Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Sebastian Gedigk

Anforderungsanalyse und Konzeption eines Annotationsdienstes für ein Digital-Humanities-System

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Sebastian Gedigk

Anforderungsanalyse und Konzeption eines Annotationsdienstes für ein Digital-Humanities-System

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

im Studiengang Bachelor of Science Informatik technischer Systeme
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Jens von Pilgrim
Zweitgutachter: Prof. Dr. Thomas Lehmann

Eingereicht am: 19. Mai 2022

Sebastian Gedigk

Thema der Arbeit
Anforderungsanalyse und Konzeption eines Annotationsdienstes für ein Digital-Humanities-System

Stichworte
Annotation, Digital-Humanities, Dehmel-Archiv, Anforderungsermittlung, Softwareentwicklung

Kurzzusammenfassung
Diese Arbeit hat das Ziel, ein Digital Humanities System (DHS) für die Staats- und Universitätsbibliothek Hamburg (SUB) und Universität Hamburg (Uni HH) zu entwickeln. Digital-Humanities sind heutzutage in den Geisteswissenschaften weit verbreitet. Dieses Themenfeld wächst stetig (Tes21). Das DHS soll die digitalisierten Briefe, Postkarten und anderweitigen Schriftverkehr des berühmten Hamburger Ehepaares Richard und Ida Dehmel darstellen. Die Digitalisierung der Objekte des Dehmel-Archiv befindet sich gerade im Aufbau. Diese Objekte sollen in dem zu entwickelten DHS annotiert werden. Dazu wird im Rahmen dieser Arbeit eine Webanwendung erstellt und eine Schnittstelle für die Annotationen bereitgestellt. Diese Webanwendung wird daher für die AuftraggeberInnen als Prototyp konzipiert. Am Ende dieser Arbeit ist eine funktionsfähige Webanwendung entstanden. Das entstandene DHS könnte für erste Tests in Forschungsgruppen eingesetzt werden und besitzt das Potential weiter ausgebaut zu werden.
Sebastian Gedigk

Title of the paper
Requirements engineering and conception of an annotationservice for a digital-humanities-system

Keywords
annotation, Digital-Humanities, Dehmel-Archiv, requirements engineering, software engineering

Abstract
This work aims to develop a digital-humanities-system(DHS) for the Staats- und Universitätsbibliothek Hamburg(SUB) and University of Hamburg(UniHH). Digital humanities are nowadays widely used in the humanities. This subject area is steadily growing (Tes21). The DHS is

intended to represent the digitized letters, postcards, and other correspondence of the famous Hamburg couple Richard and Ida Dehmel. The Dehmel Archive is currently in the process of digitizing these objects. These objects are to be annotated in the DHS to be developed. For this purpose, a web application will be created and an interface for the annotations will be provided within the scope of this work. This web application is therefore designed as a prototype for the client. At the end of this work a functional web application has been created. The resulting DHS could be used for first tests in research groups and has the potential to be further extended.

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

Abkürzungen

API Application Programming Interface

CD Continuous Delivery

CI Continuous Integration

CORS Cross-Origin Resource Sharing

DBMS Datenbankmanagementsystem

DDHS Dehmel Digital Humanities System

DHS Digital Humanities System

HAW Hochschule für Angewandte Wissenschaften Hamburg

ICC Informatik Compute Cloud

IDE Integrated Development Environment

IIIF International Image Interoperability Framework

JSON JavaScript Object Notation

JWT JSON Web Token

REST Representational State Transfer

SPA Single Page Application

SQL Structured Query Language

SUB Staats- und Universitätsbibliothek Hamburg

Uni HH Universität Hamburg

URL Uniform Resource Locator

W3C World Wide Web Consortiums

XML Extensible Markup Language

1 Einleitung

1.1 Motivation

Diese Bachelorarbeit ist in Folge eines Bachelorprojektes der Hochschule für Angewandte Wissenschaften Hamburg (HAW) entstanden. Bei dem Projekt wurde in Zusammenarbeit mit der Uni HH und der SUB ein erster Protoyp eines Digital-Humanities-System in Form einer Webanwendung entwickelt. Dieser Prototyp sollte genutzt werden, um die digitalisierten Objekte des Dehmel-Archivs von dem Server der SUB abzurufen und in einem Betrachtungstool darzustellen. Am Ende des Projekts wurden Ideen gesammelt, wie man diesen Protoyp weiter aufbauen und nutzen kann. Diese Ideenliste gab den Anstoß diesen Prototypen weiter zu entwickeln und im Rahmen einer Bachelorarbeit umzusetzen. In diesem Rahmen ist es möglich, sein erlerntes Wissen im Bereich der Anforderungsanalyse und Softwareentwicklung mit realen Kunden anzuwenden. Außerdem ermöglicht diese Webanwendung die Forschung in dem Bereich der Digital-Humanities voranzutreiben und mit aktuellen Mitteln durchzuführen. Der Bereich der Digital-Humanities ist ein wichtiger Bestandteil der Forschung der Uni HH und der SUB.

1.2 Problemstellung

Das Problem der Uni HH und SUB ist, dass sie zur Zeit kein Digital-Humanities-System besitzen, mit dem sie in der Lage sind, Annotationen zu ihren Forschungsobjekten hinzuzufügen. Dafür wurde bereits mit dem Bachelorprojekt ein erster Protoyp entwickelt, der dieses Problem beseitigen soll.
Der erste Prototyp kann die Objekte des Dehmel-Archivs darstellen und hat eine Userregistrierung möglich gemacht. Die SUB hat in einem ersten Gespräch mitgeteilt, was für sie wünschenswerte Erweiterungen sind. Wenn der Prototyp weiter entwickelt wird, möchten die Uni HH und die SUB die Möglichkeit haben, verschiedene Annotationen an den einzelnen Objekten durchzuführen. Das heißt, es sollen Vermerke und Anmerkungen an den Objekten hinzugefügt werden. Diese Annotationen sollen für jeden User einsehbar sein. Jedoch soll nicht jeder User in der Lage sein, Annotationen zu schreiben. Dazu war der Vorschlag, dass

der oder die User eingeloggt sein müssen bzw. es die Option der Moderation für die Annotationen gibt. Es soll eine Möglichkeit geben, sich alle Annotationen in einem Standardformat ausgeben zu lassen. Die Arbeit an Digital-Humanities hat einen erheblichen Mehrwert, wenn man Annotationen zu den Objekten hinzufügen kann. Die ForscherInnen sind so in der Lage ihr Wissen zu den Objekten einer breiteren Masse zur Verfügung zu stellen.

1.3 Zielsetzung

Das Ziel dieser Arbeit ist es, dem Auftrag der SUB und Uni HH gerecht zu werden. Dazu muss herausgefunden werden, ob es ein solches Annotationssystem/ -plugin für die Bildbetrachtungssoftware des Prototypen gibt. Es muss geprüft werden, ob dieses Plugin den Anforderungen entspricht, die Annotationen für jeden User einsehbar zu machen. Dazu muss ein Dienst oder Service bereitgestellt werden, auf dem die Daten hinterlegt werden können. User sollen sich einloggen können. Wie dieser Login erfolgen soll, ist ebenfalls Bestandteil der Anforderungsanalyse und Weiterentwicklungsphase des Systems. Die Annotationen sollen in einem Standardformat abgelegt werden, damit es möglich ist, bei einem Systemwechsel alle hinzugefügten Annotationen auf dem neuen System zu integrieren. Es kann so sichergestellt werden, dass man die Annotationen plattformunabhängig weiter verwenden kann.

1.4 Einführung ins Dehmel-Archiv

Das Dehmel-Archiv umfasst die Nachlässe von Richard und Ida Dehmel. Das Archiv hat fünf Hauptabteilungen und beinhaltet eine Vielzahl an (Oss22):

- Manuskripten
- Briefen
- Büchern aus der privaten Sammlung des Ehepaars
- Musikalien
- Allerlei anderen Gegenständen und Dokumenten aus ihrem Leben

Richard Dehmel wurde 1863 geboren und lebte bis 1920. Er war ein deutscher Dichter und Schriftsteller, der sich ab 1901 in Hamburg niederließ. Richard schrieb Kinderbücher und Gedichte und inspirierte damit viele kreative Menschen seiner Zeit. Im 1. Weltkrieg meldete sich Richard freiwillig für das Militär. Zusammen mit seiner Frau Ida Dehmel baute er sein

Haus zu einem Gesamtkunstwerk aus. Er starb 1920 in seinem Künstlerhaus in Blankenese an den folgen einer Venenentzündung aus der Zeit des Krieges (Wik21b).

Ida Dehmel wurde 1870 geboren und starb 1942. Sie betrieb zeitweise einen Salon in Berlin, in dem Künstler ihre Werke ausstellen und präsentieren konnten. 1901 zog sie zusammen mit Richard Dehmel nach Hamburg und heiratete ihn in dem selben Jahr. Sie engagierte sich viel für Fraunerechte und gründete den Künstlerinnenverband GEDOK (Gemeinschaft deutscher und oesterreichischer Künstlerinnen und Kunstfreundinnen). Nach dem Tod von Richard Dehmel kümmerte sie sich um den literarischen Nachlass und hielt das Dehmelhaus als Erinnerungsort instand. Ab 1933 galt sie in Deutschland als Jüdin und wurde ausgegrenzt und durfte nicht mehr veröffentlichen. Sie nahm sich 1942 aufgrund ihrer Lebenssituation im nationalsozialistischen Deutschland das Leben (Wik21a).

Das Dehmel-Archiv besitzt ca. 35.000 Briefe und anderen Schriftverkehr, der an das Künstlerpaar gesendet wurde bzw. von dem Künstlerpaar versendet wurde und erschließt diese mit der Uni HH. Dazu wurden in einem ersten Schritt etwa 4000 Objekte aus dem Archiv digitalisiert. Diese Objekte sind zum Beispiel Telegramme, Briefe und Postkarten.

1.5 Digital-Humanities

Digital Humanities bezeichnet ein Forschungsgebiet der Geisteswissenschaften. Es ist der „gezielte Einsatz digitaler Medien in allen klassischen Disziplinen der Geisteswissenschaften…“ (Kur16). Mittlerweile gibt es viele Archive im Internet. Diese Archive erlangen auf Grund der stetig wachsenden Digitalisierung mehr und mehr an Bedeutung. Heute werden viele historische Objekte digitalisiert und so im Internet einer größeren Menge an Personen zugänglich gemacht, zum Beispiel durch Museen, Bibliotheken, Universitäten oder andere staatlichen Institutionen. Dabei ist es wichtig, dass diese digitalisierten Objekte in einer adäquaten Art und Weise repräsentiert werden können. Um an ihnen arbeiten und forschen zu können, müssen die leicht abrufbar und erreichbar sein (Tes21).
Dazu gibt es bereits Tools die das ermöglichen. In dieser Arbeit wird das Tool Mirador genutzt um mit den Objekten zu arbeiten und diese darzustellen.
Das ist eine neue Art Forschung zu betreiben. Sie muss wachsen und man muss sehen, wie man das Potential des Internets mit Hilfe der Digitalisierung der Objekte vollumfänglich nutzen kann.

2 Anforderungsanalyse

Die Anforderungsanalyse ist ein wesentlicher Bestandteil der Softwareentwicklung.

„Wenn nicht alle Projektbeteiligten ein hinreichendes Verständnis dafür entwickeln, welche Teile der Arbeit ein Softwaresystem für uns übernehmen soll, erhalten wir Produkte, die nicht wirklich brauchbar sind und das Leben einfacher machen. Verschiedenste Statistiken belegen, dass 60% der Fehler in Softwaresystemen auf mangelhafte Requirements zurückgehen – Fehler, die man bei etwas mehr Aufmerksamkeit für dieses Thema leicht vermeiden könnte." (Hö14, Seite 423)

Diese Aussage zeigt, wie wichtig eine eindeutige Anforderungsanalyse ist. Dazu gehört, dass eine offene und am besten iterative Kommunikation mit den Stakeholdern stattfindet. Es erspart allen Beteiligten Zeit und Geld.

2.1 Anforderungsermittlung

Die Anforderungsermittlung beginnt mit dem Feststellen des IST-Zustands der Systeme aus dem vorangegangen Projekt. Das Projekt wurde im Sommersemester 2021 an der HAW im Department Informatik durchgeführt. In dem Projekt wurden zwei Teams gebildet, die unabhängig voneinander ein DHS entwickelt haben. Beide Teams hatten am Ende des Projekts einen Prototypen in Form einer Webanwendung fertiggestellt , der die Objekte mit Hilfe des Bildbetrachtungstools Mirador darstellen konnte. Mirador ist ein Bildbetrachtungstool und wird im Kapitel 4.2 genauer vorgestellt. Team 1 hat das Frontend sehr simpel gestaltet. Zusätzlich zu dem Frontend-Server wurde ein Backend-Server erstellt, damit man eine Userverwaltung implementieren kann. Diese Userverwaltung wurde nicht vollständig umgesetzt. Es war möglich sich zu registrieren und anzumelden, jedoch hatte der angemeldete User keine Möglichkeit mit dem System mehr zu machen als der nicht angemeldete User. Team 2 hat beim Frontend einen größeren Wert auf die Optik und einfache Bedienbarkeit der Webseite gelegt. Dieser Prototyp bestand aber nur rein aus dem Frontend-Server. Die zwei Prototypen dienten als Hilfestellung, wie man die Objekte mit dem Bildbetrachtungstool Mirador darstellen kann.

Dass die Objekte mit dem Tool Mirador angezeigt werden sollen, war bereits in dem Projekt des Sommersemesters ein Wunsch der SUB.
Die Anforderungsermittlung für das System, welches Bestandteil dieser Arbeit ist, wurde in mehreren Onlinemeetings mit folgenden Stakeholdern durchgeführt.

Name	Erreichbarkeit	Bezug zum Projekt
Prof. Dr. Julia Nantke	julia.nantke @uni-hamburg.de	Betreut das Erschließungs- und Forschungsprojekt Dehmel digital in Kooperation mit der SUB
David Maus	david.maus @sub.uni-hamburg.de	Leiter der Abteilung Forschung und Entwicklung, Staats- und Universitätsbibliothek Hamburg
Prof. Dr. Jens von Pilgrim	jenshenning.vonpilgrim @haw-hamburg.de	Professor für Programmiermethodik an der HAW Hamburg und Betreuer dieser Arbeit
Sebastian Gedigk	sebastian.gedigk @haw-hamburg.de	Entwickler des neuen Prototypen Dehmel Digital Humanities System (DDHS) und Verfasser dieser Arbeit

Tabelle 2.1: Stakeholderübersicht

Anhand der ersten beiden Systeme aus der vorherigen Projektarbeit wurde ermittelt, was die neuen Anwendungsfälle des Systems werden sollen bzw. welche Features nicht implementiert wurden. Die Meetings fanden aufgrund der COVID-19 Pandemie online statt. Die ermittelten Anwendungsfälle wurden dann in einem iterativen Prozess in den ersten drei Meetings mit den Stakeholdern spezifiziert und genauer ausgeplant.
Durch die Iteration in Bezug auf die Anforderungsermittlung hat sich bereits im Vorgänger-Projekt gezeigt, dass es so gut möglich ist, mit den Stakeholdern eine solide Kommunikationsbasis aufzubauen. Es ist dadurch möglich für die Stakeholder mehr über die Anforderungen für die Erweiterung des System nachzudenken und ihre Vorstellungen zu sammeln. Wenn Bedenken bei der Umsetzung entstehen, kann der Projektleiter die Zeit zwischen den Iterationen nutzen, um über die technische Umsetzung der Anforderungen nachzudenken und im Zweifelsfall versuchen Gegenvorschläge anzubringen oder Kompromisse vorzuschlagen.

2.2 Annotationen bei Digital-Humanities

Seit dem Mittelalter werden Annotationen in verschiedenen Kontexten getätigt, zum Beispiel im Bildungswesen, in der Religion oder in der Forschung. Der Begriff Annotation ist seit Mitte des 20. Jahrhunderts wichtiger Bestandteil in Informationssystemen und vereint verschiedene Konzepte, die sich in Art und Ausrichtung unterscheiden. In den Geisteswissenschaften bezeichnet der Begriff zusätzliche Textbeschreibungen und Ergänzungen der Autoren oder Herausgeber. Annotationen sind Werkzeuge der Forschung und bilden die Brücke zwischen Wahrnehmung und Schaffung von Wissen. Es kann so zu einem Objekt eine Übersetzung, Bildbeschreibung oder Erklärung verknüpft werden. Bei Annotationen werden bestimmte Perspektiven zu einem Objekt ausgedrückt, die nicht immer objektiv sein müssen, sondern auch den Standpunkt des Erstellers unterbewusst vermitteln können.
Im Bereich der Informatik hat der Begriff Annotation die Bedeutung Metadaten in Codebereiche einzubinden oder semantisch betrachtet menschenlesbare Inhalte für Maschinen lesbar/ darstellbar zu machen. Die Web Annotation Working Group des World Wide Web Consortiums (W3C) hat als erste versucht, das klassische Konzept einer für Menschen erzeugte Annotation in ein technisches Modell zu wandeln. Dieses Konzept kann im Internet dargestellt werden kann (NS20).
Diese Arbeit profitiert davon, dass die Professorin für „Neuere deutsche Literaturwissenschaften mit dem Schwerpunkt Digital Humanities für Schriftartefakte“ Prof. Dr. Julia Nantke Teil des Dehmel-Digital Projekts ist. Sie hat in einem der ersten Interviews bzw. Meetings für diese Arbeit erklärt, wofür die Annotationen im Bereich der Digital Humanities eingesetzt werden. Dazu gehören unter anderem folgende Bereiche:

- Visuelle Unterstützung zum betrachtenden Objekt.
- Kategorisierung der Objekte.
- Erheben tabellarischer Daten (Label, Ort, Datum).
- Herstellung von Bezug zwischen den Objekten (LGH20).

Unter Betrachtung dieser Bereiche wird versucht, Annotationen in das zu erstellende DHS zu integrieren. Sie sollen sowohl einsehbar, erstellbar und editierbar sein.

2.3 Randbedingungen

Nach den drei Interviewterminen mit den Vertretern/-innen der Uni HH und SUB wurden folgende technische Randbedingungen für den neuen Prototypen des DHS festgelegt:

Anf. Nr.	Technische Randbedingung	Erläuterung
1	System soll als Container betrieben werden	Nach der Fertigstellung der einzelnen Komponenten sollen diese als Containerimage umgewandelt und betrieben werden
2	Aktueller Stand der Entwicklung einsehbar	Damit die Uni HH und SUB stets einen Überblick haben, wie weit die Entwicklung im Projekt ist, wird das System online verfügbar gemacht.
3	Betrachtungstool Mirador	Das Bildbetrachtungstool Mirador soll eingesetzt werden für das Anzeigen der digitalisierten Objekte
4	Annotationen persistieren	Die Annotationen sollen auf einem Server persistent hinterlegt sein
5	Annotationsformat Webannotations	Die Annotationen sollen im standardisiertem Format Webannotation vorliegen

Tabelle 2.2: Übersicht der technischen Randbedingungen

Neben den technischen Randbedingungen wurden für das Projekt folgende organisatorische Randbedingungen definiert.

Organisatorische Randbedingung	Erläuterung
Projektdurchführender	Sebastian Gedigk
Zeitplan	Beginn der Entwicklung Oktober 2021, Übergabe an Herrn Maus spätestens April 2022
Vorgehensmodell	Dokumentation der Application Programming Interface (API) nach OpenApi-Standard
Entwicklungswerkzeuge	Diagramme werden mit Draw.io erstellt. Zur Dokumentation und Versionskontrolle wird GitLab der HAW genutzt.
Konfigurations- und Versionsverwaltung	GitLab der HAW
Veröffentlichung	Prototyp wird über die Informatik Compute Cloud (ICC) der HAW betrieben. Alle Repositories werden an Herrn Maus übergeben.

Tabelle 2.3: Übersicht der organisatorischen Randbedingungen

Die Abgabe des fertigen Prototypen soll im Frühjahr 2022 (bis spätestens April) statt finden. Es ist von der SUB gewünscht, dass das Projekt containerisierbar übergeben wird. Dazu wird dem Ansprechpartner auf Seiten der SUB, David Maus, Zugriff auf die Gitlab Server der HAW gegeben.

2.4 Kontextabgrenzung

Dieser Abschnitt beschreibt das Umfeld des zu entwerfenden DHS, wie es auf unterschiedliche User reagiert und wie es mit Fremdsystemen interagiert.
Das DHS nutzt die „International Image Interoperability Framework (IIIF) Image API“ der SUB, um sich von dort aus die Objekte des Dehmel-Archivs zu laden. Auf die „IIIF Image API“ wird in Kapitel 3 noch explizit eingegangen. Diese Schnittstelle beinhaltet Informationen zu dem Objekt, wie zum Beispiel Titel, Verfasser, erwähnte Personen und Datum der Erstellung. Die Objekte werden vom DHS mit einem Bildbetrachter im Browser dargestellt. Die User haben die Möglichkeit das System über den Browser zu nutzen. Es wird des weiteren gefordert, dass über das Fremdsystem Dehmel-Digital eine Verlinkung zu bestimmten Objekten erfolgen kann.

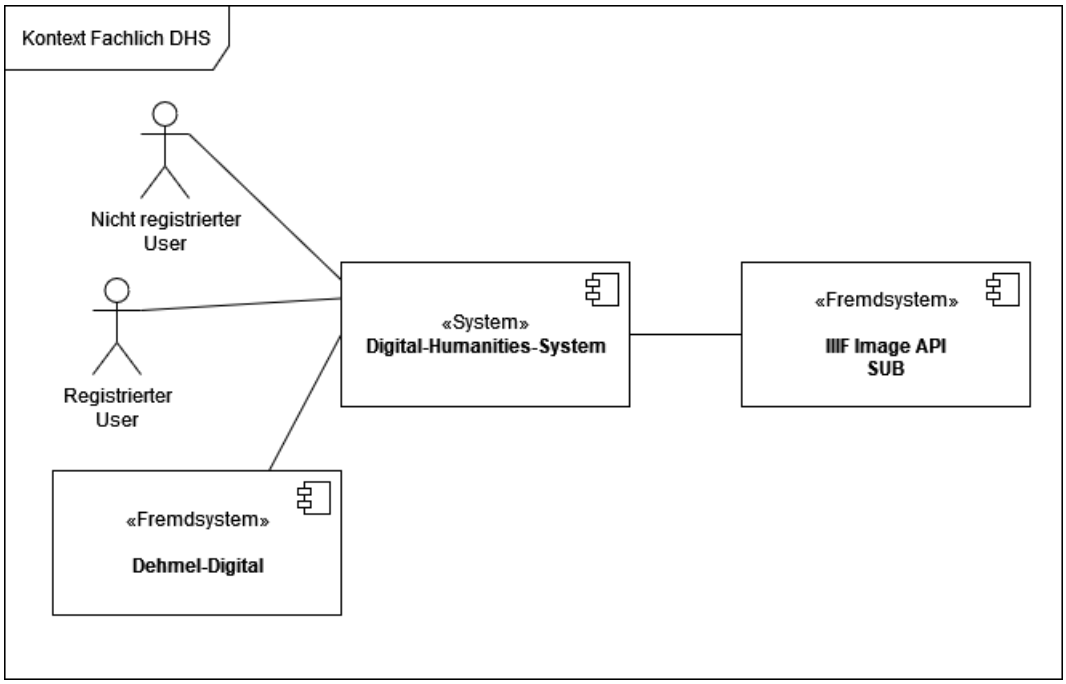

Abbildung 2.1: Fachlicher Kontext

In Abbildung 2.1 sind die Akteure zu sehen, die mit dem DHS interagieren.

Nicht registrierter User: Zu den nicht registrierten User zählt jeder User, der sich nicht beim DHS anmeldet.
Registrierter User: Die registrierten User sind User, die sich auf dem DHS angemeldet haben.
Dehmel-Digital: Dieses Fremdsystem befindet sich aktuell im Aufbau und wird ein Teil des digitalen Auftritts des Dehmel-Archivs. Es beinhaltet viele Informationen zu den Nachlässen von Richard und Ida Dehmel. Dieses System soll die Möglichkeit erhalten, direkt explizite Objekte auf dem DHS aufzurufen
IIIF Image API SUB: Diese Schnittstelle enthält alle Objekte die aus dem Dehmel Archiv digitalisiert wurden.

2.5 Anwendungsfälle

Die neuen Anwendungsfälle des Systems wurden mit den Stakeholdern gemeinsam erarbeitet.

Anf. Nr.	Anwendungsfall	Erläuterung
6	Direkteinstieg in ein bestimmtes Objekt	Es soll die Möglichkeit geben, mit einem bestimmten Objekt direkt die Seite zu öffnen.
7	Annotationen von allen Usern einsehbar	Die Annotationen sollen von registrierten und nicht registrierten Usern einsehbar sein.
8	Annotationen sollen moderierbar sein	Es soll eine Möglichkeit geben, die Annotationen zu bearbeiten/ zu löschen.
9	Annotationen mit Presets	Es soll die Möglichkeit geben für Annotationen Voreinstellungen zu nutzen bezüglich Form und Farbe.
10	Annotationen sollen exportierbar sein	Die Annotationen sollen vom Server direkt abfragbar sein.
11	Userverwaltung erstellen	Es soll möglich sein User anzulegen und zu löschen.
12	Registrierung notwendig für das Annotieren	Es dürfen nur registrierte User Annotationen erstellen oder modifizieren.

Tabelle 2.4: Übersicht der Anwendungsfälle

Die Anwendungsfälle werden in einem Anwendungsfalldiagramm dargestellt. Das Fremdsystem Dehmel-Digital verhält sich beim benutzen des DHS grundsätzlich wie ein nicht registrierter User. Es gibt daher kein Anwendungsfalldiagramm für dieses Fremdsystem. Es ist lediglich im Kontextdiagramm vorhanden, da es das DHS per direkt Link zu einem bestimmten Objekt aufrufen kann.
Damit es übersichtlich in den Diagrammen bleibt, werden zwei separate Anwendungsfalldiagramme erstellt. In den Anwendungsfalldiagrammen werden die neuen Features, die Bestandteil dieser Arbeit sind und implementiert werden sollen, farblich in grau hervorgehoben. Die alten Features sind aus der Projektarbeit aus dem Sommersemester 2021 in weiß gehalten. Die Features aus der Projektarbeit wurden aus beiden Teams genutzt. Dazu zählt das Anzeigen der Objekte auf der Landingpage und die Möglichkeit sich anzumelden.

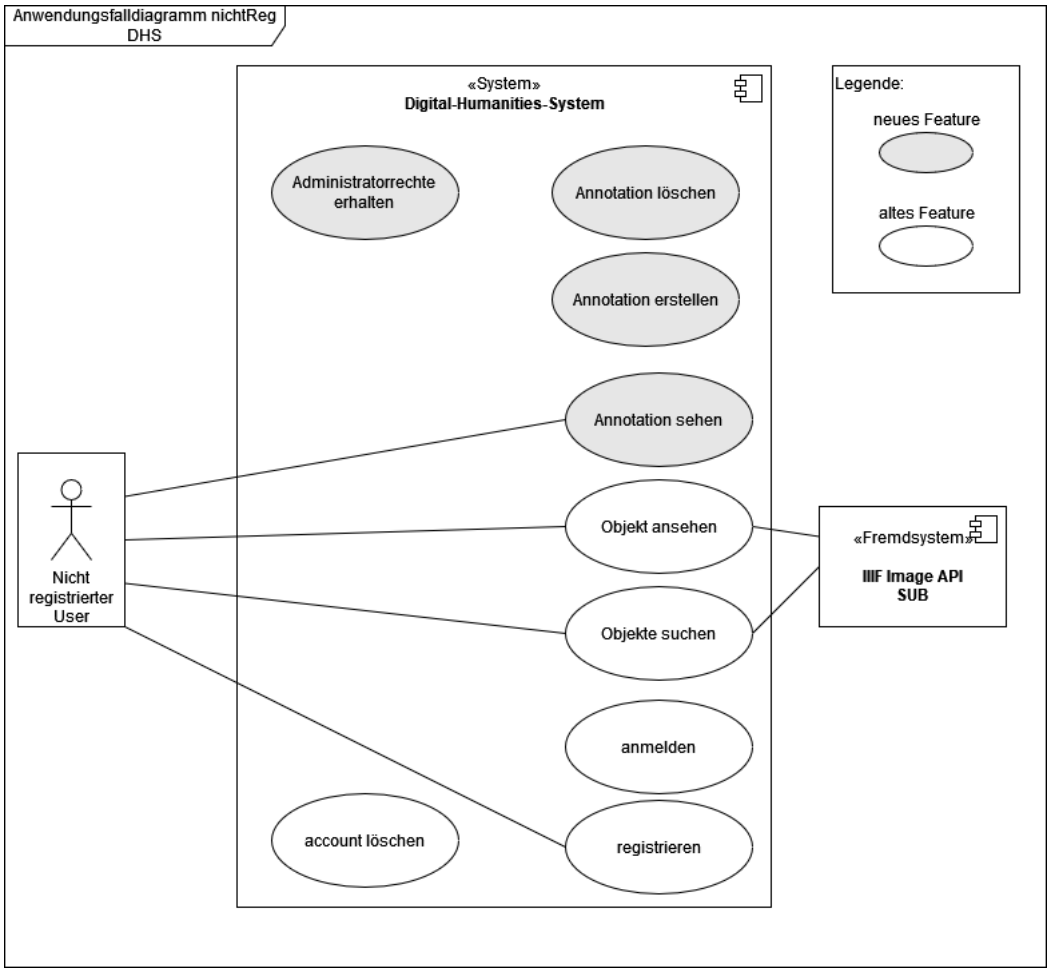

Abbildung 2.2: Anwendungsfalldiagramm nicht registrierter User

Abbildung 2.2 zeigt das DHS und die Anwendungsfälle von nicht registrierten Usern und dem Fremdsystem IIIF Image API SUB.

Nicht registrierte User: Sind alle User, die den Login-Prozess nicht durchlaufen haben. Es ist ihnen möglich die Objekte zu betrachten und zu suchen. Sie haben lesenden Zugriff auf abgespeicherte Annotationen. Die User können sich registrieren. Damit ist es ihnen möglich in den Zustand des registrierten Users zu wechseln.

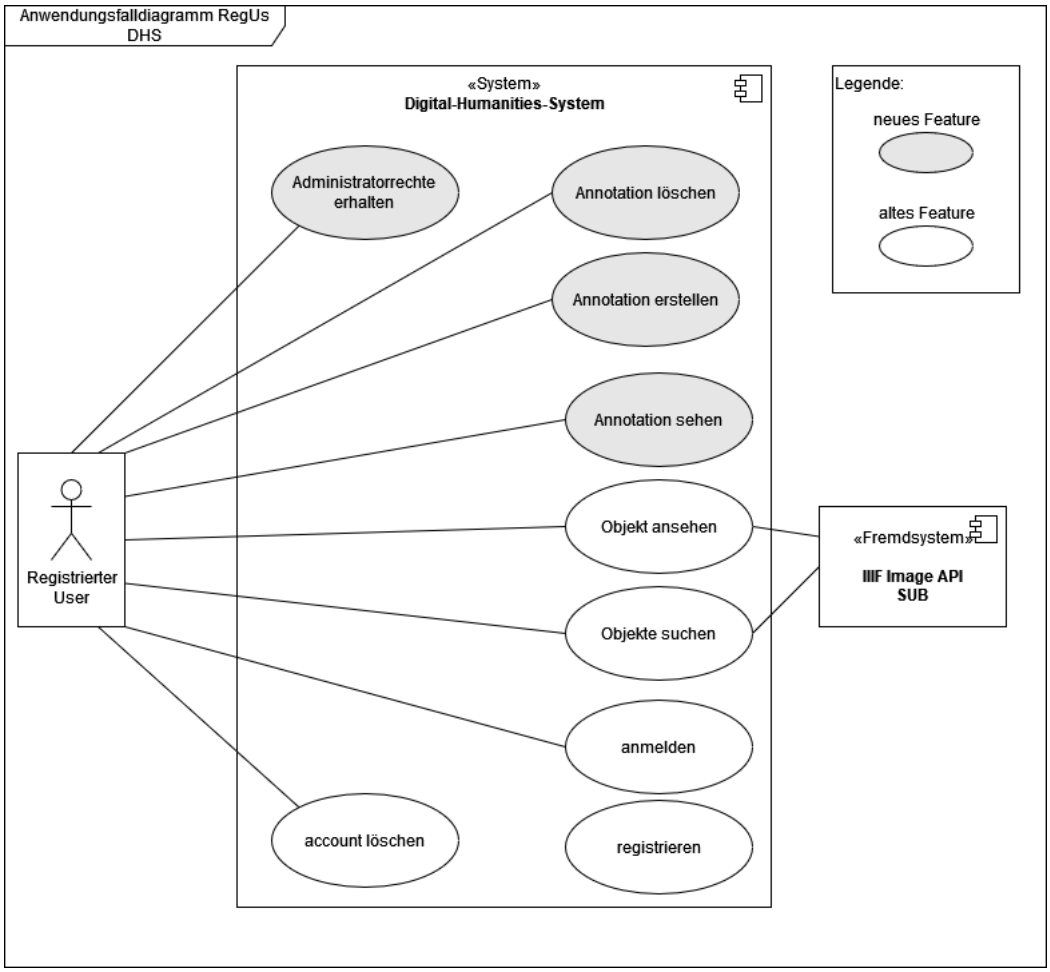

Abbildung 2.3: Anwendungsfalldiagramm registrierter User

Abbildung 2.3 zeigt das DHS und die Anwendungsfälle von registrierten Usern und dem Fremdsystem IIIF Image API SUB.

Registrierte User: Sind alle User die erfolgreich den Login-Prozess durchlaufen haben. Sie übernehmen alle Rechte der nicht registrierten User. Des weiteren können sie Annotationen erstellen und löschen. Registrierte User können Administratorrechte erhalten. Damit sind sie in der Lage andere User zu löschen.

3 Entwurf und Design

Dieses Kapitel beschreibt, wie anhand der heraus gearbeiteten Anforderungen, das System aufgebaut werden soll. Es wird dargestellt aus welchen Teilen das System bestehen soll und dient als Vorlage für die Realisierung des Systems.

3.1 Architektur

Die Architektur eines Softwaresystems beschreibt den Aufbau des Systems. Es wird beschrieben, welche Komponenten verwendet werden und wie einzelne Komponenten miteinander agieren.
In diesem Projekt wird eine hierarchische Architektur verwendet. Das bedeutet, dass einzelne Elemente in einer hierarchischen Struktur untereinander angeordnet sind. Jedes untergeordnete Element bietet seinem übergeordnetem Element Dienste an.
Dazu wird das System in mehrere Schichten mit unterschiedlichen Verantwortlichkeiten aufgeteilt. Jede Schicht kapselt seine Implementierung von der Außenwelt ab und hat definierte Schnittstellen. Durch die Schnittstellen können andere Elemente aus der darüberliegenden Schicht darauf zugreifen. Die Datenabfrage darf nur in eine Richtung erfolgen, von oben nach unten. Das heißt, dass die Daten von oben nach unten abgefragt werden und von unten nach oben weiter gereicht werden können, jedoch die untere Schicht nicht Daten der oberen Schicht abfragen darf.Zirkuläre Abhängigkeiten gilt es stets zu vermeiden, damit das System nicht in eine Abhängigkeitsschleife gerät (Sta20).
Dieser Architekturansatz hat folgende Vor- und Nachteile:

Vorteil	Nachteil
Implementierungen einer Schicht können beliebig ausgetauscht werden, solange man die gleichen Dienste anbietet.	Die Performance des Systems kann durch das durchlaufen einzelner Schichten beeinträchtigt werden.
Das System kann horizontal einfacher skaliert werden.	Die Netzwerklast bei vielen Dienstaufrufen zwischen allen Schicht kann erhöht sein.
Das Konzept der Schichten ist leicht verständlich und gibt einen überschaubaren Überblick über den Aufbau des Systems	Die Entwicklung und Wartbarkeit des Systems kann bei vielen Abhängigkeiten zwischen den Schichten komplex werden

Tabelle 3.1: Vor- und Nachteile dieses Architekturansatzes

Die Abbildung 3.1 zeigt den Aufbau der hierarchischen Architektur in diesem DHS. Es wird drei Schichten geben, und damit wird die klassische „Three Tier Architecture“ eingesetzt (Sta20, Seite 112).

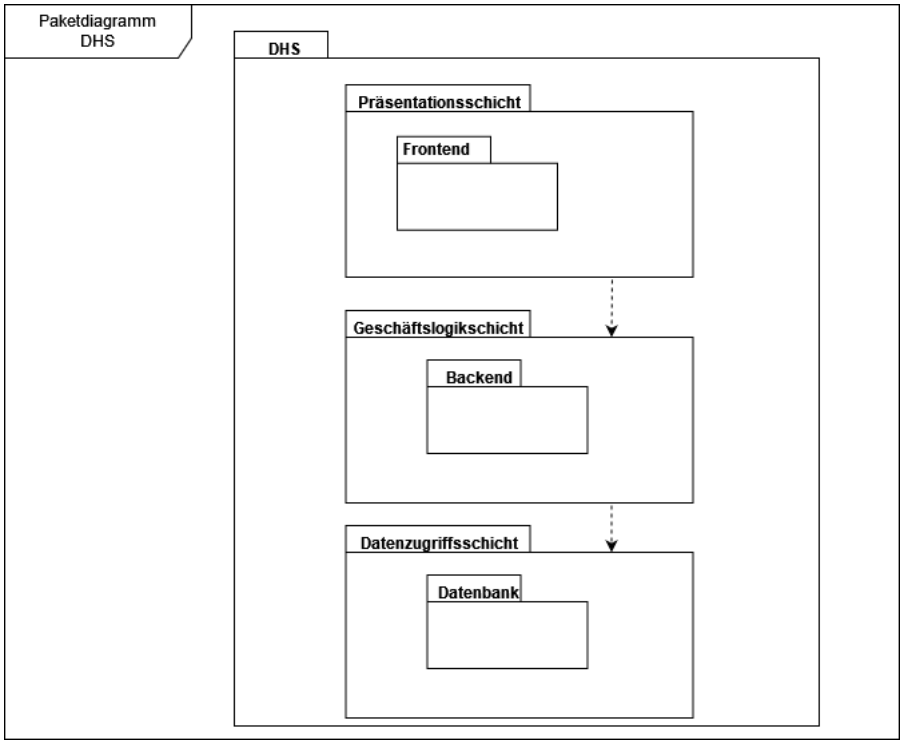

Abbildung 3.1: Paketdiagramm des DHS

Jede Schicht bedient mit einem Dienst die über ihr liegende Schicht.
Diese Architektur wird gewählt, da bereits das vorangegangene Projekt in diesem Stil aufgebaut wurde. Es ist praktikabel und ein modularer Austausch einzelner Komponenten kann gewährleistet werden. Bei der Implementation des neuen Prototypen für dieses Projekt kann man sich an einem bereits bestehenden Prototypen orientieren und so Zeit bei der Planung der Architektur einsparen.

Präsentationsschicht: Stellt die angeforderten Daten für den User in ansehnlicher Form dar. Sie dient ebenso zur Bedienung und Bereitstellung der Funktionen des Systems.

Geschäftslogikschicht: Diese Schicht wird auch Anwendungslogikschicht genannt. Sie dient der Realisierung des eigentlichen Geschäftsmodell, regelt Zugriffsbeschränkungen und leitet die Daten zwischen der Datenzugriffsschicht und der Präsentationsschicht weiter.

Datenzugriffsschicht: Diese Schicht kapselt den Zugriff auf die Daten. Die Daten werden an diesem Ort durch eine Datenbank persistent gehalten. Sie bietet Schnittstellen zum Zugriff auf die Daten für die Geschäftslogikschicht an.

Bei der Implementation dieser Architektur kann auf bereits existierendes Wissen in der „Three Tier Architecture“ aufgebaut werden. Diese Art der Architektur hat den Vorteil, dass man einfach einzelne Komponenten austauschen und updaten kann. Es ist kein monolithisches System, bei dem die gesamte Software bei der Aufdeckung von Sicherheitslücken angepasst werden muss. Wenn es ein Problem gibt, kann direkt auf die einzelne Komponente eingewirkt werden und so das restliche System weiter funktionieren.

3.2 Design

Dieser Abschnitt beschäftigt sich mit dem Design des DHS. Ein gutes Design soll ansehnlich sein und eine einfache, intuitive Bedienung des Systems ermöglichen.
Da das System in der Forschung an der SUB und Uni HH eingesetzt wird, wurde sich darauf geeinigt, dass der Prototyp eine Webanwendung sein wird. Diese soll als Webseite betrieben werden. Dazu wurde bereits im vorhergehenden Projekt eine Single Page Application (SPA) entwickelt.
Moderne Webentwicklung setzt auf den SPA Ansatz (JSD15). Single Page Application bedeutet

die Webseite wird einmal geladen und bleibt dann in ihrer Grundform gleich. Wenn ein User mit der Seite interagiert, werden nur einzelne Komponenten erneuert. Das spart Bandbreite und Zeit.
Aktuelle Statistiken zeigen, dass über 90% der User weltweit mit ihrem Mobiltelefon im Internet unterwegs sind, und noch immerhin 70% der User mit einem Laptop oder Desktop-PC im Internet surfen (Sta21). Eine SPA ist daher eine gute Wahl für eine Webanwendung, da sie sowohl von mobilen Endgeräten als auch von einem Desktop-PC gut bedient werden kann. Die Anwendung skaliert die Erscheinung dynamisch an die Größe der Auflösung des Endgerätes. In diesem Projekt werden die User hauptsächlich mit einem Tablet-PC oder PC / Laptop die Webanwendung nutzen. Es wird daher vom Design keine Anpassung an Smartphones im Vordergrund stehen. Die Webseite soll in drei Teile aufgeteilt werden.

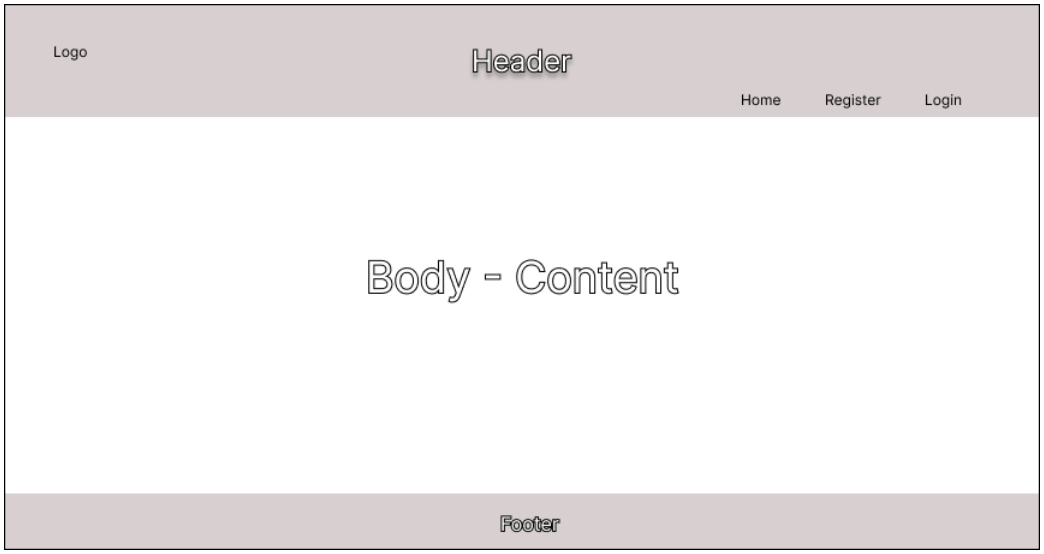

Abbildung 3.2: Entwurf für das Design der Webanwendung

Die Abbildung 3.2 zeigt das geplante Design der Webanwendung und wo Teile des Menüs platziert werden sollen.

Header: Der Header ist der obere Bereich der Anwendung. Dieser Bereich behält seine Größe bei, passt sich jedoch inhaltlich an. Wenn eine User eingeloggt ist, sieht es anders aus als bei einem Gast-User. Der linke Teil des Headers ist reserviert für die Logos der SUB und Uni HH. Der rechte Teil des Headers wird als Menübereich benutzt. Dort kann man sich entweder zur

Startseite bewegen oder registrieren bzw. einloggen.

Body: Der Body nimmt die größte Fläche der Anwendung ein. Dort wird der Inhalt angezeigt den man sehen möchte. Es können dort zum Beispiel die Objekte, das Registrierungsformular oder das Loginformular angezeigt werden. Dieser Teil der Anwendung ist sehr dynamisch.

Footer: Am unteren Teil der Anwendung ist der Footer. Dort werden Verlinkungen zum Impressum und zu anderen organisatorisch relevanten Seiten platziert. Dieser Teil der Anwendung bleibt immer gleich.

3.3 International Image Interoperability Framework

In diesem Bereich wird das IIIF Konsortium und das IIIF Framework kurz erläutert.
Es ist Teil dieses Projekts. Mit Anwendung dieses Standards wurden die Objekte des Dehmel-Archivs digitalisiert und in diesem IIIF Format abgespeichert. Die Objekte werden mit der IIIF Image API dargestellt. Daher muss das zu entwickelnde DHS so aufgebaut werden, dass es diese Schnittstelle nutzen und die Objekte darstellen kann.
Das IIIF Konsortium besteht aus Forschungsbibliotheken, Staatsbibliotheken, Archiven, Software Firmen und digitalen Agenturen mit gemeinsamen Zielen wie z.B.:

- Zur Verfügung stellen von digitalisierten Objekten auf der ganzen Welt.
- Eine gemeinsame Schnittstelle definieren und betreiben, um Interoperabilität zwischen den Teilnehmern herzustellen.
- Herstellung einer Technologie, die den Umgang mit den Objekten erleichtert.

Das IIIF Konsortium umfasst aktuell 60 Mitglieder, unter anderem die Stanford University, Yale, Cornell University und die Bayerische Staatsbibliothek.
International Image Interoperability Framework ist ein Standard um Bilder oder audio/visuelle Daten von einem Fileserver für das Internet bereit zu stellen. Dieser Standard schafft es, dass Browser in der Lage sind, mehr als einfache Bild und Audio/Video Formate darzustellen. Die IIIF Spezifikation kann mit den gängigen Webstandards kombiniert, die Funktionalitäten dieser Dateien erweitern. Bilder können einen tiefen Zoom erhalten, verglichen werden, Strukturen analysiert werden oder Annotationen erhalten. Videos können zum Beispiel mit Übersetzungen oder Annotationen beim abspielen versehen werden. Dateien können so in einem einheitlichen Stil repräsentiert werden und stehen plattformunabhängig zur Darstellung in unterschiedlichen

Betrachtungstools zur Verfügung (Con22).
Es gibt zwei Kernschnittstellen bei den IIIF API´s die IIIF Image API und die IIIF Presentation API. In diesem Projekt wird die IIIF Image API genutzt. Diese Schnittstelle gibt ein Objekt im IIIF Standard zurück. Das Objekt kann in diesem Format Meta-Daten wie zum Beispiel: Autor, Jahr der Erscheinung, Ort oder auch mehrere weitere Objekte enthalten. So kann zum Beispiel eine Postkarte mit Vorder- und Rückseite zusammenhängend übertragen werden.

4 Realisierung und Integration

In diesem Kapitel wird erklärt, wie das System realisiert wird. Es soll dargestellt werden wie die einzelnen Komponenten des Systems miteinander kommunizieren und interagieren. Die eingesetzten Technologien werden erklärt und es wird begründet, warum man diese ausgewählt hat.

4.1 Technische Umsetzung

Das DDHS wird in Anlehnung an den bestehenden Prototypen aus dem vorherigen Projekt entwickelt. Beide Systeme nutzen die gleichen Technologien, auf die in den folgenden Unterkapiteln eingegangen wird. Es wird, wie in Kapitel 3 erklärt, die „Three Tier Architecture“ eingesetzt. Diese drei Komponenten sind:

- Das **Frontend**, welches für die Darstellung im Browser zuständig ist.
- Das **Backend**, welches die Logikkomponente darstellt in Form von Userverwaltung und Annotationsverwaltung.
- Die **Datenbank**, welche dafür zuständig ist, dass die Daten persistent abgelegt werden.

Die digitalisierten Objekte werden von der SUB über eine IIIF Image API bereitgestellt. Diese Schnittstelle beinhaltet aktuell etwa 4000 Objekte mit ihren zugehörigen Metadaten.
Sämtlicher produzierter Quellcode und die Dokumentation der einzelnen Komponenten wird in der Git-Lab Instanz der HAW abgelegt. Damit wird ein zentraler Ablageort geschaffen. Es wird eine Projektgruppe erstellt, in der die einzelnen Komponenten Repositories erhielten.

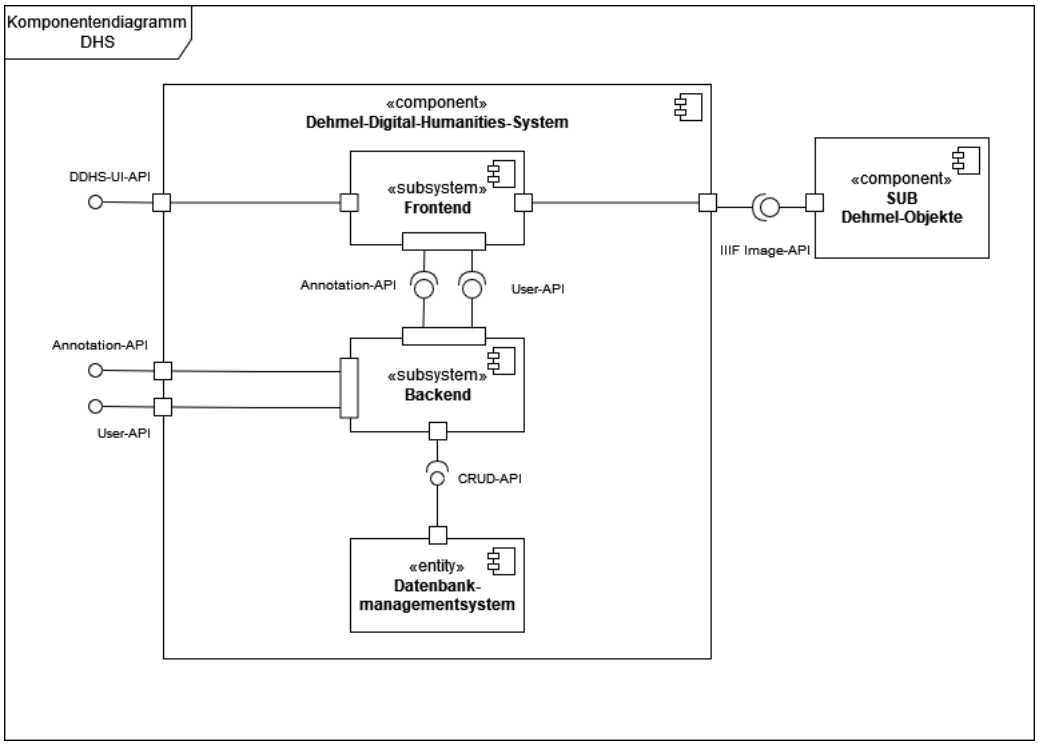

Abbildung 4.1: Komponentendiagramm des DHS

Das Komponentendiagramm aus Abbildung 4.1 zeigt den geplanten Aufbau des DDHS, welche Schnittstellen nach außen angeboten werden und mit welchen Schnittstellen das System kommunizieren wird.
Das System bezieht die Übersicht der Objekte und jedes einzelne zu betrachtenden Objekt aus der IIIF Image API der SUB. Die Übersicht bzw. die Objekte werden dann vom Frontend dargestellt. Die gesamte Webseite wird im Browser durch die DDHS-UI-API dargestellt.
Das Backend hat in diesem System zwei Funktionen. Es soll erstens eine Userverwaltung ermöglichen und zweitens die Annotationen in geeigneter Form weiterleiten. Das Frontend fragt die Annotationen für die einzelnen Objekte mittels Annotation-API vom Backend ab. Das Backend erhält die Annotationen vom Datenbankmanagementsystem (DBMS) und leitet diese dann ans Frontend. Die Userverwaltung wird vom Backend durchgeführt. Dazu werden alle nutzerspezifischen Anfragen vom Backend in Verbindung mit dem DBMS geprüft und verarbeitet. Das Ergebnis dieser Anfragen wir dann wieder an das Frontend geschickt. Die beiden API´s im Backend sind zudem von außen erreichbar. Damit ist es möglich die Annotationen und User separat zu verwalten. Die Annotationen können so gesammelt oder einzeln abgerufen werden. Das war eine Vorgabe der SUB aus der Anforderungsermittlung.

Die Datenbank ist der Speicherort für die Annotationen und Daten der User. Dort wird zu jedem Objekt ein Dokument mit allen dazugehörigen Annotationen abgelegt und zu den User ein Eintrag in der Datenbank erstellt.
Die User haben nach diesem Aufbau zwei Möglichkeiten auf Daten zuzugreifen. Zum einen über die Webseite und zum anderen über die Angebotenen API´s des Backends.

4.2 Frontend

Das Wort Frontend bezeichnet die öffentlich zugänglichen Internetseiten der Anwendung.
Das Frontend ist in diesem Projekt auf zwei Technologien aufgeteilt. Es besteht aus der Webseite, die mit dem Javascript Framework Angular angelegt wird, und dem Frontendserver, welcher auf Basis von Nginx läuft. Der Quellcode des Frontend ist auf dem Gitlabserver der HAW abgelegt (Ged22c).
JavaScript bietet die Möglichkeit Webseiten interaktiv und benutzerfreundlich zu gestalten. Wie schon im Kapitel 3 erwähnt, wird die Webseite als SPA entwickelt. Es wird einen Header- und Footerbereich geben und der Inhalt wird in der Mitte der Seite platziert. Damit man sich nicht mit der Umsetzung der Navigation und Reaktion auf Benutzereingaben beschäftigen muss, gibt es bereits eine Reihe an Frameworks, die das für die EntwicklerInnen erledigen. Das hier eingesetzte Framework heißt Angular. Es löst das Problem der Navigation und der Benutzereingaben, sodass sich die EntwicklerInnen auf die Umsetzung des Designs und der Funktionalitäten konzentrieren können. Angular ist ein JavaScript-Framework explizit zur Erstellung von SPAs. Dieses Framework wurde 2009 von Miško Hevery, einem Googlemitarbeiter, entwickelt (Wei16).
Es gibt aktuell drei Javascript-Frameworks, die häufig für SPAs eingesetzt werden. Das sind Angular, React und Vue.js. In Abbildung 4.2 wird die Häufigkeit der weltweiten Suchanfragen der letzten fünf Jahre zu den drei Frameworks dargestellt. Die Werte auf der Y-Achse sind relative Zahlen, bezogen auf das Suchinteresse im festgelegten Zeitraum. Es ist zu sehen, dass Angular den Trend seit 2018 nicht mehr anführt, sich dafür aber im soliden mittleren Bereich aufhält. Angular ist seit 2010 auf dem Markt, React und Vue.js seit 2014. Da das vorangegangene Projekt mit Angular aufgesetzt wurde, wird es in diesem Projekt ebenso genutzt. Das spart Programmieraufwand und erleichtert den Einstieg in den Webseitenaufbau, da bereits vom Entwickler erste Kenntnisse im Umgang mit Angular vorhanden sind.

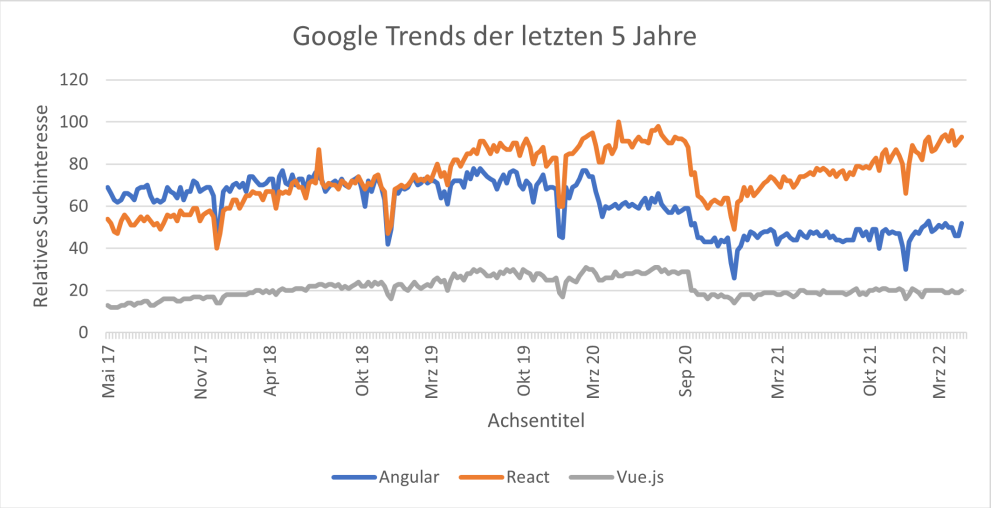

Abbildung 4.2: Google Trends der letzten 5 Jahre zum Thema Angular, React und Vue.js (Goo22)

Angular trennt die Logik von der Darstellung. Jeder Teil der Webseite wird als eigenständige Komponente im Quellcode erstellt. Dazu nutzt man die Boardmittel von Angular. Eine Komponente ist in drei Bereiche gegliedert:

- Der Stylebereich in der .css Datei. Dort werden für diese Komponente die einzelnen Styles festgelegt. Das bedeutet zum Beispiel an welchem Ort sitzt welches Bild oder wie sehen Überschriften und der Text aus.
- Der HTML- Bereich. Dort wird der Aufbau der Komponente erstellt. Das bedeutet, ob die Komponente in einem Grid aufgeteilt ist oder ob Bilder in der Komponente vorhanden sind. In diesem Bereich können Formulare genutzt werden. Zum Beispiel kann eine Komponente den Userlogin regeln. Dann werden dort die Anmeldedaten der User entgegengenommen.
- Der Logikbereich. An dieser Stelle wird die Geschäftslogik für die Komponente geschrieben. Die Daten aus dem HTML-Bereich werden hier verarbeitet. Es können zum Beispiel die Formulardaten des Userlogins entgegengenommen werden und an die Authentifizierungstelle geschickt werden. Wenn die Anmeldung erfolgreich war, können weiterführende Skripte ausgeführt werden oder im HTML-Bereich angezeigt werden. So ist ersichtlich, dass die Anmeldung geklappt hat. Ebenso besteht die Möglichkeit Fehler weiterzuleiten an den HTML-Bereich.

Die Logik in Angularkomponenten wird in Typescript geschrieben. Diese Sprache ist eine Obermenge von JavaScript.
Damit das Frontend in diesem Projekt eigenständig in einer Containerumgebung laufen kann, benötigt man einen Webserver. Dazu wird NGINX genutzt.

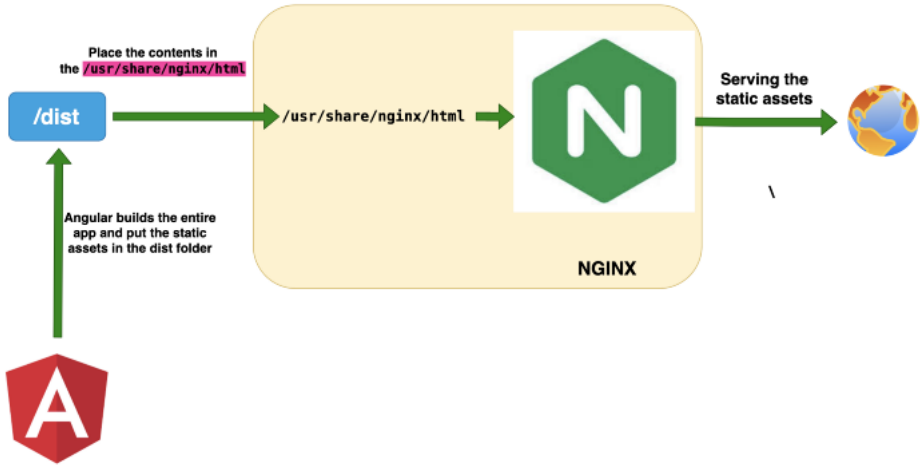

Abbildung 4.3: Angular und das Zusammenspiel mit NGINX (Bac19)

Die Abbildung 4.3 ist von einem Medium Artikel (Bac19) und zeigt, wie Angular und NGINX miteinander kombinert werden um die Webseite auszuliefern. Die SPA, die mit Angular gebaut wird, wird in den HTML Ordner Webserver von NGINX kopiert. Der Webserver stellt dann die Inhalte aus dem HTML-Ordner dar. Dieser Verbund wird als ein Containerimage erstellt und, wie später im Unterkapitel Deployment erklärt, ausgeliefert. Grundsätzlich wäre die Webseite nur über eine einzige URL erreichbar. In diesem Fall heißt die Projekt URL „https://dehmel-digitalhumanities.informatik.haw-hamburg.de/“. Damit man auf einzelne „Seiten“ der Webseite zugreifen kann, muss die Konfiguration des Webservers angepasst werden. So können alle Aufrufe zu der bestimmten Komponente der Anwendung weitergeleitet und Argumente in der URL übergeben werden.
Die digitaliserten Objekte der SUB werden in der Mitte der Webseite dargestellt. Dazu wird das Bildbetrachtungstool Mirador genutzt. Das ist eine Opensource Bildbetrachtungsplattform. Diese Plattform wurde auf Wunsch der SUB bereits im vorangegangenen Projekt genutzt. Mirador kann Objekte im IIIF Format darstellen. Man kann damit Objekte zoomen, drehen, vergleichen und Meta Informationen darstellen.
Es ist Modular aufgebaut und kann mit verschiedenen Plugins genutzt werden. Die Community

von Mirador versorgt diese Plattform mit neuen Features und Plugins. In diesem Projekt wurden folgende Plugins zum Mirador-Viewer hinzugefügt:

- **Imagetool Plugin:** Damit kann man die Darstellung des Objekts bearbeiten. Es werden damit die Helligkeit, Sättigung, der Kontrast und die Farbe verändert. Ebenfalss kann man das Objekt spiegeln und die Ausrichtung ändern. Diese Änderungen sind nur lokal und nicht in der originalen Datei des Objekts.
- **Annotation Plugin:** Dieses Plugin ermöglicht es für jedes einzelne Objekt Annotationen zu erstellen. Die Anpassungen werden im Unterkapitel Annotationsplugin erläutert.

Mirador wird regelmäßig geupdated und ist aktuell in der Version 3 verfügbar. Im Bereich der Digital-Humanities wird es unter anderem von Oxford, Stanford und einem Zusammenschluß mehrerer deutscher Forschungsgemeinschaften genutzt. Die Forschungsgemeinschaft besteht aus der Staatsbibliothek zu Berlin, der Unversitätsbibliothek Leipzig, der Herzog August Bibliothek und der Bayerischen Staatsbibliothek. Sie nutzen Mirador in einem entstehenden Handschriftenportal.

4.3 Annotation Plugin

Das Annotation Plugin fügt dem Mirador-Viewer die Option hinzu, Annotationen zu sehen, zu erstellen oder zu editieren. Es ist mit dem Plugin möglich bestimmte Positionen auf einem Objekt zu markieren und mit einer Annotation zu versehen. Dazu gibt es in dem Plugin unterschiedliche Möglichkeiten.

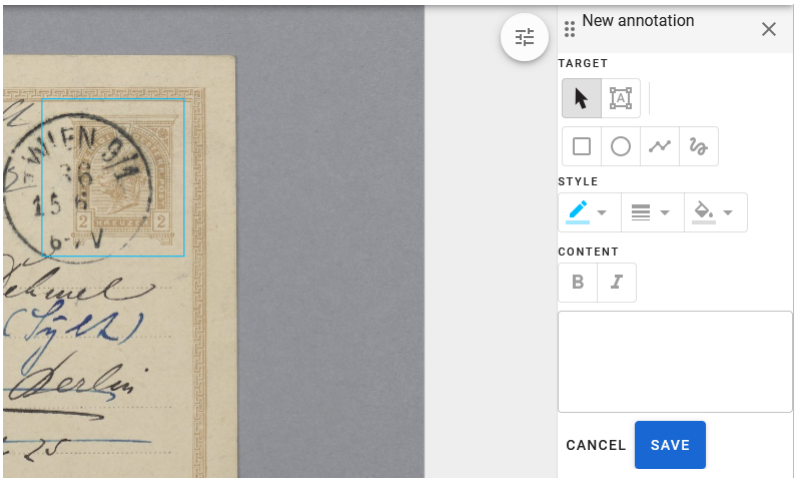

Abbildung 4.4: Ausschnitt aus der Webseite zum Erstellen einer neuen Annotation

In Abbildung 4.4 wird dargestellt, wie das ursprüngliche Plugin aussieht. Die Abbildung zeigt das Erstellen einer neuen Annotation. Es gibt dazu mehrere Optionen wie man markieren möchte. Entweder nutzt man einen Kreis, ein Rechteck, verkettete gerade Linien, den Freihandstil oder keine Markierung. Dazu kann man die Linienstärke, Farbe und Füllung auswählen. Zuletzt kann man den Text, den man für die Annotation nutzen möchte, fett oder kursiv darstellen.
Das Plugin bietet die Möglichkeit Annotationen lokal im Browser abzulegen. Diese Option bedeutet, wenn man seinen Browserspeicher löscht, verliert man alle Annotationen. Die Annotationen sind ebenfalls nur an einem einzigen Ort sichtbar. Des weiteren bietet das Plugin die Möglichkeit, einen Server zu nutzen, um die Plugins dort hinzuschicken. Diese Option ist in dem Plugin aber nicht so integriert, wie es für dieses Projekt nötig ist. Bei diesem Projekt werden zur Authentifizierung der User JWT-Token verwendet. Wie dies funktioniert und was das ist, wird in dem Unterkapitel Backend erklärt. Diese Token können mit dem ursprünglichen Plugin nicht in einer geeigneten Art und Weise erneuert werden. Ebenso sind die Abfragen der Annotationen bzw. das Übermitteln der Annotationen nicht so gelöst wie es eigentlich praktisch wäre. Deshalb wird das Plugin um einen Adapter für einen eigenen Server erweitert. So hat man die Möglichkeit direkt Änderungen am Adapter einzufügen und den passend zum Backend zu gestalten. Der Wunsch der Uni HH und SUB, war es Voreinstellungen für bestimmte Annotationen treffen zu können. Diese werden Presets im weiteren Verlauf der Arbeit genannt. Die Presets umfassen zum Beispiel das Preset Briefmarke. Es enthält bereits den fett geschriebenen Namen Briefmarke, eine rechteckige Markierung und die Farbe blau. Folgende Presets wurden im Plugin als statische Liste integriert:

- **Name:** Briefmarke **Form:** Rechteck **Farbe:** dunkel blau
- **Name:** Poststempel **Form:** Kreis **Farbe:** türkis
- **Name:** Person **Form:** Rechteck **Farbe:** hellgrün
- **Name:** Bemerkung **Form:** Freihand **Farbe:** schwarz

Abbildung 4.5: Ausschnitt aus der Webseite mit 3 Beispiel-Annotationen nach der Anpassung

Die Abbildung 4.5 zeigt ein Objekt mit Annotationen aus den vorhanden Presets. Die Annotationstexte werden auf der linken Seite dargestellt und die Markierungen sind im Objekt zu sehen. Die Annotationen können einzeln im Objekt angezeigt werden oder man kann sich alle Annotationen anzeigen lassen.
Das Annotation Plugin wird an dieses Projekt angepasst. Dazu wird das ursprüngliche Repository in eigenes GitHub Repository (Ged22d) geforked und ein neuer Branch namens addOwnServer erstellt. Das bedeutet in ein eigenes Repository abgezweigt, um eigenständig daran zu arbeiten. Der Vorteil an dieser Methode ist, dass man so dieses Plugin auf seine Anwendung anpassen und weiterentwickeln kann. Man kann das Plugin dann auf die NPM Plattform uploaden und in seinem Projekt einsetzen (Ged22e).

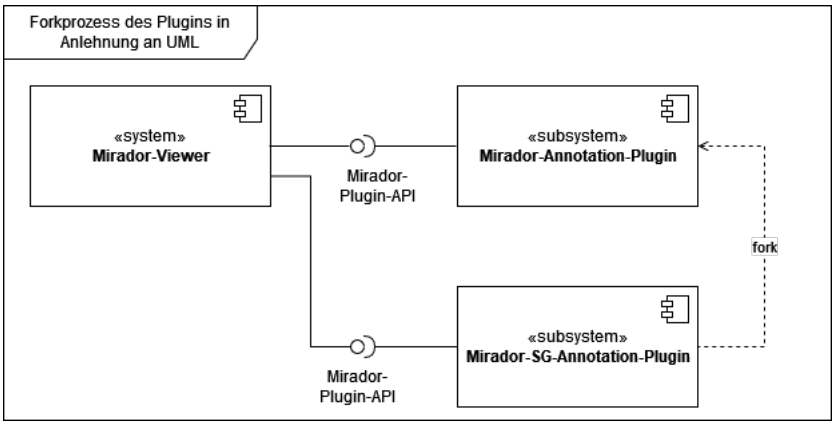

Abbildung 4.6: Forkprozess des ursprünglichen Repositories

Die Abbildung 4.6 soll verdeutlichen, dass das Mirador-SG-Annotation-Plugin vom ursprünglichen Mirador-Annotation-Plugin geforked wird. Beide Plugins nutzen die Plugin-API des Mirador-Viewers. Das angepasste Plugin kann mit dem Mirador-Viewer genauso zusammenarbeiten wie das Original. Es wird exakt so in den Viewer eingebunden wie das Original. Die sichtbare Änderung des Plugins ist die Annotationserstellungs- und Bearbeitungsseite. Die Art der Kommunikation mit dem Server ist der nicht sichtbare Anteil der Änderung.
Die Annotationen werden im W3C Webannotation Format (W3C17) erstellt und auf dem Server abgelegt. Dieses Format ermöglicht einen einfachen Austausch zwischen unterschiedlichen Bildbetrachtungstools. Es ist ein Hardware bzw. plattformunabhängiges Format, welches Annotationen nach einem bestimmten Aufbau beschreibt. Das Webannotation Format wird laut Spezifikation als JavaScript Object Notation (JSON) abgespeichert. Dateien im JSON Format dienen dem Austausch von Informationen zwischen zwei Anwendungen. Man kann Dateien in diesem Format einfach lesen und verstehen.

4.4 Backend

Das Backend bezeichnet den Teil einer Software, der im Hintergrund agiert. Es führt die Datenverarbeitung und Speicherung aus. Dazu können Daten lokal abgelegt oder an eine Datenbank weitergeleitet werden. In diesem Projekt werden die Daten an eine Datenbank weitergeleitet. Das Frontend fragt die Daten beim Backend an und das Backend leitet die Daten aus der Datenbank ans Frontend weiter. Der Quellcode des Backends ist auf dem Gitlabserver der HAW abgelegt (Ged22b).
In dieser Arbeit wird sich für das Backend-Framework Spring entschieden. Dieses Framework ist in der Programmiersprache JAVA geschrieben und wurde bereits im vorangegangenen Projekt genutzt. Daher ist der Einstieg in die Entwicklung leichter. Es ist grundsätzlich möglich, andere Frameworks für die Umsetzung des DDHS zu nutzen, solange diese mit einer Datenbank kommunizieren können. Die Schnittstellen des Backends werden als Representational State Transfer (REST)-API zur Verfügung gestellt. REST-Schnittstellen sind zustandslos. Das bedeutet, dass es irrelevant ist, welche Anfragen bereits gestellt wurden und jede Anfrage in sich geschlossen zu betrachten ist. Dadurch muss keine Sitzung zwischen Client und Server gespeichert werden.
Es gibt weitere Spring Unterprojekte mit denen man die Funtkionalitäten und die Sicherheit des Backends leichter verbessern kann (Pra17). Folgende Spring Unterprojekte wurden in diesem Projekt eingesetzt:

- **Spring-Data** Dient dazu Datenmodelle in Datenbanken zu erstellen und auf Daten zuzugreifen.
- **Spring-Security** Dient der Absicherung der Anwendung. In diesem Projekt vor allem zum geregelten Zugriff auf Daten und deren Bearbeitung.

Das Spring-Framework hat die Möglichkeit, andere Java-Frameworks zu integrieren. Dadurch hat man die Möglichkeit auf bereits bestehende Bibliotheken zuzugreifen und man kann sich zum Beispiel JSON-Dateien oder Authorisierungstoken generieren lassen. Des weiteren wurden folgenden zusätzlichen Bibliotheken eingesetzt:

- **Java-JWT** Das ist eine Bibliothek, die es ermöglicht mit Java einen JSON Web Token (JWT) zu generieren und zu decodieren. Ein JWT ist nach dem Standard RFC 7519 (JBS15) ein Authorisierungsschlüssel.
- **JSON-Simple** Das ist ein Softwarewerkzeug, dass es ermöglicht, in Java JSON Dateien zu lesen und zu erstellen
- **Lombok** Das ist eine Bibliothek, die es ermöglicht beim entwickeln sich wiederholenden Code zu sparen. Lombok ermöglicht es, durch Annotationen im Quellcode Codepassagen, wie zum Beispiel Getter, Setter oder Konstruktoren, zu ersetzen. Dieser sich wiederholende Code wird Boilerplate-Code genannt.

Damit die oben genannten Bibliotheken und Unterprojekte in das Backend integriert werden können, nutzt Spring das Buildtool Maven. Maven lässt sich in die meisten Integrated Development Environment (IDE)´s leicht einbinden bzw. wird bei der Installation mit ausgeliefert. Es erleichert das integrieren von Abhängigkeiten und kümmert sich um den gesamten Build-Prozess. Dazu kann Maven durch zusätzliche Bibliotheken erweitert werden und sich zum Beispiel darum kümmern, dass nach dem Build Prozess verschiedene Artefakte zur Verfügung stehen oder der Quellcode alle Tests durchläuft.

4.4.1 Userverwaltung

Die Userverwaltung ist eine von den zwei Kernaufgaben des Backends. Sie dient dazu, dass sich User anmelden können, um so Annotationen hinzuzufügen, zu bearbeiten oder zul öschen . Bei diesem Protoypen ist die Userverwaltung relativ klein und simpel gehalten. User können sich:

- **Registrieren** User könenn sich einen Account im DDHS anlegen.

- **Anmelden** User können sich, wenn sie registriert sind, anmelden, damit sie Annotationen bearbeiten können.
- **Abmelden** Damit können User ihren Account löschen.
- **Account bearbeiten** Die User können ihre hinterlegten Daten ändern.

Das Backend kommuniziert immer mit der Datenbank, wenn die User Tätigkeiten an ihrem Account durchführen. Die User werden im Datenmodell als AppUser dargestellt. Ein AppUser besteht aus einer Id(In dieser Anwendung repräsentiert durch die Email), einem Passwort, dem Vornamen, dem Nachnamen und der Rolle. Die AppUser können eine von zwei verschiedenen Rollen zugewiesen bekommen. Die erste Rolle ist die Forscher Rolle. Diese Rolle erlaubt es den User Annotationen zu erstellen, zu bearbeiten und zu löschen. Sie können mit der Rolle ihre eigenen Daten löschen und bearbeiten. Die zweite Rolle ist die Administrator Rolle. User mit dieser Rolle können zusätzlich zu den Möglichkeiten der Forscher andere User löschen und sich alle User und ihre Rollen anzeigen lassen. Diese Adminrechte werden erlangt in dem man ein Administratorpasswort an eine bestimmte Schnittstelle im Backend schickt. Diese Art der Rollenkontrolle sollte in einem Produktivsystem geändert werden. Es ist für den Prototypen aber ausreichend, um schnell die Übersicht der User zu erhalten oder um falsch angelegte Accounts zu löschen. Die Schnittstelle, um diese Administrator Rolle zu erhalten, ist nicht im Frontend implementiert und muss mittels eines Resttools, wie z.Bsp. Postman, genutzt werden. Damit die User nicht jedes mal ihre Zugangsdaten zum DDHS übermitteln müssen, werden JWTs eingesetzt. Zwei JWTs werden nach erfolgreicher Anmeldung generiert und an das Frontend zurück geschickt. Das Frontend kümmert sich um die Tokenverwaltung, indem es den Token im lokalen Browserspeicher ablegt. Die JWTs haben eine zeitlich begrenzte Gültigkeit. Das Backend vergibt nach erfolgreicher Anmeldung folgende zwei Token:

- **Access Token** Dieser Token ist für die Autorisierung zuständig. Er hat in diesem Projekt eine Gültigkeit von zwei Stunden
- **Refresh Token** Dieser Token wird genutzt, um bei einem abgelaufenen Access Token einen neuen zu generieren. Die Gültigkeit von diesem Token beträgt 10 Tage.

Wenn der Access Token abgelaufen ist, wird dem Frontend mitgeteilt, dass der Token nicht mehr gültig ist. Nachdem das Frontend das erfahren hat, kann es versuchen mit dem 10 Tage gültigen Refresh Token einen neuen Access Token zu generieren. Davon bekommen die User nichts mit. Das geschieht im Hintergrund. Wenn das Erneuern des Access Token scheitert, müssen die User sich erneut einloggen.

Die Authentifizierung und die Autorisierung werden im Backend über zwei Filter geregelt, die mittels Spring-Security eingebunden sind. Die Authentifizierung dient der Token Generierung und prüft die übermittelten Login Daten mit den Daten aus der Datenbank ab. Die Autorisierung erfolgt anhand des übermittelten Authorization Header und dem danach folgenden Access Token, beginnend mit dem Wort Bearer und dem Token. Dazu wird der Token mittels dem JWT-Passwort und der Überprüfungsmethode aus der Java-JWT Bibliothek auf seine Gültigkeit geprüft. In diesem Prototypen ist dies die Zugriffskontrolle auf die Daten aus der Datenbank.

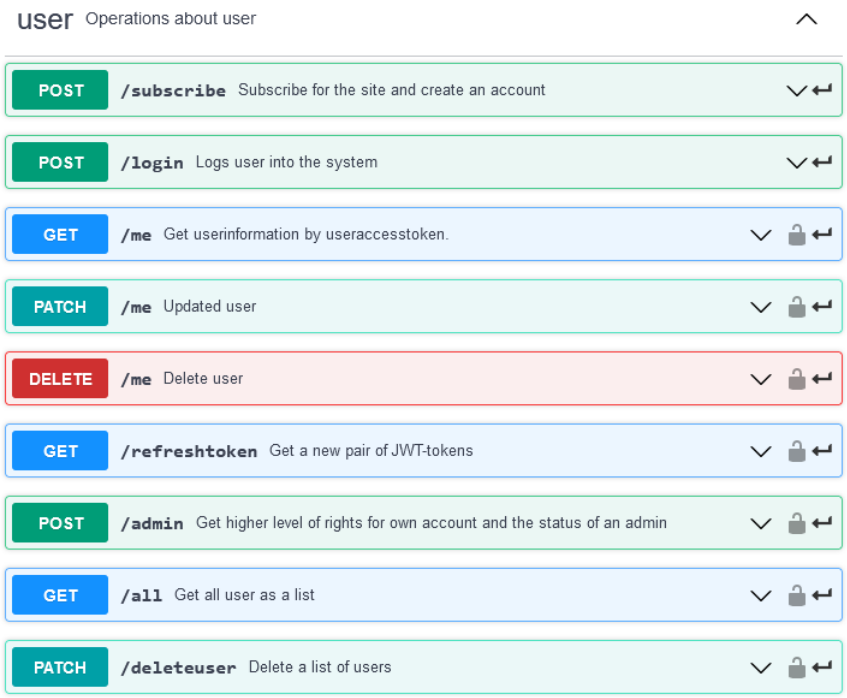

Abbildung 4.7: Übersicht der Schnittstellen für die Userverwaltung (Ged22a)

Das Backend bietet mehrere Schnittstellen für die Userverwaltung an. Diese Schnittstellen sind mittels Open API Standard dokumentiert. Diese Schnittstellendokumentation wird in dem Repository an die SUB mit übergeben. Die Abbildung 4.7 ist ein Ausschnitt aus der Schnittstellendefinition (Ged22a). Sie zeigt die Endpunkte der Userverwaltung. Der Open API Standard ermöglicht es, in der Schnittstellendefinition anzugeben, wie die Endpunkte anzusprechen sind. Es sind im Standard die Schemen der Anfragen und Antworten hinterlegt, sowie das Datenmodell der AppUser und der Annotation.

4.4.2 Annotationsverwaltung

Die Annotationsverwaltung ist die zweite von den zwei Kernaufgaben des Backends. Sie dient dazu, dass Annotationen durch die User angesehen,erstellt und gelöscht werden können.

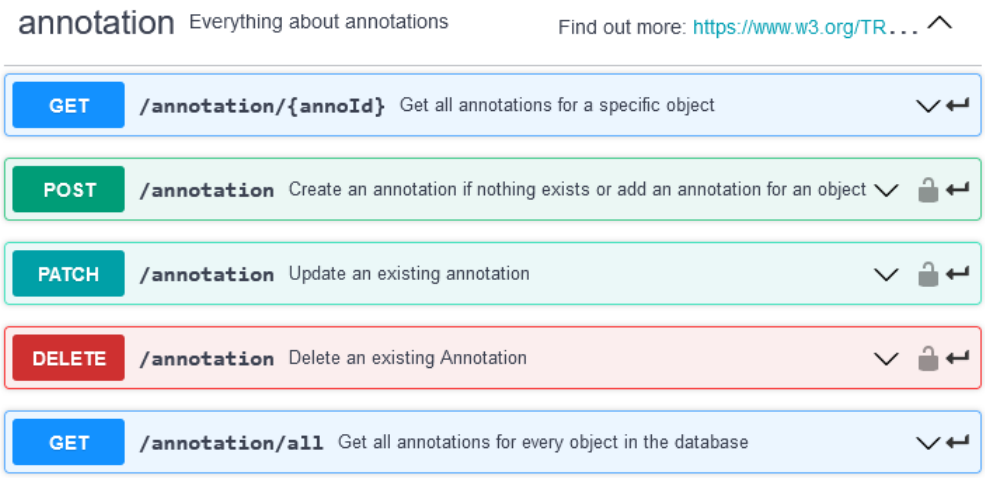

Abbildung 4.8: Übersicht der Schnittstellen für die Annotationsverwaltung

Die Abbildung 4.8 zeigt die Schnittstellen des Backends, die für die Annotationsverwaltung zuständig sind. Diese wurden ebenfalls mit dem Open API Standard dokumentiert und können im Repository und unter dem Link (Ged22a) angesehen werden. Die Annotationsverwaltung kann folgende 5 Aktionen ausführen:

- **Annotation ausgeben** Es werden alle Annotationen zu einem bestimmten Objekt ausgegeben.
- **Annotation erstellen** Angemeldete User können eine Annotation zu einem bestimmten Objekt erstellen.
- **Annotation bearbeiten** Angemeldete User können eine Annotation zu einem bestimmten Objekt verändern
- **Annotation löschen** Angemeldete User können eine Annotation zu einem bestimmten Objekt löschen.
- **Alle Annotationen erhalten** Es können alle Annotationen zu allen Objekten gesammelt ausgegeben werden.

Die Annotationen werden im JSON Format an das Backend übermittelt und auch im JSON Format an die anfragende Stelle übermittelt. Die Annotationen sind nach dem Webannotation Standard aufgebaut. Die Annotationen können von allen User angefragt werden. Jedoch können nur angemeldete User bzw. User mit einem gültigen JWT Token in der Anfrage Annotationen erstellen, bearbeiten oder löschen. Diese Richtlinie wurde während der Meetings mit der SUB und Uni HH festgelegt. Damit soll verhindert werden, dass jeder User Annotationen erstellen kann die sich nicht auf das Objekt beziehen. So erhält man mehr Kontrolle über die Annotationen.

4.5 Datenbank

Die unterste Schicht der „Three Tier Architecture“ ist die Datenzugriffsschicht. Sie wird in Form einer Datenbank mit Datenbankmanagementsystem repräsentiert. In der Datenbank werden die Informationen zentral und persistent (dauerhaft) gespeichert. Ein Datenbanksystem besteht immer aus zwei Komponenten. Die eine Komponente ist die Speicherungskomponente und die andere Komponente ist die Verwaltungskomponente.
Es gibt zwei unterschiedliche Arten von Datenbanken. Auf der einen Seite gibt es die relationalen Datenbanksysteme. In relationalen Datenbanksystemen werden die Informationen in Tabellen abgelegt. Diese Tabellen enthalten die Daten und Datenbeziehungen zueinander. Diese Relationen können die Abhängigkeiten zueinander aufdecken. Des weiteren wird in der Datenbank das Schema der einzelnen Tabellen hinterlegt. Dieses definiert die Identifikationsschlüssel und Merkmale. Die Verwaltungskomponente einer relationalen Datenbank definiert (MK16):

- **Datenzugriffsprache** Der weiteste verbreitete Standard ist Structured Query Language (SQL).
- **Mehrbenutzerbetrieb** Dies regelt, wie mehrere User gleichzeitig auf die Datenbank zugreifen können und sich laufende Transaktionen nicht gegenseitig behindern.
- **Architektur** Ermöglicht die Datenunabhängigkeit. Es werden Daten und Anwendungen von einander mit Hilfe der Verwaltungskomponente entkoppelt.

Auf der anderen Seite gibt es die nicht-relationalen Datenbanksysteme. Oft werden diese als NoSQL-Datenbanksysteme bezeichnet. Diese Systeme kommen zum Einsatz, wenn die Daten oder Informationen nicht zwangsweise einer Struktur oder einem abhängigen (relationalem) Aufbau folgen. Die zwei größten Unterschiede zu einer relationalen Datenbank sind erstens

die Daten werden nicht in einer Tabellenform gespeichert und zweitens die Datenbankzugriffssprache ist nicht SQL. Diese Art von Datenbanken werden heut zu tage viel in Big Data Anwendungen oder massiv verteilten Webanwendungen eingesetzt. In einer nicht relationalen Datenbank werden die Daten entweder als Schlüssel-Wert-Paar, als Spalte, Dokument oder als Graphen abgelegt. Ein nicht-relationales-Datenbanksystem erfüllt diese Bedingungen (MK16):

- Das Datenbankmodell ist nicht relational.
- Es gibt kein festes Datenbankschema.
- Die Datenrepliaktion wird vom Datenbanksystem unterstützt.

In dieser Arbeit wird das nicht-relationale Datenbankmodel genutzt. Diese Entscheidung wird anhand des Modells der Webannotationen getroffen, die als JSON Format dargestellt werden. Somit können sie in einer nicht-relationalen Datenbank als Dokumente abgelegt werden. Dokumente sind Text-Daten, die strukturiert sind und im JSON oder Extensible Markup Language (XML) Format vorliegen. Auf sie wird über einen eindeutigen Schlüssel oder ein Merkmal zugegriffen. Bei einer Annotation ist der eindeutige Schlüssel der Name des Objekts, welches annotiert wird. Diese Struktur ist schemenfrei und kann bei Änderung des Webannotation Standards leicht geändert werden. Ein weiterer Vorteil einer nicht relationalen Datenbank ist, dass man die Daten leicht auf mehrere andere nicht relationale Datenbanken verteilen kann. Das erleichtert die Lastverteilung des Netzwerkverkehrs und ist mit dieser Art von Datenbanken aufgrund des Dokumentenaufbaus leichter zu realisieren (MK16).

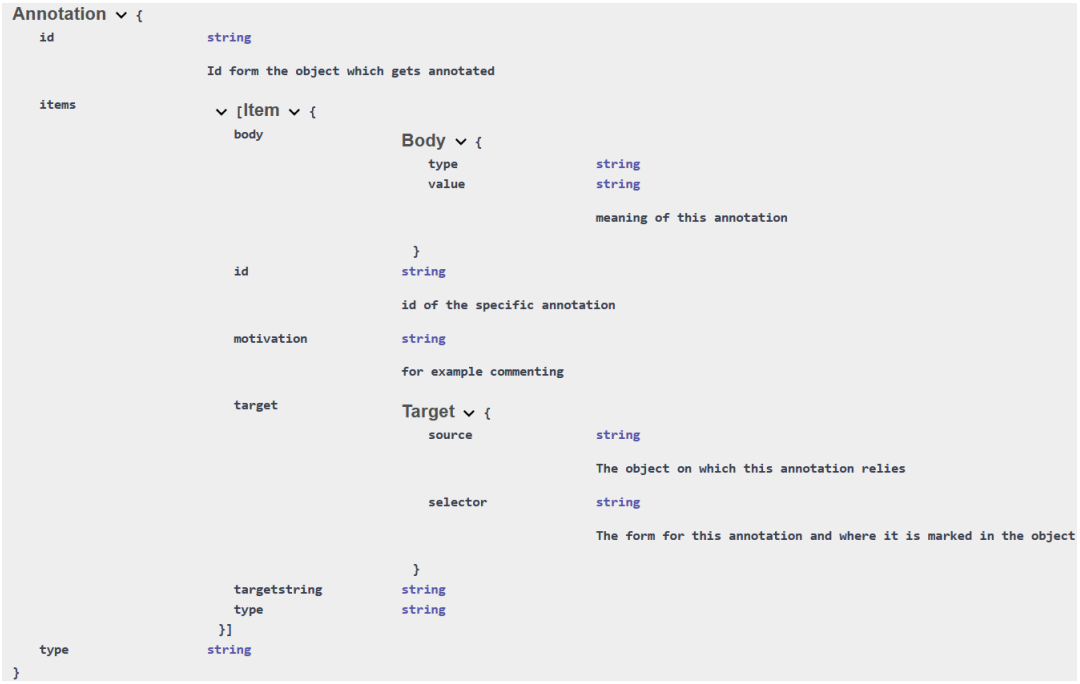

Abbildung 4.9: Übersicht über das Modell für eine Annotation

In Abbildung 4.9 wird der Aufbau einer Annotation in der Datenbank dargestellt. Die Id ist der Identifizierungsschlüssel der Annotation (Id des Objekts). Der Body ist der textuelle Teil der Annotation. Das Target definiert das Ziel der Markierung der Annotation. Der Selector des Targets gibt an, mit welcher Form das Objekt wo markiert wurde.
Die User des DDHS werden ebenfalls in dieser Datenbank gespeichert. Die User wurden in diese Datenbank integriert, damit der Verwaltungsaufwand in dem Prototypen gering bleibt. Es werden dadurch aktuell alle Daten an einem zentralen Ort gespeichert. Die AppUser hätten in einer relationalen Datenbank abgelegt werden können. Dazu wären zwei Datenbanksysteme nötig gewesen und der Aufwand diese zu betreiben zu hoch.

Abbildung 4.10: Übersicht über das Modell für einen AppUser

Das Modell der AppUser ist in Abbildung 4.10 dargestellt. Der Identifizierungsschlüssel eines AppUsers ist die Email-Adresse. In der Datenbank werden die Passwörter nicht im Klartext abgelegt. Sie werden bereits vom Backend kryptiert und dann an die Datenbank übertragen. Ein AppUser hat neben der Id und dem Passwort einen, Vornamen, Nachnamen und eine zugewiesene Rolle.
Das eingesetzte Datenbanksystem heißt MongoDB. Das Backend stellt eine Bibliothek bereit, um die Verbindung mit einer MongoDB Datenbank herzustellen. MongoDB ist ein nicht-relationales-Datenbanksystem, welches dokumentenorientiert ist. Da die Annotationen als Dokumente abgespeichert werden sollen, bietet sich die Nutzung von MongoDB an. Die Nutzung der MongoDB ist im Internet gut dokumentiert und erleichtert den Einstieg in die Verwendung.

4.6 Deployment

Deployment bezeichnet den Prozess der Installation von Software. Das Deployment des DDHS ist im Rahmen dieser Arbeit im Kubernetes-Cluster der ICC der HAW geschehen. Dazu wird mit Hilfe der Mittel des Gitlab Servers der HAW für das Frontend und Backend Pipelines eingerichtet. Diese Pipelines stellen die Continuous Integration (CI) und das Continuous Delivery (CD) sicher. CI/CD bedeutet, dass jeder neue Softwarestand automatisiert in das laufende System eingefügt und installiert wird. Wenn zum Beispiel das Backend eine neue Schnittstelle

erhalten hat, wird diese, nach dem Hochladen des Updates in das Repository, anschließend verfügbar sein.
Kubernetes ist ein Tool zur Orchestrierung von Containern. Dazu werden Container in sogenannten Pods gestartet und überwacht. Kubernetes ermöglicht es, dass Pods in einem Namespace miteinander kommunizieren können. Es erleichtert das Verwalten der einzelnen Container und bietet die Möglichkeit Passwörter und Umgebungsvariablen abzuspeichern, damit die Pods darauf zugreifen können. Namespaces definieren bei Kubernetes einen isolierten Bereich. Für diese Arbeit wird ein Namespace mit dem Namen „dehmel-digitalhumanities“ angelegt. In diesem isolierten Bereich können die Pods sich gegenseitig finden. Die Passwörter für die MongoDB und das Backend wurden in den Kubernetes Secret-Speicher für den Namespace dehmel-digitalhumanities gelegt. Damit sind sie nicht im Klartext im Repository hinterlegt und die Passwort Dateien können von jedem Pod innerhalb des Namespaces benutzt werden. Das Deployment für das Frontend ist wie folgt aufgebaut:

1. Das Dockerimage wird erstellt. Dazu wird das Dockerfile ausgelesen, in dem definiert ist, wie das Image aufgebaut ist. In dem Frontend wird zuerst die Angularanwendung erstellt. Danach wird diese erstellte Anwendung an den NGINX Server übergeben und in seine Dateistruktur integriert.
2. Das Dockerimage wird in die Container Registry der HAW hochgeladen.
3. Der Kubernetes-Cluster lädt das neue Image aus der Registry.
4. Der Frontend-Pod startet mit dem neuen Image.
5. Das Frontend ist auf aktuellem Stand verfügbar.

Damit das DDHS für die SUB und Uni HH einsehbar und der Stand der Entwicklung sichtbar ist, wird über die ICC der HAW eine Domain angefordert. Diese Domain ist über den Zeitraum der Entwicklung und bis zum Abschluss der Arbeit unter folgendem Link erreichbar.

https://dehmel-digitalhumanities.informatik.haw-hamburg.de/

Das DDHS wird mit einer Ingress Konfigurationsdatei auf die URL weitergeleitet. Ingress in Kubernetes dient als Router bzw. Proxy für die Weiterleitung von Datenverkehr im Kubernetes-Clustern aus einzelne Pods. Pods im Kubernetes-Cluster können durch Ingress über Extern verfügbare URLs verfügbar gemacht werden.
Das Backend wird ebenfalls als Pod im Kubernetes-Cluster zur Verfügung gestellt und wird wie folgt deployed:

1. Das Dockerfile erstellt ein Image. Dazu wird mit Maven der Backend-Code zu einer ausführbaren Java Datei erstellt. Die Java Datei wird dann in eine Javaumgebung zur Ausführung kopiert.
2. Das Dockerimage wird in die Container Registry der HAW hochgeladen.
3. Der Kubernetes-Cluster lädt das neue Image aus der Registry.
4. Der Backend-Pod startet mit dem neuen Image.
5. Das Backend ist auf aktuellem Stand verfügbar.

Die Datenbank ist nicht über ein Repository deployed worden, sondern wird direkt in den Kubernetes-Cluster geladen. Da die Datenbank nicht verändert wird, wird das aktuellste MongoDB Image genutzt. Der Speicher wird als PersistentVolume im Kubernetes-Cluster reserviert. Als Speichergröße wurden 900MByte definiert. Diese Größe reicht für den Prototypen aus, da keine großen Datenmengen erwartet werden.
Die SUB hat zur Übergabe Zugriff auf die Repositories des Frontend und Backend erhalten. Damit kann die SUB die Daten in ihre IT-Infrastruktur kopieren und dann von dort aus betreiben und gegebenenfalls weiter anpassen. Die SUB wird das DDHS vorraussichtlich als Dockerimages in ihrem Kubernetes-Cluster deployen. Somit wird dort das Deployment ähnlich wie in dieser Arbeit sein.

5 Evaluation

Die Evaluation in dieser Arbeit beschäftigt sich mit der Auswertung des erstellten Prototypen. Dazu wird das Testing der einzelnen Komponenten erklärt und die von der SUB und Uni HH gestellten Anforderungen an das DDHS auf ihre Umsetzung und Gültigkeit geprüft. Des weiteren werden Probleme bei der Umsetzung angesprochen und die technischen Schulden dargestellt.

5.1 Testing des Prototypen

Testen ist ein wichtiger Bestandteil der Softwareentwicklung. Automatisierte Tests sind heute der Standard. Ein Test sollte am besten:

- Ein einfaches Verhalten, eine Methode oder eine einzelne Schnittstelle einer API testen.
- Einen bestimmten Input oder Wert für einen API Aufruf haben.
- Ein beobachtbaren Output oder Verhalten haben.
- In einem kontrolliertem Bereich stattfinden, einem isoliertem Prozess.

Dies ist die Herangehensweise von Google (WMW20).
In dieser Arbeit wird der Hauptschwerpunkt auf die Umsetzung und nicht auf das vollständige Testen gelegt. Die Tests werden manuell im Frontend und teilweise automatisiert im Backend durchgeführt. Der Umfang der Tests ist gering gehalten. Es werden wie bei Google einzelne Schnittstellen auf ihre Funktion überprüft und das Verhalten dieser beobachtet. Jedoch sind in dieser Arbeit eine geringe Anzahl an Tests genutzt worden. Dies muss in einem späteren Einsatz bzw. nach der Prototypenphase angepasst werden.
Das Frontend wird im Browser getestet. Dazu wird lokal, nach Erstellung einer neuen Komponente für die Angularwebanwendung, das Frontend gestartet oder erneut geladen . Die Komponente auf ihre korrekte Funktion durch anklicken im Browser geprüft.
Das Annotation Plugin wird in seiner erweiterten Funktion in der Webanwendung getestet. Nach Änderung eines Features oder Fehlers wird ein Objekt im Mirador geöffnet und die

Funktionalität überprüft.
Die zwei vorangegangen Arten des Tests sind live Tests, die manuell ausgeführt werden. Es ist möglich visuelle Tests mit modernen Testframeworks durchzuführen, dafür fehlte die Zeit bei der Implementation des Prototypen. Da es sich um einen Prototypen handelt, wurden diese Tests so ausgeführt wie beschrieben. Bevor das System einen Live Einsatz erhält, würde es sich anbieten, automatisierte Tests für das Frontend zu integrieren.
Das Backend wird während der Entwicklung mit einem Plugin für Visual Studio Code namens Thunder Client getestet. Damit kann man REST APIs testen. Die Tests werden lokal durchgeführt bevor das Backend bzw. neue Schnittstellen deployed werden. Dazu werden die einzelnen Schnittstellen mit dem Tool angesprochen und die Antworten des Backends überprüft. Man kann so feststellen, ob das Backend erreichbar ist und mit der Datenbank kommuniziert. Damit das Backend nach dem Deployment getestet wird, gibt es im Thunder Client 3 Separate Testgruppen. Die erste Testgruppe ist für das lokale Testen der UserSchnittstelle zuständig, die zweite für die Userschnittstellen im ausgerollten Backend in der ICC und das dritte testet sowohl lokal als auch in der ICC die Annotationsschnittstellen.

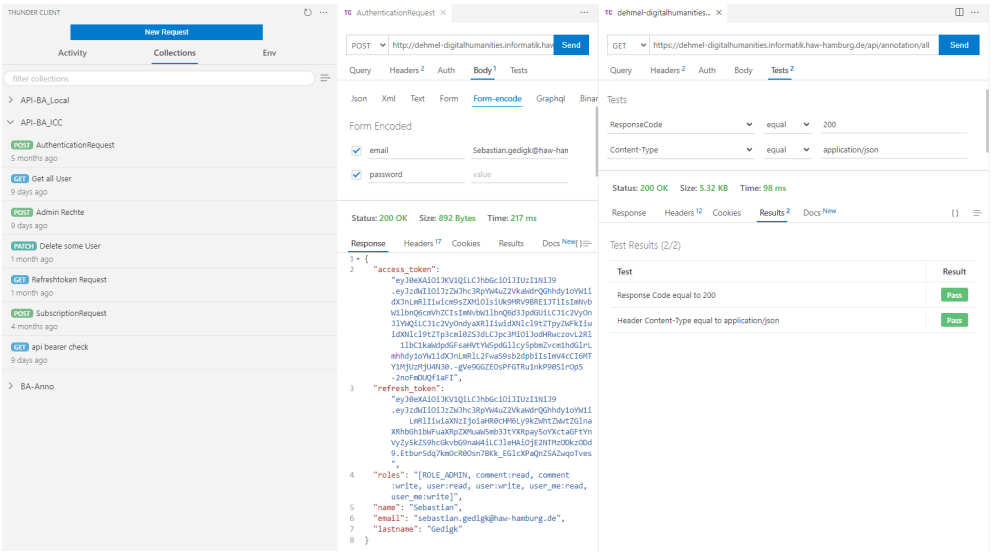

Abbildung 5.1: Ansicht Thunder Client mit Tests des Backends

In der Abbildung 5.1 wird dargestellt, wie der Thunder Client zwei API Aufrufe ausgeführt hat. Auf der linke Seite sieht man die vorher angesprochenen drei Gruppierungen zum testen. In der Mitte wird ein Login durchgeführt. Zu sehen ist die Antwort des Backends mit dem Status Code und dem Inhalt des Antwortbodys. Dieser beinhaltet den Access-Token, den Refresh-Token,

und die restlichen Account Informationen des Users. Auf der rechten Seite der Abbildung wir der erfolgreiche Abruf aller Annotationen dargestellt. Des weiteren sieht man die Tests, die mit dieser Anfrage ausgeführt wurden. Der erste Test prüft den Antwortcode ab, ob dieser 200 ist und somit die Abfrage korrekt vom Backend verarbeitet werden konnte. Der zweite Test prüft, ob die Antwort vom Typ JSON ist. Das Tool ermöglicht es, dass man die Schnittstellen abfragen kann und manuell die Antwort auswertet. Des weiteren ermöglicht es das Tool, Tests in die Abfragen einzubinden und die Auswertungen dieser Tests zu erhalten.
Die Datenbank wird in Kombination mit dem Backend geprüft, denn sie wird nur vom Backend angesprochen und nicht autark verwendet. Die Validierung der Daten wird manuell durchgeführt. Dazu werden als Beispiel User über das Backend erstellt und anschließend eine Liste aller User vom Backend angefordert. Diese Daten wurden dann gegeneinander geprüft. Wenn sie stimmen sind die Daten in der Datenbank korrekt abegelegt worden. Es bietet sich an, im späteren Einsatz das Anlegen eines Users mit einem automatisierten Test auf Korrektheit zu prüfen.

5.2 Validierung der Anforderungen

Damit das DDHS an die SUB und Uni HH übergeben werden kann, muss überprüft werden, ob die Anforderungen an das DDHS erfüllt sind. Dazu sind alle Anforderungen aus dem Kapitel 2 in einer Tabelle (Anforderungstablle) zusammengefasst worden.
Das System wird, wie in Kapitel 4.6 beschrieben, mit drei Docker-Containern betrieben. Damit ist diese Anforderung Anf. 1 erfüllt.

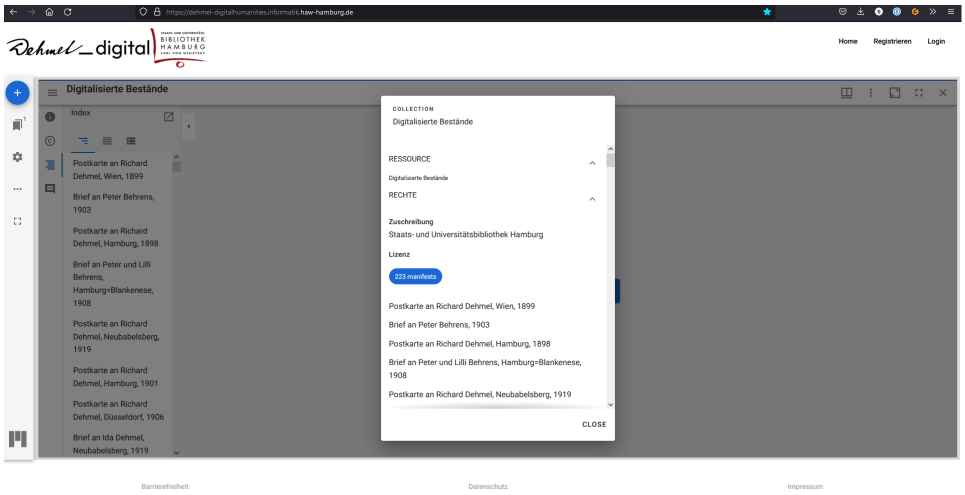

Abbildung 5.2: Ansicht der Startseite des Digital-Humanities-System

Die Abbildung 5.2 zeigt das fertige DDHS. Das DDHS wird mit dem Link aufgerufen, der von der ICC zur Verfügung gestellt wird. Damit ist es außerhalb des HAW-Netzwerks einsehbar (Anf. 2). Auf der gezeigten Startseite erhält der User die Möglichkeit, ein Objekt im Mirador-Viewer auszuwählen(Anf. 3). Nach der Auswahl wird dieses Objekt geöffnet.
Die Annotationen werden innerhalb des Kubernetes Cluster in einer zentralen Datenbank gespeichert (Anf. 4). Das Backend erhält die Annotationen im Webannotation Standard (Anf. 5) von der Datenbank und gibt die Annotationen in diesem Format an das Frontend weiter oder über die Annotation API extern aus (Anf. 10).

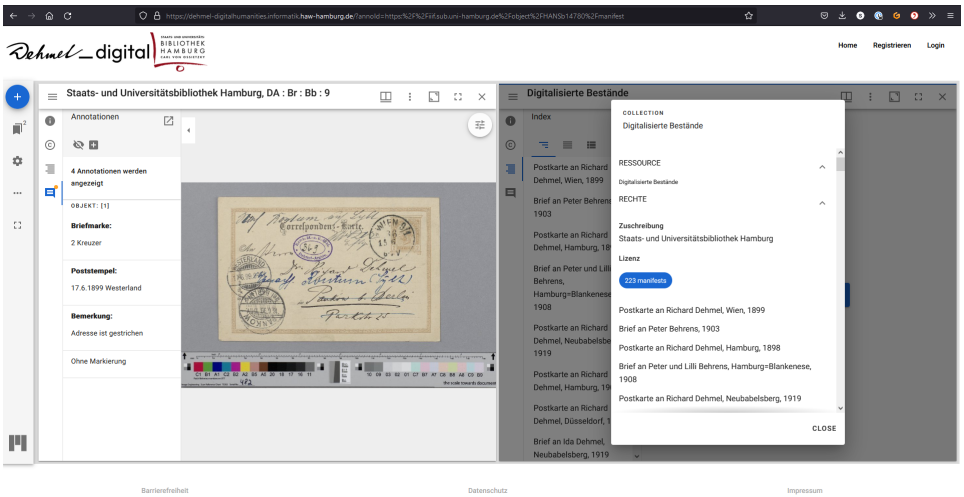

Abbildung 5.3: Ansicht direkter Einstieg in ein Objekt des Digital-Humanities-System

Die Abbildung 5.3 zeigt die Ansicht des Direkteinstiegs in ein Objekt (Anf. 6). Auf der linken Seite des Mirador-Viewers ist das gewünschte Objekt verfügbar. Es hat in der Sitzung für Abbildung 5.3 kein Login stattgefunden. Man sieht auf dem Objekt der linken Seite die zugehörigen Annotationen (Anf. 7), welche vom Backend angefordert wurden. Auf der rechten Seite kann man zusätzlich ein weiteres Objekt anzeigen lassen. Die Annotationen sollen moderierbar sein. Diese Anforderung wird teilweise eingehalten. Es ist möglich Annotationen zu löschen und zu editieren (Anf. 8). Wenn eine Annotation erstellt wird, ist sie sofort sichtbar und wird nicht erst auf Schimpfwörter geprüft oder muss von einem Moderator freigegeben werde. Daher kann sie nur durch editieren oder löschen moderiert werden.

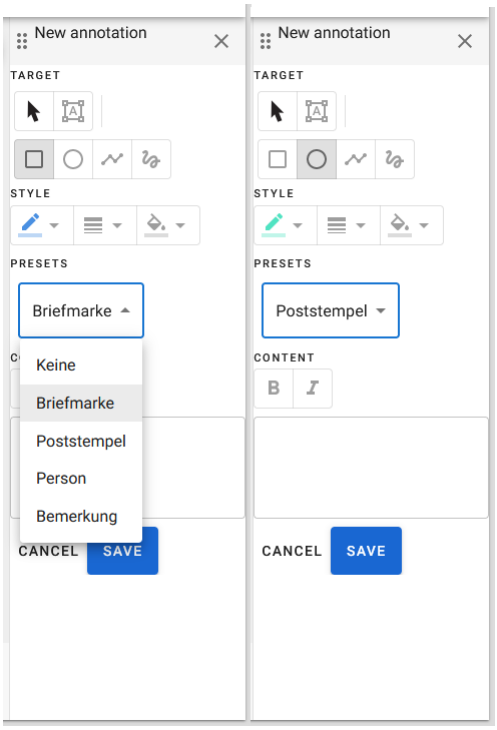

Abbildung 5.4: Ansicht des Annotation Plugin mit Presets

Wie in Abbildung 5.4 zu sehen ist, kann man mit dem weiterentwickelten Annotation Plugin Presets auswählen(Anf. 9). Die Presets sind in diesem Plugin statisch und können nur über das Plugin angepasst werden.
User können Annotationen nur erstellen, wenn sie eingeloggt sind (Anf. 12). Dazu müssen sie sich vorher registriert haben. Es ist dafür eine Userverwaltung integriert worden (Anf. 11). Sie ist aufgrund der Tatsache, dass es hier um einen Prototypen geht, simpel gehalten. Die Userverwaltung erlaubt es lediglich Namen, Vornamen, Passwort und, wenn das Administratorpasswort bekannt ist, die Rolle zu wechseln.

Anf. Nr.	Anforderung	Anforderung Erfüllt?
1	System soll als Container betrieben werden	Ja
2	Aktueller Stand der Entwicklung einsehbar	Ja
3	Betrachtungstool Mirador	Ja
4	Annotationen persistieren	Ja
5	Annotationsformat Webannotations	Ja
6	Direkteinstieg in ein bestimmtes Objekt	Eingeschränkt siehe Kap. 5.4
7	Annotationen von allen User einsehbar	Ja
8	Annotationen sollen moderierbar sein	Eingeschränkt siehe Kap. 5.4
9	Annotationen mit Presets	Ja, statisch
10	Annotationen sollen exportierbar sein	Ja
11	Userverwaltung erstellen	Ja
12	Registrierung notwendig für das Annotieren	Ja

Tabelle 5.1: Tabellarische Übersicht der Anforderungen an das DDHS

Die Tabelle 5.1 zeigt, dass grundsätzlich die Anforderungen erfüllt sind. Es gibt 2 Einschränkungen, die bei späterer Verwendung angepasst werden sollten. Der Prototyp ist mit diesen Einschränkungen trotzdem einsatzbereit und funktionsfähig.

5.3 Risiken

Die Risiken bei einer Webanwendung sind stets zu beachten. Man muss die möglichen Angriffspunkte versuchen zu minimieren. Es ist nicht möglich ein zu 100% sicheres System zu haben. Die Risiken für dieses Projekt, das Ziel eines koordinierten Angriffs zu werden, sollten geringer sein als bei einer Webanwendung mit hohem Bekanntheitsgrad. Ein potentielles Risiko ist ebenfalls falsche Recherche in Bezug auf welche Technologie wie eingesetzt werden sollte. Dazu sollte man recherchierte Studien und Statistiken immer hinterfragen und die Quellen prüfen, denn wie in dem Paper von Daniela S. Cruzes (CO17) beschrieben, kommt es in Softwarestudien zu fehlerhaften Interpretationen der Zahlen oder falschen Erhebung von Zahlen. Dadurch kann es passieren, dass falsche Entscheidungen beim Einsatz von Technologien getroffen werden. Zur Verdeutlichung welche Risiken es für dieses Projekt gibt, werden drei Beispiele aufgezeigt.

1. Potentielle Angreifer können mit einem Bruteforce-Angriff versuchen das Administratorpasswort zu erhalten. Ein Bruteforce-Angriff ist ein gezielter Angriff von einem oder mehreren Computern, der durch systematische und wiederholte Passworteingaben ver-

sucht, die richtige Passwortvariante zu erraten und so einen Administratorzugang erhält. Mit diesem Administratorzugang kann man sämtliche User löschen. Das ist ein Szenario, welches nicht unwarscheinlich ist. Der Angreifer erhält so die Vornamen und Nachnamen sämtlicher User und deren Email-Adresse. Dies kann an anderen Stellen genutzt werden, um dort mit Hilfe der Email-Adresse und einem erneutem Bruteforce-Angriff Passwörter auf anderen Seiten der User zu erhalten. Damit man das verhindern kann, ist es sinnvoll den Prototypen mit einer anderen Authentifizierungsart zu betreiben. Eine mögliche Variante wäre es mit einem Single Sign-On Dienst eines anderen Anbieter zu arbeiten, zum Beispiel mit dem Dient von Google.

2. Ein weiteres Risiko dieser Webanwendung ist Hassrede oder rechtswidriger Inhalt in den Annotationen der einzelnen Objekte. Es besteht die Möglichkeit, dass die Annotationen für illegalen oder beleidigenden Inhalt missbraucht werden könnten. Diese Problematik ist im Internet weit verbreitet und kommt oft in sozialen Netzwerken vor (Rit18). Um dagegen anzugehen, kann man versuchen einen Schimpfwortfilter im Backend zu implementieren. Eine weitere Idee ist es, dass Annotationen erst freigegeben werden müssen bevor sie veröffentlicht werden.

3. Das dritte Risiko bezieht sich auf die Verwendung von Bibliotheken für die Webanwendung. Im Dezember 2021 ist die Sicherheitslücke von Log4j veröffentlicht worden. Diese Bibliothek wurde in abgewandelter Form bis zu diesem Zeitpunkt genutzt, um im Backend Informationen zu erfassen. Mit der Veröffentlichung der Sicherheitslücke wurde auf den Einsatz dieser Logging Bibliothek verzichtet. Zusammengefasst ermöglichte diese Sicherheitslücke, dass Angreifer mit einer bestimmten Zeichenfolge bzw. bestimmten dafür entwickelten Tools, Server mit Schadsoftware infizieren konnten. Danach waren die Angreifer in der Lage, die Daten aus dem Server auszulesen oder ihn sogar zu löschen (HNSW22). Damit dies nicht geschieht, wurde diese Bibliothek aus dem Quellcode entfernt.

Diese drei genannten Risiken spiegeln einen kleinen Ausschnitt der möglichen Angriffszenarien für die in dieser Arbeit erstellte Webanwendung wider. Bei Weiterentwicklung des Prototyps oder des Plugins sollte eine Risikoanalyse vorgenommen werden, um die Risiken zu minimieren.

5.4 Probleme und technische Schulden

In diesem Unterkapitel wird auf die aufgetretenen Probleme bei der Entwicklung des DDHS eingegangen. Der weiteren werden die technischen Schulden gegenüber der SUB und der

Uni HH dargestellt.

5.4.1 Probleme

Während der Entwicklung des Plugins ist es ab einem unbestimmten Zeitpunkt nicht mehr möglich gewesen, das Plugin als lokale Abhängigkeit einzubinden. Das Plugin funktionierte nur in der Webanwendung, die in die ICC deployed wird. Nach längerem untersuchen ist der Fehler gefunden worden. Das Plugin nutzt eine andere React Version als der Rest des Projekts. Damit hatte das Plugin Probleme. Die Lösung ist das Tool Yalc (ML22) aus der NPM Bibliothek. Yalc ermöglicht es ein lokales NPM Paket in sein Projekt einzubinden, so als ob es aus der NPM Bibliothek eingebunden wird. Dazu lädt das Projekt die Abhängigkeiten für das Plugin aus der Yalc Bibliothek. Mit dieser Lösung gab es keine Probleme mehr beim lokalen einbinden des Plugins. Die Weiterentwicklung des Plugins konnte ohne Veröffentlichung in der NPM Bibliothek weitergeführt werden, bis der Stand zur Veröffentlichung erreicht war.

Das Plugin hatte Probleme die Authentifizierung am Backend einzubeziehen. Der Ansatz des Frontend ist es, mit einem Interceptor zu arbeiten. Dabei werden alle ausgehenden Anfragen zum Backend vom Interceptor abgefangen. Der Browser hat nach erfolgreichem Login die Daten des Users in Form eines JWT-Token gespeichert. Der Interceptor nimmt, falls der User eingeloggt ist, den JWT-Token und hängt die Autorisierung dem Header zu der Abfrage an. Das Problem war, dass Mirador mit dem Plugin eigenständig Abfragen sendet, ohne das Frontend mit einzubeziehen. Es war leider nicht möglich, alle ausgehenden Anfragen über das Frontend bzw. die Angular-Anwendung zu leiten. Daher wird das Plugin angepasst. Es nutzt jetzt, falls vorhanden und benötigt, den JWT-Token aus dem Browser und leitet, falls dieser abgelaufen ist, eine Aktualisierung des Token ein.
Ein weiteres Problem waren Cross-Origin Resource Sharing (CORS)-Fehler. Dieser Mechanismus ermöglicht es, dass Anwendungen/ Browserclients, die nicht auf dem selben Gerät installiert sind, mit dem Backend kommunizieren können. Dazu muss im Backend an verschiedenen Stellen die Erlaubnis erteilt werden, dass Anfragen von anderer Herkunft akzeptiert werden. Dies betrifft in der Regel Browseranfragen vom Frontend zum Backend. Die Lösung war, im Backend die Sicherheitsregeln anzupassen. Das wird in der Securitykonfiguration und den REST-Controllern der Annotaion und Userverwaltung angepasst.
Zuletzt sind Probleme beim Mirador-Viewer zu nennen. Diese beziehen sich auf die Bedienung des Tools. Es ist nicht wirklich intuitiv zu bedienen. Der Mirador-Viewer lässt es nicht zu, dass man beim Schließen der Auswahlseite der Objekte danach ein Objekt aus der Auflistung

öffnen kann. Nachdem man die Auswahlseite geschlossen hat, muss man die Webseite erneut öffnen, um ein Objekt auszuwählen. Eine weitere unschöne Lösung ist, dass man Kontextseiten beliebig oft öffnen kann. Dies geschieht zum Beispiel wenn man die Index Seite im Hilfsfenster öffnen möchte. Dann kann man sie beliebig oft öffnen. Diese Probleme sollten bei Gelegenheit den Entwicklern des Mirador-Viewers mitgeteilt werden.

5.4.2 Technische Schulden

Unter technischer Schuld wird eine unsaubere bzw. nicht ganz korrekte Umsetzung von Anforderung verstanden. Diese hat einen Mehraufwand und Änderungen der Software zur Folge. In diesem Projekt sind drei Anforderungen anzusprechen, die für den Einsatz angepasst werden sollten.
Der erste Punkt ist die Anforderung des Direkteinstiegs in ein Objekt (Anf. 6). Es ist möglich in ein explizites Objekt auf der Webseite einzusteigen, jedoch nur mit einem zweiten Auswahlfenster. Bis zur Abgabe des Projekts konnte keine Lösung dafür gefunden werden, ein Fenster im Mirador-Viewer zu erhalten und gleichzeitig die gesamte Bibliothek der Objekte einzubinden. Das zweite Miradorfenster muss manuell geschloßen werden, damit das Objekt den kompletten Platz im Viewer einnimmt.
Der zweite Punkt ist die Anforderung der moderierbaren Annotationen (Anf. 8) nur teilweise umgesetzt. Annotationen können gelöscht und bearbeitet werden, jedoch ist es nicht möglich sie erst vor der Veröffentlichung freizugeben. Ein Moderationssystem ist die beste Lösung, um sich vor dem Risiko 2 aus dem Kapitel 5.3 zu schützen.
Der dritte Punkt betrifft die Userverwaltung. Diese wird von der SUB gefordert, damit nur registrierte User Annotationen bearbeiten und erstellen können. Zusätzlich war es der Wunsch der SUB, einen Login mit OAuth oder Shiboleth einzubinden. Die Umsetzung war eher als zusätzliches Feature formuliert und nicht sehr hoch priorisiert. Darum wird unter Berücksichtigung der verfügbaren Zeit, die Entscheidung getroffen, die Einbindung dieser Authentifizierungsart auszulassen.
Diese drei Punkte sollten bei einer Weiterentwicklung des Projekts angegangen werden, um ein besseres Nutzungsverhalten zu erzeugen.

6 Fazit und Ausblick

6.1 Fazit

In dieser Arbeit wurde eine Webanwendung mit dem Namen Dehmel-Digital-Humanities-System für die Uni HH und SUB erstellt, diese Anwendung stellt digitalisierte Objekte des Dehmel-Archivs dar und ermöglicht es Annotationen vorzunehmen.

Die Webanwendung wurde nach dem klassischen „Drei Schichten Modell“ umgesetzt. Das Frontend wurde mit dem JavaScript-Framework Angular erstellt. Dies ermöglicht es, eine Webanwendung als Single Page Applikation zu erstellen. Das Backend ist mit dem Java Framework Spring Boot umgesetzt. Spring Boot ermöglicht es, schnell und einfach einen Backend Server zu konfigurieren, der die Anbindung an eine Datenbank herstellt. MongoDB ist in dieser Anwendung das Datenbankmangementsystem. Es ist nutzt ein auf NoSQL basierendes Datenbanksystem.

Es war von der Uni HH und SUB gefordert worden, dass die Objekte des Dehmel-Archivs annotierbar sein sollen. Diese Anforderung wurde mit einem Annotations Plugin umgesetzt. Dieses Plugin war bereits verfügbar, musste jedoch an die Bedürfnisse der auftraggebenden Personen angepasst werden. Dazu wurde das Plugin an das Backend angepasst. Es erhielt die Möglichkeit die Autorisierungsmethode mittels JWT-Token zu nutzen und Annotationen an ein Backend zu übermitteln. Des weiteren wurde das Tool um die Möglichkeit Presets zu nutzen erweitert.

Die einzelnen Komponenten dieser Arbeit wurden so entwickelt, dass sie als Docker-Container ausgeliefert werden. Das heißt man kann sie unabhängig vom Betriebssystem nutzen. Zusätzlich hat man die Möglichkeit die Container in eine bestehende Container-Orchestrierung einzubinden.

Damit wurden die Ziele, die diese Arbeit hatte, erreicht. Die Uni HH und SUB haben einen

funktionsfähigen Prototypen erhalten. Dieser ist in der Lage, die Objekte des Dehmel-Archivs darzustellen. Die User dieser Anwendung sind in der Lage, Annotationen an diesen Objekten durchzuführen und diese persistent abzulegen. Somit kann jeder User der Anwendung diese Annotationen sehen.

6.2 Ausblick

Dieser Prototyp hat das Potential, im realen Einsatz der Uni HH oder SUB genutzt zu werden. Ein möglicher Einsatz wurde bereits während der ersten Interviews mit Julia Nantke und David Maus besprochen. Sie können sich vorstellen, diesen Prototypen zu ersten Versuchen in Übungen an der Uni HH einzusetzen. So könnte die Nutzbarkeit der Anwendung auf die Probe gestellt werden. Durch Nutzung der Webanwendung können Fehler im System aufgedeckt werden. Ebenso ist es möglich über weitere Features für die Annotationen nachzudenken. Drei mögliche Erweiterungen könnten wie folgt aussehen:

- Die Presets für Annotationen könnten dynamisch in das Plugin integriert werden. Das bedeutet, dass je nach Einsatzweck verschiedene Forschungsgruppen eventuell andere Presets nutzen möchten. Dazu könnte man im Backend eine Schnittstelle anbieten, die diese Presets ausgibt und eine weitere Schnittstelle um Presets zu definieren.
- Die Userverwaltung könnte weiter ausgebaut werden. Dies bezieht sich nicht auf das Usermanagement, sondern auf die Funktionalität Forschungsgruppen zu erstellen. Diese Forschungsgruppen können zum Beispiel vordefinierte Presets für Annotationen nutzen, die extra für diese Gruppe angelegt werden.
- Das dritte Feature könnte eine Filterung der Objekte sein, die eine Annotation erhalten haben. Somit kann der Forschungsstand dieser Objekte leichter sichtbar gemacht werden.

Der Prototyp sollte bei einer möglichen Weiterentwicklung Tests für das Frontend und das Backend erhalten.

Das Frontend könnte mit Selenium (Tho22) getestet werden. Dieses Framework kann automatisierte Tests bei Webanwendungen durchführen. So können Webseiteneingaben auf ihre richtige Weiterverarbeitung überprüft werden.

Die Testautomatisierung des Backends könnte mittels Postman statt finden. Postman ist eine

Software zum testen von Schnittstellen. Mit diesem Tool kann man automatisierte Schnittstellentests schreiben. Der automatiserte Schnittstellentest sollte als eine Routine angelegt werden. Diese kann einmal am Tag ausgeführt werden. Dazu werden immer um 10 Uhr morgens alle Schnittstellen einmal angesprochen und mit einem Test pro Schnittstelle ausgeführt. Wenn einer dieser Tests fehlschlägt, wird eine E-mail an den Administrator versendet. Diese Tests werden gegen das ausgerollte Backend in der IT-Umgebung der SUB ausgeführt. Damit kann man überwachen, ob das Backend online ist und die Zusammenarbeit mit der Datenbank funktioniert. Die Tests schließen jedoch keine Fehlfunktionen, in bezug auf Logikfehler in der Verarbeitung der Anfragen aus. Dazu sollte man beim späteren Einsatz des Prototypen Unittests im Backend integrieren.

Abschließend kann man hoffen, dass diese Arbeit einen Prototypen hervorgebracht hat, der in einem späteren Entwicklungsprozess genutzt bzw. weiterentwickelt wird. Es besteht mit dem aktuellen Entwicklungsstand eine solide Grundlage, um Forschungen an den Digital-Humanities durchzuführen.

Literaturverzeichnis

[Bac19] Bachina, Bhargav: *How To Serve Angular Application With NGINX and Docker*. Web. https://medium.com/bb-tutorials-and-thoughts/how-to-serve-angular-application-with-nginx-and-docker-3af45be Version: 2019

[CO17] Cruzes, Daniela S.; Othmane, Lotfi b.: Threats to Validity in Empirical Software Security Research. Version: 2017. DOI 10.1201/9781315154855-10. In: Othmane, Lotfi b. (Hrsg.); Jaatun, Martin G. (Hrsg.) ; Weippl, Edgar (Hrsg.): *Empirical Research for Software Security*. CRC Press, 2017, S. 275–300. DOI 10.1201/9781315154855-10. – ISBN 9781315154855

[Con22] Consortium, IIIF: *How it works*. Web. https://iiif.io/get-started/how-iiif-works/. Version: April 2022

[Ged22a] Gedigk, Sebastian: *Dehmel-Annotation-Backend*. Web. https://app.swaggerhub.com/apis/Garry1704/Dehmel-Annotation-Backend/1.0.0-oas3. Version: 2022

[Ged22b] Gedigk, Sebastian: *dehmel-digital-mirador-backend*. Web. https://git.haw-hamburg.de/dehmel-digitalhumanities/dehmel-digital-mirador-backend. Version: 2022

[Ged22c] Gedigk, Sebastian: *dehmel-digital-mirador-frontend*. Web. https://git.haw-hamburg.de/dehmel-digitalhumanities/dehmel-digital-mirador-frontend. Version: 2022

[Ged22d] Gedigk, Sebastian: *Garry1704 / mirador-annotations*. Web. https://github.com/Garry1704/mirador-annotations/tree/addOwnServer. Version: 2022

[Ged22e] Gedigk, Sebastian: *mirador-annotations*. Web. https://www.npmjs.com/package/@garry89/mirador-annotations. Version: 2022

[Goo22] Google: *Suchaufrufe der letzten 5 Jahre zum Thema Angular, React und Vue.js.* Web. https://trends.google.com/trends/explore/TIMESERIES/1651770000?hl=de&tz=-120&cat=5&date=today+5-y&q=%2Fg%2F11c6w0ddw9,%2Fm%2F012l1vxv,%2Fg%2F11c0vmgx5d&sni=3. Version: 2022

[HNSW22] Hiesgen, Raphael; Nawrocki, Marcin; Schmidt, Thomas C. ; Wählisch, Matthias: *The Race to the Vulnerable: Measuring the Log4j Shell Incident.* http://arxiv.org/pdf/2205.02544v1. Version: 2022

[Hö14] Höher, Peter A.: *Handbuch IT-Projektmanagement : Vorgehensmodelle, Managementinstrumente, Good Practices.* Carl Hanser Verlag GmbH Co. KG https://www.hanser-elibrary.com/doi/book/10.3139/9783446441217. ISBN 978–3–446–44121–7

[JBS15] Jones, Michael; Bradley, John ; Sakimura, Nat: *JSON Web Token (JWT).* RFC 7519. DOI 10.17487/RFC7519. Version: Mai 2015 (Request for Comments)

[JSD15] Jadhav, Madhuri A.; Sawant, Balkrishna R. ; Deshmukh, Anushree: Single page application using angularjs. In: *International Journal of Computer Science and Information Technologies* 6 (2015), Nr. 3, S. 2876–2879

[Kur16] Kurz: *Digital Humanities.* Springer Fachmedien Wiesbaden, 2016. ISBN 978–3–658–11212–7

[LGH20] Lina Franken; Gertraud Koch ; Heike Zinsmeister: Annotationen als Instrument der Strukturierung. Version: 2020. DOI 10.1515/9783110689112-005. In: Nantke, Julia (Hrsg.); Schlupkothen, Frederik R. N. (Hrsg.): *Annotations in scholarly editions and research.* Berlin and Boston : De Gruyter, 2020, S. 89–108. DOI 10.1515/9783110689112-005. – ISBN 978–3–11–068911–2

[MK16] Meier, Andreas; Kaufmann, Michael: *SQL- & NoSQL-Datenbanken.* 8., überarbeitete und erweiterte Auflage. Springer Vieweg (eXamen.press). http://www.springer.com/. ISBN 978–3–662–47664–2

[ML22] MIT-License, Open-Source: *Yalc.* Web. https://www.npmjs.com/package/yalc. Version: April 2022

[NS20] Nantke, Julia (Hrsg.); Schlupkothen, Frederik R. N. (Hrsg.): *Annotations in scholarly editions and research: Functions, differentation, systematization.* De Gruyter. DOI 10.1515/9783110689112. ISBN 978–3–11–068911–2

[Oss22] Ossietzky, Staats und Universitätsbibliothek Hamburg Carl v.: *Dehmel-Archiv.* Web. https://www.sub.uni-hamburg.de/sammlungen/nachlass-und-autographensammlung/dehmel-archiv.html. Version: April 2022

[Pra17] Prasad Reddy, K. S.: *Beginning Spring Boot 2: Applications and Microservices with the Spring Framework.* Berkeley, CA : Apress, 2017 (SpringerLink Bücher). DOI 10.1007/978-1-4842-2931-6. ISBN 9781484229316

[Rit18] Ritzmann, Alexander: 1. Hintergrund:„Das ABC des Problemkomplexes Hassrede, Extremismus und NetzDG “. In: *Hassrede und Radikalisierung im Netz* (2018), S. 11

[Sta20] Starke, Gernot: *Effektive Softwarearchitekturen: Ein praktischer Leitfaden.* 9., überarbeitete Auflage. Hanser (Hanser eLibrary). DOI 10.3139/9783446465893. ISBN 978–3–446–46589–3

[Sta21] Statista: *Share of users worldwide accessing the internet in 3rd quarter 2021, by device.* Web. https://www.statista.com/statistics/607716/worldwide-artificial-intelligence-market-revenues/. Version: 2021

[Tes21] Teszelszky, Kees: Introduction: digital humanities and the use of web archives. DOI 10.1007/s42803-021-00040-5. In: *International Journal of Digital Humanities* 2 (2021), Nr. 1-3, S. 1–4. ISSN 2524–7832

[Tho22] ThoughtWorks: *selenium.* Web. https://www.selenium.dev/. Version: Mai 2022

[W3C17] W3C: *Web Annotation Data Model.* Web. https://www.w3.org/TR/annotation-model/. Version: 2017

[Wei16] Weisse, Bengt: *AngularJS & Ionic Framework: Hybride App-Entwicklung mit JavaScript und HTML5.* Hanser (Hanser eLibrary). DOI 10.3139/9783446448070. ISBN 9783446448070

[Wik21a] Wikipedia: *Ida Dehmel.* Web. https://de.wikipedia.org/wiki/Ida_Dehmel. Version: November 2021

[Wik21b] Wikipedia: *Richard Dehmel*. Web. `https://de.wikipedia.org/wiki/Richard_Dehmel`. Version: November 2021

[WMW20] Winters, Titus; Manshreck, Tom ; Wright, Hyrum: *Software engineering at Google: Lessons learned from programming over time*. First edition. Beijing China : O'Reilly Media, 2020. ISBN 9781492082798

Hiermit versichere ich, Sebastian Gedigk, dass ich die vorliegende Bachelorarbeit mit dem Thema:

Anforderungsanalyse und Konzeption eines Annotationsdienstes für ein Digital-Humanities-System

ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

Hamburg, 19. Mai 2022 Sebastian Gedigk