for Linguistic Annotation". In: *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology: Demonstrations - Volume 4*. NAACL-Demonstrations '03. Edmonton, Canada: Association for Computational Linguistics, S. 17–18. DOI: 10.3115/1073427.1073436. URL: https://doi.org/10.3115/1073427.1073436.

Musen, Mark A. (Juni 2015). „The Protégé Project: A Look Back and a Look Forward". In: *AI Matters* 1.4, S. 4–12. DOI: 10.1145/2757001.2757003. URL: https://doi.org/10.1145/2757001.2757003.

Ogren, Philip V. (2006). „Knowtator: A Protégé Plug-in for Annotated Corpus Construction". In: *Proceedings of the 2006 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology: Companion Volume: Demonstrations*. NAACL-Demonstrations '06. New York, New York: Association for Computational Linguistics, S. 273–275. DOI: 10.3115/1225785.1225791. URL: https://doi.org/10.3115/1225785.1225791.

OpenNLP (2010). „Apache OpenNLP". In: URL: http://opennlp.apache.org.

Stenetorp, Pontus u. a. (2012). „BRAT: A Web-Based Tool for NLP-Assisted Text Annotation". In: *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*. EACL '12. Avignon, France: Association for Computational Linguistics, S. 102–107.

Yimam, Seid Muhie, Iryna Gurevych, Richard Eckart de Castilho und Chris Biemann (Aug. 2013). „WebAnno: A Flexible, Web-based and Visually Supported System for Distributed Annotations“. In: *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Sofia, Bulgaria: Association for Computational Linguistics, S. 1–6. URL: https://www.aclweb.org/anthology/P13-4001.