

Universität Siegen
Naturwissenschaftlich Technische Fakultät
Lehrstuhl Betriebssysteme und verteilte Systeme



Bachelorarbeit zum Thema :

**Konzeptionierung und Entwicklung
einer Web-Applikation zur Digitalisierung
der Arbeitsprozesse einer sozialen Organisation**

**Conception and development of a web application for
digitizing the work processes of a social organization**

Zur Erlangung des Grades
Bachelor of Science (B. Sc.)

Vorgelegt von :

Mohammed Ali Anis
Glückaufstr. 52
57076, Siegen
NRW, Deutschland
Email : mohammed.anis@student.uni-siegen.de

Matrikelnummer : 1237412
Studiengang : Informatik

Abgabe : Siegen, den 29.09.2021
Erstgutachter : Prof. Dr. Roland Wismüller
Zweitgutachter : AR Dr.-Ing. Andreas Hoffmann

Selbstständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Alle nicht eigenen Gedanken und Visualisierungen sind als solche gekennzeichnet.

Diese Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und nicht anderweitig veröffentlicht. Ich habe diejenigen Paragraphen der für mich geltenden Prüfungsordnung, welche etwaige Betrugs- und Täuschungsversuche betreffen, zur Kenntnis genommen.

Der Speicherung meiner Studienarbeit zum Zweck der Plagiatsprüfung stimme ich zu. Ich versichere, dass die elektronische Version dieser Arbeit mit der gedruckten Ausgabe übereinstimmt.

Siegen, den 29.09.2021
Mohammed Ali Anis



Independent declaration

I hereby assure that I wrote the present work independently and that I have not used any other sources and aids than those given. All thoughts and visualizations that are not your own are marked as such.

This work has not been submitted to any other examination authority in the same or a similar form and has not been published elsewhere. I have taken note of those paragraphs of the examination regulations that apply to me, which concern any attempts at fraud or attempted deception.

I agree to the storage of my thesis for the purpose of plagiarism check. I guarantee that the electronic version of this work corresponds to the printed version.

Siegen, den 29.09.2021
Mohammed Ali Anis





Danksagung

An dieser Stelle möchte ich mich bei all denjenigen bedanken, die mich während der Anfertigung dieser Bachelorarbeit unterstützt und motiviert haben.

Zuerst gebührt mein Dank Herrn **Dr. Ing. Andreas Hoffmann** und Herrn **Prof. Dr. Roland Wismüller**, die meine Bachelorarbeit betreut und begutachtet haben. Für die hilfreichen Anregungen, die konstruktive Kritik und die vielen inspirierenden Ideen bei der Erstellung dieser Arbeit möchte ich mich herzlich bedanken.

Ich möchte mich bei meinen Eltern **Ing. Abdulbaseet Anis & Math. Nada Kassab**, meiner Schwester **Haya** und meinem Bruder **Ehab** bedanken, die mich immer besonders liebevoll unterstützten und stets ein offenes Ohr für meine Sorgen hatten. Ohne sie wäre das Studium für mich unmöglich gewesen.

Kurzfassung

Diese Arbeit befasst sich mit der Konzipierung, Implementierung und dem Design einer Web-Applikation mit dem Ziel, den Arbeitsprozess einer sozialen Organisation zu digitalisieren und damit zu erleichtern und besser zu organisieren.

Hierzu wird das Spring Framework in Verbindung mit HTML, CSS, JavaScript, Datenbank SQLite und verschiedene API verwenden.

Dieses Projekt kann als Vorlage für die meisten sozialen Organisationen verwendet werden, aber im Fokus stand die Erfüllung der Anforderungen der Arbeitsprozesse des Vereins für Soziale Arbeit und Kultur Südwestfalen VAKS in Siegen.

Keywords

Java, Spring Boot Framework, Google APIs, Google Calendar, Webanwendung, MVC, VAKS Siegen.

Inhaltsverzeichnis

Selbstständigkeitserklärung	I
Danksagung	II
Kurzfassung	III
Inhaltsverzeichnis	V
1 Einleitung	1
1.1 Beschreibung des Vereins	1
1.2 Ausgangssituation	2
1.3 Ziel der Arbeit	3
2 Grundlegende Software und Konzepte	4
2.1 Webapplikation	4
2.2 Client-Technologien	4
2.2.1 HTML	5
2.2.2 CSS	5
2.2.3 JS	6
2.3 Backend	6
2.3.1 Web Server Applikation	6
2.3.2 Application Logic - Business Logic	7
2.3.3 Datenbank	8
2.3.4 Application Programming Interface (API) und Framework	9
2.3.4.1 Spring Boot	9
2.3.4.2 Google Calendar API	10
2.3.4.3 Google Charts	10
2.3.4.4 JavaMail API	11
2.3.4.5 Google Dialog Flow	11
3 Konzept	12
3.1 Status Quo Prozesse	12
3.2 Anforderungsanalyse	13
3.2.1 Digitalisierung aller Dokumente	13
3.2.2 Hochladen der Dokumente Online	13
3.2.3 Elektronischer Kalender	14
3.2.4 Online Terminbuchungssystem	14
3.2.5 Statistiken des Vereins	15
3.2.6 Chatbot	16

3.3	Design	17
3.3.1	Wireframe	17
3.3.2	Architektur	20
3.3.3	Datenbank	22
4	Entwicklung und Realisierung	25
4.1	Projektstruktur	25
4.2	Berechtigungen und Befugnisse	27
4.3	Controllers	30
4.4	Models	36
4.5	Views	38
4.6	Google Calendar API	39
4.7	Google Charts	41
4.8	Google Dialog Flow	43
4.9	Fazit	45
5	Installation und Test	46
5.1	Installation	46
5.2	Test	47
6	Zusammenfassung und Ausblick	50
6.1	Fazit	50
6.2	Ausblick	51
7	Anhang	52
7.1	Dependencies	52
7.2	Datenbank (SQL)	53
7.3	Application Properties	56
	Literaturverzeichnis	VI
	Abbildungsverzeichnis	VIII
	Tabellenverzeichnis	IX
	Abkürzungsverzeichnis	X

1 Einleitung

„Aufgrund der besseren Lesbarkeit wird in dieser Arbeit das generische Maskulin verwendet.“

Mit dem folgenden Kapitel soll eine Einführung in Thema und Kontext der Arbeit gegeben werden. Zudem wird das Problem beschrieben, die Anforderungsanalyse und das Ziel festgelegt.

1.1 Die Beschreibung des Vereins

Der Verein für Soziale Arbeit und Kultur Südwestfalen (VAKS) wurde im Jahr 1985 gegründet. Der Wunsch zur aktiven Mithilfe an der Entwicklung einer sozial gerechten und multikulturellen Gesellschaft war damals schon wie auch heute noch das Hauptmotiv für das Engagement. Den Schwerpunkt der Arbeit in verschiedenen sozialen und kulturellen Bereichen bildet dabei die Hilfe und Unterstützung von Zuwanderern und Menschen ohne deutschen Pass bei dem schwierigen Prozess der Integration in unsere Gesellschaft. Die Arbeit mit Kindern und Jugendlichen steht ebenfalls im Zentrum der Arbeit. Der Verein ist zudem für mehrere offene Ganztagsgrundschulen und die verlässliche Betreuung an Grundschulen tätig [VAKS21] .

Der Verein sieht seine Arbeit sowohl als praktisch, unterstützend und beratend, als auch als gesellschaftspolitisch an. Missbräuche und Ungerechtigkeiten sollten auf allen Ebenen der Gesellschaft angegangen und zur Veränderung herangezogen werden.

1.2 Ausgangssituation

Der Verein befasst sich heute mit mehreren Themen, darunter der Einwanderung, der Bekämpfung ethnischer Diskriminierung sowie der Unterstützung von Kindern und der psychologischen und moralischen Unterstützung. Der Verein bietet auch Schul- und Sprachförderung für Kinder mit Zuwanderungsgeschichte an.

Nach der Flüchtlingskrise in den Jahren 2015 und 2016, und infolge des Drucks waren die Lebensbereiche im Allgemeinen einschließlich des Vereins betroffen, wobei der technische und technologische Aspekt des Vereins vernachlässigt wurde.

Diese Vernachlässigung verursachte eine gewisse Belastung für den Arbeitsablauf der Organisation. Der Vorgang der Terminbuchung und Terminvergabe ist sehr zeit- und fehleranfällig. Auch die Akten der Besucher des Vereins werden sind schwer zugänglich und immer noch in Papierform und nicht digitalisiert gespeichert.

Falls der Kunde ein bestimmtes Papier oder Formular zum Ausfüllen benötigt, muss es ihm zunächst per Post zugesandt werden, was eine zwei- bis dreitägige Wartezeit des Kunden erfordert.

Wenn der Kunde nur eine Auskunft braucht, muss er das telefonisch erledigen. Das kann dann nur während der Sprechzeiten erfolgen.

1.3 Ziel der Arbeit

Das Ziel der Arbeit ist die Entwicklung, Implementierung und Design einer einheitlichen Software Lösung zur Organisation und Digitalisierung des Vereins. Mit Hilfe der Software sollen vor allem die Anforderungen der verschiedenen Benutzergruppen, wie die Mitarbeiter und die Freiwilligen des Vereins und auch die Personen, die Unterstützung erhalten, zufrieden gestellt werden.

Diese Arbeit zielt darauf ab, die interne Arbeit des Vereins zu erleichtern und zu automatisieren sowie die Kommunikation zu den Besuchern und den Mitarbeitern des Vereins zu erleichtern.

Die Hauptpunkte, auf die sich die Arbeit insbesondere konzentrieren wird, sind:

- Erstellung einen elektronischen Kalender für die Mitarbeiter des Vereins.
- Entwicklung eines elektronischen Terminbuchungssystems.
- Verbesserung des Prozesses des Versendens von Dateien zwischen Kunden und Mitarbeitern des Vereins
- Darstellung von Statistiken und Infografiken zum Verein
- Integration eines Chatbot zur Beantwortung häufig gestellter Fragen.

2 Grundlegende Software und Konzepte

In diesem Kapitel werden die grundlegenden Konzepte der Technologien, die in dieser Arbeit verwendet werden, näher erläutert. Es wird ein allgemeiner Überblick über die Mechanismen und die Technologien erfolgen, die in dieser Arbeit verwendet werden.

2.1 Webapplikation

Webanwendung ist eine Client-Server-Anwendung, bei der ein Browser als Client und ein Webserver als Server fungiert. Die Webanwendungslogik wird zwischen Client und Server verteilt, die Datenspeicherung erfolgt hauptsächlich auf dem Server. Der Datenaustausch über das Netzwerk erfolgt mittels Hypertext Transfer Protocol (HTTP). Einer der Vorteile dieses Ansatzes besteht darin, dass die Benutzer nicht von einem bestimmten Betriebssystem oder einer bestimmten Hardwarekonfiguration abhängig sind. Webanwendungen sind daher plattformübergreifende Dienste.

2.2 Client-Technologien

Frontend-Entwicklung ist ein Prozess der Erstellung eines öffentlichen Teils einer Website, der in direktem Kontakt mit dem Benutzer steht. Niemand kann sagen, dass es bei der Frontend-Entwicklung nur darum geht, die Website hübsch und ansprechend bzw. auch schön zu gestalten [GurDn21]. Es ist definitiv ein sehr wichtiger Teil des Entwicklungsprozesses, aber es gibt viele verschiedene Technologien, die in den Anwendungsbereich der clientseitigen Entwicklung fallen[GurDn21]. Die Kernwerkzeuge, die dabei verwendet werden, sind: HyperText Markup Language (HTML), Cascading Style Sheets (CSS) und JavaScript (JS).

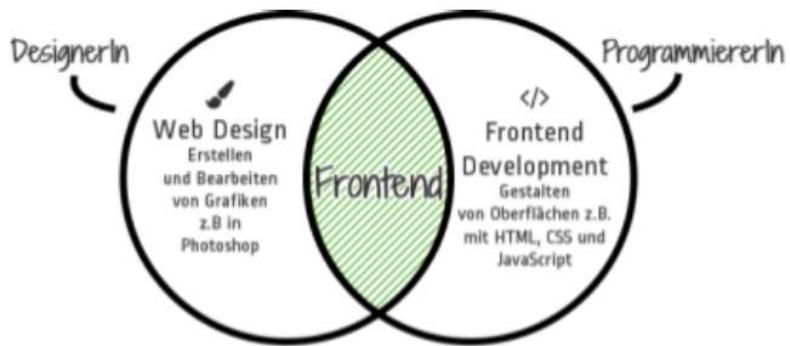


Abbildung 2.1. : Front End vs Web Design

Quelle : Frontend GmbH - https://www.frontend-gmbh.de/wp-content/uploads/2016/07/frontend_mix.png

2.2.1 HTML

HTML ist die Grundlage einer Webseite. Es ist eine Auszeichnungssprache, die die Gesamtstruktur der Seite definiert [GurDn21]. Mithilfe von Elementen oder Tags wie Liste, Tabelle, Überschrift kann der Entwickler verschiedene Teile des Inhalts kennzeichnen [W3S21]. Die HTML-Sprache wird von Browsern interpretiert und dann wird der aus der Interpretation resultierende formatierte Text auf einem Computerbildschirm oder einem mobilen Gerät angezeigt.

2.2.2 CSS

CSS ermöglicht es Webdesignern und Entwicklern, jede in HTML definierte Komponente zu stylen und abzustimmen. CSS sollte in erster Linie die Trennung des Seiteninhalts vom Seitenstil erzwingen, einschließlich Funktionen wie Schriftarten, Farben und Layout. Durch diese Trennung können mehrere HTML-Dokumente den in einer separaten CSS-Datei angegebenen Stil gemeinsam nutzen.

2.2.3 JS

JavaScript ist das Rückgrat für jede dynamische und interaktive Funktionalität auf der Webseite. Es ist eine leichtgewichtige und interpretierte Programmiersprache [TutPnt21]. Es kann beispielsweise verwendet werden, um die Gültigkeit der vom Benutzer eingegebenen Daten zu überprüfen oder die Struktur einer Webseite basierend auf den eingefangenen benutzerinitiierten Ereignissen wie Mausklicks zu ändern. Diese Technik kann verwendet werden, um alle erforderlichen Daten an einen Server zu senden und von diesem zu empfangen, ohne die Webseite zu aktualisieren.

2.3 Backend

Das Backend ist das Bindeglied zwischen den Datenbanken und dem Browser. Die Backend-Aufgaben sind die logische Funktionsweise der Seite sowie die Datenverarbeitung. Diese Art der Webentwicklung besteht normalerweise aus den Drei Teilen Server, Anwendung und Datenbank [CrsRpt21].

2.3.1 Web Server Applikation

Webserver-Software ist ein Programm, das serverseitig läuft und Daten für die Clients bereitstellt, die normalerweise im Browser dargestellt werden. Webserver-Software besteht aus mehreren Teilen. Der Kern ist jedoch ein HTTP-Server [CrsRpt21]. Es ist eine Software, die weiß, was ein Uniform Resource Locator (URL) ist und das HTTP-Protokoll versteht [NtCrft21]. Das Grundprinzip der Client-Server-Kommunikation über HTTP lässt sich anhand der folgenden Abbildung demonstrieren.

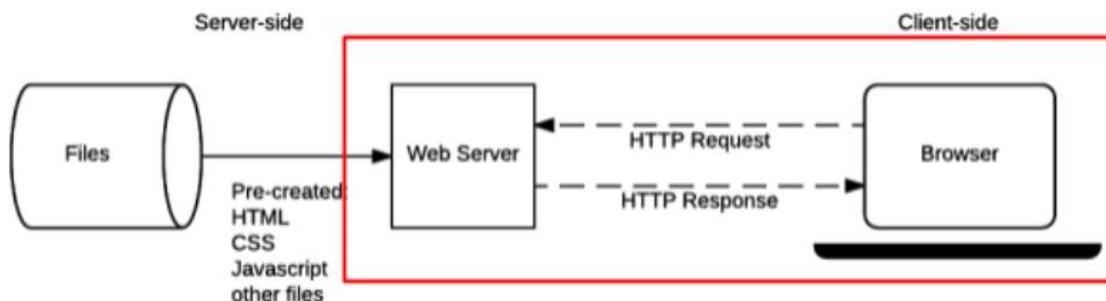


Abbildung 2.3.1 : Kommunikation zwischen Server und Client
Quelle : https://developer.mozilla.org/en-US/docs/Learn/Server-side/First_steps/Client-Server_overview/basic_static_app_server.png

Wie die Abbildung zeigt, sendet der Browser die Anfrage, erreicht dann die dedizierte Hardware und anschließen sendet der Server eine Antwort mit den angefragten Informationen. Bei Verwendung des HTTP-Protokolls werden immer bestimmte Regeln befolgt [MSDN21]. Zunächst einmal kann der Server nur auf die Anfrage antworten, die vom Client gesendet wurde. Er kann keine Anfragen an den Browser senden. Zweitens muss der Server auf jede eingehende Anfrage eine Antwort senden, zumindest eine Antwort, die eine Fehlermeldung enthält [MSDN21]. Im Allgemeinen erhält der Client eine Timeout-Meldung, die darauf hinweist, dass der Server nicht innerhalb der angegebenen Zeit geantwortet hat. Schließlich muss jede HTTP-Anfrage eine URL-Adresse enthalten, die den jeweiligen Server und Pfad angibt, an wen diese Anfrage gesendet werden soll [MSDN21].

2.3.2 Application Logic - Business Logic

Hier werden die benötigten und gesendeten Daten verarbeitet. Diese Schicht kann programmgesteuert durch viele und unterschiedliche Programmiersprachen erstellt werden, wie z.B. PHP, Java oder Python. Viele Webapplikationsframeworks wurden bereitgestellt und sind mittlerweile weit verbreitet.

Diese Frameworks erleichtern die Arbeit des Programmierers, z. B. das Koordinieren der Ausgabe und das Verwalten von Sitzungen und erleichtert

ebenso die Einrichtung der Serverkonfiguration. Daraus schließen wir, dass die Application Logic eine der wichtigsten Teile des Backend ist.

2.3.3 Datenbank

In Webanwendungen werden Daten normalerweise in Datenbanken kontrolliert und gespeichert. Die Datenbank ist eine integrierte Sammlung von logisch zusammenhängenden Datensätzen oder Dateien, die zu einem gemeinsamen Pool konsolidiert werden, der Daten für eine oder mehrere Mehrfachverwendungen bereitstellt [MoAy17].

Relationale Datenbanken verwalten Daten in Tabellen und bieten eine effiziente, intuitive und flexible Möglichkeit, strukturierte Informationen zu speichern und darauf zuzugreifen. Tabellen, auch als Relationen bezeichnet, bestehen aus Spalten, die eine oder mehrere Datenkategorien enthalten, und Zeilen, auch als Tabellendatensätze bezeichnet, die einen durch die Kategorie definierten Datensatz enthalten. Anwendungen greifen auf Daten zu, indem sie Abfragen angeben, die Operationen wie Projekt verwenden, um Attribute zu identifizieren, Auswahl zum Identifizieren von Tupeln und Joins zum Kombinieren von Beziehungen. Die derzeit am häufigsten verwendeten Datenbanken sind: Oracle, MySQL, PostgreSQL, und SQLite.

In unserem Projekt wurde SQLite als Datenbank verwendet. Die Vorteile dabei sind [SqLe21] :

- SQLite ist eine sehr einfach zu bedienende Datenbank
- Lese- und Schreibvorgänge sind bis zu 35 % schneller als auf dem Dateisystem.
- SQLite lädt nur die benötigten Daten, anstatt die gesamte Datenbank zu lesen und im Speicher zu halten.
- SQLite ist sehr einfach zu erlernen. Eine Installation und Konfiguration ist nicht notwendig. Man lädt einfach die

SQLite-Bibliotheken auf den Computer herunter und dann kann die Datenbank erstellen werden.

Als großer Nachteil von SQLite ist die schlechte Performance zu nennen, die bei sehr vielen Lese-/Schreibvorgängen auftritt.

2.3.4 Application Programming Interface (API) und Frameworks

2.3.4.1 Spring Boot

Spring Boot bietet Java-Entwicklern eine gute Plattform, um eine eigenständige und produktionstaugliche Webanwendung zu entwickeln, die einfach ausgeführt werden kann. Es kann mit minimalen Konfigurationen begonnen werden, ohne eine komplette Spring-Konfiguration einrichten zu müssen. Spring Boot konfiguriert die Anwendung automatisch basierend auf den Abhängigkeiten, die im Projekt hinzugefügt wurde, indem Sie die Annotation `@EnableAutoConfiguration` verwendet wird.

Der Einstiegspunkt der Spring-Boot-Anwendung ist die Klasse, die die Annotation `@SpringBootApplication` und die Main-Methode `main()` enthält.

2.3.4.2 Google Calendar API

Die Google Calender-API ist eine RESTful-API, auf welche über explizite HTTP- Aufrufe oder über die Google-Clientbibliotheken für die Sprachen Java, Python, PHP und Javascript zugegriffen werden kann. Die API stellt die meisten der in der Google Calendar Weboberfläche verfügbaren Funktionen bereit.

Der Google Calender bietet eine Vielzahl von Funktionen zur persönlichen Terminverwaltung.

Die Anfrage jeder Anwendung wird an die Calendar API gesendet und benötigt ein Autorisierungs-Token. Dieses Token identifiziert auch die Anwendung für Google. Die Anwendung muss OAuth 2.0 verwenden, um Anfragen zu autorisieren. Andere Autorisierungsprotokolle werden nicht unterstützt.

Die Google Calendar API ist kostenlos wenn weniger als 1 Million Abfragen pro Tag gemacht werden. Die meisten Anwendungen brauchen diese Menge an Abfragen nicht [[GogCa21](#)].

2.3.4.3 Google Chart

Die Verwendung von Google Charts bietet den Websites eine sehr schöne Möglichkeit. Diagramme anzuzeigen und zu visualisieren.

Die gebräuchlichste Art Google Charts zu verwenden, ist mit einfachem JavaScript, das in eine Webseite eingebettet wird. Indem Sie einige Google Chart-Bibliotheken laden, die anzuzeigenden Daten auflisten und Optionen zum Anpassen des Diagramms auswählen und schließlich ein Diagrammobjekt mit einer von Ihnen gewählten ID zu erstellen. Nachfolgende Erstellen Sie auf der Webseite einfach ein `<div>` mit dieser ID, um das Google-Diagramm anzuzeigen. Alle Diagrammtypen werden mithilfe der

DataTable-Klasse mit Daten gefüllt, Der DataTable bietet Sortiermethoden und kann direkt von der Datenbank ausgefüllt werden [GogCh21].

2.3.4.4 JavaMail API

Java verfügt über eine dedizierte Bibliothek für E-Mail-Nachrichten mit dem Namen JavaMail. JavaMail ist eine API, die “ein plattform- und protokollunabhängiges Framework zum Erstellen von Mail- und Messaging-Anwendungen bereitstellt” [Ora21].

Mit der JavaMail-API können Dokumente auch angehängt werden oder den CC angeben, um E-Mails an mehrere Personen zu senden. Der Betreff kann individuell geschrieben werden, da diese API praktisch die gleiche Flexibilität beim Senden von E-Mails bietet und die gleichen bekannten Dienste.

2.3.4.5 Google Dialog Flow

Dialogflow ist ein Entwicklungsframework für Google Chatbots. Es kann verwendet werden, um Konversationsschnittstellen für Websites, mobile Anwendungen, Messaging-Plattformen, oder IoT-Geräte zu erstellen und ist für Google Assistant optimiert.

Dialogflow bietet eine One Click Integration mit den meisten gängigen Messaging-Plattformen wie Whatsapp, Facebook und Twitter [GogCld21].

3 Konzept

In diesem Abschnitt werden die gemeinsamen und allgemeinen Anforderungen der sozialen Organisationen analysiert, die durch die Webanwendung umzusetzen sind.

3.1 Status Quo Prozesse

Die Arbeit des Vereins hängt von „traditionellen Methoden“ ab. Der Verein setzt bislang nur emails und Telefon als technische Unterstützung ein.

Außerdem werden die Termine für interne Besprechungen und die Termine für Sitzungen für Personen, die eine Beratung benötigen, in einem handschriftlichen Papierkalender auf dem Schreibtisch jedes Mitarbeiters oder Freiwilligen im Verein eingetragen.

Der Prozess der Buchung bzw. Vereinbarung von Terminen zwischen Mitarbeitern des Vereins und Personen, die Beratung benötigen, basiert also auf einer sehr „traditionellen Methode“, wie bspw. per Telefonanruf und im besten Fall per E-Mail.

Darüber hinaus werden alle speziellen Formulare wie Anmeldeformulare, Jahresurlaubsantragsformulare und Protokolle zwischen Verein und Personen auf herkömmliche Weise bearbeitet, indem der jeweilige Antrag ausgedruckt und mit einem Stift ausgefüllt wird.

Alle Unterlagen des Vereins und der Kunden sind in Ordnern gestapelt und in den Regalen gelagert. Sie werden allerdings nicht digital archiviert, was im Falle von Feuer oder bei Wasserschäden dazu führt, dass alle Dokumente verloren gehen können. Es macht auch den Prozess der Suche eines bestimmten Dokuments innerhalb aller Papiere zu einem schwierigen und zeitaufwendigen Prozess.

Termine werden von den Kunden auch gebucht, um vor allem Fragen zu stellen, die in der Regel in vielen Fällen gleich beantwortet werden, wie zum Beispiel „Wie viel kostet das ÖPNV-Ticket in Siegen?“ oder „Wie hoch ist die monatliche Gebühr der Rundfunkbeitrag von ARD, ZDF und Deutschlandradio?“ oder „Was sind die Voraussetzungen für eine dauerhafte Aufenthaltserlaubnis?“

3.2 Anforderungsanalyse

3.2.1 Digitalisierung aller Dokumenten

Die Digitalisierung von Dokumenten ist eine wesentliche Methode für ein papierloses Büro und bietet entscheidende Vorteile, wie z.B. die Überall-Verfügbarkeit, Zugriffsberechtigungen und die Archivierung. Gleichzeitig verstärkt sie die Teamarbeit, in dem sie Dokumente freigeben und mehreren Benutzern gleichzeitig Zugriff darauf gewähren kann.

Ein Szenario welches sich in der Vergangenheit ereignet hat ist besonders zu erwähnen. Es hat uns gezeigt, wie wichtig es ist, Dokumente zu digitalisieren. Aufgrund der Umstände der COVID-19 Pandemie, die wir zum Zeitpunkt der Arbeit immer noch durchleben, wurden die meisten Mitarbeiter des Vereins unter Quarantäne gestellt bzw. ins Home-Office geschickt. Das machte den Zugriff auf die Unterlagen sehr schwierig bis unmöglich.

3.2.2 Hochladen der Dokumente Online

Die Applikation ermöglicht den Mitarbeitern, und im gleichen Fall den Kunden des Vereins, Dokumente elektronisch hochzuladen. Dies spart Zeit und Ärger, insbesondere bei der Arbeit von zu Hause aus (Home-Office). Anstatt das Formular per Post an die betroffene Person zu senden und darauf zu warten, dass es ausgefüllt und zurück gesendet wird, kann das Formular direkt elektronisch ausgefüllt werden und an einen Bearbeiter geschickt werden.

3.2.3 Elektronischer Kalender

Der Papierkalender wird ersetzt und ein allgemeiner elektronischer Kalender für alle Mitarbeiter und Freiwilligen erstellt, in dem die Daten der offiziellen Feiertage, Sitzungen und allgemeinen regelmäßigen Ereignisse angegeben sind. Parallel dazu ein persönlicher elektronischer Kalender für jede Person in dem Verein, über den Termine organisiert werden. Mit dem elektronischen Kalender wird die Planung erleichtert, z.B. wenn ein Termin verschoben werden muss. Der Termininhaber wird elektronisch über den Vorgang informiert, sodass Zeitverschwendungen durch telefonische Information vermieden wird.

3.2.4 Online Terminbuchungssystem

Wie bereits erwähnt, arbeitet der Verein mit Menschen mit Migrationshintergrund, und viele von ihnen beherrschen die deutsche Sprache nicht gut, was leider häufig zu Missverständnissen beider Parteien führen kann.

Es kommt z.B vor, dass die Person, die einen Termin benötigt, das Datum und die Uhrzeit des Termins telefonisch nicht richtig verstehen kann. Dies ist etwas, was als dringend verbesserungswürdig angesehen wird.

Die derzeit geltende traditionelle Methode für Terminbuchungen wird durch eine elektronische Methode zur Buchung und Stornierung von Terminen ersetzt. Diese Methode ermöglicht die Buchung von Terminen, auch außerhalb der Öffnungszeiten und an Feiertagen.

Eine Bestätigungsnachricht wird an beide Parteien bezüglich des Reservierungs- oder Stornierungsprozesses gesendet.

Wenn der Termin online gebucht wird, wird der Termin automatisch im oben erläuterten elektronischen Kalender angezeigt, dadurch die Erinnerungsfunktion aktiviert wird.

Dieser Mechanismus erspart den Mitarbeitern viel Zeit und erleichtert es den Kunden einen Termin zu bekommen.

3.2.5 Statistiken des Vereins

Fast alle Organisationen verwenden Statistiken um damit Fördergelder bzw. Unterstützungen und Spenden zu beantragen. Aber auch um entsprechende Probleme und Verbesserungen innerhalb der Organisation zu erkennen und dann umzusetzen.

Beim elektronischer Speichern der Besucherinformationen des Vereins in der Datenbank wie z. B. Geschlecht, Alter, Heimatland und Bildungsstand, ist es hilfreich mittels Diagrammen einige Bezüge aufzuzeigen. z.B das Verhältnis von Männern zu Frauen, die Art des Aufenthaltes (Niederlassungserlaubnis, Flüchtlingseigenschaft, subsidiärer Schutz, Abschiebeverbot oder Duldung). Auch in Bezug auf das Herkunftsland zu ermitteln.

Daten können mithilfe von SQL Abfragen einfach erfasst und in Diagrammen dargestellt werden.

Der Verein bietet Freiwilligen und Mitarbeitern in Kooperation mit der Refugee Law Clinic Siegen (RLC)* jedes Jahr Weiterbildungsmöglichkeiten zu verschiedenen Themen, die den Personen während der Arbeit im Verein begegnen können. Anhand der Statistiken, die dieses Projekt liefern wird, ist es möglich, die Themen und Schwerpunkte, an denen der Verband aktuell Tagen arbeitet, zu erkennen und dementsprechend die Hauptrichtung der vorgestellten Workshops bzw. der Weiterbildungen festzulegen.

3.2.6 Chatbot

Für den Verein wird ein Chatbot erstellt, um die Mitarbeiter des Vereins von wiederholten Fragen zu entlasten.

Einige Menschen buchen in der aktuellen Situation einen Termin telefonisch, nur um sich nach etwas zu erkundigen. Die Antwort ist in vielen Fällen dieselbe.

Beispiele hierzu sind:

- “Wie bekomme ich ein monatliches Fahrticket für öffentliche Verkehrsmittel”
- “Wie viel kostet das Fahrticket?
- “Was sind die allgemeinen Voraussetzungen für eine dauerhafte Aufenthaltserlaubnis in Deutschland?”

Diese und weitere ähnliche Anfragen benötigen viel Zeit der Mitarbeiter und oftmais hindern sie daran, sich auf wichtige Fälle zu konzentrieren, an denen gearbeitet werden muss.

Ein Szenario: Ein Flüchtling aus Syrien, der eine Aufenthaltserlaubnis nach der Flüchtlingseigenschaft (§ 25 Abs. 1 bis 3 AufenthG) besitzt und sich in Deutschland dauerhaft aufhalten möchte. Er wird durch den Chatbot über die allgemeinen Bedingungen für den Erhalt der Aufenthaltserlaubnis informiert.

Im zweiten Fall wird einem Asylbewerber aus Syrien mit einer Duldung (§ 60a AufenthG) vom Chatbot mitgeteilt, dass er mit seiner aktuellen Aufenthaltsform keinen Antrag stellen kann.

3.3 Design

3.3.1 Wireframe

Das Programm Balsamiq wurde für die Wireframes verwendet.

Nach Fertigstellung des Designs und bei der Umsetzung des Projekts können weitere Verbesserungen und Modifikationen an der äußereren Form, den Farben und Formen der Schaltflächen, sowie an den Projekteigenschaften vorgenommen werden.

The wireframe shows a web interface for managing users. On the left, there is a sidebar with links: 'Manage User' (highlighted), 'Calendar', 'Manage Event', 'Document', and 'Statistics'. The main content area has a header 'VAKS | Siegen' with a search bar and navigation links: Home, About Us, Welcome, Service, Gallery, Contact Us, Profile, and Login. Below the header is a table titled 'Manage User' with columns: ID, First Name, Last Name, Country, Role, and Delete. The table contains four rows of data. Below the table is a 'New User' form with fields: First Name (text input with value 'Alex'), Last Name (text input with value 'Blosser'), Sex (checkboxes for Male and Female, Female is checked), Country (dropdown menu with value 'Alberta'), and DSH I or DSH II (radio buttons for Yes and No, Yes is selected). A 'Submit' button is at the bottom right of the form.

Abbildung 3.3.1 a : Manager | Manage User

In der Abbildung 3.3.1a erscheint die Oberfläche zur Verwaltung von Benutzerkonten, da sie Informationen zu den Benutzern anzeigt, einschließlich ihres Vornamens, Nachnamens, Geburtsdatums, ihrer Rolle und ihres Landes.

Es ist auch möglich, jeden Benutzer zu löschen oder zu bearbeiten.

Durch Drücken des "Add User" Buttons wird innerhalb der Oberfläche ein Bereich angezeigt, in dem Felder ausgefüllt werden können, um einen neuen Benutzer anzulegen.

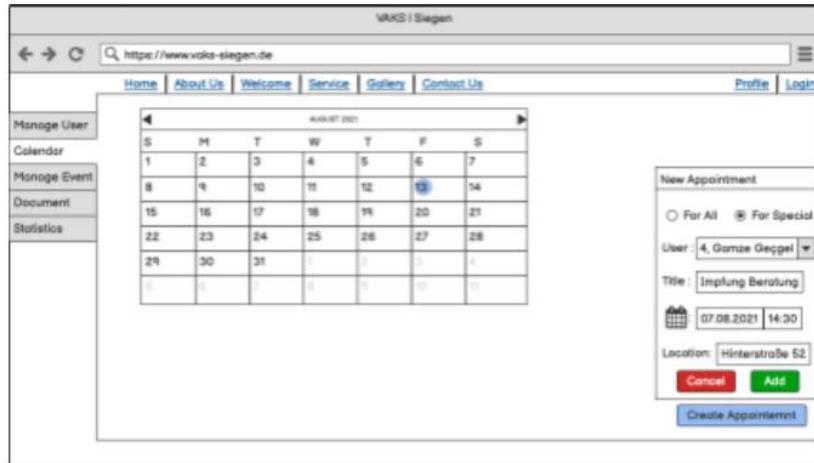


Abbildung 3.3.1 b : Manager | Calendar

Die Abbildung 3.3.1b zeigt das Interface, in dem der elektronische Kalender angezeigt wird. Durch Drücken des “Create Appointment” Buttons erscheint das Popup Interface, in dem Sie zusätzlich zu Uhrzeit, Datum und Ort auch den Terminnamen schreiben können, ob das Event von allen Mitarbeitern des Vereins geteilt wird oder ob es sich um einen bestimmten Benutzer handelt. Dies kann über den Radio Button gesteuert werden.

ID	Event Title	Date	Time	User	Delete	Edit	
1	Implfung Beratung	07.08.2021	14:30	–	4. Gomze Geppel	<input type="button" value="Delete"/>	<input type="button" value="Edit"/>
13	Ausländerbehörde	18.08.2021	09:30	–	9. Tareq Al-Majoni	<input type="button" value="Delete"/>	<input type="button" value="Edit"/>

Abbildung 3.3.1 C : Manager | Manage Event

Der Manage Event Tab, wo alle Events in einer Tabelle, die neben dem Termininhaber auch den Namen, das Datum und die Uhrzeit des Events angezeigt werden, ist in der Abbildung 3.3.1c zu sehen. Hier können auch die Events gelöscht oder bearbeitet werden. Der Filtervorgang kann durch Eingabe des Datums an dem der Benutzer die Events sehen möchte, oder durch Auswahl der ID des Benutzers erfolgen, und dann können alle seine Events angezeigt werden.

The screenshot shows a web application interface for managing documents. On the left, there is a sidebar with links: 'Manage User', 'Calendar', 'Manage Event', 'Document', and 'Statistics'. The 'Document' link is highlighted. The main content area has a header 'VAKS i Siegen' and a search bar 'Search by User: 4. Gomze Geppel'. Below the search bar is a table with columns: ID, Name, Language, Belong To, Download, and Delete. One row is visible: ID 3, Name 'Impfung Antrag', Language 'ENG', Belong To '4. Gomze Geppel', Download 'Click Here', and Delete 'Delete'. Below the table is a form titled 'Add Document': Title 'BAföG Antrag', Language 'DE', Type 'Public Documents', and For User dropdown. A 'Submit' button is at the bottom right of the form.

Abbildung 3.3.1 D : Manager | Document

Ähnlich zu dem Manage User Tab ist der Document Tab, Hier können hochgeladene Dokumente eingesehen, heruntergeladen und gelöscht werden.

Von dieser Seite aus ist es auch möglich, eine Datei hochzuladen, und es kann festgestellt werden, ob diese Datei für einen bestimmten Benutzer bestimmt ist oder ob sie öffentlich und für jedermann zugänglich ist.



Abbildung 3.3.1 e : Manager | Statistics

In der Abbildung 3.3.1e ist der Statistics Tab gezeigt, Dieser Tab erscheint nur, wenn die Rolle des angemeldeten Benutzers Manager ist, Hier werden alle Statistiken gezeigt.

3.3.2 Architektur

In dieser Arbeit wurde Spring Boot Framework verwendet, um die Webanwendung zu erstellen. Dieses Framework ist eine Erweiterungen für Spring MVC, was bedeutet, dass es kommt auch auf die MVC-Architektur ankommt. Der Begriff MVC ist ein Akronym für das Konzept Model - View - Controller und ist ein Softwaredesign Muster, Dieses Muster wird traditionell für grafische Desktop-Benutzeroberflächen (GUIs) verwendet und zum Entwerfen von Webanwendungen die erste Wahl [laND21]. Beliebte Programmiersprachen haben MVC-Frameworks integriert, die die Implementierung des Musters erleichtern.

Die drei wichtigsten MVC-Komponenten sind [OnVhf21]:

- Modell: Es enthält alle Daten und die zugehörige Logik
- View: Präsentiert dem Benutzer Daten oder verarbeitet Benutzerinteraktionen.
- Controller: Eine Schnittstelle zwischen Model- und View-Komponenten.

-
- **View** : Der View ist das Mittel zum Anzeigen von Objekten innerhalb einer Anwendung. Er umfasst bspw. das Anzeigen eines Fensters oder von Schaltflächen oder Text innerhalb eines Fensters. Er enthält alles, was der Benutzer sehen kann. Der View stellt die bereitgestellten Informationen in einem bestimmten Format dar. Im Allgemeinen wird JSP verwendet, um eine Ansichtsseite zu erstellen. Spring unterstützt jedoch auch andere Ansichtstechnologien wie Apache Velocity, JMustache, Pebble und Thymeleaf [Balng21]. Das View-Format wird durch Eingabe des nachfolgend aufgeführten Befehls in der `application.properties` bestimmt :

```
1 spring.mvc.view.suffix=.jsp
```

- **Controller** : Der Benutzer kann eine Anfrage (GET, POST, PUT, DELETE) senden, indem er mit einer Ansicht interagiert. Der Controller verarbeitet diese Anfragen und sendet sie an ein Model und erhält dann eine entsprechende Antwort vom Model. Die Antwort wird an die View gesendet.
Es kann auch die erforderliche Logik beinhalten und es fungiert als Vermittler zwischen Ansicht und Model.
Webanwendungen implementieren Controller mithilfe von Servlets. Desktopanwendungen implementieren Controller mithilfe von Listener-Klassen.
- **Model** : Das Model ist verantwortlich für die Speicherung und Verwaltung von Daten. SQL-Abfragen sind ebenso Teil des Models, also besteht die Aufgabe des Modells darin, die Daten einfach zu verwalten. Unabhängig davon, ob die Daten aus einer Datenbank, einer API oder einem JSON Objekt stammen, ist das Modell für deren Verwaltung verantwortlich.

Durch das Diagramm unten kann der Zusammenhang vereinfacht verstanden werden .

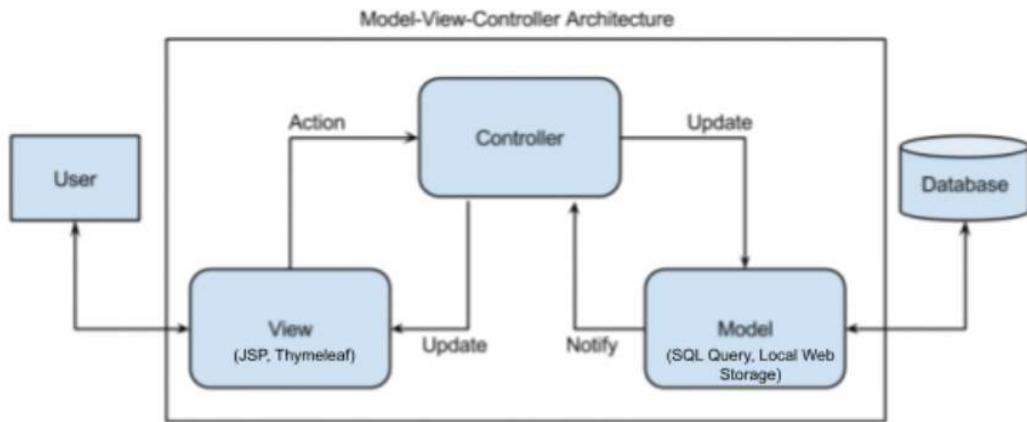


Abbildung 3.3.2 : MVC Architektur

Quelle : <https://www.methodpark.de/blog/wp-content/uploads/2018/07/Model-View-Controller-High-Level-Diagram-768x339.png>

3.3.3 Datenbank

Bei dieser Arbeit handelt es sich um ein Prototyp Projekt, daher wurde die SQLite Datenbank verwendet, aber für die Überführung des Projekts ins Leben zur Nutzung müssen einige Änderungen daran vorgenommen werden, einschließlich der Umstellung auf eine online zugängliche Datenbank.

Die Datenbankstruktur besteht aus 20 Tabellen. Man kann sagen, dass die User Tabelle der Kern der Datenbankstruktur.

Die Hauptfunktion der Calendar Tabelle besteht darin, die ID jedes elektronischen Kalenders in das Feld `string_calendar_id` aufzunehmen. Der Primärschlüssel in dieser Tabelle ist das Feld `calendar_id`, die User_Calendar Tabelle nimmt jede der `user_id` aus der User Tabelle zusätzlich zu der `calendar_id` aus der Calendar Tabelle als Referenz (Fremdschlüsseln), durch die jeder elektronische Kalender und sein Besitzer identifiziert werden.

Gleiches gilt für den Rest der Tabellen bspw. die `event_id` wird in der Event Tabelle gespeichert und diese Events werden an die Benutzer in der User_Event Tabelle verteilt. Bei der User_Event Tabelle ist der `calendar_id` Feld Fremdschlüssel zu der

Tabelle `Calendar`, dadurch wurde Jedes Event mit einem eigenen Kalender gekennzeichnet.

In der Tabelle `Document` werden Informationen zu den hochgeladenen Dokumenten bzw. Dateien wie der Name, die Sprache, das Dateiformat der Dokumente gespeichert.

Die `Role` Tabelle enthält vier Datensätze, die nicht geändert werden können und beim Erstellen der Datenbank automatisch konfiguriert werden. Die Rollen sind Manager, Worker, Volunteer und Customer entsprechend der IDs 1, 2, 3 und 4, wobei die `role_id` ist der Primärschlüssel der Tabelle.

Nach der Erläuterung der Haupttabellen ergibt sich ein allgemeines Bild über den Aufbau und die Entstehung und Gestaltung der Datenbank.

Das Datenbankschema wird in der folgende Abbildung 3.3.3 angezeigt.

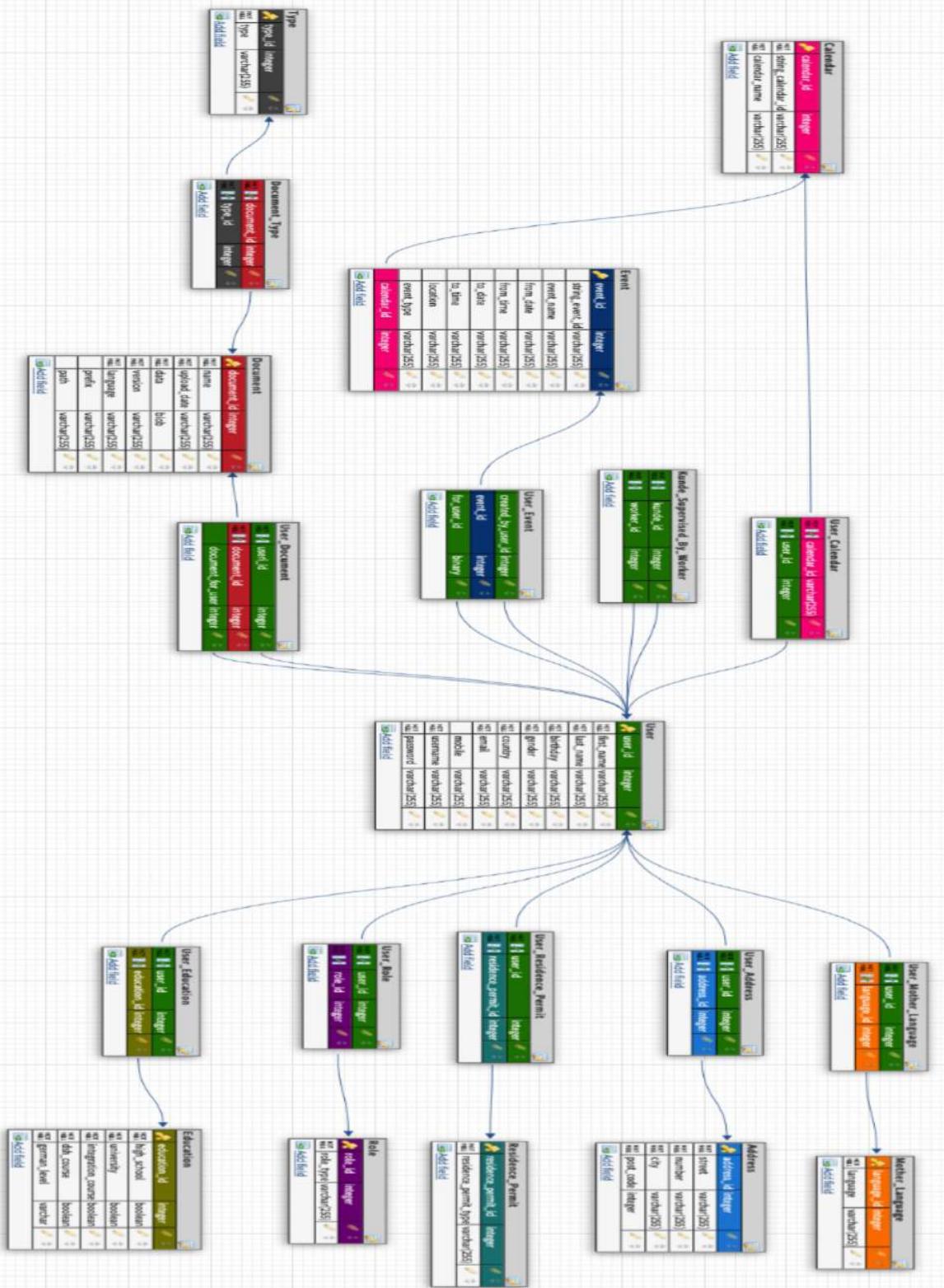


Abbildung 3.3.3 : Das Datenbankschema

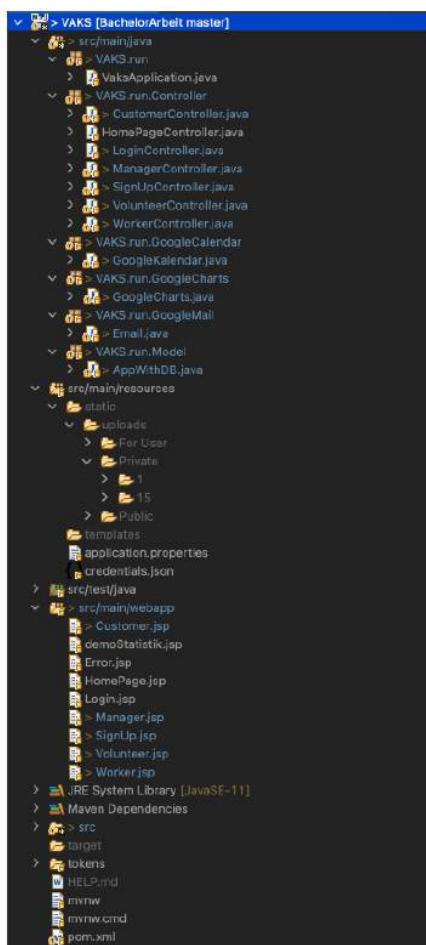
4 Entwicklung und Realisierung

In diesem Kapitel werden die oben genannten Ziele nach dem MVC-Modell (siehe 3.3.2 Architektur) umgesetzt. Bestimmte Teile werden erklärt, die ein allgemeines Verständnis des Implementierungsprozesses ermöglicht.

4.1 Projektstruktur

Die Projektstruktur ist klar definiert und die Komponenten sind entsprechend ihrer Rollen getrennt. Die Strukturübersicht ist in der folgenden Abbildung zu sehen:

Abbildung 4 : Projektstruktur



Die Klasse `VaksApplication.java` ist der Runpoint der Webanwendung. Es enthält die gesamte Initialisierung, Konfigurationen und Server-Start Logik. Durch die Ausführung dieser Klasse wird der Server konfiguriert und beginnt zu arbeiten. Über ihn kann auf die Anwendungsseiten zugegriffen werden.

Alle Operationen, die für das Mapping, die Weiterleitung (Forwarding), Umleitung (Redirecting) und auch die Projektlogik verantwortlich sind, sind in dem Paket Controller `VAKS.run.Controller` enthalten.

Das Paket Model `VAKS.run.Model` enthält die Klasse, aller Operationen, die für das Speichern, Löschen und Ändern von Daten in der Datenbank verantwortlich sind, sowie die Aufgabe, die Anwendung mit der Datenbank zu verknüpfen und Informationen

aus der Datenbank zu erhalten.

Die JSP Files `.jsp` befinden sich im Ordner `webapp` Sie enthalten neben JavaScript-Funktionen alle Bilder, Farben und Schriftarten (Seitenformat). Alles, was der Benutzer sieht, ist in diesen Dateien unter dem Pfad `src/main/webapp` enthalten.

Im Pfad `src/main/resources/static` und im Ordner `uploads` werden Dateien wie Bilder, PDF, und Word Dateien vom Anwendungsbenuzer gespeichert.

Dieser Ordner ist so aufgebaut, dass die Dateien für jeden Benutzer nach seiner persönlichen Nummer `ID` abgelegt werden. Alle öffentlichen Dokumente werden im Ordner `uploads/public` gespeichert. Als Beispiel der Benutzer, der die `ID 15` hat, wurden seine Dateien im Pfad `src/main/resources/static/uploads/For User/15/*` abgelegt.

Beim Anlegen eines neuen Benutzers wird automatisch und sofort ein neuer Ordner mit seiner `ID` erstellt.

Die Klasse `GoogleCalendar.java` in dem Paket `VAKS.run.GoogleCalendar` ist verantwortlich für alle Aktionen im Zusammenhang mit dem Kalender zum Erstellen, Löschen oder Ändern eines Termins, Es ist verantwortlich für die API-Funktionen und das Senden von Anfragen an die API.

Die Klasse im Paket `VAKS.run.GoogleMail` ist verantwortlich für die Bestätigungs Nachrichten, die an den Kunden beim Erstellen, Löschen oder Ändern einen Termin gesendet werden.

Die Statistikoperationen und Datenbankabfragen zu den Statistikgraphen werden innerhalb der Klasse `GoogleCharts.java` bei dem Paket `VAKS.run.GoogleCharts` ausgeführt.

4.2 Berechtigungen und Befugnisse

In den meisten sozialen Einrichtungen ist die Organisationsstruktur ähnlich: Es gibt Führungskräfte (Manager), Ehrenamtliche (Volunteers), feste Mitarbeiter (Worker) und zusätzlich Kunden (Customer). Es ist offensichtlich, dass die Führungskraft mehr Befugnisse hat als die normalen Angestellten und die einfachen Angestellten mehr als die Freiwilligen, da dieses Konzept oft hierarchisch ist. In der folgenden Tabelle sind die Befugnisse von **Manager**, **Worker**, **Volunteer** und **Customer** aufgeführt.

		User		
		Create	Edit	Delete
Manager	Manager	✓	✓	✓
	Worker	✓	✓	✓
	Volunteer	✓	✓	✓
	Customer	✓	✓	✓
Worker	Manager	✗	✗	✗
	Worker	✗	✗	✗
	Volunteer	✓ (Nur für sie Verantwortliche)	✓ (Nur für sie Verantwortliche)	✓ (Nur für sie Verantwortliche)
	Customer	✓ (Nur für sie Verantwortliche)	✓ (Nur für sie Verantwortliche)	✓ (Nur für sie Verantwortliche)
Volunteer	Manager	✗	✗	✗
	Worker	✗	✗	✗
	Volunteer	✗	✗	✗
	Customer	✗	✗	✗
Customer	Manager	✗	✗	✗
	Worker	✗	✗	✗
	Volunteer	✗	✗	✗
	Customer	✓ (Nur für ihn selbst)	✗	✗

Tabelle 4.2a: Manage User Befugnisse

In der **Tabelle 4.2a** kann man sehen, dass der Manager beispielsweise einen neuen Worker, Volunteer oder auch Manager hinzufügen kann, aber das Gegenteil ist nicht der Fall, da es für einen Volunteer nicht möglich sein soll, einen neuen Manager zu erstellen.

In der folgenden Tabelle werden die gemeinsamen oder öffentlichen Veranstaltungen, die alle Mitarbeiter des Vereins haben, als "Public" bezeichnet. Der Termin mit einem bestimmten Benutzer oder seinem Vorgesetzten wurde für den Angestellten als "Private" bezeichnet.

Der Tabelle zufolge kann der Manager alle Events (Public und Private) hinzufügen, löschen und bearbeiten.

Anders als der Worker, der lediglich die von ihm hinzugefügten Events ändern kann.

Es ist auch anzumerken, dass der Volunteer absolut nicht in der Lage ist, Public Events hinzuzufügen, zu ändern oder zu löschen.

		Event		
		Create	Edit	Delete
Manager	Public	✓	✓	✓
	Private	✓	✓	✓
Worker	Public	✓	✓ (Nur die von ihm eingefügte)	✓ (Nur die von ihm eingefügte)
	Private	✓	✓ (Nur für sie Verantwortliche)	✓ (Nur für sie Verantwortliche)
Volunteer	Public	✗	✗	✗
	Private	✓	✓ (Nur die von ihm eingefügte)	✓ (Nur die von ihm eingefügte)
Customer	Public	✗	✗	✗
	Private	✓ (Nur für ihn selbst)	✓ (Nur für ihn selbst)	✓ (Nur für ihn selbst)

Tabelle 4.2b: Manage Event Befugnisse

Ähnlich wie in der **Tabelle 4.2b** erwähnt werden öffentliche Dokumente, auf die jeder zugreifen kann, als Public bezeichnet, und die Dateien eines bestimmten User wurden als Private bezeichnet.

Das gleiche Prinzip gilt in ähnlicher Weise für die Prozesse des Hochladens und Herunterladens von Dokumenten.

Der Manager kann wie üblich auf alle Dateien zugreifen, sie herunterladen und ändern, während der Mitarbeiter beispielsweise nur auf die Dokumente der Kunden zugreifen und diese ändern kann, aber nur für die er verantwortlich ist.

		Document		
		Upload	Download	Delete
Manager	Public	✓	✓	✓
	Private	✓	✓	✓
Worker	Public	✓	✓	✓
	Private	✓ (Nur für sie Verantwortliche)	✓ (Nur für sie Verantwortliche)	✓ (Nur für sie Verantwortliche)
Volunteer	Public	✓	✓	✓ (Nur die von ihm eingefügte)
	Private	✓	✓	✓
Customer	Public	✗	✗	✗
	Private	✓ (Nur für ihn selbst)	✓ (Nur die für ihn selbst)	✓ (Nur die für ihn selbst)

Tabelle 4.2c: Manage Document Befugnisse

	Calendar
	Besitzen
Manager	✓
Worker	✓
Volunteer	✓
Customer	✗

Tabelle 4.2d: Besitzung eines Kalenders

Da die Statistiken viel Privatsphäre der Besucher wie Herkunftsland, Alter usw. preisgeben, wurde beschlossen, den Zugriff auf die Statistiken nur dem Manager zu ermöglichen.

	Statistiken
	Zugriff
Manager	✓
Worker	✗
Customer	✗
Volunteer	✗

Tabelle 4.2e: Zugriff auf Statistiken

4.3 Controllers

ManagerController

Diese Klasse ist für die logischen Operationen des ManagerControllers und für die Operationen der Umleitung und Weiterleitung verantwortlich.

In diesem Controller sind mehrere POST- und GET- Methoden verfügbar. Dadurch werden die Prozesse des Hinzufügens eines neuen Benutzers, des Hinzufügens eines Termins für einen Kunden, Termine zu löschen oder zu ändern, und die Bearbeitung der User, also alle Anfragen werden auch an die APIs gesendet. Beispielhaft wird die addUser() Methode erklärt :

```
1  @RequestMapping(value = "AddUser")
2      public String addUser(@RequestParam(required = true) String userName, email, ...)
3          throws SQLException {
4
5          AppWithDB.sqlAddUserByManager(userName, email, ...);
6
7          VAKS.run.GoogleMail.Email.giveSupervisorInfoEmail(userName, email);
8
9          return "redirect:/Manager#User";
10
11 }
```

Wie oben zu sehen ist, ist die Methode `addUser()` - der Name sagt viel von der Funktionalität - für das Hinzufügen eines neuen Benutzers verantwortlich. Beim Ausfüllen des `Form` im View `Manager.jsp` und durch Drücken des Submit-Buttons, erhält dieser Controller eine `POST`-Anfrage und Attributen wie Username, Email, Firstname, ..., und Lastname wurde erfasst, anschließend werden diesen auf das Model `AppWithDB.java` übertragen, Dort werden diese Daten über eine `INSERT-SQL-Abfrage` in der Datenbank gespeichert. Danach wird über die Klasse `Email()` über die Methode `giveSupervisorInfoEmail()` eine E-Mail an den registrierten Kunden gesendet, in der erklärt wird, wer für ihn verantwortlich ist inkl. Informationen wie E-Mail-Adresse und Telefonnummer die über ihn gespeichert sind. Schließlich leitet die Methode zur relativen URI `“localhost:8080/Manager#User”` weiter, wo der Manager den neu hinzugefügten User sehen kann (siehe Abbildung 3.3.1: Manager | Manage User).

WorkerController

Genau wie im vorherigen Absatz gezeigt, ist der `WorkerController.java` dem `ManagerController.java` sehr ähnlich.

Gemäß der Berechtigungstabelle (siehe 4.2 Berechtigungen und Befugnisse) führen die Methoden die in dieser Klasse angegebenen Funktionen unter Berücksichtigung der im vorherigen Absatz erklärten Methode `addUser()` aus. Diese Methode ist innerhalb dieses Controllers vorhanden, jedoch mit weniger Befugnissen, da zwar neue Benutzer hinzugefügt werden, jedoch nur als Volunteer oder Customer.

Die für das Löschen des Benutzers verantwortliche Methode wird hier erklärt:

```
1  @RequestMapping(value = "Worker/deleteUser")
2      public String showTable(@RequestParam(required = false) String user_id_hidden, String
3      actionBtnValue) throws SQLException, IOException {
4
5          if (actionBtnValue.equals("Delete") && !user_id_hidden.equals("cancel")) {
6              AppWithDB.sqlDeleteUserByManager(user_id_hidden);
7          }
8          return "redirect:/Worker#User";
9      }
```

Wenn der Mitarbeiter die Löschtaste drückt, um einen Benutzer zu löschen, schließt er das Senden einer `POST`-Anfrage innerhalb des `Form` in der View `Worker.jsp` ab.

Der Controller erhält zunächst die `ID` des zu löschen Benutzers dann werden diesen auf das Modell `AppWithDB.java` übertragen. Dort werden diese Daten über eine `DELETE`-SQL-Abfrage aus der Datenbank gelöscht.

Hier wird auch geprüft, ob der gelöschte User die Rolle Volunteer hat. Falls ja, wird auch sein Google Calendar gelöscht.

Beim Senden einer `DELETE`-Anfrage an die Google Calendar API.

Mit der `DELETE`-Anfrage wurde bereits die `calendarID` an die Google Calendar API gesendet. Diese wurde beim Anlegen des Benutzers in der Datenbank gespeichert.

VolunteerController

Hier erklären wir, wie eine Datei von der Clientseite auf die Serverseite hochgeladen wird, das heißt, in einfacher Sprache, wie ist der Prozess des Hochladens von Dateien, Dafür erkläre ich die Methode `uploadFile()` :

```
1  @RequestMapping(value = "Volunteer/uploadDocument", method = RequestMethod.POST)
2      public String uploadFile(@RequestParam(required = false) String name, String
3 language, String Type, String document_for_user, MultipartFile browse) throws SQLException,
4 IllegalStateException, IOException {
5
6         DateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss.SSS");
7         Date date = new Date();
8         String now = dateFormat.format(date);
9         String prefix = browse.getContentType().split("/")[1];
10
11        if (Type.equals("Public")) {
12
13            String baseDir = "/Users/mohammedalianis/git/BachelorArbeit/
14 VAKS/src/main/resources/static/uploads/Public/";
15
16            String uploadedFile = baseDir + name + " <> " + now + "." + prefix;
17            String finalPath = uploadedFile;
18            AppWithDB.sqlUploadDocumentByManager(name + " <> " + now, now,
19 language, Type, uploadedFile, document_for_user, prefix,
20                                     finalPath);
21            browse.transferTo(new File(finalPath));
22        } else if (Type.equals("For User")) {
23
24
25
26
27        return "redirect:/Volunteer#Documents";
28    }
```

Hier wird geprüft, ob die hochzuladende Datei “Public” ist oder es ist für einen bestimmten Benutzer “Private” deklariert. Beim Ausfüllen des Form im View `Volunteer.jsp` und durch Drücken des Upload-Buttons, erhältet diesen Controller eine POST-Anfrage mit Attributen wie Name, Language, Type des Dokuments und auch das File Objekt.

In diesem Schritt war es schwierig, eine Duplizierung von Dateien zu vermeiden, da es möglich ist, dass der Benutzer beim Hochladen einer bestimmten Datei denselben Namen wie die zuvor hochgeladene Datei hat.

Der Benutzer hat beispielsweise eine Kopie seiner Aufenthaltskarte mit der Name "Aufenthaltstitel" hochgeladen und nach Ablauf dieser Aufenthaltserlaubnis und sobald er eine neue Aufenthaltskarte erhalten hatte, beschloss er eine Kopie der neuen Aufenthaltskarte hochzuladen, die die neue Datei mit hoher Wahrscheinlichkeit genau so benennt wie die alte Datei, also auch "Aufenthaltstitel".

Um die Duplizierung von Dateien zu vermeiden habe ich mich entschieden, das Datum des Datei-Uploads zusammen mit Sekunden und Sekundenbruchteilen zum Namen der hochzuladenden Datei hinzuzufügen. Dieses Zeichen <> wird als Trennzeichen zwischen dem Dateinamen und dem Upload-Datum verwendet, als Beispiel:

Name : Aufenthaltstitel

Prefix : pdf

Date : 2021-07-6_13:39:03.718

Type : public

=>

baseDir:src/main/resources/static/uploads/public

finalPath:src/main/resources/static/uploads/public/Aufenthaltstitel<>

2021-07-6_13:39:03.718.pdf

Mit dem Befehl `transferTo(new File(finalPath))` wurde die Datei auf die Serverseite hochgeladen. Das Verzeichnis jeder gespeicherten Datei wird auch in der Datenbank gespeichert, dies macht es einfach, die Datei zu finden, wenn sie heruntergeladen werden.

CustomerController

Hier wird die Methode gezeigt, die Dateien nach dem Herunterladen von der Serverseite löscht und es ist auch eine Methode, die beim CustomerController verfügbar ist.

Durch die SELECT-SQL-Abfrage wird der Pfad der zu löschen Datei aus der Datenbank geholt.

Dieser Vorgang findet erst dann statt, wenn der Customer den Delete Document Button drückt. Dann wird die ID auf das Modell AppWithDB.java übertragen, und dort werden diese Daten über eine DELETE-SQL-Abfrage von der Datenbank gelöscht.

Nachdem der Pfad der zu löschen Datei vom Server abgerufen hat und mit diesem Befehl `file.delete()` und mit Hilfe des Pfades wird die Datei gelöscht und dann wieder nach View Customer.jsp weitergeleitet.

```
1  @RequestMapping(value = "Customer/deleteDocument")
2      public String deleteDocument(@RequestParam(required = true) String
3          document_id_hidden, String actionBtnValue) throws SQLException {
4
5
6
7      String query = " Select path from Document where document_id = '" +
8      document_id_hidden + "'";
9
10
11
12
13
14
15      r = s.executeQuery(query);
16      AppWithDB.sqlDeleteDocumentByManager(document_id_hidden);
17      File file = new File(path);
18      file.delete();
19      return "redirect:/Customer#Documents";
20  }
```

4.4 Models

Wie bereits in Kapitel 3 erwähnt (siehe 3.3.2 Architektur) ist das Model verantwortlich für die Verwaltung der Daten und die Beziehung der Anwendung zur Datenbank. Dadurch werden die meisten SQL-Abfragen (SELECT, UPDATE, INSERT und DELETE) erledigt.

Hier findet der Prozess des Hinzufügens von Benutzern, Kalender, Events und der Verteilung von Daten auf die Datenbanktabellen statt.

Durch das Modell ist jeder Benutzer mit seinem elektronischen Kalender innerhalb der Tabelle verknüpft und jedes Event wird mit seinem Besitzer und seinem eigenen Kalender klassifiziert.

Die Methode zum Löschen des Benutzers `sqlDeleteUserByManager(String user_id)` :

Die Methode erhält die `ID` des zu löschen Benutzers, Es wird dann geprüft, ob der zu löschen Benutzer die Rolle Manager, Worker oder Volunteer innehaltet.

Dann wird durch `DELETE`-SQL-Abfrage der Benutzer aus allen Tabellen, außer der Tabelle `Calendar` und der Tabelle `User_Calendar`, gelöscht.

Falls die Rolle des Users nicht Customer war, bedeutet es, dass der Benutzer auch einen elektronischen Kalender hat, Hier wird die `Calendar_ID` aus der Tabelle `User_Calendar` angeliefert. Diese `Calendar_ID` wird an die Google Calendar-API unter der `GoogleKalendar.java` Klasse gesendet.

Damit ist der Kalender schon gelöscht.

Falls der User die Role Customer hatte, wurden die Tabellen `Calendar` und `User_Calendar` ignoriert. Es wird keine Anfrage an die Google Calendar API gesendet, da der User keinen elektronischen Kalender hat.

```

1  public static void sqlDeleteUserByManager(String user_id) throws SQLException, IOException {
2      String string_calendar_id = null;
3      String role_type = null;
4
5      String query1_1 = "Select address_id, role_type from User NATURAL JOIN
6          Address NATURAL JOIN User_Address Natural Join Role Natural Join User_Role where user_id='"+
7          user_id + "';";
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119

```

Ich möchte an dieser Stelle darauf hinweisen, dass alle Methoden innerhalb des Model AppWithDB.java die selbe Logik ausführen, die hier gezeigt wurde. Deshalb habe ich darauf verzichtet, alle Methoden zu erklären und detailliert darzustellen und nur eine exemplarisch vorgestellt, durch die der Mechanismus der Arbeit des Modells gezeigt wurde.

Die vollständige Dokumentation mittels javadoc wird dieser Arbeit beigelegt, wodurch die Funktionalität aller existierenden Methoden und alle Parameter für jede Methode separat angezeigt werden.

4.5 Views

In dieser Arbeit wird für die Views `.jsp` verwendet. Jede View enthält die CSS- und einige Javascript-Funktionen um den Seiten etwas Ästhetik und Design zu geben und die Oberfläche benutzerfreundlich zu gestalten.

Der Pfad `VAKS/src/main/webapp` enthält alle Views wie die `HomePage.jsp`, `Login.jsp`.

Bei der Eingabe von Benutzername und Passwort in View `HomePage.jsp`, wird in dem `LoginController.java` durch SELECT-SQL-Abfrage die Rolle des Benutzers überprüft, natürlich nur wenn das Passwort und der Benutzername korrekt eingegeben wurden, Anschließend wird von dort entsprechend der Rolle des Benutzers, der sich anmelden möchte, in die entsprechende View umgeleitet.

Die meisten Zeilen in Views sind mit HTML-Markup belegt und nur ein Teil hat JSP-Syntax zum Rendern einiger dynamischer Daten. Das Codebeispiel für einen kleinen Teil einer Table ist unten zu sehen :

```
1 <table id="DocumentTable" class="table table-bordered">
2   <thead>
3     <tr>
4       <th>ID</th>
5       <th>Name</th>
6       <th>Language</th>
7       <th>Type</th>
8       <th>Delete</th>
9     </tr>
10    </thead>
11    <tbody>
12      .
13      .
14      .
15      <% HashMap<String, String> data = AppWithDB.getDocumentData(); %>
16      <tr>
17        <td <%=data.get("document_id")%> </td>
18        <td <%=data.get("document_name")%> </td>
19        <td <%=data.get("document_language")%> </td>
20        <td <%=data.get("document_type")%> </td>
21        <td ><input type="submit" Value="Delete" ... </td>
22      </tr>
```

In der Codebox wird angezeigt, wie eine Tabelle über HTML erstellt und wie sie durch JSP Scriptlet und JSP Expression gefüllt wird.

JSP Scriptlet : Da es innerhalb dieser `<% %>` möglich ist, Java-Code innerhalb einer JSP Seite zu schreiben.

JSP Expression : Der im `<%= %>` platzierte Code wird in den Ausgabestream der Antwort geschrieben. Man braucht also nicht `out.print()` schreiben, um Daten zu schreiben. Es wird hauptsächlich verwendet, um die Werte von Variablen oder Methoden zu drucken.

Auf die gleiche Weise werden alle dynamischen Inhalte des Projekts erstellt.

4.6 Google Calendar API

Um auf Google-Konten zuzugreifen, muss das Konto zunächst gemäß den entsprechenden Berechtigungen authentifiziert werden. Laut der offiziellen Website von Google können alle Google APIs mit OAuth 2.0 authentifiziert werden. Es ist ein Login-Prozess erforderlich, um die Berechtigungen zu authentifizieren. Dieser Vorgang bietet gemäß den Berechtigungen, denen der Benutzer zugestimmt hat, einen eingeschränkten Zugriff.

Auf diese Weise wird ein Access Token namens "StoredCredential" generiert. Dabei ist zu beachten, dass dieses Token eine bestimmte Gültigkeitsdauer hat und nach Ablauf erneuert werden muss. Es ist möglich, dieses Token zu erneuern oder seine Gültigkeit zu verlängern, ohne sich erneut anmelden zu müssen. Ein Refresh Token wird automatisch generiert, sobald der vorherige Token abgelaufen ist. Das Prinzip der Refresh Token wurde im aktuellen Projekt nicht übernommen, aber es ist geplant, diesen Mechanismus zu übernehmen, wenn die Webanwendung in den laufenden Betrieb eingeführt wird.

Jeder Kalender und auch jedes Event haben ihre eigene ID, die in der Datenbank gespeichert wird. Jedes Event ist über seine ID mit einem eigenen Kalender verknüpft.

Beim Hinzufügen, Löschen oder Ändern eines Events erfolgt der Prozess parallel über die Datenbank durch JDBC und auch am elektronischen Kalender durch die API.

Beim Erstellen des Kalenders gibt es Parameter, die über die API übergeben werden, welche eine genauere Beschreibung geben und den Kalender anpassen. Beispielsweise kann über den Parameter summary ein Name für den Kalender angegeben werden. Eine Beschreibung des Kalenders kann weitestgehend definiert werden durch die description. Der Erstellungsort und die Zeitzone können auch über die location bzw. timeZone angegeben werden.

Ebenso wird jedes Event auch durch Parameter beschrieben, die über die API übergeben werden. Das Start- und Enddatum sowie die Uhrzeit des Events werden durch den Parameter start.dateTime und end.dateTime bestimmt, der Eventsort ebenfalls durch den Parameter location .

```
1 public static void deleteEvent(String eventID, String calendarID) throws IOException,
2                                     GeneralSecurityException {
3
4
5     // Delete an event
6     service.events().delete(calendarID, eventID).execute();
7     System.out.println("Event has been deleted");
8 }
```

Wie oben gezeigt, benötigt die Methode zum Löschen eines Events deleteEvent(String eventID, String calendarID) die ID des Events selbst, sowie die ID des Kalenders, zu dem der zu löschenende Termin gehört. Daher werden alle diese Informationen in der Datenbank gespeichert, um später leicht darauf zugreifen zu können.

Dabei ist zu beachten, dass die Google Calendar das RFC3339 System als Format für Datum und Uhrzeit übernimmt.

4.7 Google Charts

Um einige Diagramme anzuzeigen, wurde in dieser Arbeit Google Charts verwendet, da hiermit das Einbetten sehr leicht ist und Google Charts kostenlos verwendet werden kann.

Google Charts zu verwenden, ist mit einfachem JavaScript möglich, das in eine Webseite eingebettet wird.

Statistiken und Datenerhebungen werden innerhalb von der Klasse `GoogleCharts.java` geladen, die sich innerhalb des Pakets `VAKS.run.GoogleCharts` befindet. Diese Operation erfolgt durch eine `SELECT-` SQL-Abfrage .

```
1  public static Map<String, Integer> residenceCount() throws SQLException {
2      ResultSet r = null;
3      String Query = "SELECT residence_permit_type, COUNT(residence_permit_id) AS
4                      Number FROM . . . ";
5
6
7
8
9
10     while (r.next()) {
11         graphData.put(r.getString("residence_permit_type"),
12             Integer.parseInt(r.getString(2)));
13     }
14
15
16
17     return graphData;
18 }
```

In der View Manager.jsp muss der Loader selbst geladen werden, was in einem separaten Script-Tag mit `src="https://www.gstatic.com/charts/loader.js"` geschieht, Danach wird die Methode aufgerufen und die Werte werden dort verteilt.

Hier muss eine ID gegeben sein, hier `Residence_Permit`, um das Google-Diagramm anzuzeigen, Die Darstellung erfolgt durch ein `<div>`-Tag.

```
1  <script type="text/javascript" src="https://www.gstatic.com/charts/loader.js"></script>
2  <script type="text/javascript">
3      google.charts.load("current", {packages: ["corechart"]});
4      google.charts.setOnLoadCallback(drawChart);
5      function drawChart() {
6          var data = google.visualization.arrayToDataTable([
7              ['Residence Type', 'Count'],
8              <%
9                  for (Map.Entry<String, Integer> pair : residencePermitAnswer.entrySet()) {
10                      %>
11                      ['<%=pair.getKey()%>', <%=pair.getValue()%>],
12                      <%
13                      }
14                      %>
15                  ]);
16          var options = { . . . }
17      };
18      var chart = new
19          google.visualization.PieChart(document.getElementById('Residence_Permit'));
20          chart.draw(data, options);
21      }
22  </script>
23
24
25
26
27
28
29
30  <div id="Residence_Permit" style="width: 900px; height: 500px;"></div>
```

4.8 Google Dialog Flow

<iframe> wurde in die Website integriert, um die Anfragen von Nutzern und Besuchern zu beantworten und in Natural Language kommunizieren zu können.

Der Eingabeprozess besteht entweder aus einem Satz oder mehrere Wörter. Der Chatbot empfängt die Keywords, verarbeitet sie und analysiert sie mit dem NLP Natural Language Process.

Allerdings stellt Google Entwicklern keine Informationen zur Funktionsweise der Methoden bei der Verarbeitung dieser Keywords zur Verfügung.

Der Chatbot versucht zunächst grundlegende Informationen über den Benutzer wie Name, Nationalität und Art der Aufenthaltserlaubnis zu erhalten.

Die Hauptidee bei der Erstellung dieses Chatbots besteht darin, häufig gestellte Fragen zu beantworten. Dafür hat Google ein spezielles Tool zum Erstellen eines FAQ's Chatbots, welche "Knowledge Connector" heißt.

Um dem Projekt die Frequently Asked Questions und deren Antworten zur Verfügung zu stellen, kann der Link zur Webseite mit den Fragen und Antworten übergeben werden. Google Dialog Flow analysiert wiederum die HTML-Seite und extrahiert daraus Frage und Antwort. Anstelle der URL kann alternativ eine CSV-Datei übergeben werden, sodass diese Datei aus zwei Spalten ohne Kopfzeile besteht.

Die erste Frage steht in der ersten Spalte und ihre Antwort in der zweiten Spalte. Die zweite Frage steht in der zweiten Zeile der ersten Spalte und ihre Antwort befindet sich in derselben Zeile der zweiten Spalte und so weiter[GoGDF21].

Durch Dialog Flow Messenger wird ein Code generiert, ein <script>, das kopiert und auf der Seite eingefügt werden kann, zu der der Chatbot hinzugefügt werden soll. Der Chatbot kann durchaus auf mehr als einer Seite integriert werden.

```

1 <script
2   src="https://www.gstatic.com/dialogflow-console/fast/messenger/bootstrap.js?v=1"></script>
3 <df-messenger
4   intent="WELCOME"
5   wait-open="1"
6   chat-title="VAKS Bot"
7   agent-id="25531a17-59eb-44b8-906e-8df67aac36ae"
8   language-code="de"
9 ></df-messenger>
```

Es gibt eine Reihe von einstellbaren Parametern. Der Parameter `language-code` definiert beispielsweise die Sprache der Benutzeroberfläche des Chatbots, `wait-open` gibt die Zeit in Sekunden für den Start des Begrüßungsprozesses an.

Das Aussehen der Box des Chatbots kann durch CSS festgelegt werden.

```

1 df-messenger {
2   --df-messenger-bot-message: #878fac; /*Hintergrundfarbe für Nachrichten*/
3   --df-messenger-button-titlebar-color: #df9b56; /*Farbe Schaltfläche, die Titelleiste*/
4   --df-messenger-chat-background-color: #fafafa; /*Farbe für den Hintergrund*/
5   --df-messenger-font-color: white; /*Schriftfarbe für das Texteingabefeld*/
6   --df-messenger-send-icon: #878fac; /*Farbe des Symbols "Senden" im Texteingabefeld*/
7   --df-messenger-user-message: #479b3d; /*Hintergrundfarbe für User Benachrichtigungen*/
8 }
```

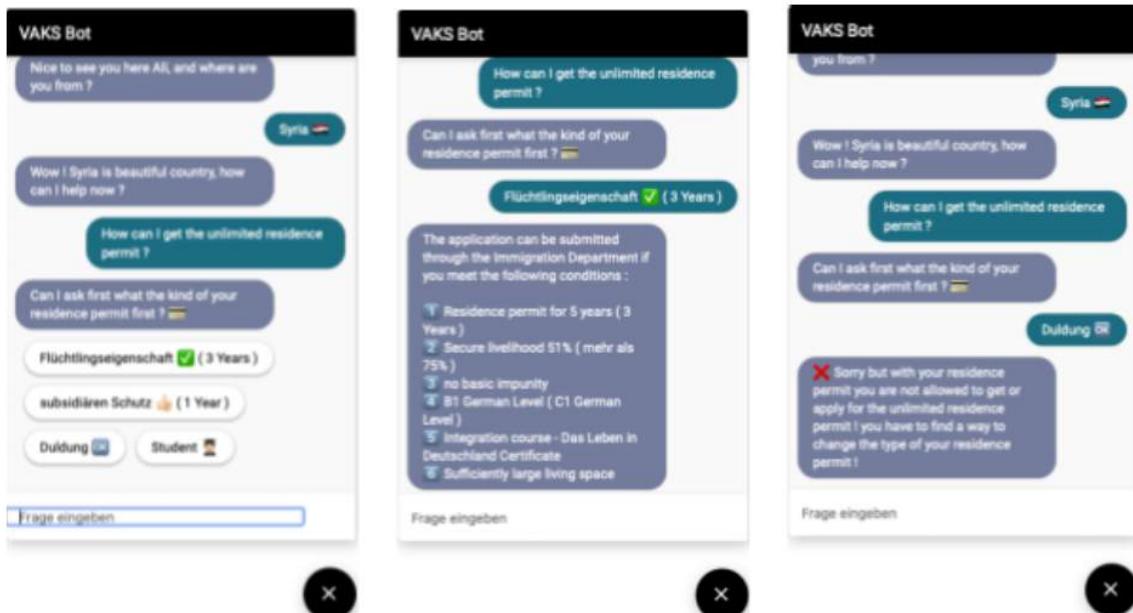


Abbildung 4.8 : Chatbot Szenario

4.9 Fazit

Am Ende dieses Kapitels möchte ich durch eine einfache Abbildung die Architektur des Projekts aufzeigen :

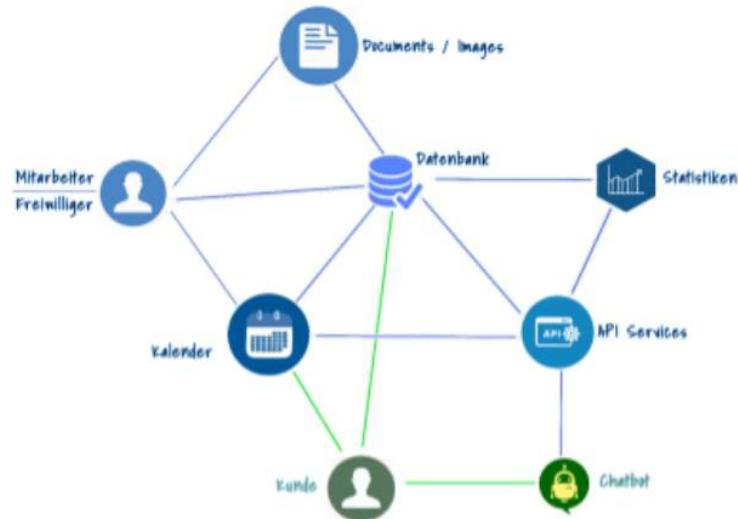


Abbildung 4.9 : die Architektur des Projekts

Wie oben gezeigt, zeigt sich die Wechselbeziehung zwischen allen Komponenten des Projekts. Man kann als Beispiel die Verknüpfung zwischen der Datenbank und dem API mit den Statistiken nennen, die wie bereits erwähnt die Daten über die Datenbank sammeln und an die API senden.

Beim elektronischen Kalender wird ersichtlich wie die Beziehung zwischen der API und der Datenbank zusätzlich zu den vier Benutzerrollen ist.

Es wird zudem deutlich wie wichtig die Datenbank ist, das sie mit allen Elementen verbunden und das Bindeglied zwischen ihnen ist.

5 Installation und Test

5.1 Installation

Zuerst wurde die Anwendung auf dem lokalen Computer mit dem Betriebssystem macOS Big Sur und auf einem Macbook Air erstellt. Mit dieser einfachen Anwendung wurden viele der Funktionen eingeführt und konfiguriert, die später im größeren Kontext verwendet werden. Zu diesem Zweck wurde das Maven Plugin mit einem eingebetteten Anwendungsserver verwendet.

Die Anwendung wurde mit dem Git-Repository verbunden, das den gesamten Quellcode enthält und die Versionskontrolle für die Anwendung durchführt. Die Datenbank wurde so erstellt und konfiguriert, dass sie zuverlässig und skalierbar ist und mit der Anwendung verbunden wurde.

Die Anwendung kann von meinem Github durch das Klonen des Repositorys BachelorArbeit heruntergeladen werden. Durch die Eingabe der folgenden Befehle im Terminal kann der Prozess durchgeführt werden :

```
cd folder/to/clone-into/  
git clone https://github.com/mohammedali-anis/BachelorArbeit.git
```

Apache Maven muss installiert sein oder kann über den folgenden Befehl durch Homebrew installiert werden:

```
brew install maven
```

Bei der Eingabe des Befehls `mvn spring-boot:run` im selben Verzeichnis, in dem der Klonen-Prozess durchgeführt wurde, dann ist die Anwendung über Port 8080 zugänglich.

Die Homepage kann über den Browser über die URI `localhost:8080/` aufgerufen werden.

5.2 Test

Eine sehr wichtige Phase der Programmierung und Programmentwicklung ist das Testen. Durch diesen Prozess werden Fehler und Fallstricke im Programm entdeckt und festgestellt, ob das Programm die durchzuführenden Anforderungen erfüllt. Hierdurch kann die Anwendung stark verbessert werden.

Die Anwendung wurde von insgesamt sechs Personen getestet. Der Hauptzweck bestand darin, Probleme mit der Anwendung aufzudecken und neue potenzielle funktionale Anforderungen zu finden oder bereits bestehende funktionale Anforderungen zu ändern.

Je nach Rolle des Benutzers werden die Aufgaben in Gruppen eingeteilt, dass heißt der User, der Volunteer, der Worker und der Manager.

User

- Konto Verwalten : Sign Up, Login, Logout.
- Termine : Termine Buchen, Termine Verwalten (Verschieben, löschen).
- Dokumente : Hochladen, Herunterladen, Löschen.

Volunteer

- Termine : Termine Buchen, Termine Verwalten (Verschieben, löschen).
- Dokumente : Hochladen, Herunterladen, Löschen.

Worker

- Konto Verwalten : Sign Up, Login, Logout.
- Termine : Termine Verwalten (Verschieben, löschen).
- Dokumente : Hochladen, Herunterladen, Löschen.

Manager

- Konto Verwalten : Sign Up, Login, Logout.
- Termine : Termine Verwalten (Verschieben, löschen).
- Dokumente : Hochladen, Herunterladen, Löschen.
- Statistiken : Ansehen

Den Personen, die die Anwendung ausprobiert haben, wurden einfache Fragen gestellt :

1. Wie wird das Design der Anwendung bewertet?
2. Ist die Anwendung einfach zu bedienen?
3. Ist die App nützlich?

In der Tabelle sehen Sie die Ergebnisse der einfachen Auswertung :

	Personen					
Fragen	Customer 1	Customer 2	Volunteer 1	Worker 1	Worker 2	Manager 1
1	OK	OK	✓	OK	✗	OK
2	✓	✓	✓	✓	OK	✓
3	OK	✓	✓	✓	✓	✓

✓ : Gut

OK : Ausreichend

✗ : Schlecht

Tabelle 5.2: Tests-Umfrage

Die Umfrage ergab, dass das Design der Anwendung verbesserungswürdig ist, aber dennoch einfach zu bedienen ist und seinen Zweck erfüllt [BkVh18].

Unten sind einige Kommentare von Leuten, die es getestet haben :

Manager 1 : “*Im Abschnitt Manage User war es sehr nützlich, eine Search Box auch mit der Filterfunktion zu haben, um das Auffinden des Benutzers zu erleichtern, nach dem Sie suchen möchten. Das Schöne ist, dass an diesem Punkt im Bereich Manage Event gearbeitet wurde und es sehr gut funktioniert*”.

Worker 2 : “*Es ist vorzuziehen, die Wahlfreiheit zwischen der Oberflächensprache z. B. zwischen Deutsch und Englisch zu haben, als nur eine Sprache zu haben*”.

Customer 1 : “*Das Tolle ist, dass ich nicht mehr anrufen muss, um einen Termin zu buchen, sondern das über die App machen kann. Ebenso benötige ich beim Versenden eines bestimmten Papiers keine Papierkopien mehr, es reicht ein digitales Foto vom Handy des Dokuments und ich kann es problemlos an den für mich zuständigen Mitarbeiter senden*”.

6 Zusammenfassung und Ausblick

6.1 Fazit

Das Ziel der Arbeit war es den Arbeitsprozess einer sozialen Organisation zu digitalisieren.

Die Anforderungsanalyse, zusätzlich zum praktischen Teil, nahm viel Zeit in Anspruch, da die verschiedenen Akteure befragt werden mussten.

Das Ergebnis der detaillierten Recherche zu den Techniken, die im praktischen Teil des Projekts verwendet wurden, sowie deren Behandlung im theoretischen Teil wurden dargestellt, damit der Leser die Struktur und Architektur der Anwendung versteht.

Die Anwendung wurde mit Spring Boot und verschiedenen APIs entwickelt. Serverseitiges Rendering mit CSS und Javascript wurde verwendet, um Benutzeroberflächen zu erstellen.

Der Code wurde in verschiedene Abschnitte wie Controller und View strukturiert, wodurch er leichter zu verstehen und unabhängig entwickelt und bei Bedarf gewartet und weiterentwickelt werden kann. Natürlich wurde die Anwendung kontinuierlich getestet. Die Anwendung ist eine Prototypanwendung, das heißt das bei der Anwendung noch unerwartete Fehler oder Verhalten auftreten können.

Trotz der Tatsache, dass das Projekt zu einem sauberen, organisierten und funktionalen Prototyp führte, gibt es daher immer mehrere verschiedene Technologien und Ansätze, die verwendet werden könnten, sowie einen sehr großen Raum für Verbesserungen.

6.2 Ausblick

Bevor die Anwendung in den Echtbetrieb übertragen wird, ist es notwendig, den Anwendungsrahmen zu verbessern. So ist z.B. die Anwendung z.Zt. nur auf eine Sprache abgebildet. Es wäre sehr sinnvoll wenn die Anwendung mehrsprachig wie z.B. Türkisch, Arabisch, Persisch wäre.

Die Webanwendung ist auch noch nicht für mobile Endgeräte optimiert.

Zielgruppe dieser Arbeit sind, wie bereits erwähnt, Beschäftigte in einer sozialen Organisation sowie Menschen mit Migrationshintergrund, die Unterstützung und Beratung benötigen.

In der Regel achtet dieser Personenkreis nicht so sehr auf E-Mails im Gegensatz zu SMS oder WhatsApp Nachrichten. Ein Smartphone ist immer dabei, daher ist es in Zukunft in dieser Anwendung geplant, Textnachrichten an die betroffene Person zu senden.

Im Folgenden werden einige Ideen erläutert, die zur Verbesserung der Webanwendung implementiert werden könnten:

- Die Möglichkeit Termine über den Chatbot selbst zu buchen und zu bearbeiten.
- Der Vorgang des Einbettens jedes hochgeladenen Dokuments mit einem QR-Code.
- Eine Plattform entwickeln, über die Besucher des Vereins mit ihren Verantwortlichen kommunizieren zu können.
- Entwicklung der Forum-Plattform, die es Mitarbeitern ermöglicht, bestimmte Erfahrungen zu veröffentlichen, die möglicherweise nicht sehr häufig vorkommen, und durch die andere Mitarbeiter im Falle eines ähnlichen in der Plattform erwähnten Ereignisses Erfahrungen sammeln können.

7 Anhang

7.1 Dependencies

```
1 <dependencies>
2     <dependency>    <!Uses Tomcat as the default embedded container -->
3         <groupId>org.springframework.boot</groupId>
4         <artifactId>spring-boot-starter-web</artifactId>
5     </dependency>
6
7     <dependency>    <!Starter for testing Spring Boot applications -->
8         <groupId>org.springframework.boot</groupId>
9         <artifactId>spring-boot-starter-test</artifactId>
10        <scope>test</scope>
11    </dependency>
12
13    <dependency>    <!--Core Tomcat implementation -->
14        <groupId>org.apache.tomcat.embed</groupId>
15        <artifactId>tomcat-embed-jasper</artifactId>
16        <scope>provided</scope>
17    </dependency>
18
19    <dependency>    <!-- Sqlite Dependency -->
20        <groupId>org.xerial</groupId>
21        <artifactId>sqlite-jdbc</artifactId>
22        <version>3.30.1</version>
23    </dependency>
24
25
26    <dependency>    <!-- To Refresh without restart the server -->
27        <groupId>org.springframework.boot</groupId>
28        <artifactId>spring-boot-devtools</artifactId>
29        <optional>true</optional>
30    </dependency>
31
32
33    <!-- Google Calendar API Dependency -->
34    <dependency>
35        <groupId>com.google.api-client</groupId>
36        <artifactId>google-api-client</artifactId>
37        <version>1.23.0</version>
38    </dependency>
39
40    <dependency>
41        <groupId>com.google.oauth-client</groupId>
42        <artifactId>google-oauth-client-jetty</artifactId>
43        <version>1.23.0</version>
44    </dependency>
45
46    <dependency>
47        <groupId>com.google.apis</groupId>
48        <artifactId>google-api-services-calendar</artifactId>
49        <version>v3-rev305-1.23.0</version>
50    </dependency>
51
52    <dependency>    <!-- Gmail -->
```

```

53     <groupId>javax.mail</groupId>
54     <artifactId>mail</artifactId>
55     <version>1.4.7</version>
56   </dependency>
57 </dependencies>

```

7.2 Datenbank (SQL)

Unten ist der SQL-Code, über den die Datenbanktabellen erstellt und die grundlegenden Eingaben konfiguriert werden :

```

1 BEGIN TRANSACTION;
2 CREATE TABLE IF NOT EXISTS "Mother_Language" (
3     "language_id"      INTEGER NOT NULL UNIQUE,
4     "language"         varchar,
5     PRIMARY KEY("language_id" AUTOINCREMENT)
6 );
7 CREATE TABLE IF NOT EXISTS "User_Mother_Language" (
8     "user_id"          INTEGER,
9     "language_id"      INTEGER,
10    FOREIGN KEY("user_id") REFERENCES "User"("user_id"),
11    FOREIGN KEY("language_id") REFERENCES "Mother_Language"("language_id")
12 );
13 CREATE TABLE IF NOT EXISTS "Address" (
14     "address_id"      INTEGER NOT NULL UNIQUE,
15     "street"          varchar,
16     "number"          varchar,
17     "city"            varchar,
18     "post_code"        INTEGER,
19     PRIMARY KEY("address_id" AUTOINCREMENT)
20 );
21 CREATE TABLE IF NOT EXISTS "User_Address" (
22     "user_id"          INTEGER,
23     "address_id"      INTEGER,
24     FOREIGN KEY("user_id") REFERENCES "User"("user_id"),
25     FOREIGN KEY("address_id") REFERENCES "Address"("address_id")
26 );
27 CREATE TABLE IF NOT EXISTS "User_Residence_Permit" (
28     "user_id"          INTEGER,
29     "residence_permit_id"  INTEGER,
30     FOREIGN KEY("residence_permit_id") REFERENCES
31 "Residence_Permit"("residence_permit_id"),
32     FOREIGN KEY("user_id") REFERENCES "User"("user_id")
33 );
34 CREATE TABLE IF NOT EXISTS "Role" (
35     "role_id"          INTEGER,
36     "role_type"        varchar,
37     PRIMARY KEY("role_id" AUTOINCREMENT)
38 );
39 CREATE TABLE IF NOT EXISTS "User_Role" (
40     "user_id"          INTEGER,
41     "role_id"          INTEGER,
42     FOREIGN KEY("user_id") REFERENCES "User"("user_id"),
43     FOREIGN KEY("role_id") REFERENCES "Role"("role_id")

```

```

44 );
45 CREATE TABLE IF NOT EXISTS "User_Education" (
46     "user_id"      INTEGER,
47     "education_id" INTEGER,
48     FOREIGN KEY("user_id") REFERENCES "User"("user_id"),
49     FOREIGN KEY("education_id") REFERENCES "Education"("education_id")
50 );
51 CREATE TABLE IF NOT EXISTS "Document_Type" (
52     "document_id"  INTEGER,
53     "type_id"      INTEGER,
54     FOREIGN KEY("type_id") REFERENCES "Type"("type_id"),
55     FOREIGN KEY("document_id") REFERENCES "Document"("document_id")
56 );
57 CREATE TABLE IF NOT EXISTS "Kunde_Supervised_By_Worker" (
58     "kunde_id"      INTEGER,
59     "worker_id"     INTEGER,
60     FOREIGN KEY("worker_id") REFERENCES "User"("user_id"),
61     FOREIGN KEY("kunde_id") REFERENCES "User"("user_id")
62 );
63 CREATE TABLE IF NOT EXISTS "Residence_Permit" (
64     "residence_permit_id"  INTEGER NOT NULL UNIQUE,
65     "residence_permit_type" varchar,
66     PRIMARY KEY("residence_permit_id" AUTOINCREMENT)
67 );
68 CREATE TABLE IF NOT EXISTS "Education" (
69     "education_id"  INTEGER,
70     "high_school"   boolean,
71     "university"    boolean,
72     "integration_course" boolean,
73     "dsh_course"    boolean,
74     "german_level"  varchar,
75     PRIMARY KEY("education_id" AUTOINCREMENT)
76 );
77 CREATE TABLE IF NOT EXISTS "User" (
78     "user_id"      INTEGER NOT NULL UNIQUE,
79     "first_name"   varchar,
80     "last_name"    varchar,
81     "birthday"     varchar,
82     "gender"       varchar,
83     "country"      varchar,
84     "email"        varchar UNIQUE,
85     "mobile"       varchar,
86     "username"     varchar UNIQUE,
87     "password"     varchar,
88     PRIMARY KEY("user_id" AUTOINCREMENT)
89 );
90 CREATE TABLE IF NOT EXISTS "User_Calendar" (
91     "user_id"      INTEGER,
92     "calendar_id"  INTEGER,
93     FOREIGN KEY("calendar_id") REFERENCES "Calendar"("calendar_id"),
94     FOREIGN KEY("user_id") REFERENCES "User"("user_id")
95 );
96 CREATE TABLE IF NOT EXISTS "Calendar" (
97     "calendar_id"  INTEGER,
98     "string_calendar_id" TEXT UNIQUE,
99     "calendar_name" varchar,
100    PRIMARY KEY("calendar_id" AUTOINCREMENT)
101 );
102 CREATE TABLE IF NOT EXISTS "Type" (
103     "type_id"      INTEGER,

```

```

104      "document_type"  varchar,
105      PRIMARY KEY("type_id" AUTOINCREMENT)
106 );
107 CREATE TABLE IF NOT EXISTS "User_Event" (
108     "event_id"          INTEGER,
109     "created_by_user_id"  INTEGER,
110     "for_user_id"        INTEGER,
111     FOREIGN KEY("event_id") REFERENCES "Event"("event_id"),
112     FOREIGN KEY("created_by_user_id") REFERENCES "User"("user_id"),
113     FOREIGN KEY("for_user_id") REFERENCES "User"("user_id")
114 );
115 CREATE TABLE IF NOT EXISTS "Event" (
116     "event_id"          INTEGER,
117     "string_event_id"    INTEGER,
118     "event_name"         varchar,
119     "from_date"          varchar,
120     "from_time"          varchar,
121     "to_date"             varchar,
122     "to_time"             varchar,
123     "location"            varchar,
124     "event_type"          TEXT,
125     "calendar_id"        INTEGER,
126     PRIMARY KEY("event_id" AUTOINCREMENT),
127     FOREIGN KEY("calendar_id") REFERENCES "Calendar"("calendar_id")
128 );
129 CREATE TABLE IF NOT EXISTS "User_Document" (
130     "document_id"        INTEGER,
131     "user_id"             INTEGER,
132     "document_for_user"   INTEGER,
133     FOREIGN KEY("user_id") REFERENCES "User"("user_id"),
134     FOREIGN KEY("document_id") REFERENCES "Document"("document_id")
135 );
136 CREATE TABLE IF NOT EXISTS "Document" (
137     "document_id"        INTEGER,
138     "document_name"       varchar,
139     "document_language"   TEXT,
140     "document_data"       BLOB,
141     "document_version"    varchar,
142     "document_upload_date"  varchar,
143     "prefix" TEXT NOT NULL,
144     "path"   TEXT,
145     PRIMARY KEY("document_id" AUTOINCREMENT)
146 );
147
148 INSERT INTO "Mother_Language" ("language_id","language") VALUES (1,'Arabic');
149 INSERT INTO "Mother_Language" ("language_id","language") VALUES (2,'Persian');
150 INSERT INTO "Mother_Language" ("language_id","language") VALUES (3,'Englisch');
151 INSERT INTO "Mother_Language" ("language_id","language") VALUES (4,'Turkish');
152 INSERT INTO "Mother_Language" ("language_id","language") VALUES (5,'French');
153 INSERT INTO "Mother_Language" ("language_id","language") VALUES (6,'Urdu');
154 INSERT INTO "Mother_Language" ("language_id","language") VALUES (7,'German');
155
156 INSERT INTO "Role" ("role_id","role_type") VALUES (1,'Manager');
157 INSERT INTO "Role" ("role_id","role_type") VALUES (2,'Worker');
158 INSERT INTO "Role" ("role_id","role_type") VALUES (3,'Volunteer');
159 INSERT INTO "Role" ("role_id","role_type") VALUES (4,'Customer');

```

7.3 Application Properties

```
1 # Set Port Number and suffix
2 server.port=8080
3 spring.mvc.view.suffix=.jsp
4 # Set Port Number and suffix
5
6 #Set the upload file limit to unlimited !
7 spring.http.multipart.max-file-size=-1
8 spring.http.multipart.max-request-size=-1
9 spring.servlet.multipart.max-file-size=-1
10 spring.servlet.multipart.max-request-size=-1
11 #Set the upload file limit to unlimited !
12
```

Literaturverzeichnis

- [VAKS21] vaks.de: *Leitbild 2010*. <https://www.vaks.info/ueber-uns/leitbild/>, Abruf: 17.05.2021.
- [GurDn21] Codesido I. What is front-end development?. Guardian official website; 28 September 2009.
<https://www.theguardian.com/help/insideguardian/2009/sep/28/blogpost>. Abruf: 05.08.2021.
- [W3S21] World Wide Web Consortium. What is CSS? [online]. W3 official website.
<https://www.w3.org/standards/webdesign/htmlcss#whatcss>
Abruf: 05.08.2021.
- [TutPnt21] JavaScript – Overview [online]. TutorialsPoint official website.
https://www.tutorialspoint.com/javascript/javascript_overview.htm
Abruf: 05.08.2021.
- [CrsRpt21] ULTIMATE GUIDE – Front End Development vs Back End Development [online].
<https://www.coursereport.com/blog/front-end-development-vs-back-end-development-where-to-start>
Abruf: 05.08.2021.
- [NtCrft21] Web Server Survey [online]. Netcraft official; 19 September 2016
<https://news.netcraft.com/archives/2016/09/19/september-2016-web-server-survey.html>
Abruf: 05.08.2021.
- [MSDN21] MSDN official website [online]. MSDN official website;
https://developer.mozilla.org/en-US/docs/Learn/Server-side/First_steps/Web_frameworks
Abruf: 05.08.2021.
- [MoAy17] Monelli Ayyavaraiah, Arepalli Gopi - Database Management System - 1. Edition - Horizon Books (A Division of Ignited Minds Edutech P Ltd) ; 2017 - P. 51.
- [SqLe21] SQLite official website [online];
https://www.sqlite.org/aff_short.html
Abruf: 09.08.2021
- [GogCa21] Google for Developer [online]; Google Calendar API
<https://developers.google.com/calendar/api/guides/overview>
Abruf: 10.08.2021
- [GogCh21] Google for Developer [online]; Using Google Charts
<https://developers.google.com/chart/interactive/docs>
Abruf: 10.08.2021
- [Ora21] Oracle JavaMail API
<https://docs.oracle.com/de-de/iaas/Content>Email/Reference/javamail.htm>
Abruff: 24.09.2021

-
- [GogCld21] Google Cloud Platform [online]; Dialogflow Documentation
<https://cloud.google.com/dialogflow/docs>
Abruf: 10.08.2021
- [IaND21] What Are The Benefits of MVC? [online]; Ian Davis
<https://blog.iandavis.com/2008/12/what-are-the-benefits-of-mvc/>
Abruf: 16.08.2021
- [OnVhf21] On2Vhf | MVC Framework
<https://de.on2vhf.be/mvc-framework-tutorial>
Abruf: 17.08.2021
- [Balng21] Template Engines for Spring, Baeldung 2020
<https://www.baeldung.com/spring-template-engines>
Abruf: 19.08.2021
- [GoGDF21] Google Cloud [online]; Google Dialog Flow | Guide
<https://cloud.google.com/dialogflow/es/docs/how/knowledge-bases#create-kb-web>
Abruf: 24.08.2021
- [BkVh18] KNOWLES,Bran;HANSON,VickiL.The wisdom of old technology (non)users. Communications of the ACM. 2018, vol. 61, no. 3, pp. 72– 77. Available from DOI: 10.1145/3179995.

Abbildungsverzeichnis

2.1	Front End vs Web Design	5
2.3.1	Kommunikation zwischen Server und Client ...	7
3.3.1a	Manager Manage User	17
3.3.1b	Manager Calendar	18
3.3.1c	Manager Manage Event	18
3.3.1d	Manager Document	19
3.3.1e	Manager Statistics	20
3.3.2	MVC Architektur	22
3.3.3	Datenbankschema	24
4.1	Projektstruktur	25
4.8	Chatbot Szenario	24
4.9	Die Architektur des Projekts	18

Tabellenverzeichnis

4.2a	User Befugnisse	27
4.2b	Event Befugnisse	28
4.2c	Document Befugnisse	29
4.2d	Besitzung eines Kalenders	30
4.2e	Zugriff auf Statistiken	30
5.2	Test-Umfrage	48

Abkürzungsverzeichnis

VAKS	Der Verein Für Soziale Arbeit und Kultur Südwestfalen
RLC	Refugee Law Clinic
SQL	Structured Query Language
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
CSS	Cascading Style Sheets
JS	Javascript
API	Application Programming Interface
ÖPNV	Öffentlicher Personennahverkehr
MVC	Model - View - Controller
JSP	Java Server Page
JDBC	Java Database Connectivity
OAuth	Open Authorization
NLP	Natural language processing
FAQs	Frequently Asked Questions