HTL Leonding
next level

HÖHERE TECHNISCHE
BUNDESLEHRANSTALT
LEONDING

LeoMail - Connect, Communicate, Coordinate

DIPLOMARBEIT

verfasst im Rahmen der

Reife- und Diplomprüfung

an der

Höheren Abteilung für Informationstechnologie - Medientechnik

Eingereicht von:
Tommy Neumaier
Lana Šekerija

Betreuer:
Prof. Natascha Rammelmüller
Prof. Michael Palitsch-Infanger

Projektpartner:
X-Net Services GmbH

Leonding, April 2025

Ich erkläre an Eides statt, dass ich die vorliegende Diplomarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe.

Die Arbeit wurde bisher in gleicher oder ähnlicher Weise keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Die vorliegende Diplomarbeit ist mit dem elektronisch übermittelten Textdokument identisch.

Leonding, April 2025 T. Neumaier & L. Šekerija

Danksagung

An dieser Stelle möchten wir allen herzlich danken, die uns während der Erstellung unserer Diplomarbeit *LeoMail - Connect, Communicate, Coordinate* auf vielfältige Weise unterstützt haben.

Zuerst gilt unser Dank einander: Die enge Zusammenarbeit war in vielen herausfordernden Momenten eine große Hilfe und hat unsere gemeinsame Arbeit inhaltlich und persönlich bereichert.

Ein besonderer Dank gilt Herrn Prof. Michael Palitsch-Infanger, welcher uns bei der Ausarbeitung der schriftlichen Arbeit wesentlich unterstützte und mit wertvollen Rückmeldungen zur Weiterentwicklung beitrug, und Frau Prof. Natascha Rammelmüller, welche uns mit ihrer umfassenden technischen und organisatorischen Begleitung wesentlich vorangebracht hat.

Ebenso danken wir Herrn Prof. Christian Aberger für seine hilfreichen Hinweise und seine technische Unterstützung, die das Gelingen unserer Arbeit wesentlich mitgeprägt hat.

Ein herzliches Dankeschön auch an Herrn Prof. Peter Bauer, der sich mit großem Engagement eingebracht hat und uns durch seine Anregungen, Ideen und Gespräche stets motiviert hat.

Abschließend möchten wir uns bei Herrn Nikolaus Dürk und Herrn Wolfgang Eibner von der Firma *X-Net Services GmbH* bedanken. Durch das von ihnen ermöglichte Sommerpraktikum im Jahr 2024 konnten wir bereits im Rahmen eines vierwöchigen Pflichtpraktikums wesentliche Teile unseres Projekts umsetzen und dadurch ein deutlich ausgereifteres Ergebnis erzielen. Besonders hervorzuheben ist dabei die wertvolle Unterstützung durch das gesamte Team der X-Net.

Ohne die Hilfe und das große Engagement all dieser Personen wäre die erfolgreiche Umsetzung unserer Diplomarbeit in dieser Form nicht möglich gewesen.

Vielen herzlichen Dank!

Abstract

Communication within schools, especially with numerous stakeholders, can often become inefficient and impersonal. Typically, school-wide messages and updates are managed manually, leading to outdated information and generic mass emails.

With *LeoMail – Connect, Communicate, Coordinate*, we present a software solution designed to unify and optimize school communications. The application facilitates continuous updating of contact data and the creation of reusable, personalized emails, significantly enhancing both internal and external stakeholder communication.

Die Kommunikation in Schulen kann besonders bei einer Vielzahl von Beteiligten oft ineffizient und unpersönlich werden. Schulweite Nachrichten und Aktualisierungen werden häufig manuell verwaltet, was zu veralteten Informationen und unpersönlichen Massen-E-Mails führt.

Mit *LeoMail – Connect, Communicate, Coordinate* stellen wir eine Softwarelösung vor, welche die schulische Kommunikation vereinheitlicht und optimiert. Die Anwendung ermöglicht eine kontinuierliche Aktualisierung von Kontaktdaten sowie die Erstellung wiederverwendbarer, personalisierter E-Mails und verbessert somit die Kommunikation mit internen und externen Stakeholdern erheblich.

Inhaltsverzeichnis

1 Einleitung

1.1 Ausgangslage

Die HTL Leonding organisiert jedes Jahr eine Vielzahl an Projekten, darunter den Project Award, den Company Thesis Day und den jährlichen Maturaball. Diese Projekte erfordern ein hohes Maß an Organisation und Zeitmanagement. Dabei ist es essenziell, dass sich die Schule und das Organisationsteam professionell präsentieren.

Da die Organisation von Projekten viel sorgfältige Arbeit erfordert, können durch Unachtsamkeit Fehler passieren, die schnell unprofessionell wirken können. Ein professionelles Auftreten ist vor allem in der Kommunikation mit externen Personen oder Unternehmen entscheidend. Dafür wird meistens über die E-Mail-Adresse Kontakt aufgenommen. Dabei sind sowohl die Struktur als auch die persönliche Anrede der Mail sehr wichtig.

Dem Maturaballkomitee 2024 unterlief bei der Ansprache von Sponsoren ein solcher Fehler. Es wurden E-Mails an sehr viele Unternehmen geschickt, wobei durch ein Missgeschick beim Versenden der Mails die Möglichkeit bestand, dass die Empfänger alle anderen E-Mails und Ansprechpersonen der Sponsoren sehen konnten. Dies geschah, weil die einzelnen Personen beim E-Mail-Versand unter „An“ und nicht unter „BCC“ aufgelistet wurden. Außerdem wurde eine generalisierte Anrede benutzt, obwohl viele der Sponsoren bereits vorher persönlich angesprochen wurden. Dies ließ das Maturaballkomitee und die Schule unprofessionell und unorganisiert wirken.

Um diese Fehler zu vermeiden, haben wir LeoMail an der HTL Leonding ins Leben gerufen.

1.2 Problemstellung

Derzeit verfügt die HTL Leonding über kein zentrales E-Mail-Verwaltungssystem, das es ermöglicht, personalisierte und gruppenbasierte E-Mails effizient zu versenden. Bei der Zusammenarbeit mit zahlreichen Projektbeteiligten stellt dies ein großes Problem dar.

Die Personalisierung von E-Mails wird dadurch stark eingeschränkt oder sogar komplett vernachlässigt, weil es viel Zeit in Anspruch nimmt, jeden Empfänger individuell anzusprechen. Daher greift man tendenziell zu generalisierten Anreden wie „Sehr geehrte Damen und Herren“, was den professionellen Auftritt der Schule schwächt.

Jedes Jahr werden an der Schule viele Projekte durchgeführt, bei denen häufig beinahe gleichbleibende Ansprechpersonen und externe Partner kontaktiert werden. Dies führt zu zusätzlichem Aufwand, da jede E-Mail-Adresse der Ansprechpersonen erneut recherchiert und kontaktiert werden muss. Zudem muss der E-Mail-Text jedes Jahr neu formuliert werden, was zeitintensiv ist.

1.3 Aufgabenstellung

Die Anforderung besteht darin, ein System zu entwickeln, das die Erstellung und den Versand sowohl personalisierter als auch gruppenbasierter E-Mails ermöglicht. Diese Software soll über eine benutzerfreundliche Oberfläche verfügen, die den Nutzern das Erstellen und die parallele Verwaltung von Projekten erlaubt. Der erste Schritt des Nutzers wäre die Erstellung eines Projekts, das verwaltet werden kann. Hier hat man die Möglichkeit, E-Mails von einer bestimmten E-Mail-Adresse zu versenden und Vorlagen sowie Gruppen für dieses Projekt zu erstellen.

Projekte

Jedes Projekt stellt ein eigenes Objekt dar, welches individuell konfigurierbar ist. Es soll aber auch eine Möglichkeit bestehen, Projekte zu duplizieren, wobei eine Kopie erstellt werden soll, damit die Originalversion nicht überschrieben wird. Dies gibt dem Nutzer die Möglichkeit, viel Zeit zu sparen, wenn er wiederkehrende Projekte erstellt. Hier sollen Vorlagen sowie Gruppen und versendete E-Mails dem Nutzer als Kopie zur Verfügung gestellt werden, welche wiederum vom Nutzer bei Änderungen angepasst werden können. Jedes Projekt kann mehrere Projektmitglieder haben, die vom Projektadministrator hinzugefügt werden können. Dies bedeutet, dass mehrere Projektmitglieder gleichzeitig an einem Projekt arbeiten können.

Externe Personen

Im System sollte es möglich sein, externe Personen, die nicht in der internen Schuldatenbank vorhanden sind, einzufügen, zu verwalten und diese dann auch in Gruppen hinzuzufügen. Dies soll die Möglichkeit bieten, bei Projekten mit externen Stakeholdern effizienter Kontakt aufzunehmen.

Vorlagen

Für jedes Projekt soll es möglich sein, individuelle E-Mail-Vorlagen zu erstellen. Diese können individuell angepasst werden und bieten die Möglichkeit, personalisierte E-Mails zu versenden. Es soll aber auch möglich sein, allgemeine Anreden zu benutzen, um breitere Zielgruppen anzusprechen. Die Vorlagen sollen auch alle Formatierungsoptionen und Anpassungen bieten, um jede E-Mail schön formatiert darzustellen.

Gruppen

Ein besonders wichtiger Punkt ist die Möglichkeit, die Gruppen, an die die E-Mails versendet werden, zu personalisieren. Die Software soll die Möglichkeit bieten, jede Gruppe individuell zu konfigurieren und zu verwalten. Um maßgeschneiderte E-Mails zu verschicken, soll die Software die Anpassung der Anrede, des Inhalts und anderer relevanter Details sicherstellen.

Geplante Mails

Zusätzlich soll die Software die Möglichkeit bieten, Mails zu terminieren. Nutzer sollen beim Versenden der Mail die Funktion haben, die Nachricht zu einem späteren Zeitpunkt zu verschicken. Die geplanten E-Mails sollten jederzeit bearbeitet und angepasst werden können, solange sie noch nicht versendet wurden. Dies bietet dem Nutzer oder Projektteam Flexibilität und die Möglichkeit, Änderungen auch im letzten Moment vorzunehmen.

Zusammenfassung der Hauptfunktionen

Zusammenfassend soll die Software folgende Hauptfunktionen bieten:

1. **Erstellung und Verwaltung von mehreren Projekten**: Jedes Projekt kann individuell konfiguriert und verwaltet werden.
2. **Projektmitglieder hinzufügen**: Der Projektadministrator hat die Möglichkeit, Projektmitglieder zu Projekten hinzuzufügen.
3. **Hinzufügen von externen Personen**
4. **Erstellung von E-Mail-Vorlagen**: Vorlagen können für jedes Projekt individuell erstellt und personalisiert werden.
5. **Erstellung von Gruppen**: Gruppen können spezifisch konfiguriert werden, um personalisierte Nachrichten zu versenden.
6. **Terminierung der E-Mails**: E-Mails können zu einem späteren Zeitpunkt verschickt und bis dahin jederzeit bearbeitet werden.

1.4 Zielsetzung

Das Ziel ist es, der HTL Leonding eine Software zur Verfügung zu stellen, die den E-Mail-Versand effizient gestaltet und dadurch erheblich Zeit bei der Organisation von Projekten einspart. Mit unserer Software möchten wir sicherstellen, dass die HTL Leonding stets einen professionellen Eindruck hinterlässt.

Unsere Lösung soll es ermöglichen, personalisierte und gruppenbasierte E-Mails problemlos zu versenden, mehrere Projekte gleichzeitig zu verwalten und externe Personen in Gruppen hinzuzufügen. Darüber hinaus soll sie die Erstellung und Nutzung von E-Mail-Vorlagen sowie die Terminierung von E-Mails unterstützen.

Das letztendliche Ziel ist die Bereitstellung einer technisch sauberen Software, die sämtliche Anforderungen erfüllt.

Nicht-Ziele

Es soll sichergestellt werden, dass die Diplomarbeit nicht den Eindruck eines Event-Management-Tools erweckt. Dies soll strikt getrennt werden, um den Zweck der Software hervorzuheben.

Nice-to-have-Ziele

Die Software kann vielseitig erweitert werden. Eine Idee wäre, ein Vier-Augen-Prinzip zu implementieren, bei dem ein Projektmitglied die E-Mail vor dem Versenden überprüft und sie erst nach seiner Bestätigung verschickt wird. Dieses Prinzip würde sicherstellen, dass jede E-Mail doppelt geprüft wird und somit inhaltliche oder andere Fehler leichter vermieden werden können.

1.5 Aufgabendifferenzierung

Um die Applikation umzusetzen, wird eine klare Definition für die Aufteilung der verschiedenen Aufgaben benötigt.

Tommy Neumaier

Tommy Neumaier beschäftigt sich im Rahmen der Diplomarbeit mit den Themengebieten CI/CD und Deployment, User-Authentifizierung, den E-Mail-Protokollen und der Bereitstellung von Daten über Backend-APIs wie REST und WebSockets.

Lana Šekerija

Lana Šekerija beschäftigt sich in der Diplomarbeit mit den Aspekten des User Interface/User Experience-Design (UI/UX), insofern auch mit den Prinzipien von Farben und Design. Zusätzlich implementiert sie mittels der verwendeten Frontend-Technologien die Web-Anwendung und arbeitet an der Integration des Backends in jene Anwendung.

2 Umfeldanalyse

Prinzipiell lässt sich vereinfacht und verallgemeinert sagen, dass jeder E-Mail-Client ein Konkurrenzprodukt für *LeoMail - Connect, Communicate, Coordinate* ist. Denn in der Theorie könnte eine E-Mail, welche beispielsweise eine Vielzahl von Kontaktpersonen von Unternehmen personalisiert erreichen soll, auch einzeln an jede geschickt werden. Dies würde allerdings einige Stunden oder sogar Arbeitstage dauern. In der Folge werden also nur jene (Online-)Dienste näher beschrieben, welche spezielle Funktionen für die Kommunikation mit einer Vielzahl an Personen oder Gruppen bieten.

2.1 Microsoft Outlook

Outlook, welches erstmals im Januar 1997 von Microsoft veröffentlicht wurde, ist heute einer der funktional breitesten Mail-Client-Dienste. Durch die Integration mit anderen Microsoft-Services, speziell denen des Office 365-Pakets wie Word, Excel, PowerPoint oder Access, ist er zu einem der beliebtesten Services in dieser Nische aufgestiegen [54].

Die nachfolgende Funktionalitätsbeschreibung kann sowohl auf die Web- als auch auf die Desktopversion bezogen werden.

Abbildung 1: Logo von Microsoft Outlook [47]

Für sogenannte Bulk-Emails bietet der Mail-Client auch die Funktion an, Serienfelder ("Mail Merge Fields") anzulegen. Hier können Personen- oder sonstige Daten, die in einer tabellarischen Struktur wie bspw. Excel oder Access angelegt wurden, automatisch pro Empfänger eingearbeitet und versendet werden [54].

Weiters können hier auch "Contact Lists", dt. *Kontaktgruppen*, angelegt werden. An diese können nun E-Mails versendet werden, ebenfalls mit Serienfeldern oder generalisiert.

Hier allerdings fehlt die Möglichkeit, gleich bestehenden Kontakten (die durch den Administrator des Mailservers bspw. automatisch eingebunden wurden) Attribute zuzuweisen und diese wiederzuverwenden – das muss immer über neue bzw. wiederverwendete Excel-/Access-Listen geschehen, auch die Möglichkeit, Open-Source-Datenbanken oder -Tabellenformate zu verwenden, fehlt gänzlich.

Zusätzlich ist es nicht möglich, klassische Vorlagen direkt im Client zu lagern und mit anderen zu teilen – diese können zwar zu Dateien mit der Endung *.txpt* exportiert werden, allerdings führt das beim Weiterleiten dieser Datei beispielsweise zu Problemen, wenn ein Mac-Betriebssystem eine von einem Windows-Betriebssystem oder vice versa gespeicherte Vorlage verwenden möchte, da die Metadaten der Datei nicht standardisiert sind und die Verknüpfung zu der Datenquelle nicht einwandfrei funktioniert [18].

Es gibt außerdem nicht die Möglichkeit, gemeinsam an sogenannten Projekten zu arbeiten, also gemeinsam Vorlagen zu bearbeiten, versendete E-Mails unter einer gemeinsamen E-Mail-Adresse zu versenden, ohne den Konfigurationsprozess dieser durchlaufen zu müssen. Das führt im besten Fall nur dazu, dass doppelt gearbeitet wird und im schlimmsten Fall dazu, dass die Kommunikation – speziell bei Projekten und nicht einzelnen Situationen – nicht korrekt aufgeteilt wird und E-Mails doppelt in möglicherweise verschiedenen Versionen versendet werden.

Zu guter Letzt gibt es bei der Verwendung von Outlook auch größere Datenschutzbedenken – seit einem Update im Dezember 2023 wird nämlich bei der Anmeldung mit einem E-Mail-Konto, unabhängig davon, ob Microsoft auch der Mail-Provider ist, das Konto nicht direkt mit dem jeweiligen Provider authentifiziert, sondern als *Man in the Middle* auch über die Microsoft Cloud geleitet [44, 42]. Das geschieht, um ein sogenanntes intelligentes Postfach mit Unterstützung künstlicher Intelligenz via Cloud Computing zu ermöglichen. Dabei werden ebenfalls Anmeldedaten in der Cloud hinterlegt, sodass Microsoft, unabhängig von der Nutzung des Postfachs, auch selbständig darauf zugreifen könnte. Zusätzlich muss Microsoft als US-amerikanisches Unternehmen dem Auslandsgeheimdienst der Vereinigten Staaten, der NSA, auch ohne richterlichen Beschluss im Rahmen des *USA PATRIOT ACT* alle personenbezogenen Daten aushändigen, die sie verlangen [14].

2.2 Mozilla Thunderbird

Thunderbird, welches von der Mozilla Foundation erstmals im Juli 2003 veröffentlicht wurde, ist ein Open-Source-Mail-Client, welcher seinen Hauptfokus im Gegensatz zu bestehenden Alternativen auf den Aspekt der Privatsphäre und Datensicherheit legt [59].

Abbildung 2: Logo von Mozilla Thunderbird [49]

Das untermauert Mozilla auch mit den zahlreichen Sicherheitsoptionen: E-Mails können beispielsweise auch mittels OpenPGP-Key verschlüsselt werden, Script-Features wie JavaScript werden standardmäßig in E-Mails deaktiviert. Auch große Organisationen wie das französische Militär setzen auf Thunderbird und tragen zu seiner Entwicklung bei.

Ähnlich wie Outlook hat auch Mozilla die Möglichkeit, Kontakte zu erstellen; die Möglichkeit, diese in Listen zu gruppieren, fehlt allerdings gänzlich.

Das Feature, Serienfelder zu verwenden, fehlt in der Standardversion von Thunderbird allerdings komplett – es ist über Erweiterungen installierbar, was aber dem Plug-N-Go-Prinzip von benutzerfreundlichen Anwendungen entspricht [13]. Auch hier können weitere Attribute zu den Kontakten nur über externe tabellarisch strukturierte Listen gespeichert werden – dafür funktionieren hier auch Open-Source-Dateiformate wie CSV.

Die Integration von externen Services ist in Thunderbird nicht gegeben, was allerdings auch dem Fokus auf Datenschutz widersprechen würde. Allerdings beinhaltet Thunderbird einen integrierten Firefox-Browser, welcher ebenfalls Open-Source ist.

Ebenfalls existiert die gemeinsame Bearbeitung und Sicherung von E-Mails nicht; ähnlich wie bei Outlook gibt es in dieser Hinsicht ausschließlich den Export von E-Mail-Vorlagen.

2.3 Sonstige

Es existieren eine Vielzahl von Mail-Clients, die grundlegende Versandfunktionen unterstützen, allerdings haben diese – bis auf wenige Paid-Dienste – keine ausgereiften Möglichkeiten zur Massenkommunikation.

2.4 Fazit

Insgesamt ist die Auswahl an Mail-Clients, die zumindest grundlegende Features in Hinsicht auf serielle Kommunikation und Kooperation beim Verfassen und Bearbeiten von E-Mails bieten, mit *Microsoft Outlook* und *Mozilla Thunderbird* etwas dünn.

3 Projektorganisation

Die Organisation eines Software-Projekts unterscheidet sich aufgrund seiner Charakteristiken maßgeblich von der Entwicklung herkömmlicher, analoger Produkte. Durch die Dynamik und die Anpassungsfähigkeit eines Software-Produktes, im Gegensatz zu materiellen Produkten, ist ein Software-Produkt nie absolut fertiggestellt. Im laufenden Einsatz werden bei allen Produkten laufend Fehler oder Erweiterungsmöglichkeiten entdeckt, diese sind aber bei analogen Produkten kaum bis gar nicht erweiterbar, während Software im Gegensatz dazu sehr einfach ausgetauscht oder erweitert werden kann.

3.1 Software Development Life Cycle

Der sogenannte **Software Development Life Cycle** (SDLC) beschreibt den klassischen Ablauf der Entwicklung eines Software-Produktes [10, 3].

Requirement Engineering

Im ersten Schritt wird eine Anforderungserhebung durchgeführt – also erhoben, welche Anforderungen seitens des Kunden an die Software gestellt werden. Dazu führen verschiedenste Stakeholder des Auftragnehmers, wie der Projektmanager oder auch mit dem Kunden vertraute Software-Entwickler, Gespräche mit dem Auftraggeber. Hier ist oft auch die Erfahrung in der Praxis von Bedeutung, denn die Erfahrung der Software-Entwickler kann dem Kunden maßgeblich dabei helfen, zu verstehen, was er tatsächlich benötigt und wie Arbeitsabläufe des Kunden beispielsweise digitalisiert oder optimiert werden können, da bei branchenfremden Kunden oft noch keine klare Vorstellung der tatsächlichen Anforderungen vorliegt [87].

Oft werden nicht nur Gespräche geführt, sondern auch Fragebögen mit sich im Inhalt der Antwort überschneidenden Fragen gestellt, um die Wünsche des Kunden überprüfen zu können und seine Konstanz bezüglich dieser sicherzustellen.

Mit dem Entwicklungsprozess vertraute Kunden bringen bereits bei der Auftragssuche ein Lastenheft mit, welches die Anforderungen an das Produkt bereits konkretisiert.

Entwurf und Planung

Im nächsten Schritt zieht sich der Auftragnehmer zurück und plant die Umsetzung des geforderten Produktes. Weiters wird ebenso eine Aufwandsschätzung erstellt, anhand derer auch die Kalkulation des Finanzrahmens des Produktes erfolgt [8].

Im Rahmen des Entwurfs eines Software-Projektes erfolgt außerdem die Erstellung eines Pflichtenheftes – in diesem wird beschrieben, wie die Anforderungen des Lastenheftes, welches der Auftraggeber erstellt, konkret erfüllt werden sollen. Dieses ist äußerst nützlich für beide Seiten, da es die konkreten Funktionen schriftlich festlegt und man sich seitens beider Projektpartner darauf berufen kann.

Der Umfang der Planung hängt generell auch sehr stark davon ab, welche Projektorganisationsmethode man wählt: Während bei der konventionellen Projektorganisation grundsätzlich bereits ein sehr konkreter Entwurf vorliegt, plant sich das Produkt bei der agilen Projektorganisation sozusagen im Prozess „von selbst“.

Dieser Phase kann ebenfalls die Einrichtung von etwaigen Entwicklungsressourcen zugeschrieben werden: Im Rahmen der rechtlichen Möglichkeiten durch den Projektvertrag werden beispielsweise einzelne Software-Bereiche an andere Unternehmen ausgelagert, es wird etwaige Entwicklungshardware angeschafft oder bereitgestellt und ein Projektteam wird zusammengestellt.

Noch vor der Zusammenstellung des Projektteams muss sich auch auf konkrete Technologien, auch genannt einen **Tech Stack**, verständigt werden. Dies kann beispielsweise durch eine Nutzwertanalyse geschehen.

Nutzwertanalyse

Die Nutzwertanalyse ist ein analytisches Werkzeug, welches es erleichtert, bei einem Software-Produkt zwischen mehreren alternativen Vorgehensweisen aufgrund qualitativer und quantitativer Kriterien eine Entscheidung zu treffen. Beispielhaft ist die durchgeführte Nutzwertanalyse zur Entscheidung der Backend-Technologie.

Zunächst werden erstmal die Alternativen aufgelistet, welche zur Debatte stehen. Im Anschluss daran verständigt man sich auf Bewertungskriterien, welche man dann – im Idealfall in der Gruppe – gewichtet, um die Priorität gewisser Kriterien in die Bewertung miteinfließen zu lassen.

Danach kommt es zur Bewertung der Kriterien: sinnvoll ist hier beispielsweise ein inverses Notensystem von 1–5 oder 1–10, wobei die Bewertung umso besser ist, je höher die Zahl. Auch hier ist es wieder zweckmäßig, innerhalb des Projektteams oder auch gemeinsam mit dem Auftraggeber zu entscheiden, und bei knappen Entscheidungen auch mehrere Durchläufe mit verschiedenen Kriterien und Gewichtungen durchzuführen.

Implementierung

Nach der Planung und der Abnahme des Projektentwurfs durch den Auftraggeber erfolgt die Implementierung des Projektes durch das vom Auftragnehmer zusammengestellte Projektteam. Auch hier unterscheidet sich der genaue Vorgang von der gewählten Projektorganisationsmethode.

Gerade bei der Implementierung stellen Versionskontrolle sowie sogenannte IDEs eine enorme Abhilfe dar [11].

Testing

Im Laufe der Implementierung kann und sollte parallel auch bereits die Anwendung getestet werden – dabei ist vor allem zwischen sogenannten Usertests, Systemtests und Anwendungstests zu unterscheiden [7].

Usertests

Usertests können in drei Phasen unterteilt werden:

Anfangs wird eine Funktion meist direkt im Prozess der Implementierung seitens des Entwicklers der Funktion lokal am Endgerät des Software-Entwicklers getestet. Hierbei variiert die Qualität des Testens stark, oft können Fehler nicht entdeckt werden, gerade im Zusammenspiel mit weiteren abhängigen Funktionen.

Danach wird im Rahmen des Mergings von Feature-Branches (über die Versionskontrolle) die Funktion von mehreren Entwicklern überprüft. Hierbei werden mehr Fehler

aufgedeckt als in der vorherigen Phase, allerdings können Problematiken eines Produktivsystems (wie beispielsweise Race Conditions) weiterhin nicht abgebildet werden.

Um diese abzudecken, kommt es nun zum Staging der Anwendung: Hierbei wird die Anwendung erstmals als „Pseudo“-Produktivsystem für eine Gruppe von Entwicklern oder auch für den Kunden zur Verfügung gestellt, um erste Erfahrungswerte für die Produktionsumgebung zu sammeln und auch etwaige fatale Fehler vor dem konkreten Einsatz festzustellen.

Systemtests

Systemtests werden durchgeführt, um die Belastbarkeit der Ressourcen beziehungsweise Systeme zu testen, auf denen die Anwendung laufen soll. Hierbei werden entweder manuelle Tests durchgeführt oder ein extremer Nutzertraffic simuliert, um die Resilienz eines Systems auch zu sogenannten „Peak-Zeiten“ überprüfen zu können.

Anwendungstests

Anwendungstests werden direkt in der Anwendung programmiert oder können auch außerhalb als eigenständige Applikation ausgeführt werden. Dazu zählen beispielsweise Schnittstellentests, welche die für die Anwendung relevanten REST-Schnittstellen mit vordefinierten Anfragen auf die zu erwartenden Ergebnisse prüfen, oder auch automatisierte Usertests, welche das Verhalten von Nutzern in der Anwendung mittels Tastatureingaben oder Mausklicks simulieren sollen.

Bereitstellung

Bei der Bereitstellung ist die eigentliche Entwicklung abgeschlossen, alle Tests wurden positiv abgeschlossen und die Projektabnahme ist bereits prinzipiell erfolgt. Hier wird nun das Produkt öffentlich zugänglich gemacht oder dem Unternehmen zur Nutzung zur Verfügung gestellt.

Wartung und Weiterentwicklung

Die Wartung einer Anwendung ist die letzte Phase. Zwar kann es sein, dass seitens des Auftraggebers keine spätere Wartung durch den Auftragnehmer mehr vorgesehen ist – beispielsweise, wenn der Auftraggeber selbst dazu im Stande ist, etwaige Wartungen durchzuführen; in der Regel passiert dies aber nicht.

In der Wartung werden etwaige Fehler, welche im Produktiveinsatz auftreten, behoben. Auch Software-Updates, die kritisch sind (also beispielsweise erforderlich für den Betrieb der Ordnung) oder auch aufgrund von Sicherheitsbedenken notwendig sind, werden durchgeführt.

Außerdem kann es auch zur Weiterentwicklung durch die Entwicklung neuer Funktionen kommen: Bei sauberem Vorgehen durchläuft jede neue Funktion erneut den **Software Development Life Cycle** in kleinem Rahmen, in der Praxis ist dies jedoch oft nicht der Fall.

Durch die Wartung und die Weiterentwicklung wird hier außerdem ein weiterer Charakterzug eines Software-Produktes definiert: Ein Software-Projekt ist meist endlos und wird per se nie absolut abgeschlossen.

3.2 Organisationsmethodik

Für die Organisation und Entwicklung von Projekten gibt es in der Software-Entwicklung prinzipiell zwei Ansätze: den konventionellen sowie den agilen Entwicklungsansatz.

Bei der konventionellen Methode, die auch oft "klassische"Methode genannt wird, wird das finale Produkt so gut wie möglich geplant. Ein gutes Beispiel hierfür sind die Produktion von Autos, Hardware oder sonstigen dinglichen Produkten, wo eine konkrete und weitreichende Planung zur Vermeidung von unnötigen Mehrkosten sinnvoll ist.

Durch diese strikte Planung und Konzeption offenbart sich in der klassischen Produktentwicklungsform ein großes Problem: Software-Projekte müssen meist an den Zeitgeist und die damit verbundenen Herausforderungen flexibel angepasst werden. Das wird durch eine agile Methode, wie beispielsweise Scrum, mit iterativen Prozessen gewährleistet [9].

Statt aufeinanderfolgenden Planungs-, Umsetzungs- und Produktionsphasen laufen diese nun parallel zueinander ab: In sogenannten „Sprints“ wird jeweils ein „minimum viable product“ festgelegt – also ein funktionierendes, minimalistisches Produkt, welches in sogenannten „User Stories“ beschrieben wird.

Diese User-Stories beschreiben die Anwendung aus Sicht des Nutzers: Jeden Sprint soll eine davon erfüllt werden. Am Ende eines Sprints steht immer ein sogenanntes „Sprint Review“, an dem das Team als Einheit (oder vertreten durch eine Person) mit den

Auftraggebern das zuletzt umgesetzte Produkt begutachtet und bewertet. Parallel dazu oder direkt im Anschluss wird ein sogenanntes „Sprint Planning“ durchgeführt, bei dem die umzusetzende User Story für den nächsten Sprint festgelegt wird [9].

Nach dem Sprint Planning verteilt ein sogenannter „Scrum Master“ aus der Mitte des Teams die Aufgaben für die jeweilige User Story möglichst gleichmäßig auf das Entwicklungsteam.

In professionell arbeitenden Unternehmen, je größer diese sind, kann das agile Projektmanagement in dieser Form sehr gut umgesetzt werden – in weniger perfekten Umgebungen kann das „Arbeiten nach Vorschrift“ hingegen schwieriger werden.

Tatsächliche Organisationsmethodik bei LeoMail

Im Projekt LeoMail wurde grundsätzlich ein agiler Organisationsansatz als Grundlage gewählt. Idealerweise zeichnet sich agiles Projektmanagement durch die strikte Einhaltung fester Zeitzyklen – beispielsweise zweiwöchige Sprints – sowie die konsequente Definition und Umsetzung von User Stories aus. In der Praxis konnte dieser Idealzustand jedoch nicht vollständig erreicht werden.

Auftraggeber war in erster Linie Prof. Peter Bauer, Abteilungsvorstand der HTL Leonding, während die beiden Diplomarbeitsbetreuer, Prof. Michael Palitsch-Infanger und Prof. Natascha Rammelmüller, eine beratende Funktion innehatten. Das Team verfügte von Beginn an über einen relativ hohen Gestaltungsspielraum: Nach einer initialen Ideensammlung wurde der erste Prototyp entwickelt. Erst im Anschluss daran erfolgte das erste Feedback, welches in die weitere Projektentwicklung einfloss – sinngemäß entsprach dies einer Iteration im agilen Vorgehen.

Rückblickend hätte eine strengere organisatorische Umsetzung des agilen Ansatzes in mehreren Bereichen Vorteile gebracht:

- **Strikte Aufgaben-Definition:** Es wurde primär an den angestrebten Feature-Zielen gearbeitet, während eine detaillierte Aufgabendefinition zugunsten der Flexibilität vernachlässigt wurde.
- **Ausformulierung von User Stories:** Die explizite Formulierung von User Stories, die aus Nutzersicht die Anforderungen beschreiben, hätte zu einer noch klareren Strukturierung beitragen können.

- **Feste Sprintzyklen:** Die Sprints erfolgten unregelmäßig. Eine feste zeitliche Struktur, entweder in Abstimmung mit dem Auftraggeber oder zumindest innerhalb des Teams, hätte den agilen Entwicklungsprozess optimiert.

3.3 Versionskontrolle

Was ist Git?

Git ist eine Open-Source-Software zur Versionskontrolle (Version Control System, VCS) von Software-Projekten, welche 2005 erstmals unter der Leitung des Linux-Erfinders Linus Torvalds veröffentlicht wurde. Sie ist heutiger Industriestandard und hat heute ein De-facto-Monopol auf diesem Markt.

Mittels Git können unterschiedliche Versionen von Software-Projekten lokal und remote als sogenannte Repositorys gespeichert werden. Änderungen, auch Commits genannt, werden als Increments erfasst und können jederzeit eingesehen und rückgängig gemacht werden – was die Basis für die heutige Software-Entwicklung darstellt, da Projekte heute immer schneller wachsen und die Menge an Code pro Applikation durchschnittlich steigt [11].

Umgangssprachlich wird Git oftmals mit den Onlinediensten GitHub oder GitLab gleichgesetzt, was jedoch falsch ist, da diese nur die kooperative Zusammenarbeit mittels Git ermöglichen, indem sie Repositories im Internet bereitstellen.

GitLab vs. GitHub vs. Gitea

GitLab, GitHub und Gitea sind Onlinedienste, die Teams bei der kooperativen Zusammenarbeit mittels Git unterstützen und Repositories online zur Verfügung stellen. Oftmals wird bei der Auswahl zwischen der Open-Core-Software GitLab und der proprietären Software GitHub von ideologischen Kriterien gesprochen, da sie sich sehr stark in ihrer Funktionalität ähneln. Sie haben ähnliche Preismodelle und ähneln sich in ihrer Struktur. Seitens GitHub werden immer wieder Bedenken bezüglich des Datenschutzes laut, da es im Besitz von Microsoft ist und auf Basis der von Nutzern erstellten Repositories deren KI-Assistenten Copilot trainiert. Gitea ist aufgrund seiner Open-Source-Zugänglichkeit von dieser Diskussion grundsätzlich ausgeschlossen [84].

Während GitHub rein cloudbasiert ist und das Hosten eigener Instanzen auf Maschinen nicht möglich ist, bietet GitLab diese Möglichkeit zum Self-Hosting zusätzlich zur cloudbasierten Variante, während Gitea nur auf eigenen Rechnern gehostet werden kann.

Alle drei bieten die Möglichkeit zur Erweiterung – vor allem GitLab und GitHub mit einer Vielzahl von Third-Party-Tools oder Plugins. GitHub und GitLab profitieren von einer großen Community und einer breiten Palette von Dokumentationen, während Gitea als Neuling dieser drei Systeme diesen Umfang noch nicht bieten kann.

Gitea hatte, besonders aufgrund der Leichtgewichtigkeit und seines Open-Source-Charakters, einen Vorteil bei der Auswahl des VCS-Providers. Da allerdings Gitea keinerlei Unterstützung für CI/CD-Pipelines und DevOps bietet und hier weitere Software benötigt wird, schied es bereits am Projektstart aus.

Anfangs wurde mit einer Self-hosted GitLab-Instanz der Partnerfirma von LeoMail, X-Net Services GmbH, gearbeitet – was im späteren Verlauf der Diplomarbeit, speziell im Hinblick auf das Deployment auf die LeoCloud und die zu erwartende Weiterentwicklung durch schulinterne Personen, zur Entscheidung für GitLab führte. Die potenzielle und einfache Wartbarkeit durch zukünftige Diplomarbeitsteams ist somit besser gegeben. Eine Verwendung einer eigenen GitLab- oder Gitea-Instanz für ein einziges Software-Projekt ist zudem nicht zielführend.

Tatsächliche Versionskontrolle bei LeoMail

LeoMail verwendet nun ein öffentliches GitHub-Repository zur Versionskontrolle. Die aktuellen Entwicklungsstände sind im „dev“-Branch einsehbar, während die aktuelle Stable-Version immer im „main“-Branch abrufbar ist. Der Inhalt des „main“-Branchs entspricht zugleich immer der aktuell veröffentlichten Version auf der LeoCloud, auf welcher die produktive Web-Anwendung dauerhaft verfügbar ist.

Dieses Repository ist unter https://github.com/htl-leo-itp-2325-4-5BHITM/leomail dauerhaft erreichbar.

3.4 Ideenfindung

Der Startschuss zur Ideenfindung wurde mit der Identifikation der Problemstellung begonnen: In Zukunft soll verhindert werden, dass E-Mails – speziell jene, die zur

Kommunikation mit Stakeholdern von außen dienen – ohne inhaltliche sowie personalisierungsbezogene Fehler versendet werden.

Dabei wurde in allen möglichen Konstellationen stets Brainstorming als Kreativitätsmethode gewählt. Brainstorming konzentriert sich methodisch stark auf eine Problemstellung oder ein Ziel, worauf alle Teilnehmer mit verschiedenen Perspektiven und Visionen ihren Blick richten und Lösungsansätze entwickeln. Die Vorschläge werden unabhängig von ihrer Ausgereiftheit nicht kritisiert und möglichst von anderen Teilnehmern ergänzt, bis man zu umsetzbaren und realistischen Lösungswegen kommt [80].

Brainstorming erweist sich als besonders effektiv, wenn Personen mit unterschiedlichen Hintergründen und Erfahrungen daran teilnehmen. Im Fall dieser Diplomarbeit trug das Partnerunternehmen X-Net Services GmbH wertvolle Ideen und Erfahrungen aus einem vergleichbaren Kundenprojekt bei. Die Betreuung durch Prof. Peter Bauer und Prof. Natascha Rammelmüller bot eine fundierte Einschätzung der konkreten Anforderungen der Schule. Das Projektteam brachte vor allem visionäre und kreative Impulse ein.

Einige Ideen, welche beim Brainstorming ermittelt wurden, wurden original oder abgeändert in das Feature-Set übernommen, während andere verworfen wurden. Weniger Ideen wurden auch in die Erweiterungsvorschläge für zukünftige Diplomarbeitsgruppen von LeoMail aufgenommen.

3.5 Feature-Beschreibung

Kontaktpersonen

Kontaktpersonen sind natürliche Personen, an welche man E-Mails versenden kann. Diese werden entweder automatisch aus der Benutzerdomäne der Schule importiert (beispielsweise Schüler und Lehrkörper) oder können manuell angelegt werden.

Sie dienen als Quelle für Informationen beim Rendern und Generieren der einzelnen E-Mails. Eingegeben werden müssen etwaige (akademische) Titel der Personen, Vor- und Nachname, eine E-Mail-Adresse sowie das Geschlecht.

Kontaktpersonen sind neben den Projekten die einzige Entität, welche global über die gesamte Anwendung hinweg angelegt wird.

Gruppen

Diese Kontaktpersonen werden außerdem automatisch oder manuell in Gruppen zusammengefasst. Dies ist relevant, wenn beispielsweise Kontaktpersonen als Empfänger von E-Mails gruppiert werden sollen und man manche E-Mails immer wieder an dieselben Personengruppen sendet. Den Gruppen werden Namen zugeordnet.

Projekte

Veranstaltungen, wiederkehrende Prozesse oder repetitive Aufgaben können als Projekte angelegt werden. Diesen Projekten können weitere Schulbenutzer zugeordnet werden, was eine kollaborative Zusammenarbeit im Bereich der Kommunikation ermöglicht. Diesen Projekten werden entsprechende Gruppen und Vorlagen zugeordnet sowie die gesendeten E-Mails aus den jeweiligen Vorlagen.

Vorlagen

Vorlagen spiegeln die vorgefertigten E-Mail-Nachrichten wider, welche je nach Empfängerdaten generiert und versendet werden können. Diese Vorlagen werden mittels eines Rich-Text-Editors bearbeitet, wobei alle möglichen Styling-Optionen zur Verfügung stehen. Der Betreff der E-Mail wird ebenfalls hinterlegt.

Es kann auch festgelegt werden, dass diese Vorlage das Hochladen von Dateien erfordert – diese müssen dann vor dem Versenden der E-Mail hochgeladen werden. Den Vorlagen können zudem Namen zugeordnet werden, um sie schneller in der Übersicht finden zu können.

(Geplante) Mails

Aus den Vorlagen können nun E-Mails versendet werden. E-Mails können auch dahingehend konfiguriert werden, dass sie zu einem bestimmten Zeitpunkt automatisch versendet werden. Beim Versenden werden nun auch die Empfänger, also einzelne Kontaktpersonen oder Gruppen, ausgewählt sowie die jeweilige Vorlage. Sollte für das Versenden dieser Vorlage ein Dateiupload notwendig sein, wird dies im Versendungsprozess ermöglicht. Vor der endgültigen Bestätigung besteht die Möglichkeit, die einzelnen personalisierten E-Mails auf Fehler zu überprüfen.

Persistenzentitäten

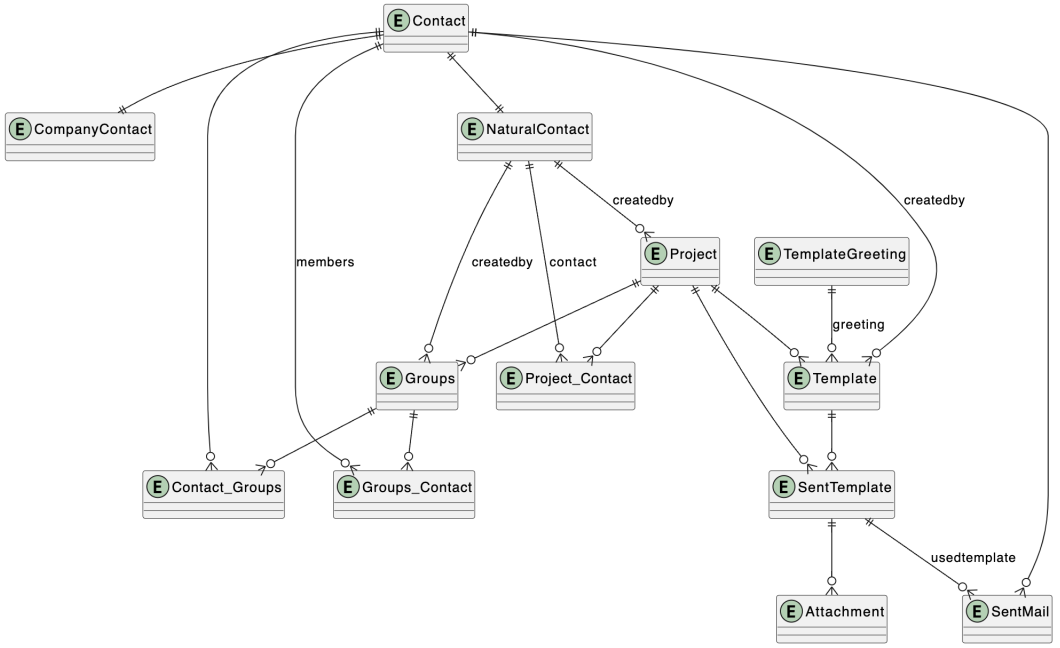

Abbildung 3: Entity-Relations-Diagramm im Backend

Systemübersicht

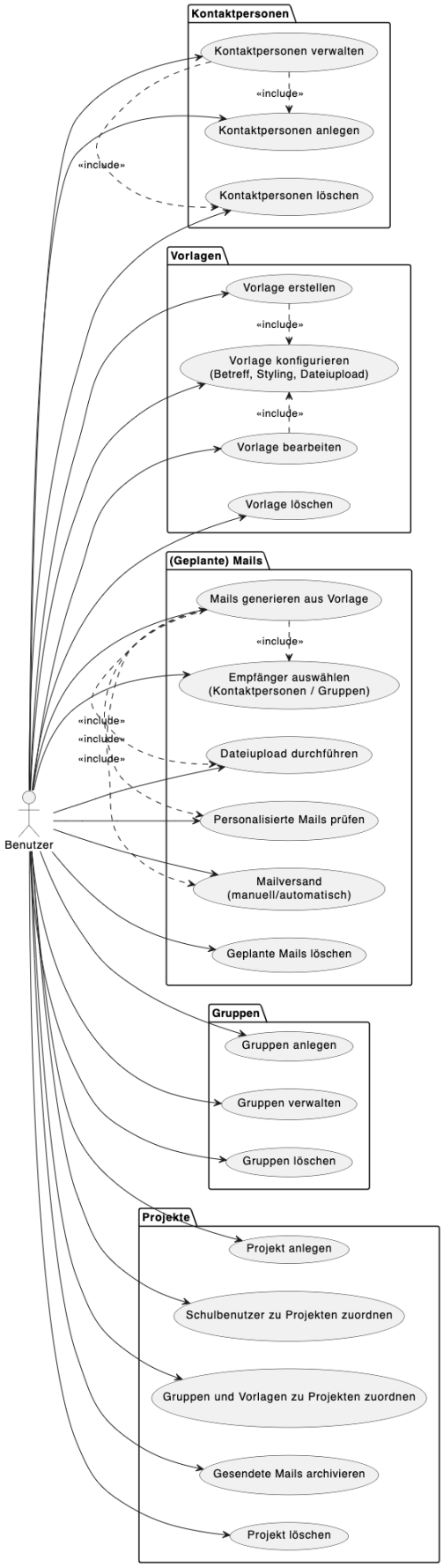

Abbildung 4: Use-Case-Diagramm der Web-Anwendung

3.6 Projektverlauf

Projektstart

Durch den Fehler, der dem Maturaballkomitee 2024 unterlaufen ist, ergab sich ein Kontakt zur Partnerfirma X-Net Services GmbH. Nach telefonischem sowie E-Mail-Kontakt und zwei Meetings ergab sich ein Praktikumsplatz für die Bearbeitung der praktischen Diplomarbeit.

Beim ersten Meeting wurde gemeinsam mit Hr. Wolfgang Eibner (Software-Leiter) sowie Hr. Nikolaus Dürk (Geschäftsführer der X-Net) ein erster Entwurf eines Konzeptes der Anwendung definiert. Nach einer darauffolgenden finalen Zusage für den Praktikumsplatz und die Kooperation wurde seitens des Projektteams der erste Kontakt mit der Schulleitung, Hr. Prof. Peter Bauer, gesucht und ein Anwendungsbereich für diese Anwendung festgelegt.

Zunächst stand es im Raum, dass diese Diplomarbeit auch Funktionen im Bereich der allgemeinen Veranstaltungsorganisation beinhalten sollte – dies wurde jedoch, mit Verweis auf die vorhandene Kommunikationsproblematik, wieder verworfen, sodass der Fokus erneut auf die einheitliche E-Mail-Kommunikation innerhalb und außerhalb der Schule gelegt wurde.

Das zweite Meeting in der X-Net Services GmbH glich einer „großen“ Runde: Anwesend waren Hr. Wolfgang Eibner, Hr. Nikolaus Dürk, Hr. Prof. Peter Bauer, Fr. Prof. Natascha Rammelmüller sowie das Projektteam. Hier wurde gemeinsam der technische und funktionale Rahmen des Projekts finalisiert sowie ein Überblick über die in Frage kommenden Technologien gegeben.

Im Anschluss wurde bereits damit begonnen, einen Design-Entwurf für die Web-Applikation zu erstellen. Nach einem ersten Mockup gab es ein Abstimmungstreffen mit Hr. Prof. Peter Bauer, bei dem nochmals kleinere Änderungen vorgenommen wurden. Kurz darauf wurde der finale Diplomarbeitsantrag nach einmaliger Änderung eingereicht.

Festgelegte Meilensteine

Meilenstein	Soll-Datum	Ist-Datum
Die Features wurden endgültig formuliert.	22.03.2024	21.03.2024
Erste Design-Skizzen wurden in Figma erstellt.	15.04.2024	09.04.2024
Das finale Mockup der App wurde in Figma erstellt.	28.05.2024	27.05.2024
Die Anwendung kann mit ersten Grundfunktionen bedient werden.	12.07.2024	11.07.2024
Das Backend und das Frontend wurden fertiggestellt.	09.08.2024	09.08.2024
Das Backend und das Frontend wurden ausgiebig getestet und letzte Fehler wurden behoben.	01.09.2024	01.09.2024
Die Inhalte für die schriftliche Arbeit wurden festgelegt.	30.09.2024	28.08.2024
Die ersten Textproben für die schriftliche Arbeit wurden geschrieben.	15.10.2024	01.10.2024

Tabelle 1: Festgelegte Meilensteine der Diplomarbeit

Umsetzungsphase

Die technische Umsetzung ist weitestgehend im dafür vorgesehenen Kapitel beschrieben. Zu einem großen Teil bestand diese Zeit aus eigenständigem Arbeiten; bei größeren erreichten Meilensteinen gab es einzelne Meldungen an die Projektbetreuung.

Anwendungstests

Die Anwendung soll in ihrer Handhabung und Funktionalität in kleinem Umfang mit einer kleinen Nutzerzahl getestet werden. Mit Stand des 27. März 2025 ist geplant, dass das Sekretariatspersonal der Schule probeweise die wöchentliche „Wocheninformation“ für die Lehrkräfte mittels LeoMail versendet.

4 Technologien

4.1 Frontend

Während der Planungsphase wurde React zunächst aus reinem Interesse als Frontend-Technologie ausgewählt. Vor Beginn der praktischen Umsetzung der Diplomarbeit wurde diese Entscheidung jedoch noch einmal überprüft. Verschiedene Technologien wurden verglichen, um die am besten geeignete Technologie für dieses Projekt zu finden. Die Entscheidung fiel schließlich auf eines von drei Frontend-Frameworks: Angular, Vue.js und React.

Angular

Angular wurde von Google entwickelt und ist ein Framework, welches für die Entwicklung einer gesamten Plattform für Webanwendungen genutzt wird [41].

Abbildung 5: Logo von Angular [5]

AngularJS war die erste Version von Angular, die seit 2009 ein Open-Source-Projekt ist. Angular ist eine Weiterentwicklung von AngularJS, doch viele Features basieren auf dem Vorläufer. Alle sechs Monate erhält Angular einen neuen Versionssprung mit neuen Features und Breaking Changes.

TypeScript dient bei Angular als neue Code-Basis und hat Angular auf eine komponentenbasierte Architektur ausgerichtet. Damit verschiedene Teams gleichzeitig an der Logik bzw. Architektur oder am Design arbeiten können, wurde die View als HTML separat entwickelt.

Angular liefert integrierte DOM-Sanitizer, die Cross-Site-Scripting (XSS) Bugs verhindern. Für die Darstellung auf verschiedenen Endgeräten ist es wichtig, dass die Technologie plattformübergreifend ist. Diese Grundvoraussetzung erfüllt Angular und

ist durch seine partielle Aktualisierung bei User-Interaktionen bekannt. Somit ist es besser für Single-Page-Applikationen geeignet.

Angular benutzt das Prinzip des bidirektionalen Data Binding, welches nur bei Veränderung der View aktualisiert wird. Hierbei ist der Ladeprozess langsamer, aber bietet die Synchronisation von View und Datenmodell auf eine automatisierte und fehlerunanfälligere Weise.

React

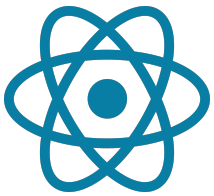

Abbildung 6: Logo von React [76]

Meta (ehemals Facebook) entwickelte 2011 die Open-Source-JavaScript-Bibliothek React, welche heute noch die Grundlage für viele Produkte und Dienste weltweit ist. React ist eine leistungsstarke Open-Source-JavaScript-Bibliothek, die zur Entwicklung von mobilen und Web-Anwendungen verwendet wird. Sie ermöglicht das Erstellen wiederverwendbarer Komponenten und ist ausschließlich für das Rendern der Komponenten einer Anwendungsschicht verantwortlich. Deshalb ist sie kein Framework, sondern eine Bibliothek.

Durch die Einführung von deklarativen und komponentenbasierten Ansätzen revolutionierte React die UI-Entwicklung. Zur Optimierung der Leistung verwendet React im Gegensatz zum manuellen Document Object Model (DOM), wie es in jQuery zu sehen ist, ein virtuelles DOM. Außerdem bietet React den Entwicklern eine strukturierte und effiziente Möglichkeit, interaktive, datengesteuerte Oberflächen zu erstellen. Diese Eigenschaften zeigen, dass React gut für komplexe Projekte geeignet ist.

React führte JSX ein, eine Syntaxerweiterung, die es ermöglicht, JavaScript mit HTML-Code zu kombinieren. [62]

React prahlt im Gegensatz zu anderen Technologien mit seiner großen Popularität. Seit 2014 bis Januar 2023 zeichnet sich weltweit eine klare Tendenz ab.

Als sehr umfangreiche Library bietet React im Bereich SEO bessere Grundvoraussetzungen als andere Technologien. Als komponentenbasierte Library bietet React hohe Flexibilität und Anpassungsfähigkeit gemäß den Projektanforderungen. Für das Bereitstellen des User Interfaces und der Funktionen sind die Komponenten verantwortlich. Styles werden standardmäßig ausgelagert. React bietet eine ausführliche Dokumentation,

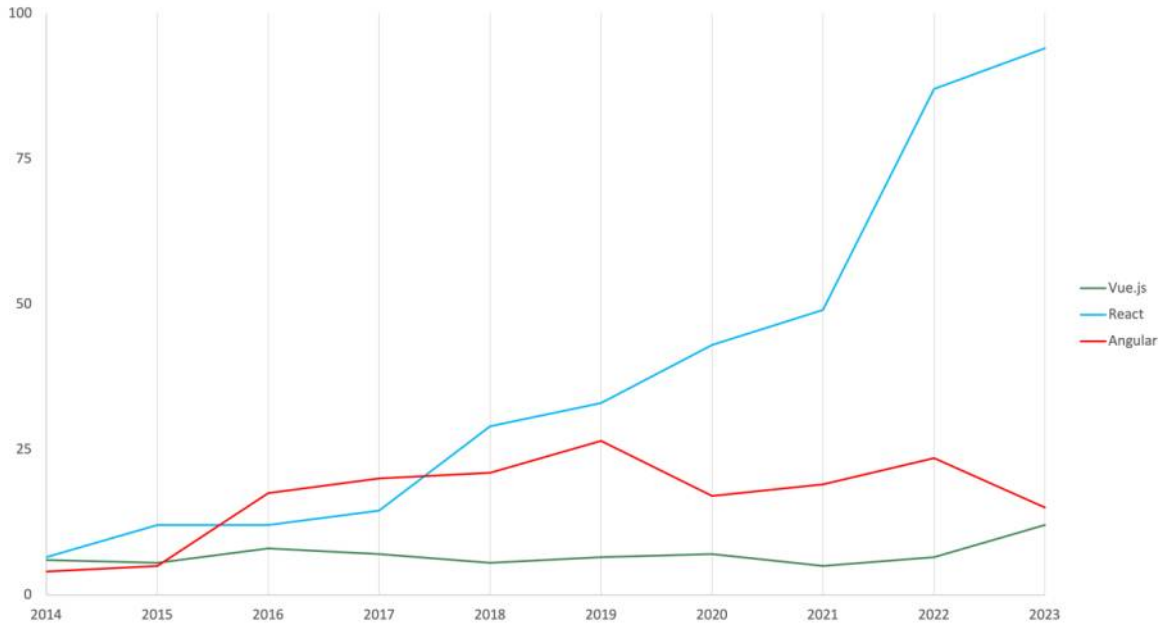

Abbildung 7: Beliebtheit von Vue.js, React und Angular (Quelle: dotnetpro.de)

die durch die große Reichweite und eine aktive Community sowie die Unterstützung von Meta regelmäßig aktualisiert wird.

Vue.js

Vue.js wurde ursprünglich 2014 von Evan You entwickelt. Es ist ein modernes JavaScript-Framework, welches Entwicklern die Erstellung interaktiver Benutzeroberflächen und die einfache und effiziente Erstellung von Single-Page-Anwendungen (SPAs) ermöglicht. [4]

Abbildung 8: Logo von Vue.js [94]

Die Technologie basiert auf TypeScript, kann aber für die Entwicklung sowohl TypeScript als auch JavaScript verwenden. Für die Komponenten nutzt Vue.js das Prinzip der Trennung zwischen Oberfläche und Logik. Dies soll die Möglichkeit bieten, unabhängig voneinander an verschiedenen Schichten zu arbeiten.

Vue.js ist ein sehr einsteigerfreundliches Framework, da es auf der einfachen Syntax von HTML und JavaScript basiert. Neue Versionen oder Features werden vom Gründer Evan You auf GitHub dokumentiert. Durch die geringere Popularität lässt sich schließen, dass die Entwickler-Community kleiner ist als bei den anderen Technologien.

Um kurze Ladezeiten und schnelle Benutzerinteraktionen zu gewährleisten, nutzt Vue.js das virtuelle Document Object Model (DOM). Außerdem vereint es diese Technologie

mit dem Prinzip des bidirektionalen Data Binding, welches nur bei Veränderungen an der View aktualisiert wird. Da sich Vue.js je nach Anwendung skalieren lässt, ist dessen Performance auch die schnellste.

Vue.js ist plattformübergreifend und bietet eine barrierefreie Umsetzung. Es ist außerdem für Endverbraucher mit Screenreadern relevant. Zusätzlich bietet es eine leichte Integration in bestehende Projekte, da es sowohl als separates Framework als auch in Kombination mit anderen JavaScript-Bibliotheken oder -Frameworks verwendet werden kann.

Entscheidung

Um die richtige Technologieentscheidung zu treffen, wurde eine Tabelle erstellt, die die wichtigsten Features für dieses Projekt auflistet. Letztendlich fiel die Wahl auf Vue.js. Ein sehr wichtiger Aspekt bei der Entscheidung war die Lernkurve, bei der Vue im Gegensatz zu den anderen Technologien am besten abschneidet. Dies bietet einen möglichst schnellen und produktiven Start ins Projekt. Die anderen Technologien bieten zwar aufgrund ihrer größeren Popularität eine umfangreichere Dokumentation, doch Vue konnte nach Recherchen auch mit genügend erforderlichen Informationen überzeugen. [61]

Alle drei gewährleisten eine gute Wartbarkeit, was das Aktualisieren und Überarbeiten der Komponenten erleichtert. Ein weiterer gewünschter Faktor war die Möglichkeit, persönliche Unterstützung im Umfeld zu bekommen. Der Kooperationspartner X-Net Services GmbH arbeitet schon länger mit Vue.js und stünde jederzeit für Fragen oder Hilfestellungen zur Verfügung. Es gäbe zwar eine Professorin, die bei Fragen zu React zur Verfügung stünde, doch wäre der Kontakt nur über E-Mail oder Online-Meetings möglich, was viel Aufwand und Zeit kosten würde.

Wichtige Features	**React**	**Angular**	**Vue.js**
Lernkurve	●	✓	✓
Dokumentation	✓	✓	●
Wartbarkeit	✓	✓	✓
Popularität	✓	●	✗
Performance	●	✗	✓
Support im Arbeitsumfeld	●	●	✓

Abbildung 9: Eigene Darstellung - Vergleich der Top 3 Frontend-Frameworks

4.2 Backend

Bei den Backendtechnologien fiel die Entscheidung, ähnlich wie bei der Entscheidung um die Frontend-Technologien ([61]), aufgrund der Frameworks.

Es gibt eine Vielzahl an beliebten und bekannten Backendtechnologien, darunter fallen beispielsweise:

- C# ASP.NET
- Java Spring Boot
- node.js express
- PHP
- Go Mux

Die Entscheidung wurde allerdings schließlich zwischen Quarkus und Django mittels einer Nutzwertanalyse gefällt [85]. Die Grundlage für die Entscheidung zwischen den beiden Technologien war, dass Quarkus einerseits an der HTL Leonding gelehrt wird und Django von unserer Kooperationsfirma *X-Net Services GmbH* verwendet wird.

Was ist Backend?

Während das Frontend sich darauf fokussiert, die Funktionalität einer Anwendung für einen Benutzer verwendbar zu machen, stellt ein Backend alle notwendigen Informationen dieser Anwendung zur Verfügung und dient der Datenverarbeitung. Oft werden zur Speicherung und zum späteren Abruf der Daten auch Datenbanken mit dem Backend verbunden.

Direkten Zugriff auf das Backend haben in der Regel nur Administratoren sowie die Backend-Entwickler selbst – der Benutzer wird nie direkten, unkontrollierten Zugriff auf die Daten einer Anwendung haben, sondern nur eingeschränkt die für ihn relevanten Daten sehen.

Python Django

Django ist ein Open-Source-Framework, welches die Entwicklung von Web-Anwendungen erleichtert [34].

Im Gegensatz zu Quarkus ist es kein reines Backend-Framework, sondern bietet als sogenanntes Fullstack-Framework auch die Möglichkeit, das Frontend direkt in derselben Anwendung zu integrieren [20].

Abbildung 10: Logo von Python Django [51]

Das Python-Framework setzt außerdem das sogenannte "MVC"(also "Model-View-Controller")-Pattern konsequenter um als andere Frontend-Frameworks wie Angular oder Vue.js [34]. In der Backend-Sparte des Fullstacks trifft man zudem auf das Prinzip der abstrakten Datenbankschemen: Dadurch können nahtlose Wechsel zwischen verschiedenen Datenbankdialekten, beispielsweise von MariaDB zu PostgreSQL, durchgeführt werden – Änderungen an der Datenbankstruktur, die über einen ORM (*Object-Relation-Mapper*) konfiguriert werden, werden auch am Datenbankserver angepasst.

Da Django in den letzten Jahren wesentlich an Bekanntheit erlangt hat und aus dem Schatten seines Bruders Flask gesprungen ist, existieren zahlreiche Bibliotheken, die Optimierungs- und Erweiterungsmöglichkeiten bieten [34]. Django bringt allerdings auch die Nachteile der Programmiersprache Python mit sich: So ist Python in der Regel beträchtlich langsamer in der Übersetzung in Maschinencode als traditionellere Sprachen wie Java oder C# [45].

Quarkus - Supersonic Subatomic Java

Quarkus ist, genau wie Django, ein Open-Source-Framework, welches besonders für die Entwicklung moderner Backend-Applikationen verwendet wird. Es basiert auf der bekannten Programmiersprache Java [33].

Ganz besonders hervorzuheben ist die Geschwindigkeit: Quarkus kompiliert im nativen GraalVM-Build um bis zu 5-mal schneller als eine Spring-, und um bis zu 3-mal schneller als eine Django-Anwendung [89]. Auch verfolgt Quarkus den Ansatz der abstrakten Datenbankschemen, sodass ein fliegender Wechsel zwischen verschiedenen Datenbankdialekten nahtlos möglich ist – ebenso wie das Entwerfen der

Abbildung 11: Logo von Quarkus [52]

Datenbankstruktur und das Verwalten der Datensätze über
einen OR-Mapper [33].

Mit Quarkus kann sowohl ein reaktiver als auch ein blockierender imperativer Code-Ansatz gewählt werden [33]. Zudem bietet das Framework zahlreiche Integrationen bekannter Java-Bibliotheken, wie beispielsweise JPA, Apache Kafka, Eclipse MicroProfile oder Spring. Dadurch können beispielsweise Anwendungen, die zuvor auf einem Spring-Boot-Backend liefen, einfach auf eine Quarkus-Anwendung übertragen werden. Im Gegensatz zu Python bietet Java allerdings den Nachteil, dass es striktere Coding-Conventions erfordert.

Entscheidung

Die Entscheidung zwischen Quarkus und Django wurde mittels einer Nutzwertanalyse [85] durchgeführt.

Zur Auswahl standen die Möglichkeiten, eine neue Quarkus-Anwendung oder eine neue Django-Anwendung zu verwenden. Zusätzlich bestand aufgrund der Kooperation mit der X-Net Services GmbH auch die Möglichkeit, eine bereits bestehende, ähnliche Anwendung auf Basis von Django weiterzuentwickeln.

Die Nutzwertanalyse wurde mit folgenden Kriterien und Gewichtungen durchgeführt:

	Kriterium	**Gewichtung**
1	Wie lange dauert es, bis man produktiv arbeiten kann?	10%
2	Kann das Projekt von Schüler:innen oder Lehrkräften zukünftig gewartet werden?	20%
3	Unterstützung durch die Kooperationsfirma X-Net	15%
4	Können bestehende Komponenten übernommen werden?	10%
5	Ist diese Technologie im schriftlichen Teil gut verarbeitbar?	10%
6	Wie umfangreich ist die Dokumentation der Sprache/des Frameworks?	30%
7	Wie sehr sind wir an der zusätzlichen Qualifikationen interessiert?	5%

Tabelle 2: Nutzwertanalyse: Kriterien und Gewichtungen

Nach ausführlicher Diskussion mit Hr. Prof. Peter Bauer und Fr. Prof. Natascha Rammelmüller sind wir schlussendlich zu folgendem Ergebnis gekommen:

Technologie	Quarkus, neu	Django, neu	Bestehendes Projekt
Kriterium 1	4	2	2
Kriterium 2	4	2	2
Kriterium 3	2	4	5
Kriterium 4	2	3	5
Kriterium 5	3	4	5
Kriterium 6	4	4	2
Kriterium 7	3	3	2
Gesamt (gewichtet)	3,35	3,25	3,05

Tabelle 3: Bewertung von Technologien anhand der Kriterien

Quarkus erhielt hier die meiste Punktzahl und wurde deshalb als Backend-Technologie ausgewählt.

4.3 Datenbank

Für die auszuwählende Datenbanktechnologie war vor allem das bereits vorhandene Wissen über die jeweiligen Dialekte relevant. Die Verwendung einer NoSQL-Technologie war von Anfang an ausgeschlossen, da die Anwendung strikt schematische Datensätze hat, welche relationale Beziehungen zueinander aufweisen.

Durch den Unterricht an der HTL Leonding in der Abteilung Informationstechnologie fiel die Entscheidung zwischen Oracle- und PostgreSQL.

PostgreSQL ist eine Open-Source-DBMS-Lösung, welche mit allen herkömmlichen Server-Betriebssystemen kompatibel ist. Der Dialekt ist weit skalierbar und bietet zahlreiche Features an, wie beispielsweise die Unterstützung von JSON oder einer Volltextsuche [88].

Heutzutage hat PostgreSQL auch ein Pendant zu Oracles "PL/SQL"– nämlich PL/pgSQL –, das in Syntax und Handhabung sehr ähnlich ist.

Oracle SQL ist eine proprietäre Lösung des gleichnamigen Unternehmens Oracle. Es zählt ebenso wie PostgreSQL zu den weit verbreiteten Datenbankdialekten und gilt als eine der sichersten Datenbanktechnologien [88].

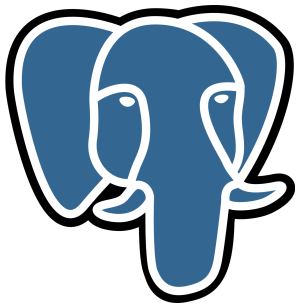

Abbildung 12: Logo von PostgreSQL [50]

Entscheidung zwischen Oracle SQL und PostgreSQL

Die Entscheidung wurde vor allem auf Basis von Flexibilität sowie Ressourcenbedarf getroffen.

PostgreSQL ist in seiner Konfiguration deutlich flexibler und eignet sich daher eher für eine Web-Anwendung wie LeoMail, die sich in ihrer Funktionalität mit der Zeit stark weiterentwickeln kann und somit auch in Zukunft andere oder weitere Anforderungen an ihr Datenbanksystem stellen wird.

Zwar liefert Oracle SQL bei einfachen Query-Abfragen eine weitaus schnellere Antwortzeit als PostgreSQL – bei komplexeren Abfragen zeigt jedoch die Open-Source-Lösung bessere Ergebnisse. Zudem weist Oracle, insbesondere im Idle-State, eine höhere Arbeitsspeichernutzung auf.

Ein weiterer Vorteil spricht für PostgreSQL: Die einfache Integration in Entwicklungsumgebungen, da PostgreSQL beispielsweise via Docker problemlos ausgeführt werden kann, wohingegen Oracle als Docker-Image, insbesondere auf macOS-Geräten, oft zusätzliche Installationen erfordert.

Die endgültige Entscheidung fiel daher auf PostgreSQL, da es sich um ein Open-Source-Projekt handelt.

4.4 Identity Management - Keycloak

Keycloak ist eine selfhosted Open-Source-Lösung für sogenanntes Ïdentity and Access-Management, die auf dem auch für LeoMail verwendeten Quarkus-Framework basiert [86].

Es ist vor allem für sein umfangreiches Feature-Set bekannt: Es bietet eine Admin Console, mit der die gesamte Konfiguration vorgenommen werden kann, und kann sowohl für die einfache Authentifizierung als auch für die Autorisierung (Rechteverwaltung) in spezifischen Softwareprojekten eingesetzt werden.

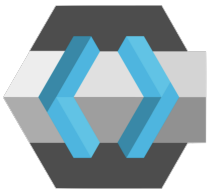

Abbildung 13: Logo von Keycloak [46]

Für LeoMail ist Keycloak insbesondere aufgrund der einfachen Integration von Identity Providern, wie dem LDAP-

Service der HTBLA Leonding, von Relevanz. Dadurch können sowohl die Nutzerdaten für den Login als auch für die Synchronisierung der LDAP-Nutzer mit der Anwendung verwendet werden.

Für den Login wird der sogenannte "Direct Access GrantFlow verwendet, bei dem der Login direkt in der eigenen Anwendung erfolgt und ein Äccess Token", das zusätzlich Metadaten des Nutzers enthält, an den Client zurückgegeben wird [38]. Sobald dieser Access Token abläuft, kann der Nutzer mittels des mitgelieferten Refresh Tokens einen neuen Access Token anfordern. Alternativ gibt es den Standard-Flow, bei dem der Login über eine eigene Login-Seite des Keycloak-Servers erfolgt [39].

Keycloak bietet zudem ein Rollensystem, das die Zuordnung von Nutzern zu Rollen ermöglicht [37]. Für die Entwicklung eines Quarkus-Backends ist Keycloak besonders geeignet, da es das OpenID Connect-Authentifizierungsprotokoll nutzt, was mittels einer Quarkus-Maven-Dependency das Schützen von Routen durch einfache Annotationen erlaubt [40, 67].

Die Entscheidung fiel zudem, weil an der Schule bereits ein Keycloak in Betrieb ist, sodass lediglich ein eigener Realm und Client für die Anwendung erstellt werden mussten. Alle vergleichbaren Alternativen sind zudem kommerzialisiert und bieten, wenn überhaupt, nur eingeschränkte Gratis-Pläne an.

4.5 File Storage - MinIO

MinIO ist eine Open-Source-Lösung zum Speichern und Verwalten von Dateien [58]. Für diese Technologie spricht vor allem, dass es einen MinIO-Client für Quarkus gibt, welcher das Hochladen, Herunterladen und Löschen von Dateien mittels einfacher Methodenzugriffe ermöglicht [66].

Zur Verwaltung kann – beispielsweise bei Fehlern oder Unstimmigkeiten – auch die Admin Console von MinIO genutzt werden [56].

Prinzipiell spricht auch die S3-Kompatibilität von MinIO sowie die Vielzahl von Erweiterungen, die unter anderem eine Kompatibilität mit Azure oder Google Cloud gewährleisten, für diese Lösung [58, 57].

Abbildung 14: Logo von MinIO [48]

5 Umsetzung

5.1 Vorbereitungen

Vor der Entwicklung eines Projektes wie LeoMail, welches, unter Berücksichtigung des Bearbeitungszeitraums und des verfügbaren Arbeitspensums, als sehr umfangreiches Projekt beschrieben werden kann, müssen sowohl in organisatorischer als auch technischer Sicht vor der Entwicklung einige Vorbereitungen getroffen werden.

Auf organisatorischer Seite wurden Aspekte wie die genaue Definition der Features und die davorgehende Ideenfindung, als auch die Koordination mit der Schulleitung im Laufe der Zeit enorm wichtig.

Vor dem Projektstart musste außerdem, wie näher in 3.6 beschrieben, das Design fertiggestellt werden, um die Entwicklungszeit der Frontend-Anwendung in Vue unnötig zu verlängern.

Aus technischer Sicht war es notwendig, einerseits lokal auf den Geräten des Projektteams die notwendige Software einzurichten und zu installieren, andererseits war es notwendig, Dienste wie Keycloak oder den Mailserver zu konfigurieren.

Einrichtung der Endgeräte

Jeder Schritt, der nachfolgend beschrieben wird, wurde jeweils auf beiden Endgeräten des Projektteams durchgeführt.

Das Vorbereiten der Geräte begann mit dem Aktualisieren aller für die Entwicklung des Projektes relevanten Anwendungen, um Fehlern durch inkompatible Versionen vorzubeugen.

Das betraf vor allem jene Anwendungen:

- IntelliJ Ultimate Edition und WebStorm als Entwicklungsumgebungen
- Git
- Docker (sowie dem eventuell bereits installierten PostgreSQL-Image)

Konfiguration der Entwicklungsumgebungen

Im weiteren Verlauf wurden in beiden Entwicklungsumgebungen, abhängig von der Notwendigkeit, Plugins installiert.

In IntelliJ wurden zahlreiche Erweiterungen installiert und konfiguriert, die gewisse Prozesse in der Entwicklung des Backends mit Quarkus vereinfachen können. Einen großen Anteil daran hatte "Database and SQL Tools", welche die direkte Verbindung mit der Datenbank innerhalb von IntelliJ erlaubt. Hier können sowohl SQL-Statements in einer Konsole ausgeführt werden, als auch die Inhalte und Struktur von Tabellen, Sequenzen und der Datenbank allgemein mittels Klicken angezeigt werden. Weiters wurden Support-Erweiterungen für Jakarta EE, als auch für Maven als Build-Tool installiert.

In WebStorm wurde einerseits der Vue.js-Support installiert, um für mehr Übersicht während der Entwicklung zu sorgen. Zusätzlich wurde IntelliVue installiert, welches zusätzlich zur JetBrains-Erweiterung eine erweiterte *Intellisense*, also Unterstützung der Entwicklungsumgebung während der Programmierung, bietet und bei der Erstellung neuer Vue-Files den initialen Code einer Komponente erzeugt.

In beiden Programmen wurde GitHub Copilot installiert: es hinkt in der Qualität des erzeugten Codes seinen Alternativen, wie ChatGPT oder Gemini, hinterher, kann aber bei einfachen und vielfach verbreiteten Code-Snippets und Algorithmen den Arbeitsaufwand verringern [35, 36].

Zusätzlich wurde in WebStorm noch eine Build Configuration erstellt, damit die Vue-Anwendung mittels Knopfdruck gestartet, neu gestartet und gestoppt werden kann. In IntelliJ war das nicht notwendig, da dieses bei Quarkus-Anwendungen diese automatisch konfiguriert.

Erneuerung und Installation der SSH- und GPG-Keys

Zur Verwendung in GitHub sowie Git Secret wurden die SSH- und GPG-Schlüssel erneuert.

Die Schlüssel können mit den folgenden Kommandozeilenbefehlen erzeugt werden:

Listing 1: Befehl zur Erzeugung der Schlüssel

```
1 # SSH-Key-Pair
2 ssh-keygen
3
4 # GPG-Key
5 gpg --full-generate-key
```

Während der Befehl zur Erzeugung des Schlüsselpaares ohne Installation weiterer Tools sowohl auf Windows als auch auf macOS zur Verfügung steht, muss für die Ausführung des gpg-Befehls GnuPG, beziehungsweise das Windows-Äquivalent GPG4win, installiert werden.

Die SSH-Keys zu erneuern war dabei nicht zwingend notwendig; aus Sicherheitsgründen ist es allerdings in gewissen Zeitabständen ratsam, damit im Falle eines gestohlenen privaten Schlüssels niemand unerlaubten Zugriff auf bestimmte Services oder fremde Maschinen erlangen kann. Die GPG-Keys haben Ablaufdaten und wurden, um spätere Verwirrung aufgrund von Fehlern bezüglich der Aktualität der Schlüssel zu vermeiden, auch erneuert.

Notwendig ist das in diesem Fall deshalb, da die SSH-Schlüssel in der GitLab-Instanz der X-Net Services GmbH notwendig waren, um mit dem Git-Repository remote zu arbeiten [24]. Die GPG-Schlüssel signieren die Commits und verifizieren, dass sie tatsächlich vom jeweiligen Autor kommen [27] - dies hat zwar in der privaten Self-hosted-Instanz von GitLab keine Relevanz, im öffentlichen Repository auf GitHub allerdings schon.

Konfiguration von Git Secrets

Um sensible Daten wie beispielsweise Passwörter, Tokens oder private Keys im Sourcecode des Projektes vor der "Öffentlichkeitßu schützen, müssen diese im Commit verborgen werden.

Git Secrets löst dies, indem es beim Initialisieren eines Git-Secrets-Repository ein *.gitsecret/*-Verzeichnis anlegt [81], in dem Verweise auf jene GPG-Schlüssel erstellt werden, denen seitens des Erstellers des Secrets-Repositorys vertraut wurde. Vor dem Hinzufügen beziehungsweise Vertrauen eines weiteren Schlüssels muss jener auf dem Endgerät lokal importiert werden.

Die Dateien, in denen nun vertrauliche Daten liegen, können mittels Befehl verborgen werden. Diese Dateien werden nun zur *.gitignore*-Datei hinzugefügt und im selben Verzeichnis wird jeweils eine Datei mit der Endung *.secret* erstellt, mit jenem Inhalt die Originaldatei nach einem etwaigen Pull wiederhergestellt werden kann.

Nun muss bei etwaigen künftigen Commits darauf geachtet werden, dass der Befehl zum Verbergen immer ausgeführt wird, bevor ein Commit geschieht. Um diesen Prozess zu automatisieren, kann man sich sogenannte Git Hooks zunutze machen [6].

Git Hooks definieren die Shell-Befehle, die bei bestimmten Git-Kommandos durchgeführt werden. Einer davon ist der *pre-commit*-Hook, welcher vor jedem Commit ausgeführt wird. Hier ist es notwendig, den Befehl zum Verstecken der Originaldateien einzubauen. Diese Hooks liegen im *.git/hooks*-Verzeichnis. Initial haben alle Hooks ein *.sample* als Dateiendung, welches nach der Änderung des Inhalts entfernt wird, damit die Änderung wirksam wird. Es ist ratsam, die Sample-Hooks abzusichern, um bei etwaigen Fehlern ein Backup zu haben.

Bei der Installation von Git Secrets gab es auf dem Windows-Endgerät Probleme: der Entwickler bietet ein PowerShell-Skript an, welches den Service installieren und im *PATH* zur Verfügung stellen sollte. Dies hat im Projektverlauf nicht geklappt, weshalb ein Workaround notwendig war. Schlussendlich wurde Git Secrets dann innerhalb einer WSL-Ubuntu-Maschine installiert und somit verwendet.

Listing 2: Befehle zur Nutzung von Git Secrets

```
# Initialisieren des Git-Secrets-Repository
git secret init

# Vertrauen eines Schluessels mittels der hinterlegten E-Mail-Adresse
git secret tell <E-Mail-Adresse>

# Hinzufuegen von Dateien mit sensiblen Daten
git secret add <Datei(pfad)>

# Verstecken der Originaldateien
git secret hide

# Wiederherstellen der Originaldateien
git secret reveal
```

Einrichtung der VPN-Verbindung

Seitens der Partnerfirma X-Net Services GmbH wurden uns OpenVPN-Dateien zur Verfügung gestellt, da das interne GitLab nur über das lokale Netzwerk erreichbar ist.

Angewandt wurden diese bei dem Windows-Endgerät über *OpenVPN*, während auf dem macOS-Endgerät die Anwendung *Tunnelblick* installiert wurde, damit die Verbindung hergestellt werden konnte.

Ausführen des Datenbank-Containers

Mittels Docker wurde auf den Arbeitsgeräten eine PostgreSQL-Datenbank für die Entwicklung ausgeführt [16]. Dieser Container basierte auf dem offiziellen Docker-Image für PostgreSQL *postgres* in der Version 17.2.

Damit keine Änderungen zwischen den Geräten notwendig wurden, hat man dieselben Umgebungsvariablen in Bezug auf Zugangsdaten der Datenbank gewählt.

Listing 3: Ausführen des Datenbank-Docker-Container

```
1 docker run -p 5432:5432 -d -e POSTGRES_PASSWORD=postgres postgres:17 --name
      POSTGRES_17
```

Git-Repository

Das Git-Repository wurde, wie bereits eingangs erwähnt, am GitLab-Provider der Partnerfirma X-Net Services GmbH erstellt. Notwendigerweise wurden in den Nutzerkonten, die uns zur Verfügung gestellt wurden, die SSH-Keys und die GPG-Keys hinterlegt.

Um eine gute Arbeitsgrundlage zu schaffen, wurde von Tommy Neumaier sowohl ein Vue- als auch ein Quarkus-Projekt erstellt, und dieses initial auf das Remote-Git-Repository mittels Push geladen. Damit sollten Merge-Konflikte gleich von Anfang an vermieden werden.

Weitere Schritte waren für den Anfang nicht notwendig, da das Deployment und die dazugehörige Pipeline erst nach der Fertigstellung des Prototypen über das neue Remote-Repository auf GitHub umgesetzt wurden.

Mailserver (mailcow)

mailcow ist ein Open-Source-Mailserver, welcher als Docker-Image auf jedem System ausführbar ist.

Dieser wurde anfangs benötigt, da für die ersten Versionen der Anwendung ein Mailserver benötigt worden ist, der ohne maximale Grenze für die Anzahl an versendeten Mails konfiguriert ist. Dies erfüllen Outlook, Gmail oder weitere E-Mail-Provider nicht.

Die Mailer-Extension von Quarkus besitzt zwar eine "mockVersandvariante, welche, anstatt die E-Mail tatsächlich zu versenden, die versendete E-Mail samt Empfängerdaten auf der Konsole des ausgeführten Programms anzeigt. Für die Entwicklung von LeoMail war es relevant zu prüfen, ob die E-Mails auch mit dem Design der Vorlage übereinstimmen; dadurch reichte dies nicht aus.

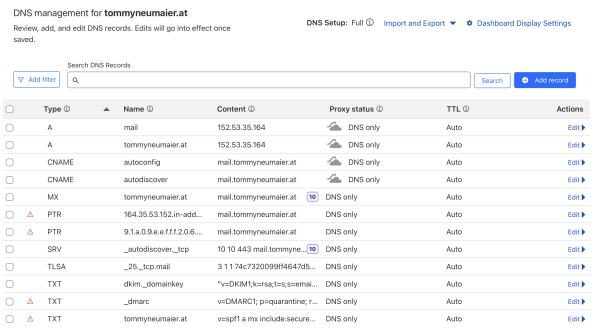

Abbildung 15: DNS-Einstellungen für den temporären Mailserver

mailcow wurde auf einer für die Diplomarbeit angemieteten virtuellen Maschine installiert. Dafür wurde ein Ubuntu 24.04-LTS "minimalSystem verwendet, mit keinen weiteren installierten Paketen außer Docker und docker-compose.

Als Mail-Domäne wurde im Projekt während der anfänglichen Entwicklung die private Domain *tommyneumaier.at* verwendet. Damit diese für den Mailserver verwendet werden konnte, mussten einige DNS-Einstellungen getroffen werden. Diese wurden über cloudflare.com verwaltet [65].

Beschreibung der DNS-Records

Die Mailserver-Installation sollte zur Entwicklung möglichst den Voraussetzungen eines marktrelevanten E-Mail-Providers entsprechen, deshalb wurden folgende Records gesetzt:

- A
- CNAME
- SRV
- MX
- PTR
- TXT

Der *A*-Record sorgt dafür, dass sowohl die Domain "tommyneumaier.atäls auch die Subdomain "mail.tommyneumaier.atäuf den richtigen Server auflöst; in diesem Fall jenen, der unter der öffentlichen IP-Adresse 152.53.35.164 erreichbar ist.

Sowohl der *CNAME*, als auch der *SRV*-Record, dienen dafür, dass E-Mail-Clients, wie beispielsweise die oben beschriebenen (2 Umfeldanalyse) Outlook und Thunderbird, nach

Eingabe der E-Mail-Adresse automatisch die Mailserver-Konfiguration erkennen können und der Nutzer diese nicht manuell eingeben muss. Prinzipiell ist dies für LeoMail irrelevant, da die SMTP-Einstellungen ohnehin in der Konfiguration der Backend-Anwendung konfiguriert sind, der Vollständigkeit halber wurde es allerdings trotzdem hinzugefügt.

Der *MX*-Record sorgt dafür, dass die E-Mails, die an eine E-Mail-Adresse mit dem Suffix *@tommyneumaier.at* enden, an den Mailserver *mail.tommyneumaier.at* gesendet werden.

Der *PTR*-Record ist für die inverse Auflösung der Domain zuständig: nach der Auflösung der Domain durch die IP-Adresse, wie beim A-Record, folgt eine Rückauflösung der IP-Adresse durch die Domain. Das ist gängige Praxis vieler Spamfilter und kann verhindern, dass die gesendeten E-Mail-Adressen des Absenders im Spamordner landen.

Die *TXT*-Records haben alle gänzlich unterschiedliche Funktionen:

- Der Record, der *dkim* im Name beinhaltet, enthält den öffentlichen Schlüssel eines DKIM-Schlüsselpaares. Mit diesem kann der Empfänger verifizieren, dass die E-Mail tatsächlich nicht durch einen "Man-in-the-Middle"verändert wurde. Dieses Schlüsselpaar wird in der mailcow Admin Console erstellt.
- Der SPF-Record, erkennbar durch *v=spf1* im Content, verifiziert, dass die gesendete E-Mail-Adresse tatsächlich vom angegebenen Mailserver gesendet wurde.
- Der TXT-Eintrag mit *dmarc* als Name entspricht der Richtlinie, wie mit E-Mails umgegangen werden soll, die entweder die DKIM- oder SPF-Prüfung nicht bestehen. Mit DMARC können außerdem auch Berichte erstellt werden, wie viele eigene E-Mails die Prüfungen bestehen oder nicht.

Konfiguration des mailcow-Servers

Für die Benutzung des Mailservers ist kein großer Konfigurationsaufwand notwendig.

Die Erstellung eines DKIM-Schlüssels für die Signatur der E-Mail, wie oben beschrieben, ist empfohlen, aber nicht notwendig.

Für mailcow-Instanzen, die tatsächlich im Produktivsystem genutzt werden, wäre es zu empfehlen, das Standardpasswort des Admin-Kontos zu ändern, beziehungsweise den Standardnutzer nach der Erstellung eines neuen Administratorenkontos zu löschen. Da

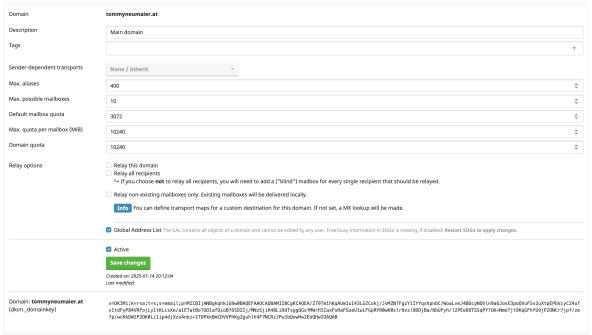

Abbildung 16: Konfiguration der Mail-Domäne in der mailcow-Admin-Oberfläche

LeoMail im Produktiveinsatz allerdings den Outlook-Mailserver der HTBLA Leonding nutzt, ist dies hier ebenfalls nicht relevant.

Die Anlegung einer Mail-Domäne ist, genauso wie die anschließende Erstellung einer Mailbox, notwendig. Diese Einstellung kann unter dem Menüpunkt *Mail/Configuration* vorgenommen werden.

Identity Provider (Keycloak)

Keycloak ist, wie bereits in 4.4 beschrieben, eine IAM-Anwendung, welche das User-Management von Anwendungen jeglicher Art zentralisiert und beinahe vollständig übernimmt.

Die Installation einer eigenen Keycloak-Instanz blieb zwar aufgrund eines Ausfalls, wie in 8.X beschrieben, nicht aus, hat aber wegen einer bereits von der Schule zur Verfügung gestellten Instanz keine Bedeutung für die Umsetzung der Diplomarbeit.

Die Konfiguration des Identity Providers, also der LDAP-Schnittstelle der Schule, wurde im Rahmen eines anderen Schulprojekts vom Projektteam bereits durchgeführt und konnte einfach übernommen werden.

Damit mussten keine großen Konfigurationen vorgenommen werden. Es musste ausschließlich ein neuer *Client* im bereits vorhandenen Realm erstellt werden [31]. Ein "Clientïm Keycloak-Fachjargon ist sozusagen die Entität für eine einzige Applikation. Innerhalb dieses Clients können die User, die im Realm hinterlegt sind, eigene Rollen oder Gruppen zugewiesen bekommen.

Bei der Konfiguration eines Clients muss neben der Auswahl einer logisch herleitbaren *Client ID*, die im Idealfall dem Namen der Anwendung gleicht, die Konfiguration

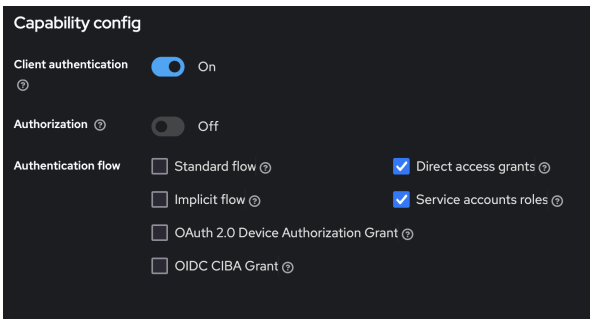

Abbildung 17: Capability-Konfiguration des LeoMail-Keycloak-Clients

der sogenannten *Capability*, also welche Funktionen dieser Client übernehmen darf, vorgenommen werden.

Wichtig ist hierbei bei herkömmlichen Anwendungen, dass der standardmäßig aktivierte Check bei *Client authentication* abschließend ebenfalls aktiviert ist. Dies ermöglicht nämlich die grundlegende Authentifizierung eines Anwendungsnutzers gegenüber dem Keycloak-Authentifizierungsservice.

Je nach Auswahl des Authentifizierungsflows muss ebenfalls *Direct access grants* aktiviert werden [32]. Keycloak bietet hauptsächlich zwei Möglichkeiten, wie eine Anwendung die Authentifizierung lösen kann, an: den *Redirect Flow*, bei welchem man bei der Anmeldung auf eine Login-Maske von Keycloak weitergeleitet wird. Die meisten Anwendungen nutzen allerdings den sogenannten "Direct Access Grant Flow", welcher die Authentifizierung über eine eigene, in der Anwendung integrierte Login-Maske ermöglicht. Dieser liefert ohne Redirect, bei gültigen Anmeldedaten, einen Access- sowie einen Refresh-Token.

Sofern also die Authentifizierung über eine eigene Login-Maske gewünscht ist, muss letztere Box aktiviert werden.

Weiters relevant ist *Service account roles*. Dieser Check aktiviert einen Nutzer, welcher de facto der Ädmin-Nutzer"für die Applikation ist. Über diesen kann beispielsweise via Backend dann im Keycloak nach Nutzern gesucht werden, es können Rollen erstellt, zugewiesen oder auch gelöscht werden. Diesem ßervice account"müssen allerdings auch, wie jedem anderen Nutzer, die jeweiligen Berechtigungen im ÜserReiter von Keycloak hinzugefügt werden.

Konfiguration im Backend

Vor dem Start der tatsächlichen Entwicklung müssen in einem Quarkus-Backend noch erste Konfigurationsschritte in den sogenannten *application.properties* durchgeführt werden.

Datenbankverbindung

Listing 4: Konfiguration der Datenbankverbindung im Backend

```
1 quarkus.datasource.username=postgres
2 quarkus.datasource.password=postgres
3 quarkus.datasource.db-kind=postgresql
4 quarkus.datasource.jdbc.url=jdbc:postgresql://127.0.0.1:5432/postgres
5
6 quarkus.hibernate-orm.database.generation=drop-and-create
```

Die *datasource*-Einstellungen entsprechen den bereits oben beschriebenen Umgebungsvariablen beziehungsweise Standardeinstellungen des Docker-Containers [71].

quarkus.hibernate-orm.database.generation beschreibt das Verhalten der Datenbank bei einem Neustart des Backends. In der Entwicklung ist es hier sinnvoll, *drop-and-create* zu verwenden, da hier die Datenbank von Grund auf bei jedem Neustart oder Änderung des Backends neu strukturiert wird und etwaige prozedurale Änderungen an der Datenbankstruktur automatisch angepasst werden. Es kann allerdings auch das Testen der Anwendung erschweren, wenn beispielsweise für gewisse Testzwecke Daten aus einem Anwendungstestdurchlauf benötigt werden: hierzu könnte beispielsweise auf *update* umgestellt werden.

CORS-Anfragen

Listing 5: Konfiguration der CORS-Anfrageverarbeitung im Backend

```
1 quarkus.http.cors=true
2 quarkus.http.cors.origins=http://localhost:5173
3 quarkus.http.cors.methods=GET,POST,OPTIONS,PUT,DELETE
4 quarkus.http.cors.headers=Authorization,Content-Type,Content-Disposition
```

CORS beschreibt jene Anfragen, die von einer anderen Domäne gestellt werden [72]. In diesem Fall ist es relevant, CORS-Anfragen von der Domäne des Frontends zu erlauben, da dieses ansonsten keine Anfragen an das Backend stellen kann.

In *quarkus.http.cors.methods* und *quarkus.http.cors.headers* werden zusätzlich noch Restriktionen bezüglich der erlaubten Anfragemethoden sowie der erlaubten HTTP-Header bei diesen Anfragen angewandt.

SMTP-Servereinstellungen

Listing 6: Mailer-Mock-Konfiguration im Backend

```
quarkus.mailer.mock=true
```

Mit der *mock*-Einstellung wird festgelegt, ob E-Mails, die im Sourcecode via Mailer an einen Empfänger gesendet werden, tatsächlich versendet werden sollen oder ausschließlich auf der Konsole ausgegeben werden sollen. Im Falle von *true* werden sie nur auf der Konsole ausgegeben, sollte diese Einstellung nicht konfiguriert sein oder auf *false* gesetzt sein, wird die E-Mail bei gültiger SMTP-Konfiguration tatsächlich versendet.

5.2 Komponenten einer Anwendung

In Software-Anwendungen ist die Trennung und sinnvolle Aufteilung des Sourcecodes essenziell für die Wartbar- und Lesbarkeit eines Programms.

Backend

Entities

Entitäten im Backend repräsentieren die Geschäftsobjekte. Diese werden durch einen Object-Mapper in Tabellen mitsamt ihren dazugehörigen Constraints übersetzt. Prozedural werden sie als POJOs, also sogenannte *Plain Old Java Objects*, verwendet.

In den sogenannten Entity-Klassen werden ebenfalls die Beziehungen zu jeweils anderen Entitäten definiert.

Listing 7: LeoMail: Ausschnitt der Vorlagen-Entität

```
@Entity
public class Template extends PanacheEntityBase {

    @Id
    @GeneratedValue(strategy = GenerationType.UUID)
    public String id;

    @Column(length = 128, nullable = false, unique = true)
    public String name;

    @ManyToOne(fetch = FetchType.LAZY)
    public Project project;

    @CreationTimestamp
    public LocalDateTime created;

    ...

    @ManyToOne(fetch = FetchType.LAZY)
    public Contact createdBy;

    @ManyToOne(fetch = FetchType.LAZY)
    public TemplateGreeting greeting;
```

```
24
25      ...
26  }
```

Repositories

Repositories sind für alle *CRUD*-Operationen in der Datenbank verantwortlich: sie erstellen, lesen, überschreiben oder löschen Daten aus der Datenbank. Sie sind die Schnittstelle zwischen der Datenbank und den *Service*-Klassen, die die eigentliche Geschäftslogik enthalten. Durch Panache wird in Quarkus die Erstellung solcher Repository-Klassen stark vereinfacht.

Listing 8: LeoMail: Ausschnitt aus dem User-Repository

```
1  @ApplicationScoped
2  public class UserRepository implements PanacheRepository<NaturalContact> {
3      ...
4  }
```

Services

Services beinhalten die eigentliche Geschäftslogik: hier werden Validierungen, Transaktionen oder komplexe Algorithmen durchgeführt. Für Datenbankzugriffe verwenden sie, wie bereits erwähnt, Repository-Klassen.

Resources

Resources sind nun die Schnittstelle zur Geschäftslogik für die Nutzer der Anwendung: an diese Klassen werden die REST-Anfragen seitens des Frontends gestellt.

Diese Schnittstellen können prinzipiell in zwei verschiedene Kategorien unterteilt werden:

- **Öffentliche Routen**: Diese Routen sind auch von nicht eingeloggten Nutzern verwendbar, beispielsweise eine Login-Route oder eine öffentlich zugängliche Statistik.
- **Geschützte Routen**: Diese Routen sind nur von eingeloggten Nutzern verwendbar, teilweise auch nur mit bestimmten Berechtigungen.

Listing 9: LeoMail: Ausschnitt aus der Status-Resource

```
1  @Path("/status")
2  public class StatusResource {
3
4      ...
```

```
5
6       @GET
7       @Path("/import")
8       public Response getImportStatus() {
9           return Response.ok(Map.of("importing",
                importStatusService.isImporting())).build();
10      }
11  }
```

Frontend

Components

Vue-Komponenten sind die grundlegenden Bausteine der Benutzeroberfläche im Frontend. Typischerweise werden sie als Single-File-Components (.vue-Dateien) implementiert. Eine solche Datei vereint das HTML-Template, Styles (CSS/SCSS) und die zugehörige TypeScript-Logik.

Größere Seitenansichten werden aus mehreren kleineren Komponenten zusammengesetzt, um die Anwendung in logisch getrennte Teile zu gliedern. Ein Beispiel ist die Login-Komponente, welche das Anmeldeformular der Anwendung realisiert. Sie enthält Eingabefelder für E-Mail und Passwort sowie eine Schaltfläche zum Absenden. Die Logik der Komponente fängt das Abschicken des Formulars ab und ruft eine Methode des Auth-Stores (oder Auth-Services) auf, um die Anmeldedaten an das Backend zu übermitteln.

Listing 10: LeoMail: Ausschnitt aus der LoginView

```
1  <template>
2      <form @submit.prevent="login">
3          <input v-model="email" type="email" placeholder="E-Mail" />
4          <input v-model="password" type="password" placeholder="Passwort" />
5          <button type="submit">Login</button>
6      </form>
7  </template>
8
9  <script setup lang="ts">
10     import { ref } from 'vue';
11     import { useAuthStore } from '@/stores/auth.store';
12
13     const authStore = useAuthStore();
14     const email = ref('');
15     const password = ref('');
16
17     function login() {
18       authStore.login({ email: email.value, password: password.value });
19     }
20 </script>
```

Services

Die Service-Schicht im Frontend abstrahiert die Kommunikation mit dem Backend [12]. HTTP-Anfragen (z.B. via Axios) werden nicht direkt in den Vue-Komponenten gestellt,

sondern über zentrale Service-Klassen gekapselt. Dadurch kann die Logik für API-Aufrufe wiederverwendet werden und der Komponenten-Code bleibt schlanker.

Typischerweise existiert eine generische Service-Klasse, die einen konfigurierten *Axios-Client* (etwa mit Basis-URL) bereitstellt und grundlegende Methoden wie GET und POST definiert. Auf dieser Basis werden spezifische Services für einzelne Domänen erstellt – zum Beispiel ein *AuthService* für Authentifizierungs-Endpunkte oder ein *UserService* für Benutzerverwaltung.

Diese Services rufen intern die Methoden der Basisklasse auf, um Daten vom Backend zu laden oder zu senden. Komponenten und Stores nutzen anschließend diese Services, anstatt direkt axios aufzurufen, was die Wartung und Fehlersuche vereinfacht.

Listing 11: LeoMail: Ausschnitt aus der Service-Klasse

```
1  import axios from 'axios';
2
3  export class Service {
4    private client = axios.create({ baseURL: import.meta.env.VITE_API_URL });
5    constructor(private endpoint: string) {}
6
7    async getAll() {
8      return (await this.client.get(this.endpoint)).data;
9    }
10
11   async create(data: object) {
12     return (await this.client.post(this.endpoint, data)).data;
13   }
14 }
```

Stores

Im Frontend wird *Pinia* als State-Management-Lösung eingesetzt, um globalen Anwendungszustand zentral zu halten [90]. Ein Pinia-Store kapselt den State (Daten) sowie Methoden zu dessen Änderung (Actions) und optionale Getter für abgeleitete Werte. Mehrere Komponenten können so konsistent auf gemeinsame Daten zugreifen, ohne diese umständlich über Props oder Events austauschen zu müssen.

Ein Beispiel ist der App-Store, der das momentane Projekt trägt. Somit kann, ohne dass via GET-Parameter in der URL geführt werden muss, jede Komponente der Anwendung jederzeit wissen, welches Projekt vom Nutzer momentan bearbeitet wird.

Komponenten binden den Store mittels *useAppStore()* ein und können so Actions auslösen oder auf State-Daten reagieren. Änderungen im Store sind reaktiv, d.h. wenn z.B. der Login-Status gesetzt wird, aktualisieren sich alle verbundenen Komponenten (wie Navigationsleisten oder geschützte Seiteninhalte) automatisch.

Listing 12: LeoMail: Ausschnitt aus der App-Store-Klasse

```
import { defineStore } from 'pinia';

export const useAppStore = defineStore('app', {
    state: () => ({
        project: '',
    }),
    persist: {
        enabled: true,
        strategies: [
            {
                key: 'project',
            },
        ],
    }
});
```

Configs

Zentrale Konfigurationsdateien sorgen für konsistentes Verhalten im Frontend. Ein wichtiger Bestandteil ist die Router-Konfiguration (z.B. in router.ts), in der alle Routen der Single-Page-Application definiert sind. Hier lassen sich auch globale *Navigation Guards* einrichten, um bestimmte Routen nur authentifizierten Benutzern zugänglich zu machen.

Im Frontend wird auch ein Request-Interceptor definiert, der bei jeder Anfrage automatisch das Authentifizierungs-Token aus dem Store als Authorization-Header hinzufügt. Durch diese zentralisierte Konfiguration müssen die Services diese Parameter nicht bei jedem Aufruf erneut setzen.

Listing 13: LeoMail: Ausschnitt aus der Router-Config

```
import { createRouter, createWebHistory } from 'vue-router';
import { useAuthStore } from '@/stores/auth.store';
import LoginPage from '@/modules/auth/Login.vue';
import DashboardPage from '@/modules/dashboard/Dashboard.vue';

const routes = [
  { path: '/login', component: LoginPage },
  { path: '/dashboard', component: DashboardPage, meta: { requiresAuth: true } }
];

const router = createRouter({
  history: createWebHistory(),
  routes
});

router.beforeEach((to, from) => {
  const auth = useAuthStore();
  if (to.meta.requiresAuth && !auth.isAuthenticated) {
    return '/login'; // umleiten zur Login-Seite
  }
});
```

5.3 Authentifizierung und User-System

Die Authentifizierung wird, wie bereits erwähnt, mittels Keycloak umgesetzt.

Token Handling

Für die Authentifizierung, die für die Dauer der aktiven Anwendung anhält, sind zwei Verfahren notwendig, welche sich immer weiter wiederholen:

Access Token und Refresh Token beantragen

Mittels einer Backend-Route, die über einen Quarkus REST-Client eine Anfrage an den Keycloak-Auth-Server an die Token-URL mit den Zugangsdaten des Nutzers stellt, werden der *Access-Token* sowie ein *Refresh-Token* beantragt.

Mit dem Access-Token kann sich der Nutzer, sofern er ihn über den *Authorization*-HTTP-Header in der Anfrage mitsendet, gegenüber dem Backend verifizieren. Wenn der Access-Token gültig ist, wird die Anfrage gestattet, ansonsten wird ein *401 Unauthorized*-Fehler zurück an den Client gesendet.

Listing 14: Quarkus: Beispiel einer Rückmeldung der Login-Route

```
{
  "access_token": "eyJh...",
  "expires_in": 300,
  "refresh_expires_in": 0,
  "refresh_token": "eyJh...",
  "token_type": "Bearer",
  "id_token": "...",
  "not-before-policy": 0,
  "session_state": "a95ddb79-d8bb-44d5-84ff-8e998d3812e2",
  "scope": "openid profile offline_access email"
}
```

Tokens aktualisieren

Access Tokens laufen nach einer bestimmten Zeit ab - bei LeoMail ist diese Zeit auf 5 Minuten konfiguriert. Danach kann für einige Zeit - noch weitere 15 Minuten - mittels des Refresh-Tokens ein neuer Access-Token (sowie ein neuer Refresh-Token) beantragt werden.

Funktion der LDAP-Nutzer in der Anwendung

Die Nutzer aus der Schuldomäne, also die Nutzer, die im Keycloak automatisch aus dem LDAP-System der Schule importiert werden, sind allesamt als natürliche Kontaktpersonen in der Anwendung verfügbar. Sie können allerdings nicht bearbeitet werden.

5.4 Versenden von Vorlagen

Der Mailversand war eine größere Herausforderung, denn die Quarkus Mailer Extension konnte hier nicht angewandt werden: diese kann nämlich nur mit einer Sender-E-Mail-Adresse arbeiten; ein definiertes Feature von LeoMail lautete, dass sowohl von einer definierten Projekt-E-Mail-Adresse als auch von der persönlichen E-Mail-Adresse gesendet werden kann. [73]

Nach einer kurzen Umfeldanalyse, um nach Alternativen für die Quarkus-Extension zu suchen, wurde schnell die Jakarta-Mail-Dependency gefunden [23]. Diese ermöglicht das prozedurale Verbinden verschiedener Konten zu einem SMTP-Server.

Zwar können die Verbindungseinstellungen nicht, wie bei der Quarkus-Erweiterung, in den *application.properties* festgelegt werden, da allerdings in der Entwicklung der Leitsatz *Single Source of Truth* gilt, wurden für die Einstellungen sogenannte *Custom Configuration Properties* in der Quarkus-Konfiguration angelegt.

Listing 15: Konfiguration der Mail-Verbindungsdaten in Quarkus

```
1 # Mail
2 jakarta.mail.host=smtp-mail.outlook.com
3 jakarta.mail.port=587
```

Überprüfen der persönlichen Zugangsdaten

Beim ersten Anmelden eines Nutzers in der Anwendung muss initial ein Passwort angegeben werden, mit welchem sich der Nutzer mit seiner persönlichen Schul-E-Mail-Adresse, also *v.nachname@students.htl-leonding.ac.at* oder *v.nachname@htl-leonding.ac.at*, anmeldet, siehe Benutzerhandbuch.

Hier wird ein JSON-Objekt mit der E-Mail-Adresse, welche automatisch aus dem *JSON Web Token* ausgelesen wird, und dem eingegebenen Passwort an die */api/auth/save-outlook-password* gesendet. Diese ruft eine Service-Methode auf, welche eine SMTP-Verbindung öffnet. Sollte diese scheitern, wird eine *MessagingException* geworfen, und

es wird *false* zurückgeworfen. Bei erfolgreichem Verifizieren der Zugangsdaten mit dem Outlook-SMTP-Server wird *true* zurückgegeben, was dazu führt, dass dieses Passwort mit einem symmetrischen Verschlüsselungsalgorithmus verschlüsselt wird. Dieses verschlüsselte Passwort wird im Anschluss im Datensatz der *NaturalContact*-Instanz des Nutzers gespeichert.

Listing 16: Algorithmus zur Verifizierung der Outlook-Zugangsdaten

```
1  public boolean verifyOutlookCredentials(String email, String password) {
2          Properties properties = new Properties();
3          properties.put("mail.smtp.host", mailHost);
4          properties.put("mail.smtp.port", mailPort);
5          properties.put("mail.smtp.auth", "true");
6          properties.put("mail.smtp.starttls.enable", "true");
7
8          Session session = Session.getInstance(properties);
9
10         try {
11             Transport transport = session.getTransport("smtp");
12             transport.connect(mailHost,
                   email.replace("students.htl-leonding.ac.at",
                   "htblaleonding.onmicrosoft.com"), password);
13             transport.close();
14
15             return true;
16         } catch (MessagingException e) {
17             return false;
18         }
19 }
```

In der Zeile 12 der Abbildung 11 wird bei der Verbindung mit dem SMTP-Server *@students.htl-leonding.ac.at* mit *@htblaleonding.onmicrosoft.com* ausgetauscht. Das ist deshalb so handzuhaben, da erstere Variante ein neuer Alias für die alten E-Mail-Adressen ist, und wenige Konten diesen nicht verwenden. Mit dem Austauschen der Domain wird zuverlässig dafür gesorgt, dass die Konten sich bei korrekten Zugangsdaten korrekt verifizieren können.

Anlegen einer Projekt-E-Mail-Adresse

Der Prozess zur Einrichtung einer Projekt-E-Mail-Adresse funktioniert exakt gleich - ein verschlüsseltes Passwort wird nach der Überprüfung mit derselben Methode im Projekt abgespeichert.

Mail Scheduling

Für das Scheduling der E-Mails wurde die Quarkus-Extension *Scheduler* [75] verwendet.

In Vorbereitung gab es kleinere Veränderungen in der SentTemplate- und in der SentMail-Entity. Es wurden folgende Tabellenspalten in den Entitäten hinzugefügt:

Listing 17: Quarkus: Entity-Attribute für das Mail-Scheduling

```
# SentTemplate
public LocalDateTime scheduledAt;
public LocalDateTime sentOn;

# SentMail
public boolean sent = false;
```

Wenn ein E-Mail-Template direkt versendet werden soll, dann ist *scheduledAt null* und *sentOn* wird auf das aktuelle Datum gesetzt. Wenn ein E-Mail-Template an einem bestimmten Zeitpunkt versendet werden soll, dann ist *scheduledAt* auf diesen Zeitpunkt gesetzt und *sentOn null*.

Die simpelste und ressourcensparendste Vorgehensweise für das Versenden nach Zeit ist ein Scheduler: dieser prüft nun mittels eines festgelegten Intervalls, welcher im Backend von LeoMail auf 15 Sekunden gesetzt ist, ob es noch ungesendete E-Mail-Vorlagen gibt, deren *scheduledAt*-Zeitpunkt bereits in der Vergangenheit liegt.

Die maximal 15-sekündige Verzögerung, die bei dem Senden einer Vorlage nun auftreten kann, ist zu vernachlässigen: einerseits ist es auf diesem Zeitniveau ohnehin auch schwer, einzuschätzen, wie lange der SMTP-Server benötigt, um die E-Mails zu versenden; außerdem gibt die Benutzeroberfläche von LeoMail ohnehin nur die Möglichkeit, bis auf die Minute genau E-Mails zu planen.

Listing 18: Quarkus: Scheduling-Algorithmus

```
@Scheduled(every = "15s")
@Transactional
public void checkScheduledTemplates() {
    List<SentTemplate> usedTemplates = getAllScheduledUsedTemplates();

    for (SentTemplate usedTemplate : usedTemplates) {
        if ((usedTemplate.scheduledAt.isBefore(java.time.LocalDateTime.now()) ||
            usedTemplate.scheduledAt.isEqual(java.time.LocalDateTime.now())) &&
            usedTemplate.sentOn == null) {
            mailRepository.sendMail(usedTemplate.id);
        }
    }
}
```

Template Rendering

Das Rendern der Vorlagen passiert über die Quarkus-Template-Extension *Qute* [74]. Diese Technologieentscheidung wurde vor allem auf Basis der Erweiterbarkeit gefällt, da somit in zukünftigen Versionen mit relativ geringem Aufwand auch Serienfelder eingebaut werden können. Der Sourcecode für Serienfelder ist in den Ausmaßen eines Prototyps bereits gegeben.

Die Vorlage wird im Frontend als HTML an das Frontend gesendet - Standardserienfelder, wie beispielsweise *firstName*, können in geschwungenen Klammern an das Backend gesendet werden und werden dort auf Basis der Daten der Empfänger geparsed.

Ähnliches passiert mit der Anrede: deren Wert ebenfalls ein *String* mit dem Inhalt eines *Qute*-Templates ist, der vor der Vorlage hinzugefügt wird.

Qute funktioniert so, dass für jede einzelne E-Mail eine eigene *TemplateInstance* erstellt werden muss, da natürlich jede einzelne E-Mail andere Namen der Adressaten beinhaltet. Diese kann nach dem Ersetzen der einzelnen Variablen durch die realen Werte in einen String gerendert werden.

Listing 19: Quarkus: Qute-Template-Rendering

```
1  private List<TemplateInstance> buildTemplateInstances(String templateId,
       List<Contact> contacts, boolean personalized) {
2          Template template = Template.findById(templateId);
3          if (template == null) throw new IllegalArgumentException("Template not
               found.");
4
5          String combinedTemplate = template.greeting.templateString + "<br>" +
               template.content;
6          io.quarkus.qute.Template quteTemplate = engine.parse(combinedTemplate);
7          List<String> templateVariables =
               extractTemplateVariables(combinedTemplate);
8
9          List<TemplateInstance> instances = new ArrayList<>(contacts.size());
10
11         for (Contact contact : contacts) {
12             TemplateInstance instance = quteTemplate.instance();
13             for (String variable : templateVariables) {
14                 String value = getContactValue(contact, variable);
15                 instance.data(variable, value != null ? value : "");
16             }
17             instance.data("personalized", personalized);
18             instance.data("sex", contact instanceof NaturalContact naturalContact ?
19                     (naturalContact.gender != null ?
                           naturalContact.gender.toString() : "") : "");
20             instances.add(instance);
21         }
22         return instances;
23     }
```

Verwenden von Anhängen

Nach der Fertigstellung des ersten Prototyps wurde mit Hr. Prof. Peter Bauer vereinbart, dass ein Dateiupload für die Anhänge benötigt wird. Es soll in den Vorlagen bereits deklariert werden, ob beim Versand der Mail auch ein Dateiupload von möglicherweise mehreren Dateien notwendig ist.

Die Dateien werden via *Multipart Form Data* an das Backend gesendet: hier sind auch die mehreren Dateien, inklusive Dateitypen und Größen, enthalten, ebenso wie die Metadaten zum Versand der E-Mail, wie beispielsweise die gewählte Vorlage, die Empfänger oder der geplante Versandzeitpunkt.

Diese Dateiinformationen werden im Anschluss auf den *MinIO*-File Storage hochgeladen [55]. Dies funktioniert über die Quarkus-Extension *MinIO Client*. Hierzu werden Dateiname, Dateityp und Dateigröße genommen, und aus dem Dateinamen und einer zufällig generierten eindeutigen ID wird ein Objektname generiert, unter welchem dann die Datei im File Storage zu finden ist. Daraus wird eine Hibernate-Entität erzeugt und persistiert. Jede dieser Entitäten wird der jeweiligen *SentTemplate*-Entität hinzugefügt.

Über eine */api/mail/attachment*-Route können die Dateianhänge auch einzeln, via der Entitäts-ID, heruntergeladen werden. Das ist im Frontend in der Mail-Detailansicht eingebunden. Dieser Anwendungsbereich ist auch der Grund für die Integration von MinIO - wenn die Dateien nicht in der Anwendung wieder abrufbar sein müssten, würde auch ein einmaliger Upload zum Versand der E-Mail ausreichen.

Die *MinIO*-Upload, -Download oder -Löschfunktion ist intuitiv über einfache Methodenzugriffe ausführbar.

Um die Konsistenz zwischen File Storage und der Datenbank beizubehalten, wird bei den *Attachment*-Entitäten mit einem Entity Listener gearbeitet. Nach dem Löschen einer Entität wird automatisch die Datei ebenfalls auf dem Dateiserver gelöscht. Durch die Datenbanktransaktion, in welcher der Upload von Dateien gehandhabt wird, wird auch sichergestellt, dass bei Einfügen eines Datensatzes in den persistenten Speicher immer auch eine hochgeladene Datei vorhanden und verfügbar ist.

Listing 20: Quarkus: AttachmentEntityListener

```
1  public class AttachmentEntityListener {
2
3      @PreRemove
4      public void preRemove(Attachment attachment) {
5          StorageService storageService =
               CDI.current().select(StorageService.class).get();
6          try {
7              storageService.deleteFile(attachment.filePath);
8          } catch (Exception e) {
9              throw new RuntimeException("...", e);
10         }
11     }
12 }
```

Listing 21: Quarkus: Einbindung des Entity Listeners

```
1  @Entity
2  @EntityListeners(AttachmentEntityListener.class)
3  public class Attachment extends PanacheEntityBase {
4      ...
5  }
```

Versand der E-Mail

Der Versand der E-Mail wird nun, wie bereits erwähnt, über die *Jakarta Mail Dependency* gehandhabt.

Abhängig davon, ob nun die Projekt- oder die persönliche E-Mail-Adresse zum Versand verwendet wird, wird nun dieses Passwort seitens des Backends mit dem privaten Schlüssel entschlüsselt, um sich mit dem Outlook-SMTP-Server zu authentifizieren. Auch hier wird bei E-Mail-Adressen mit dem *@students.htl-leonding.ac.at*-Suffix aus Vorsichtsgründen dieser ersetzt.

Nun wird die E-Mail Schritt für Schritt zusammengebaut. Besonders wichtig ist die Setzung des *Content-Type*-Headers: denn die E-Mail-Adressen werden aus HTML-Vorlagen versendet und dementsprechend muss dieser auf *text/html* gesetzt werden. Bei der Setzung des Contents muss außerdem die Zeichencodierung auf *UTF-8* gesetzt werden - würde hier, wie standardmäßig, *ASCII* verwendet werden, könnten beispielsweise spezifische Zeichen wie Emojis oder Umlaute nicht verwendet werden.

Danach werden die *Attachment*-Entitäten des *SentTemplate* durchlaufen und jedes Attachment wird nochmal als InputStream vom FileStorage heruntergeladen. Jeder Anhang wird nun einzeln, mit seinem InputStream sowie seinem Content-Type (also Dateityp), an die E-Mail angehängt.

Die gerenderten Nachrichten, wie in ***Template Rendering*** beschrieben, werden nun mitsamt den E-Mail-Anhängen und dem in der Vorlage gesetzten Betreff an die Empfänger einzeln gesendet.

5.5 Deployment, CI/CD und Produktiveinsatz

Für das Deployment wurde die schulinterne LeoCloud verwendet, ein Self-hosted Kubernetes-Cluster, das innerhalb des Schulnetzwerks betrieben wird.

Sicherheit der Pipeline

Sicherheit spielt in Pipelines eine große Rolle: vor allem bei Zugangsdaten oder -schlüsseln. Diese liegen, im Falle von LeoMail, im Entwicklungsprofil ungeschützt. Daher ist es wichtig, dass die Daten, die schlussendlich im Produktivsystem verwendet werden, mittels Geheimschlüssel geschützt werden.

Hierzu können, vor allem beim Deployment mit GitHub Actions, die GitHub Repository Secrets verwendet werden [28]. Die Secrets dienen, beginnend mit einem Dollar-Symbol und endend mit einem Namen, eingeschlossen von geschwungenen Klammern, als Ersatz für die Konfigurationswerte in Klartext. Schlüssel und Werte der Secrets können auf GitHub unter den Einstellungen des Repositorys festgelegt werden.

Docker-Images bauen und zur Verfügung stellen

Sowohl für das Backend als auch für das Frontend müssen Docker-Images gebaut werden.

Im Falle des Backends wird hierbei im *Dockerfile* die ausführbare *JAR*-Datei aus dem *target/*-Ordner in den Container kopiert.

Da die LeoCloud im Schulnetzwerk liegt, genauso wie der Authentifizierungsserver *Keycloak*, wird bei Auflösen der Domain des Authentifizierungsservers normalerweise versucht, intern zu routen und nicht über die externe IP-Adresse. Aufgrund einer Fehlkonfiguration kam es hier zu Problemen und es wurde notwendig, die IP-Adresse der Maschine, auf der *Keycloak* betrieben wird, in die */etc/hosts*-Datei zu schreiben, um die Erreichbarkeit von *Keycloak* durch das Backend zu ermöglichen.

Weiters wird anschließend die ausführbare Java-Datei ausgeführt und das Backend gestartet.

Listing 22: Quarkus: Dockerfile

```
1 FROM eclipse-temurin:21-jre
2 RUN mkdir -p /opt/application
3 COPY target/*-runner.jar /opt/application/backend.jar
4 RUN id
5 CMD ["sh", "-c", "echo '\n10.191.112.13\tauth.htl-leonding.ac.at\n' >> /etc/hosts
      && java -jar backend.jar"]
6 WORKDIR /opt/application
```

Im Frontend wird auf Basis eines *NGINX*-Webservers im Dockerfile gearbeitet. Der Inhalt des *dist/*-Verzeichnisses, in welchem die Web-Anwendung nach dem Production-Build liegt, wird in den */usr/share/nginx/html*-Ordner verschoben. Weiters wird eine *default.conf* kopiert, welche den Hostnamen festlegt sowie das Stammverzeichnis der Webanwendung.

Listing 23: Vue.js: Dockerfile

```
1 FROM nginx:stable
2 COPY dist/ /usr/share/nginx/html/
3 COPY src/configs/deployment/default.conf /etc/nginx/conf.d/default.conf
```

Diese Dockerfiles werden im GitHub Actions Workflow nach dem Ausführen der Build-Befehle von Backend und Frontend zu jeweils einem Docker-Image gebaut und auf *GHCR* [26], einer Docker-Image-Registry von GitHub, gepusht.

Konfiguration der Deployment-Files

Um nun die konkreten Pods, welche man sich als kleine virtuelle Maschinen vorstellen kann, auf denen die Maschinen laufen, zu starten, müssen für jede Anwendung nun Deployment-Files erstellt werden.

Das bezieht sich im Falle von LeoMail auf:

- Vue.js Frontend
- Quarkus Backend
- PostgreSQL Datenbank
- MinIO File Storage

Beispiel anhand von PostgreSQL

```
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: postgres
5    namespace: student-it200274
6  spec:
7    replicas: 1
8    selector:
9      matchLabels:
10       app: postgres
11   template:
12     metadata:
13       labels:
14         app: postgres
15     spec:
16       containers:
17         - name: postgres
18           image: postgres:17
19           ports:
20             - containerPort: 5432
21           envFrom:
22             - secretRef:
23                 name: postgres-secrets
24           volumeMounts:
25             - name: postgres-storage
26               mountPath: /var/lib/postgresql/data
27           resources:
28             requests:
29               memory: "1Gi"
30               cpu: "500m"
31             limits:
32               memory: "2Gi"
33               cpu: "1"
34       volumes:
35         - name: postgres-storage
36           persistentVolumeClaim:
37             claimName: postgres-pvc
```

Relevante Konfigurationsabschnitte:

- **image: postgres:17**: Docker-Image, welches innerhalb des Pods ausgeführt wird
- **containerPort: 5432**: Der Port, unter welchem der Dienst des Docker-Images läuft
- **secretRef**: Definition, welche Secrets als Umgebungsvariablen verwendet werden. Bei dieser Variante müssen die Namen der einzelnen Secrets exakt den Umgebungsvariablen entsprechen
- **volumeMounts**: Verzeichnis, welches innerhalb des definierten Volumes gespeichert wird.
- **volumes**: Abschnitt, in dem definiert wird, welcher PersistentVolumeClaim verwendet wird, um die Daten persistent (über mehrere Pod-Lifecycle hinweg) zu speichern. Dieser PersistentVolumeClaim muss zuvor angelegt werden.

Neben der Konfiguration des Deployments an sich, muss ebenfalls ein *Service*, als auch (in diesem Fall) ein *PersistentVolumeClaim* angelegt werden.

Ein Service sorgt dafür, dass der Pod auch von anderen Pods im selben Namespace via dem Service-Namen sowie dem Dienstport erreichbar ist.

Ein PersistentVolumeClaim legt ein Volume an, in welchem bestimmte Ordner oder der ganze Dienst persistent gespeichert werden kann. Das führt dazu, dass etwaige gespeicherte Daten, wie in PostgreSQL z.B. die Datenbankeinträge oder die Datenbankstruktur, bei Systemneustarts oder -ausfällen nicht verloren gehen.

Damit die Web-Anwendung auch unter dem Hostnamen richtig erreichbar ist, muss ein Ingress für den Namespace konfiguriert werden. Hier können bestimmte Pfade zu bestimmten Services zugeordnet werden. Damit kann definiert werden, unter welchem Pfad das Front- oder das Backend erreichbar ist.

Deployments zur Verfügung stellen

Nach der Konfiguration aller Deployment-Files können diese nun im Cluster zur Verfügung gestellt werden.

Zu Beginn ist es notwendig, aus den GitHub-Repository-Secrets die Base64-codierte Version der kubectl-Konfiguration zur Interaktion mit dem Kubernetes-Cluster zu importieren und anzuwenden [25].

Listing 24: Deployment: Kubectl-Config anwenden

```
- name: Set up kubeconfig
run: |
    mkdir -p $HOME/.kube
    echo "${{ secrets.KUBECONFIG_BASE64 }}" | base64 -d > $HOME/.kube/config
    chmod 600 $HOME/.kube/config
```

Die Kubernetes Secrets müssen mithilfe der, wie oben beschriebenen, Umgebungsvariablen angelegt werden.

Listing 25: Deployment: Kubernetes Secrets anlegen

```
- name: Apply Kubernetes secrets
run: |
  kubectl create secret generic postgres-secrets \
    --from-literal=POSTGRES_DB="${POSTGRES_DB}" \
    --from-literal=POSTGRES_USER="${POSTGRES_USER}" \
    --from-literal=POSTGRES_PASSWORD="${POSTGRES_PASSWORD}" \
    --dry-run=client -o yaml | kubectl apply -f -
```

Im Anschluss können die Deployment-Ressourcen angewandt werden. Bei Änderungen werden diese automatisch neu gestartet, ansonsten ist es notwendig, sie manuell neu zu starten.

Listing 26: Deployment: Anwendung einer Ressource (bspw. PostgreSQL und Ingress)

```
- name: Apply Postgres and Ingress Resources
if: matrix.service == 'backend'
run: |
  kubectl apply -f .github/k8s/postgres.yaml
  kubectl apply -f .github/k8s/ingress.yaml
```

Vorbereitungen für den Produktiveinsatz

Nach dem erfolgreichen Durchlauf der Pipeline und einer erreichbaren Anwendung werden nun automatisch alle Keycloak-Nutzer in die Datenbank der Applikation importiert.

Im Dev-Build hingegen werden nur die Nutzer des Projektteams importiert. Dies liegt daran, dass der Code für den Import der Keycloak-Nutzer nur dann ausgeführt wird, wenn sich das Backend im Production-Profil befindet.

Mit ähnlicher technologischer Vorgehensweise werden außerdem initiale Daten, wie die *TemplateGreeting*-Entitäten, im Produktionseinsatz angelegt.

6 Design

Wie bei jedem Designprozess liegt der Schlüssel zum Erfolg darin, sorgfältig darüber nachzudenken, was man erreichen möchte.

Das Ziel war es, eine simple und optisch ansprechende Benutzeroberfläche zu designen. Diese soll den Usern ein angenehmes Erlebnis ermöglichen, ihnen aber gleichzeitig die Möglichkeit geben, erforderliche Dinge schnell und einfach zu erledigen. Der Fokus liegt hierbei auf dem Service und den Funktionen der Software.

Bei der Umsetzung des Designs wurde auf wichtige Regeln eines guten UI-Designs geachtet. Wichtige Regeln eines guten UI-Designs:

Was ist User-Interface-Design?

Das User Interface (UI) beschreibt den Bereich einer Software, Website oder App, mit dem Nutzer interagieren. Es geht dabei primär um die optische und ästhetische Gestaltung der Benutzeroberfläche. Ziel ist es, durch die gezielte Auswahl von Farben, Schriften und Formen eine ansprechende und benutzerfreundliche Umgebung zu schaffen
.

Elemente des UI-Designs Die Gestaltung der Oberfläche umfasst verschiedene Elemente, darunter:

- Layout
- Visuelle Hierarchien
- Farbschema
- Typografie
- Bilder
- Icons
- Animationen
- Illustrationen

Zusätzlich spielen interaktive Elemente eine zentrale Rolle im UI-Design. Diese Elemente sorgen dafür, dass Nutzer aktiv mit der Oberfläche interagieren können. Zu den interaktiven Komponenten gehören [83]:

- Input-Elemente (Dropdowns, Buttons, Formularfelder...)
- Output-Elemente (Warnmeldungen, Erfolgs- sowie Fehlermeldungen...)
- Hilfs-Elemente (Navigation, Information, Container...)
- Slider
- Bild-Text-Kombination
- Filme
- Interaktive Karten
- Grafiken

Bei der Gestaltung werden stets wichtige Regeln des UI-Designs beachtet.

Wichtige Regeln eines guten UI-Designs

Hierarchie

Wenn auf einer Seite alles gleich wichtig erscheint, ist eigentlich gar nichts mehr wichtig. Deshalb sind visuelle Hierarchien ein großer Bestandteil eines guten Designs. Man muss visuelle Hinweise verwenden, die den Usern zeigen, worauf sie zuerst, zweitens, drittens usw. achten sollen. Um sicherzustellen, dass sich die wichtigsten Elemente vom Rest abheben, kann man dies entweder durch Dinge wie Maßstab oder Farbe bewirken. Es gibt auch die Möglichkeit, typografische Hierarchien zu verwenden, mittels unterschiedlicher Schriftarten, -größen und -stärken [64].

Progressive Offenlegung

Ein Nutzer kann sehr schnell mit vielen Fragen und Funktionen überwältigt werden, was dazu führen kann, dass man einen Verlust von Nutzer:innen erleidet. Deshalb müssen gute Designer das Interface so gestalten, dass sich der Nutzer gut orientieren kann und weiß, wie viele Schritte für ihn noch zu tätigen sind.

Es soll eine angemessene Menge an Informationen bereitgestellt werden, um einen mehrspurigen Prozess zu ermöglichen, ohne den Nutzer zu überfordern. Deshalb ist das Priorisieren von durchdacht aufgenommenen und ausgeschlossenen Daten auf der Benutzeroberfläche wichtig [64].

Gleichgewicht

Um das User Interface stimmiger zu gestalten, ist es wichtig, auf die Balance zu achten. Dies kann auf verschiedene Weise erreicht werden [30]:

- **Symmetrische Balance**

  Sie ist die einfachste Form der Balance und wird erreicht, wenn Elemente auf beiden Seiten einer zentralen vertikalen Achse gleich sind. Damit meint man zum Beispiel zwei Textblöcke, die gleich aussehen, aber vom Inhalt her nicht identisch sein müssen.

- **Asymmetrische Balance**

  Dies ist das Gegenteil der symmetrischen Balance. Hier sind die Elemente auf beiden Seiten nicht gleich. Die asymmetrische Balance kann auch durch die Trennung der vertikalen Achse erreicht werden, jedoch soll diese nicht zentriert platziert werden. Dadurch erzielt man ein ausgewogenes Erscheinungsbild, welches das schmalere Objekt „schwerer“ wirken lässt.

Konsistenz

Dass die Größe einer Schaltfläche oder die einheitliche Farbe eine große Rolle spielen, ist für viele Nutzer zwar nicht offensichtlich, kann jedoch dennoch Verwirrung stiften. UI-Designer müssen sich stets an die Konsistenz ihres Designs halten. Ist eine UI-Taste größer als die andere, führt das nicht nur zu Verzögerungen, sondern auch zu Verunsicherung. Nutzer fragen sich, warum dieses Element anders ist als alle anderen. Wenn man eine solche Unregelmäßigkeit in das Design einbaut, braucht es eine gute Begründung dafür, warum man vom etablierten Muster abweicht.

Die Konsistenz schafft außerdem Vertrauen bei den Nutzern. Sie müssen nicht mehr darüber nachdenken, wie einzelne Elemente zu bedienen sind, sondern können sich vollständig auf ihre Aufgabe konzentrieren. [29]

Kontrast

Um die Nutzer strategisch auf wichtige Inhalte aufmerksam zu machen, verwendet man starke Kontraste. Diese lenken die Aufmerksamkeit der Nutzer und fördern schnelleres Handeln. Ein gutes Beispiel hierfür ist die Verwendung der Farbe Rot bei einer Schaltfläche wie „Daten löschen“, um den Nutzer auf ein „Achtung“ hinzuweisen. Bei einer Schaltfläche wie „Konto speichern“ hingegen wäre die Farbe Grün ideal, da sie signalisiert, dass die Handlung keine negativen Konsequenzen hat. [29]

Barrierefreiheit

Jeder vierte Nutzer weltweit ist von einer Sehbehinderung betroffen. Daher muss darauf geachtet werden, die Farben und die Leuchtkraft so zu kontrastieren, dass das Design auch für Benutzer mit Sehbehinderung eindeutig und benutzerfreundlich ist. Die Farben können problemlos mit verschiedenen Prüfwerkzeugen oder Plugins getestet werden.

Um sicherzustellen, dass alle relevanten Aspekte berücksichtigt wurden, sollte man die Bestimmungen der Web Content Accessibility Guidelines (WCAG) umsetzen, einschließlich [64]:

- Bereitstellung alternativer Texte
- Verwendung geeigneter Auffüllungsregeln
- Kompatibilität mit unterschiedlichen Technologien
- ordentlicher Tastaturnavigation
- ausreichendem Kontrast zwischen Vordergrund- und Hintergrundfarbe

Nähe

Im Design wird darauf geachtet, dass Dinge, die zusammengehören, auch zusammenbleiben. Nutzer empfinden nahestehende Elemente als zusammengehörig, was zu einem natürlichen Benutzerfluss und einer intuitiveren Benutzererfahrung führt.

Ein Beispiel dafür ist ein Streaming-Dienst, bei dem die Wiedergabe-, Vorspulen- und Rückspultasten nebeneinander angeordnet sind, während die Schaltfläche zum

Ausschalten hier nicht zu finden ist. Der Grund dafür ist, unabsichtliche Klicks zu verhindern und gleichzeitig das Seherlebnis nicht zu unterbrechen [30].

Ausrichtung

Was ist ein Design ohne Ordnung und gute Lesbarkeit?

Ein gutes Design basiert auf einem ordentlichen und strukturierten System, das auch als Gittersystem bezeichnet wird. Dieses System lässt die Oberfläche nicht nur professioneller wirken, sondern verbessert durch seine konsistente Ausrichtung die Lesbarkeit und Vorhersehbarkeit. Dadurch entsteht für den Nutzer mehr Ordnung und eine leichtere sowie schnellere Navigation innerhalb der Software. [29]

Unterschied zwischen UI und UX-Design

Man unterscheidet zwischen User Interface (UI) und User Experience (UX). Während sich das UI mit der Gestaltung der Benutzeroberfläche befasst, beschäftigt sich das UX-Design mit der Schaffung eines positiven Nutzererlebnisses. Dennoch ist das UI stets ein wesentlicher Bestandteil des UX-Designs [82].

UX-Design

User Experience, auch UX genannt, befasst sich – wie der Name bereits sagt – mit den Nutzererfahrungen. Es berücksichtigt die Emotionen, Gedanken und Bedürfnisse eines Nutzers während der Interaktion mit einem System. Ziel ist es, diesen Prozess zu optimieren und komplexe Systeme einfacher sowie verständlicher zu gestalten. Man möchte den Nutzer so schnell wie möglich zu seinem gewünschten Ziel führen.

UX-Design ist für den wirtschaftlichen Erfolg vieler digitaler Produkte verantwortlich. Dies wird meistens dann erreicht, wenn es dem Nutzer nicht bewusst auffällt, sondern er intuitiv und reibungslos durch das System geleitet wird. Es spart nicht nur unübersichtliche, sondern auch komplizierte Darstellungen. Daher sollte die Bedeutung eines gut durchdachten UX-Designs niemals unterschätzt werden [19].

6.1 Logo

Bevor man ein Logo entwirft, sollte man sich ausreichend Gedanken darüber machen, was dieses Logo repräsentieren soll und welchen Eindruck es bei den Nutzern hinterlassen möchte.

Hier sind die Schritte aufgelistet, die durchgeführt wurden, um das finale LeoMail-Logo zu erstellen:

Grundidee

Die Grundidee war es, ein Logo zu kreieren, welches die HTL Leonding gut widerspiegelt, aber auch die Software grafisch verdeutlicht. Dabei wurde darauf geachtet, dass das Logo modern und leicht wiedererkennbar ist. Eines der wichtigen Aspekte war es, die Farben des HTL-Leonding-Logos beizubehalten, um die verschiedenen Fachrichtungen der Schule zu repräsentieren. Außerdem sollte das Logo eine professionelle Wirkung erzielen, passend zu einem E-Mail-Verwaltungssystem, das technisch als auch optisch überzeugt. Zusätzlich wurde festgelegt, dass das Logo aus einem Icon, dem App-Namen und einem App-Slogan bestehen soll, um einen klaren Wiedererkennungswert zu schaffen.

Namensfindung

Der Name der Software, **LeoMail**, wurde schnell festgelegt. Da die HTL Leonding bereits eine klare Namensstruktur für ihre Software verwendet – wie beispielsweise *LeoCloud* für die schuleigene Cloud – fiel die Namensfindung entsprechend leicht. Die Einheitlichkeit schafft ein durchgängiges Konzept, das für die Nutzer klar und verständlich ist. Der Name setzt sich aus dem einheitlichen Bestandteil *Leo*, der den Bezug zur Schule herstellt, und der Haupteigenschaft der Software, einem E-Mail-Verwaltungssystem, zusammen. Die einfache Kombination aus *Leo* und *Mail* sorgt dafür, dass der Name leicht auszusprechen und zu merken ist [17].

Regeln bei der Namensfindung

Bei der Namensfindung wurden außerdem folgende Regeln beachtet:

- **Wortkombination**

Die Nutzung von Wortkombinationen verleiht dem Namen Einzigartigkeit. Diese Methode eignet sich besonders gut für Software, die zwei oder mehrere Konzepte vereint, um diese gleichzeitig widerzuspiegeln. Bei der Namensfindung von *LeoMail* wurden die Wörter *Leonding* und *E-Mail-Verwaltungssystem* kombiniert, wodurch der Name *LeoMail* entstand.

- **Kurz, prägnant, knackig**

  Ein zentraler Punkt war es, den Namen kurz und prägnant zu halten, damit er einfach zu merken ist. Dies ist bei *LeoMail* gelungen, da die Hauptnutzer der Software – die Schüler der HTL Leonding – bereits mit der einheitlichen Namensstruktur vertraut sind.

- **Einprägsam und schreibbar**

  Es war entscheidend, einen Namen zu wählen, der leicht auszusprechen ist und von den Nutzern richtig geschrieben werden kann. *LeoMail* besteht aus zwei einfachen, klar verständlichen Wörtern, was es den Nutzern erleichtert, den Namen korrekt auszusprechen und zu schreiben.

- **Verzicht auf Umlaute, Zahlen und Bindestriche**

  Bei *LeoMail* wurde bewusst auf Umlaute, Zahlen und Bindestriche verzichtet, da diese den Namen unnötig kompliziert machen können. Umlaute sind besonders ungünstig, da sie für Domains problematisch sein können. In solchen Fällen müsste auch die alternative Schreibweise reserviert werden, um Verwechslungen zu vermeiden. Außerdem erleichtert der Verzicht auf Sonderzeichen und Bindestriche die Aussprache und Schreibweise für alle Nutzer.

Slogan

Der Slogan „connect-communicate-coordinate“ wurde bewusst gewählt, um die Funktionen der LeoMail-Software klar und prägnant zu erfassen. Die Entscheidung wurde basierend auf die Aufgaben, die die Software den Nutzern bietet, getroffen. Zuerst steht die Funktion „connect“ im Vordergrund. Diese soll die Kommunikationsfähigkeit, mit anderen in Kontakt zu treten, widerspiegeln. Die nächste Eigenschaft „communicate“ soll zeigen, dass LeoMail nicht nur eine Software zum Nachrichtenversenden ist, sondern auch eine effiziente Kommunikation innerhalb von Projekten unterstützt. Als Letztes wird noch der Begriff „coordinate“ genannt. Dieser Begriff betont die Fähigkeit der

Software, Informationen gut zu verwalten und zu organisieren, was besonders in der Schule oder bei der Organisation von verschiedenen Veranstaltungen wichtig ist. Der Slogan bringt den Nutzen der Software auf den Punkt, bietet aber auch eine klare und leicht verständliche Struktur. Außerdem klingt der Slogan nicht nur gut, sondern wirkt durch die dreifache Alliteration des Buchstabens „C“ flüssig und sorgt für einen leicht merkbaren Klang.

Schriftwahl

Die Wahl der Schriftart für das Logo fiel relativ leicht. Die Idee war, dieselbe Schriftart wie im HTL-Leonding-Logo zu verwenden. Da die genaue Schriftart nicht bekannt war, wurde das Logo in Adobe Illustrator importiert und die Schriftart mit den vorhandenen Schriftarten verglichen. Die Suche war schnell abgeschlossen, da es sich um die Schriftart Arial handelte.

Es wurde sorgfältig darauf geachtet, die Schrift möglichst originalgetreu zu gestalten. Dabei spielte auch der Buchstabenabstand eine wichtige Rolle, der schließlich auf 17px festgelegt wurde. So wurde sichergestellt, dass die Buchstaben weder beim Skalieren ineinander übergehen noch ihre Lesbarkeit verlieren, während das Logo weiterhin wie ein zusammenhängendes Wort wirkt.

Die Wahl der Schriftart für den Slogan stellte sich zunächst als Herausforderung heraus. Letztendlich erwies sich jedoch die Google-Schriftart „Manrope“ als die passendste Option. Diese wurde von Google Fonts heruntergeladen und in Adobe Illustrator integriert. Für den finalen Slogan entschied man sich für „Manrope ExtraLight“ mit einem Buchstabenabstand von 18. Anschließend wurde die Schrift unter dem Hauptlogo platziert, wodurch ein harmonisches Gesamtbild entstand.

Ergebnis

Abbildung 18: Finaler Textbaustein des Logos

Logo-Icon

Bei der Entwicklung des perfekten Logos wurden verschiedene Grundregeln beachtet [43]:

- **Halte es einfach**

  Ein gutes Logo sollte immer einfach, klassisch und vermieden werden, da sie das Logo nur für kurze Zeit attraktiv wirken lassen. Für eine weniger bekannte Software oder Marke empfiehlt es sich, ein Symbol zu wählen, das gut mit ihr assoziiert werden kann. Wenn dies nicht eindeutig ist, ist eine Kombination aus Text- und Bildmarke die bessere Wahl.

  Nach der Erstellung des Grundgerüsts sollte sparsam mit Elementen wie Schatten, Prägungen und Farben umgegangen werden. Dabei ist es wichtig, zu prüfen, wie das Logo auf farbigen Hintergründen wirkt, um mögliche Probleme zu vermeiden.

- **Anpassungsfähigkeit**

  Ein Logo muss in allen Größen und Farben reproduzierbar sein. Um dies zu gewährleisten, sollte es als Vektorgrafik erstellt werden, damit es ohne Qualitätsverlust beliebig skaliert werden kann. Zudem sollte das Logo für Social-Media-Plattformen geeignet sein, idealerweise in quadratischer Form und nicht nur in horizontaler oder vertikaler Ausrichtung.

  Das Design muss auch in verschiedenen Varianten funktionieren, etwa farbig oder schwarz-weiß. Wichtig ist außerdem, dass die Lesbarkeit auf unterschiedlichen Hintergründen erhalten bleibt und das Logo bei jeder Skalierung seine Proportionen beibehält.

- **Die richtige Farbe**

  Ein Logo sollte in einer einzigen Farbe funktionieren – entweder in Schwarz oder Weiß. Wenn das nicht der Fall ist, weist das Design Schwächen auf. Bevor mit Farben experimentiert wird, muss sichergestellt sein, dass das Logo in einer neutralen Farbe gut wirkt und Schrift sowie Formen klar erkennbar sind.

  Die Farbwahl sollte anschließend sorgfältig getroffen werden, da sie die Marke und ihre Botschaft stark beeinflusst. Dabei spielt die Farbpsychologie eine zentrale

Rolle. Grundsätzlich empfiehlt es sich, zwei Farben zu verwenden, auch wenn einige Unternehmen wie Google erfolgreich mit vier Farben arbeiten.

Außerdem ist es wichtig zu bedenken, dass Farben weltweit unterschiedlich wahrgenommen werden. Ebenso spielt der Einsatzort des Logos eine Rolle: Eine Farbe, die online hervorragend aussieht, kann im Druck ganz anders wirken.

- **Branding und Identifikation**

  Ein Logo vermittelt etwas über die Software, sei es ihre Philosophie oder ihre Funktionalität. Dadurch wird das Logo einprägsamer und verdeutlicht, was die Software bietet oder was sie ausmacht. Häufig entfaltet sich die Botschaft eines Logos erst bei genauerem Hinsehen.

- **Zeitlos und ansprechend**

  Eine der wichtigsten Eigenschaften, die ein gutes Logo ausmachen, ist die Zeitlosigkeit. Diese muss gewährleistet werden, ohne dass die Gefahr besteht, dass das Logo an Reiz verliert. Deshalb sollte auf Trends verzichtet werden, da diese schnell wieder verschwinden.

  Das bedeutet jedoch nicht, dass Trends komplett ausgeschlossen werden müssen. Viele Unternehmen erzielen durch die Nutzung von Design-Trends große Aufmerksamkeit. Dennoch sollte der Fokus eher auf klassischen Elementen liegen. Ein gutes Logo braucht Zeit und Geduld, um ihm ein einzigartiges Design zu verleihen.

Entwicklung des LeoMail-Logo-Icons

Eines war bei der Entwicklung des Logo-Icons von Anfang an klar: Die vier Grundfarben des Schullogos der HTL Leonding mussten berücksichtigt werden. Da der Name „LeoMail“ bereits festgelegt war, sollte auch das Logo-Icon einen Bezug zur HTL Leonding haben oder zumindest auf die Schule hinweisen. Dies sollte die Software stärker mit der Schule verbinden und auf sie verweisen. Die vier Grundfarben repräsentieren die Fachrichtungen der Schule: türkis, blau, orange und rot.

Nach mehreren Ideen und Überlegungen wurde schnell klar, dass der Begriff „Mail“ gut als Logo umsetzbar war. Die Frage war nun, wie man dies sinnvoll mit den vier Farben kombinieren könnte. Zunächst wurde das Logo von Google als Beispiel für

die Farbgestaltung betrachtet, doch schnell wurde klar, dass dieses Design nicht dem gewünschten Ergebnis entsprach.

Es folgte die Überlegung, wie man die Farben miteinander verbinden könnte. Dabei entstand die Idee, die Farben von Informatik und IT-Medientechnik sowie von Medizintechnik und Elektronik zu kombinieren. So entstanden zwei Farbpaletten: eine orange-rote und eine türkis-blaue. Weitere Recherchen zeigten, dass die Farben durch einen Verlauf harmonieren könnten. Allerdings wurde schnell klar, dass ein Verlauf von blau zu rot nicht passend wäre. Daher musste entschieden werden, die beiden Farbverläufe voneinander zu trennen.

Farbpalette

Türkis-Hellblau (IT-Medientechnik)	# 86b5d8	
Blau (Informatik)	# 0b519a	
Orange (Medizintechnik)	# e78614	
Rot (Elektronik)	# a2192b	

Abbildung 19: Farbpalette des LeoMail-Icons

Ab diesem Punkt stand fest:

- Das Logo wird in Form einer Mail dargestellt.
- Es werden zwei Farbverläufe verwendet: einer von türkis zu blau und einer von orange zu rot.
- Die Farbverläufe müssen getrennt bleiben.

Es wurde viel experimentiert und jede mögliche Darstellungsweise einer Mail auf Papier skizziert, bis schließlich eine Form gefunden wurde, die zum gewünschten Ergebnis führte. Diese Form stellt eine alternative Darstellung einer Mail dar. Auf den ersten Blick ist nicht sofort erkennbar, was es darstellt, aber bei genauerem Hinsehen erscheint es klar als eine Mail.

Dadurch, dass das Icon bewusst an ein typisches E-Mail-Symbol angelehnt wurde, lässt es sich auch als solches erkennen. Die Grundidee eines Briefumschlags wurde beibehalten,

jedoch ist die Form leicht versetzt und unterschiedlich angeordnet, was einen gewissen Wiedererkennungswert schafft.

Für einen der Farbverläufe wurde orange-rot gewählt und für den anderen türkis-blau. Anschließend wurde das Logo in Adobe Illustrator umgesetzt und die Farben so angepasst, bis das gewünschte Ergebnis erreicht war.

Ergebnis

Abbildung 20: Finales LeoMail-Icon

Design-Software

Für die Umsetzung des kompletten Logos wurde Adobe Illustrator verwendet. Dieses Tool ermöglicht es, mit verschiedenen Werkzeugen innovative und beeindruckende Designs wie Logos zu entwerfen. Da bereits Erfahrung mit diesem Tool vorhanden war, gab es keine Herausforderung, sich damit vertraut zu machen.

Abbildung 21: Logo von Adobe Illustrator [2]

Adobe Illustrator ist ein Teil der Adobe Creative Cloud, was bedeutet, dass es eng mit anderen Adobe-Programmen wie Photoshop und InDesign zusammenarbeitet. Illustrator ist ein sogenanntes Vektor-Grafik-Programm, das besonders gut für Vektorgrafiken geeignet ist. Es umfasst aber auch viele andere nützliche Funktionen wie:

- Erstellen von professionellen Grafiken und Diagrammen
- Gestaltung von hochwertigen Icons
- Erstellen von Logos
- Designen mit Hilfe von KI

- Vektorisieren von Bildern
- Bearbeitung und Gestaltung von Schriftarten und Texten
- Zeichnen von Illustrationen wie Webdesign, Werbung usw.

Da Vektorgrafiken aus Linien und Formen bestehen, können sie in jede Größe skaliert werden, ohne dass ein Qualitätsverlust auftritt [1].

Hindernisse

Eine der größten Herausforderungen bei der Logo-Entwicklung war die Auswahl der vier Grundfarben, die im Logo enthalten sein sollten. Das Hauptproblem bestand darin, diese Farben so zu kombinieren, dass das Logo nicht zu bunt wirkt. Obwohl das Ziel war, das Design einfach und schlicht zu halten, stellte es eine große Herausforderung dar, diese Aspekte mit den vielen Farben zu vereinen, ohne das Logo unnötig kompliziert wirken zu lassen. Eine weitere Schwierigkeit bestand darin, das Icon nicht wie ein typisches Mail-Icon aussehen zu lassen, sondern es innovativer und einzigartiger zu gestalten. Trotz dieser Hindernisse ist es gelungen, ein optisch ansprechendes und einzigartiges Logo zu entwickeln.

Finales Gesamtergebnis

Abbildung 22: Finaler Entwurf des Logos

6.2 Content-Strukturierung

Das Ziel der Content-Strukturierung besteht darin, ein ansprechendes und benutzerfreundliches Layout zu entwickeln. Der Schwerpunkt liegt darauf, die Inhalte klar zu präsentieren, sodass Nutzer sofort erkennen, welche Funktionen zur Verfügung stehen. Eine gezielte Platzierung der Elemente erzeugt zudem eine visuelle Hierarchie, die den Blick der Nutzer lenkt.

LeoMail sollte aufgrund ihrer umfangreichen Funktionalitäten einfach und minimalistisch gestaltet werden. Dieses minimalistische Design steigert nicht nur die Ästhetik, sondern lenkt auch die Aufmerksamkeit auf die zentralen Funktionen der Software. Eine sorgfältig abgestimmte Farbgestaltung trägt zusätzlich zur Verbesserung der Lesbarkeit und Übersichtlichkeit bei.

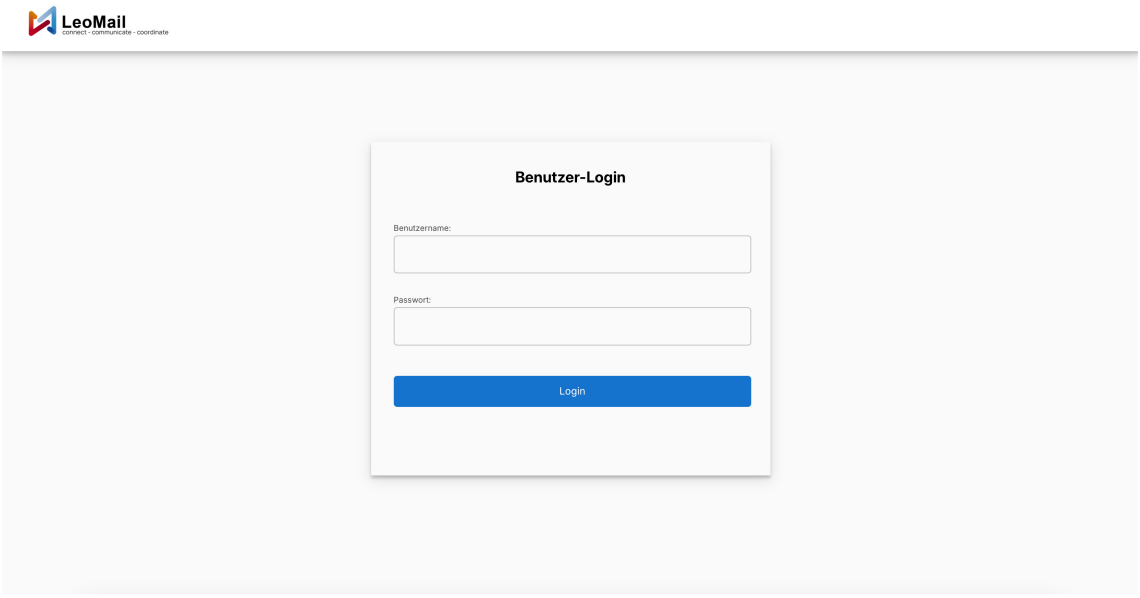

Abbildung 23: Darstellung der Anmeldeseite (Login-Page) der Webanwendung

Auf der Login-Seite wurde das Logo im linken Bereich des Headers platziert. Das zentrierte Login-Formular zieht die Aufmerksamkeit der Nutzer gezielt auf die Hauptfunktion der Seite. Der Login-Button sticht durch seinen auffälligen blauen Hintergrund hervor, der sich klar vom sonst schlichten Farbschema abhebt. Dank der zentralen Ausrichtung des Inhalts wird den Nutzern sofort klar, was zu tun ist. Der Fokus liegt ausschließlich auf dem Einloggen, ohne dass zusätzliche Elemente ablenken.

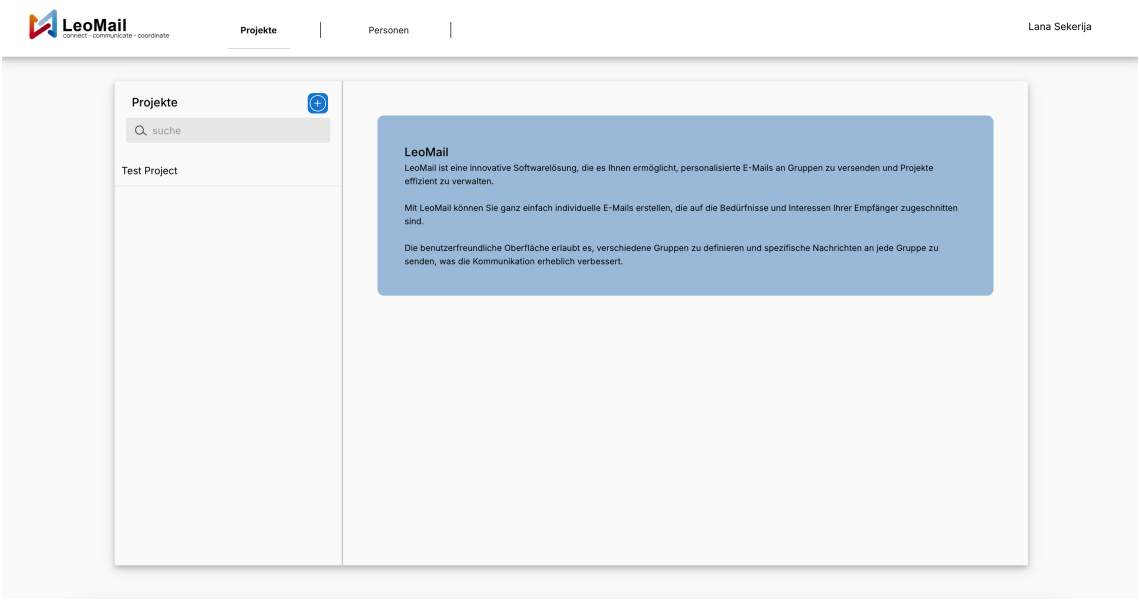

Abbildung 24: Darstellung der „Projekte“-Seite

Das Ziel der Projekt-Seite bestand darin, die einzelnen Projekte übersichtlich darzustellen und die beiden Unterseiten sinnvoll zu verlinken. Im oberen Bereich befindet sich eine standardmäßige horizontale Navigationsleiste mit dem Logo links und den Links zu den Unterseiten direkt daneben. In der rechten oberen Ecke wird die Nutzerinformation angezeigt, was durch die Darstellung des Namens die Personalisierung unterstützt.

Die linke Sidebar, die zur Projektübersicht dient, sorgt für eine klare Struktur, erleichtert die Orientierung und minimiert Verwirrung. Eine integrierte Suchfunktion in der Seitenleiste spart Zeit und verbessert die Nutzererfahrung. Das schlichte Layout verhindert visuelle Überforderung und lenkt den Fokus auf die wesentlichen Funktionen. Der großzügige Einsatz von weißem Leerraum verleiht dem Layout eine klare, professionelle und moderne Optik. Diese strukturierte Gestaltung reduziert zudem die Klickwege erheblich.

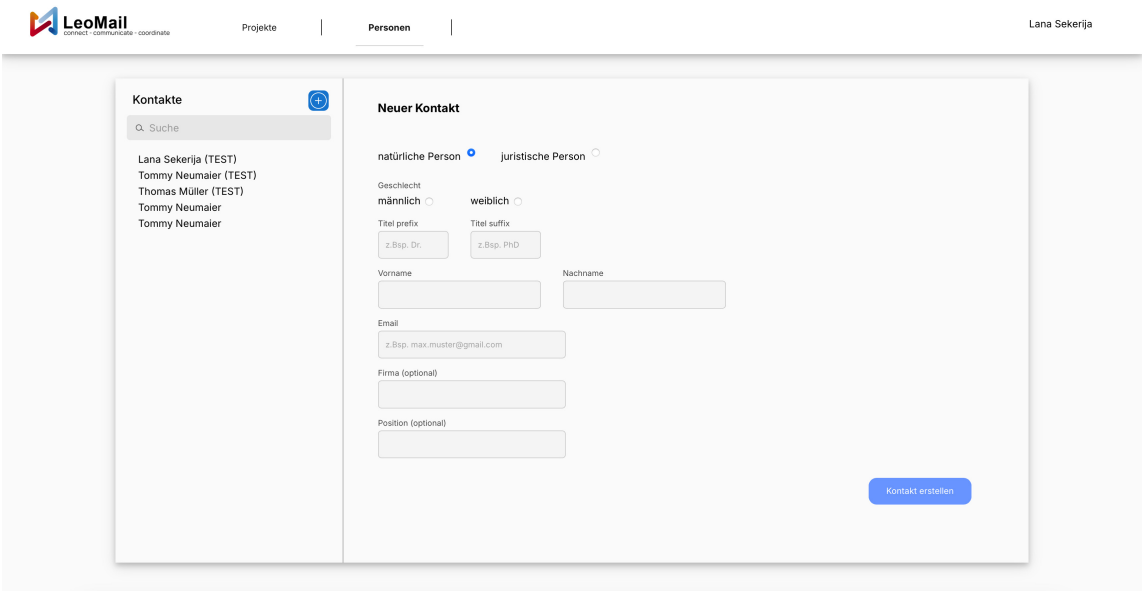

Abbildung 25: Darstellung der „Personen“-Seite

Dieses Layout wird auch auf der Personen-Seite umgesetzt, wobei in der linken Sidebar die Personen gelistet sind und sich auf der rechten Seite ein Formular befindet. Der gesamte Inhalt wird als eine große Box dargestellt, die durch gezielte Schatteneffekte hervorgehoben wird.

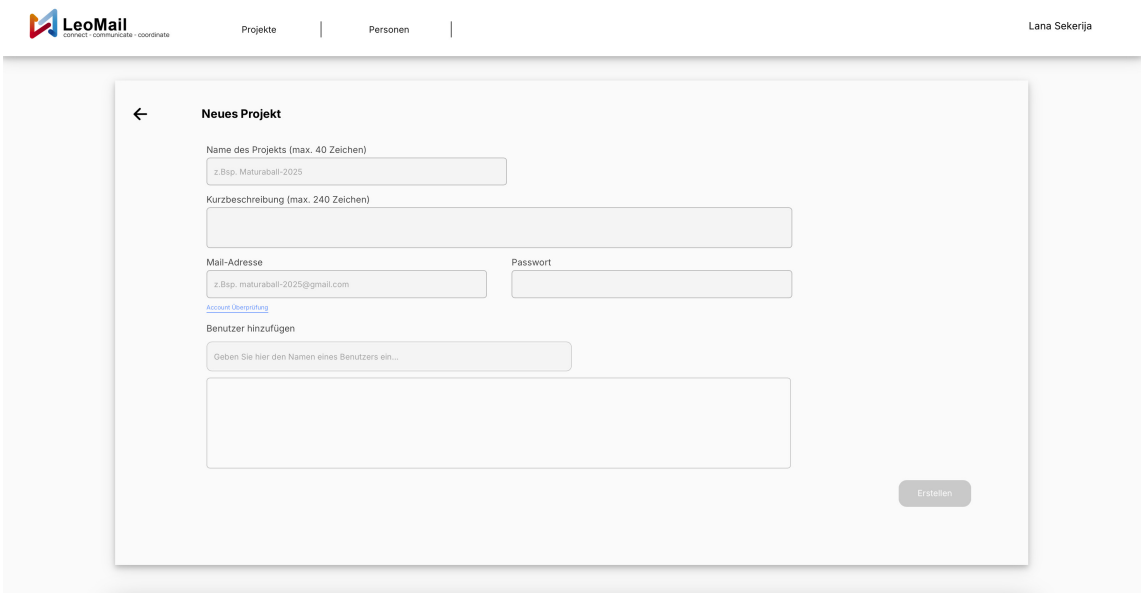

Abbildung 26: Darstellung der „Projekt erstellen“-Seite

Dieses Box-Layout wird ebenfalls auf der Seite „Projekt erstellen“ verwendet.

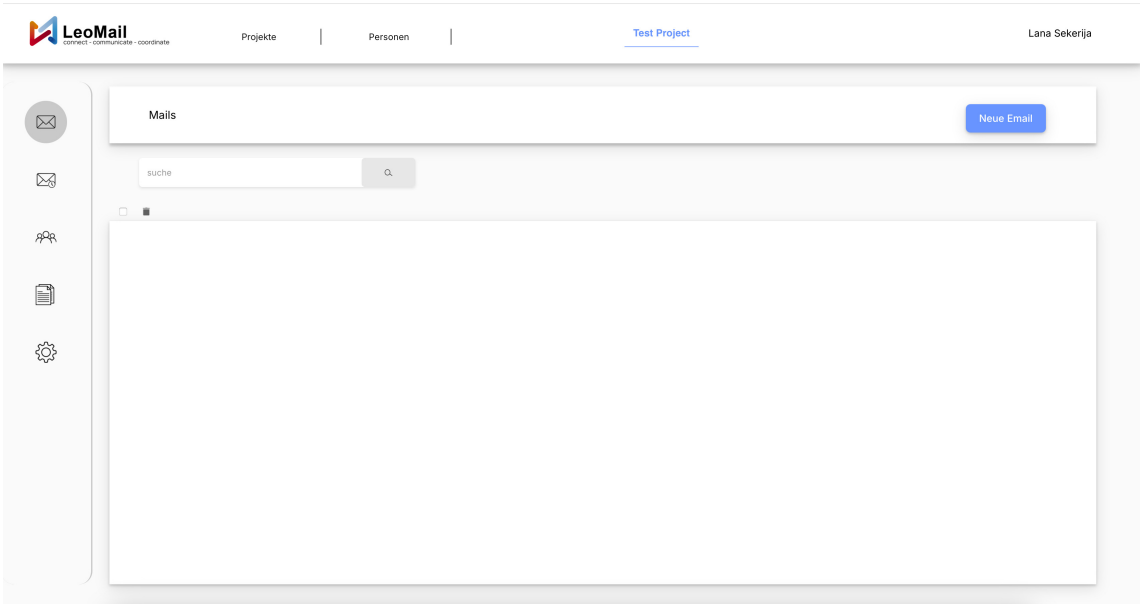

Abbildung 27: Darstellung der „Mail“-Seite

Die Mails-Seite folgt derselben Struktur wie die Geplanten-Mails-Seite. Die obere Navigationsleiste bleibt konsistent, während die linke Sidebar als eine Art sekundäre Navigation dient, die die aktuelle Seite hervorhebt. Diese Sidebar zieht sich auch durch die untergeordneten Seiten. Rechts davon befindet sich der Hauptinhalt mit der Überschrift der jeweiligen Unterseite und der auffälligen Schaltfläche „Neue Mail“, die Nutzer direkt zum Handeln einlädt.

Im Hauptinhalt liegt der Fokus auf der Verwaltung von E-Mails. Eine Suchleiste ermöglicht eine schnelle Filterung und erleichtert so die Bedienung. Trotz zahlreicher Funktionen bleibt die Seite übersichtlich und benutzerfreundlich.

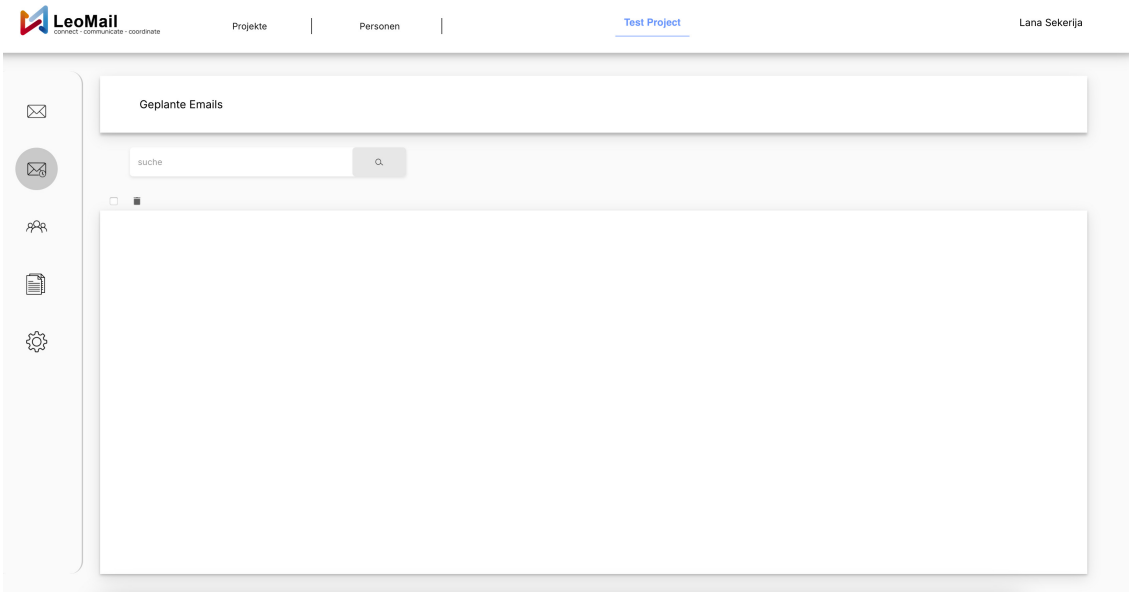

Abbildung 28: Darstellung der „Geplante Mails“-Seite

Die Geplanten-Mails-Seite ist ähnlich strukturiert, verzichtet jedoch auf die Schaltfläche „Neue Mail“. Beide Seiten verfügen unterhalb der Suchleiste über zwei zentrale Funktionen: eine Auswahl-Checkbox und eine Löschfunktion. Darunter wird die Liste der Mails angezeigt, wobei jede Mail in einer horizontalen Box dargestellt ist. Diese Mail-Boxen sind in einer übergeordneten, scrollbaren Box aufgelistet, wodurch die Inhalte klar gegliedert bleiben.

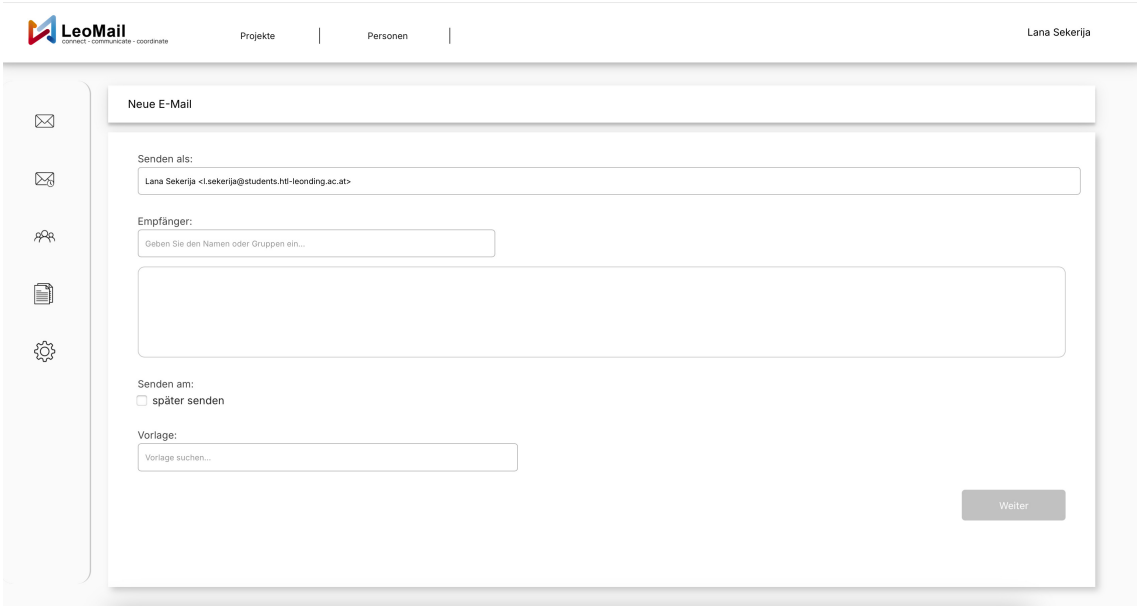

Abbildung 29: Darstellung der „Neue Mail“-Seite

Bei der Erstellung einer neuen Mail wurde die bestehende Struktur der Mail-Seite nur minimal angepasst. Die Überschrift bleibt an ihrer gewohnten Position, während die darunterliegende Inhaltsbox größer gestaltet wurde, um ausreichend Platz für das Formular zu bieten.

Die großzügige Gestaltung der Box sorgt dafür, dass alle Felder übersichtlich angeordnet und leicht zugänglich sind. Dadurch wird der gesamte Prozess nicht nur visuell ansprechend, sondern auch effizient und benutzerfreundlich.

Dieses Layout fügt sich harmonisch in das Gesamtdesign der Anwendung ein, was für eine konsistente Nutzererfahrung sorgt.

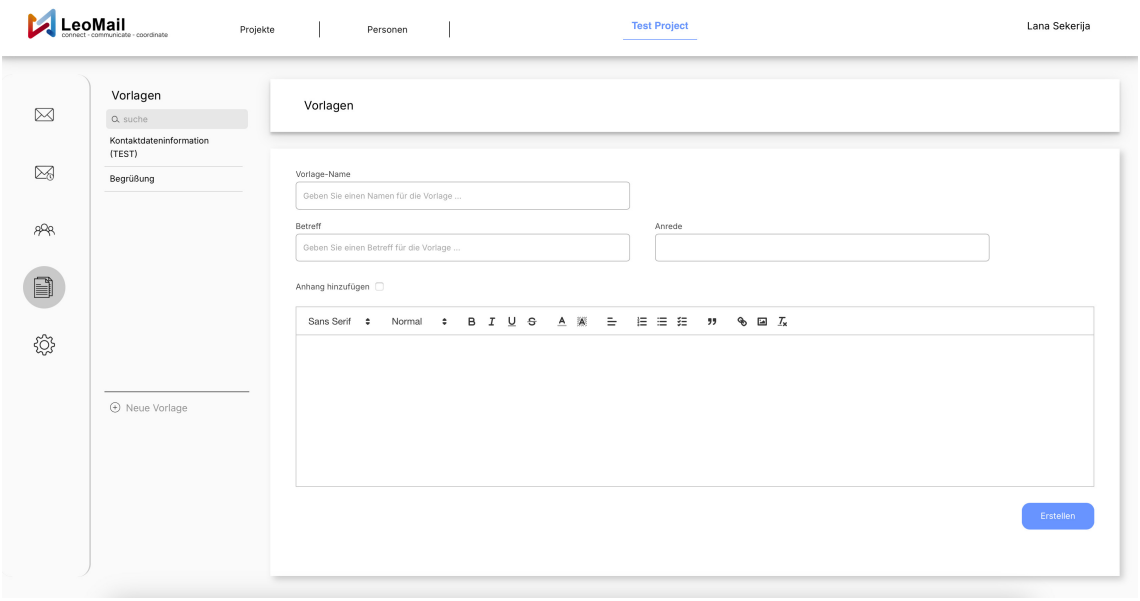

Abbildung 30: Darstellung der „Vorlagen“-Seite

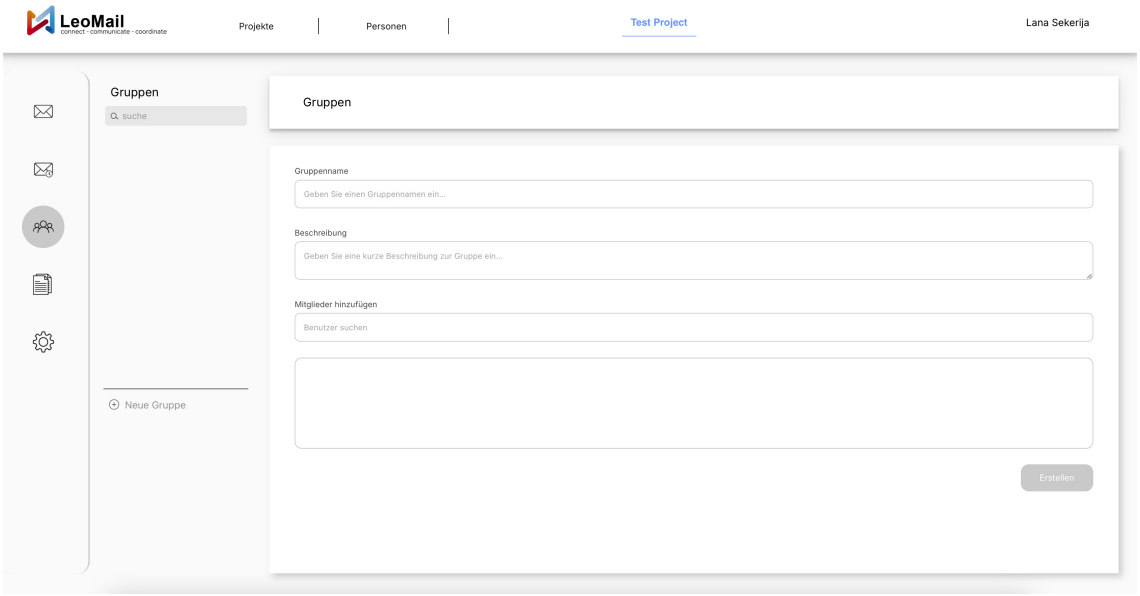

Abbildung 31: Darstellung der „Gruppen“-Seite

Die Vorlagen- und Gruppen-Seite sind ebenfalls im gleichen Layout gestaltet. Im Header befindet sich die konstante Navigationsleiste. Der Inhalt folgt derselben Struktur: Links eine Navigationsleiste und rechts eine Box, die in zwei separate Bereiche unterteilt ist. Im linken Bereich werden entweder die Vorlagen oder die Gruppen aufgelistet, ergänzt durch eine Suchleiste, die die Nutzung erheblich beschleunigt. Der rechte Bereich enthält ein Formular zum Anlegen, Aktualisieren und Löschen von Daten. Dieses Layout sorgt für eine übersichtliche Darstellung der Funktionen und trennt diese visuell klar voneinander.

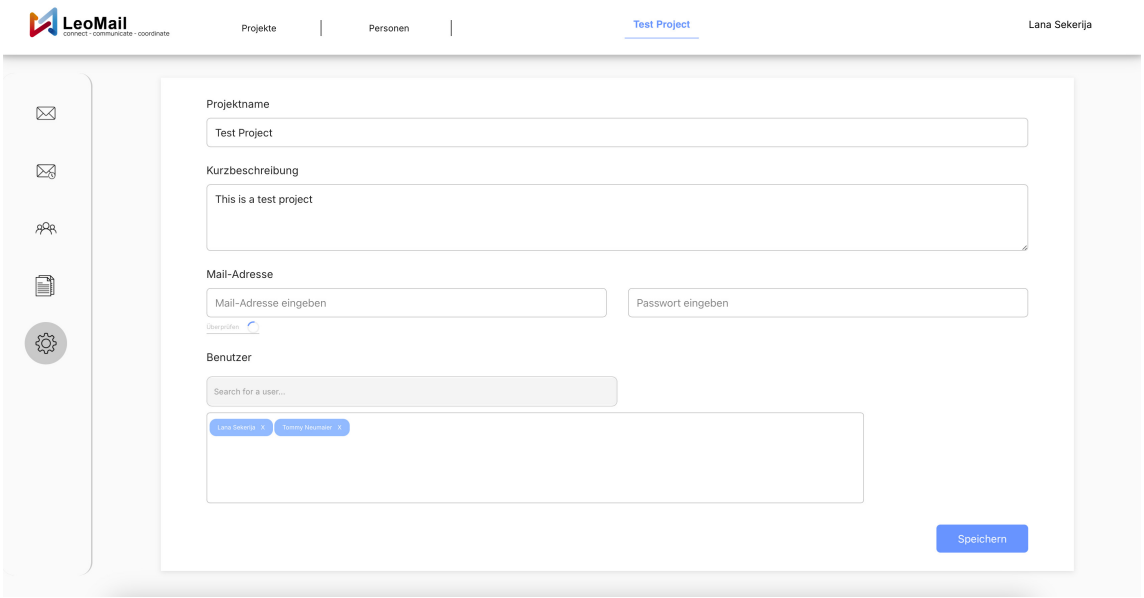

Abbildung 32: Darstellung der „Projekteinstellungen“-Seite

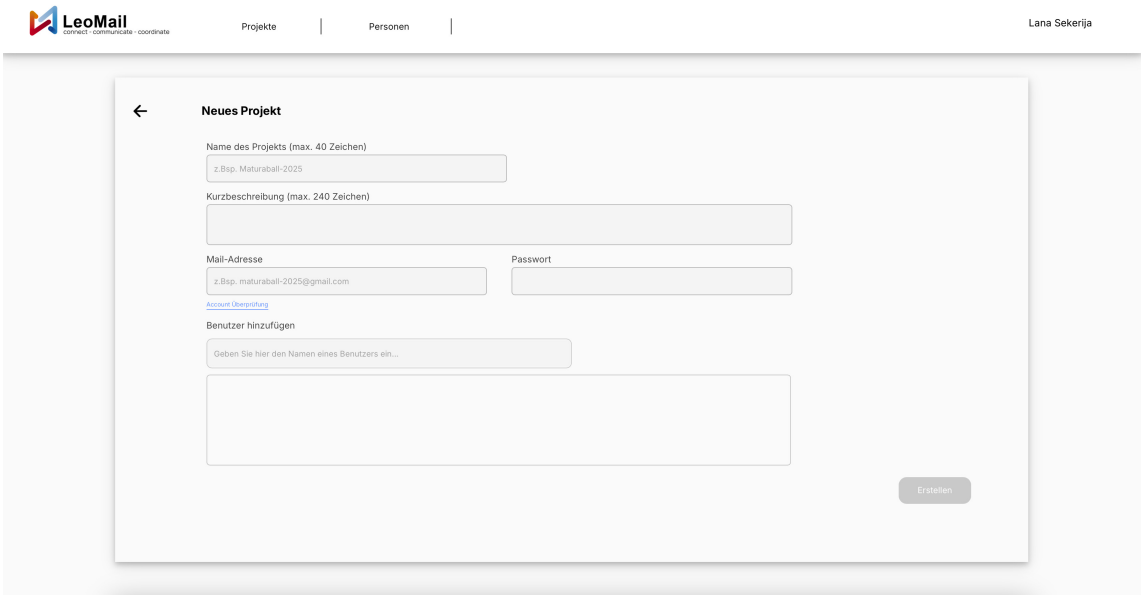

Abbildung 33: Darstellung der „Projekt erstellen“-Seite

Beim Erstellen von Projekten und in den Projekteinstellungen wird ein ähnliches Layout verwendet: eine schlichte Box mit einem Formular. In den Projekteinstellungen ist zusätzlich die linke Navigationsleiste sichtbar, während bei der Projekterstellung die Box zentriert dargestellt wird, um den Fokus direkt auf das Formular zu lenken.

6.3 Schriftwahl

Bei der Entscheidung für die Schriftwahl wurde großer Wert auf eine klare und gut lesbare Schrift gelegt. Nach ausgiebigem Testen verschiedener, von HTML bereitgestellter

Schriftarten fiel die endgültige Wahl auf die Google-Font-Schriftart „Inter“. Diese Schriftart erfüllt wichtige Typografie-Grundregeln im Webdesign und überzeugte daher als optimale Lösung.

Die folgenden Typografie-Regeln spielten bei der Schriftwahl eine entscheidende Rolle [15]:

Wahl der Schriftart

Da LeoMail primär als funktionale Software dient, stand von Anfang an fest, dass die gewählte Schriftart in erster Linie lesbar und funktional sein muss. Die Schrift sollte nicht durch eine auffällige Gestaltung ablenken, sondern den Fokus der Nutzer auf die Funktionen der Software lenken. Daher wurde die Schriftart sorgfältig ausgewählt, um sicherzustellen, dass sie sowohl zum Charakter der Software als auch zum Inhalt passt.

Anzahl der Schriftarten

Bei LeoMail wurde bewusst auf die Verwendung einer einzigen Schriftart gesetzt. Dies unterstützt eine schnelle und klare Vermittlung des Inhalts. Die Nutzung mehrerer Schriftarten birgt das Risiko von Unübersichtlichkeit und einer eingeschränkten Lesbarkeit. Daher gilt die Regel: Je weniger verschiedene Schriftarten eingesetzt werden, desto besser ist die Nutzerfreundlichkeit.

Schriftkombination

Für LeoMail war die Kombination unterschiedlicher Schriftarten keine Option, da der Fokus auf Klarheit und Funktionalität liegt. In anderen Fällen kann die Kombination jedoch sinnvoll sein, da sie durch den Kontrast typografische Spannung erzeugen kann. Beispielsweise ergänzen sich Serifenschriften (Antiqua) und serifenlose Schriften (Grotesk) oft gut, da ihre unterschiedlichen Merkmale visuelles Interesse wecken können.

Schriftgrößen

Die Schriftgrößen bei LeoMail wurden sorgfältig auf die Design-Hierarchie abgestimmt. Um ein ausgewogenes Verhältnis zwischen Überschriften und Mengentext herzustel-

len, wird empfohlen, die Schriftgröße der Überschrift etwa doppelt so groß wie die des Fließtextes zu wählen. So wird die Hierarchie betont und der Text bleibt klar strukturiert.

Anzahl der Schriftgrößen

Einheitlichkeit bei den Schriftgrößen war ein zentraler Aspekt des Designs von LeoMail. Für gleiche Text-Hierarchien wurden konsequent identische Schriftgrößen verwendet. Dies sorgt nicht nur für bessere Lesbarkeit, sondern auch für eine aufgeräumte und übersichtliche Gestaltung. Die Reduktion auf wenige Schriftgrößen trägt entscheidend zu einem klaren und benutzerfreundlichen Interface bei.

Schriftauszeichnung

Eine weitere wichtige Typografie-Regel, die bei LeoMail häufig angewendet wurde, ist die Hervorhebung durch die Schriftstärke. Überschriften wurden überwiegend in fett hervorgehobener Schrift gestaltet. Dies unterstreicht den Kontrast und definiert die Hierarchie der Textelemente klar.

Da LeoMail eine Software ist, die nur aus wenig oder gar keinem Fließtext besteht, entfallen typografische Regeln wie Schriftausrichtung, Zeilenabstand oder Zeilenlänge. Der Fokus liegt vielmehr auf der strukturierten und funktionalen Darstellung der Inhalte, die den Nutzer optimal unterstützen soll.

Eigenschaften der Schriftart „Inter“

„Inter“ ist eine Schriftart, die für eine beeindruckende Bandbreite an Anwendungen entwickelt wurde. Von der Nutzung in detaillierten Benutzeroberflächen über Marketingmaterialien bis hin zu Beschilderungen zeigt sie sich anpassungsfähig und funktional. Ihre Vielseitigkeit macht sie zu einer idealen Wahl für Projekte, bei denen sowohl Funktionalität als auch Ästhetik im Vordergrund stehen.

Diese Schriftart umfasst rund 2000 Glyphen, die 147 Sprachen abdecken. Dies ermöglicht eine internationale Einsetzbarkeit. Außerdem bleibt „Inter“ mit einer breiten Palette an Zeichen in vielen Sprachräumen gut lesbar und ästhetisch ansprechend.

Von einem Thin 100 bis hin zu einem Heavy 900 bietet die Schriftart „Inter“ eine breite Auswahl an Gewichten. Dies ermöglicht, die Schriftart an unterschiedliche Desi-

gnanforderungen anzupassen. Da jede Glyphe sorgfältig für bestimmte Gewichtsklassen entworfen wurde, bleiben Qualität und Lesbarkeit unabhängig vom Gewicht erhalten. Außerdem bietet „Inter“ die Kursivfunktion, die für zusätzliche Flexibilität sorgt. Diese Eigenschaften machen die Schrift professionell und zuverlässig [92].

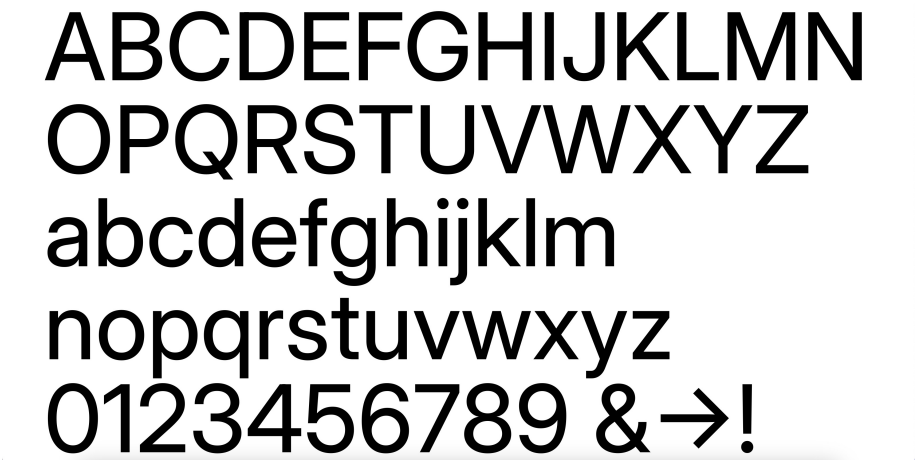

Abbildung 34: Inter Kalligraphie [92]

Warum „Inter“ ideal zu LeoMail passt

Die klare, funktionale und moderne Gestaltung von „Inter“ harmoniert hervorragend mit LeoMail und bildet die perfekte Grundlage für ein durchdachtes Design. Sie sorgt dafür, dass die Benutzer nicht von den wesentlichen Funktionen abgelenkt werden, sondern diese optimal in den Vordergrund gerückt werden. Gleichzeitig verleiht sie der gesamten Benutzeroberfläche ein professionelles und ansprechendes Erscheinungsbild, das sowohl Ästhetik als auch Funktionalität ideal miteinander verbindet.

6.4 Farbwahl

Bei der Farbwahl wurde LeoMail bewusst schlicht und minimalistisch gehalten. Die Benutzeroberfläche zeichnet sich durch überwiegend weiße Flächen aus, die entweder durch dezente Schattierungen oder leicht graue Hintergründe hervorgehoben werden. Dieses reduzierte Farbschema unterstützt die klare Funktionalität der Software und vermeidet Ablenkungen für den Nutzer. Wie bei allen anderen Designentscheidungen wurde auch bei der Farbgestaltung großer Wert darauf gelegt, die Benutzererfahrung so intuitiv wie möglich zu gestalten und die Funktionalität in den Mittelpunkt zu rücken.

Um ein harmonisches und ausgewogenes Erscheinungsbild zu schaffen, wurde mit einer Palette aus verschiedenen Grautönen unterschiedlicher Helligkeit gearbeitet. Diese sorgfältig abgestimmten Farben erzeugen ein einheitliches und ansprechendes Designbild, das die visuelle Struktur unterstützt und die Übersichtlichkeit fördert. Jede Seite von LeoMail wurde mit einem hellgrauen Grundton (`#FBFBFB`) gestaltet, der leicht als Hintergrundschicht wahrgenommen wird und die im Vordergrund stehenden weißen Elemente wirkungsvoll zur Geltung bringt.

Für die Akzentfarbe, die hauptsächlich bei Schaltflächen und interaktiven Elementen verwendet wird, wurde eine Farbwahl getroffen, die die Verbindung zu den Fachrichtungen der IT-Medientechnik und Informatik unterstreicht. Nach mehreren Tests und Anpassungen entstand eine ausgewogene blaue Farbgebung (`#78A6FF`), die wichtige Funktionen optisch hervorhebt und dabei angenehm dezent bleibt, um die Augen nicht zu überlasten. Diese Akzentfarbe verleiht LeoMail zugleich einen modernen und professionellen Charakter.

Weiters wurden verschiedene Aspekte zur richtigen Farbwahl beachtet [78]:

Begrenzte Farbpaletten

Bei der Farbwahl gilt, genau wie bei der Auswahl von Schriftarten: „Weniger ist mehr“. Für den Anfang reichen 2–3 Hauptfarben, ergänzt durch Basisfarben wie Weiß, Schwarz oder Grau. Es empfiehlt sich, auf eine reduzierte Farbpalette zurückzugreifen und erst bei Bedarf weitere Farben hinzuzufügen.

Gute Lesbarkeit

Um dem Nutzer den Text leichter lesbar und übersichtlicher zu machen, sollte man sowohl auf eine gute Farbkombination als auch auf einen ausreichenden Farbkontrast achten. Manche Farben sind auf bestimmten Hintergründen besser oder schlechter lesbar.

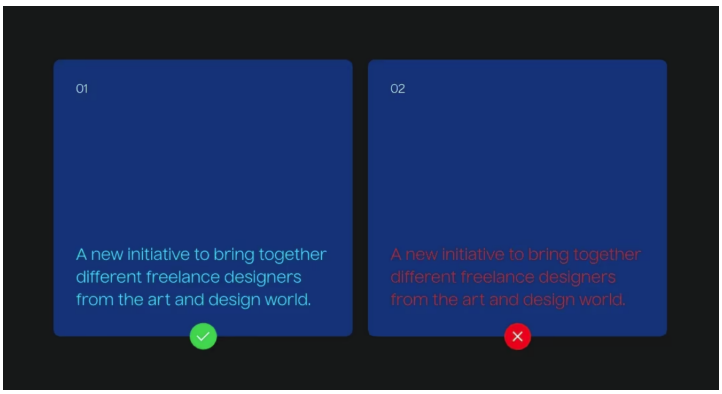

Abbildung 35: Beispiel zur UI-Lesbarkeit [78]

In der Grafik wird deutlich, dass trotz gleicher Farbhelligkeit der rote Text schwieriger zu lesen ist.

Ausreichender Kontrast

Ein bewusster Einsatz von Farbkontrasten unterstützt ebenfalls die Lesbarkeit. Farbkontrast beschreibt den Unterschied zwischen der Helligkeit und Dunkelheit einer Farbe. Selbst bei identischem Farbton kann die Lesbarkeit stark variieren, wenn die Helligkeit unterschiedlich abgestuft ist.

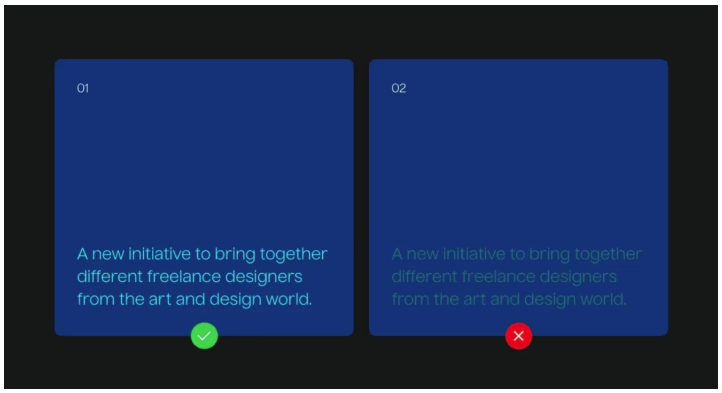

Abbildung 36: Beispiel zum UI-Farbkontrast [78]

Der gleiche türkise Farbton wird je nach Helligkeitsabstufung unterschiedlich schwer gelesen.

Konsequente Nutzung der Farbpalette

Um ein stimmiges und harmonisches Design zu gewährleisten, sollte die Farbpalette konsequent verwendet und als Guideline eingesetzt werden. So können wiederkehrende

Elemente stets dieselbe Farbe erhalten, was für ein einheitliches Erscheinungsbild sorgt.

Nutze die Farbpsychologie

Es kann äußerst hilfreich sein, sich mit der Farbpsychologie auseinanderzusetzen. Diese zeigt, welche Emotionen und Stimmungen durch verschiedene Farben ausgelöst werden. Abhängig davon, welche Botschaft oder Werte man mit seiner Software oder Website transportieren möchte, spielt dieser Aspekt eine entscheidende Rolle bei der Farbwahl. Die Farbe Blau, die oft mit Vertrauen und Zuverlässigkeit assoziiert wird, verkörpert eine der zentralen Eigenschaften von LeoMail.

6.5 Erster Design-Entwurf

Beim ersten Grunddesign der Softwarestruktur wurden die Vorstellungen und Ideen des Designs umgesetzt. Trotz einiger Änderungen blieb vieles vom ursprünglichen Grunddesign erhalten.

Eine Projektübersicht war im ersten Entwurf nicht enthalten, da die Idee erst während eines Meetings zur Designbesprechung vom Abteilungsvorstand der IT-Medientechnik und Informatik der HTL Leonding entwickelt wurde. Die Aufteilung einzelner Projekte und deren separate Verwaltung stellte zunächst eine designtechnische Herausforderung dar, da sie übersichtlich und nutzerfreundlich gestaltet werden sollte. Diese Aspekte wurden jedoch erst bei der Umsetzung des finalen Designs berücksichtigt.

Der erste Entwurf umfasste Seiten für Mail, Kalender, Gruppen, Vorlagen und Profileinstellungen.

Die Designentwürfe von LeoMail wurden mithilfe von Figma grafisch dargestellt.

Figma Design

Figma ist eine webbasierte Designplattform, mit der Designer, Projektmanager, Produktmanager und Entwickler gemeinsam an Ideen arbeiten, Feedback in Echtzeit erhalten und hochwertige Prototypen erstellen können – ganz ohne Programmcode. Dabei profitieren Teams von leistungsstarken Designtools, einer intuitiven Benutzeroberfläche und einer nahtlosen Versionskontrolle, sodass niemand den Überblick über Änderungen verliert.

Abbildung 37: Logo von Figma [22]

Die Plattform ermöglicht es, Standards für Stile, Komponenten und Variablen festzulegen, um sicherzustellen, dass alle Produkte und Marken auch bei größerem Umfang konsistent bleiben. Mit Funktionen wie Auto-Layout, Verzweigungen (Branching) oder intelligenter Ebenenanordnung lassen sich Designs schneller anpassen und Varianten unkompliziert austesten. So sparen Designer wertvolle Zeit und können sich intensiver ihren kreativen Aufgaben widmen.

Durch die direkte Einbindung externer Stakeholder wird es einfacher, Feedback einzuholen und Abstimmungen zu treffen – alles in einem einzigen Tool. Dank Community- und Partnerressourcen wie Plug-ins, Widgets und Vorlagen ist es außerdem möglich, Figma noch weiter an die eigenen Bedürfnisse anzupassen und Arbeitsabläufe zu automatisieren. Auch Integrationen mit anderen Tools (z.B. Asana, Microsoft Teams) sind verfügbar, damit Organisation und Kommunikation reibungslos ablaufen.

Zusätzlich zu den Designfunktionen bietet Figma FigJam, ein Online-Whiteboard, das Teams beim Brainstorming und bei der Entscheidungsfindung unterstützt. Wer seine Entwürfe anschließend in Programmcode übersetzen möchte, kann den Dev Mode nutzen, der die Zusammenarbeit mit Entwickler beschleunigt.

Diese Funktionen bieten für LeoMail umfassende Möglichkeiten, um ein reibungsloses, konsistentes und zugleich zügiges Design- und Entwicklungsumfeld zu schaffen [21].

Mail-Seite

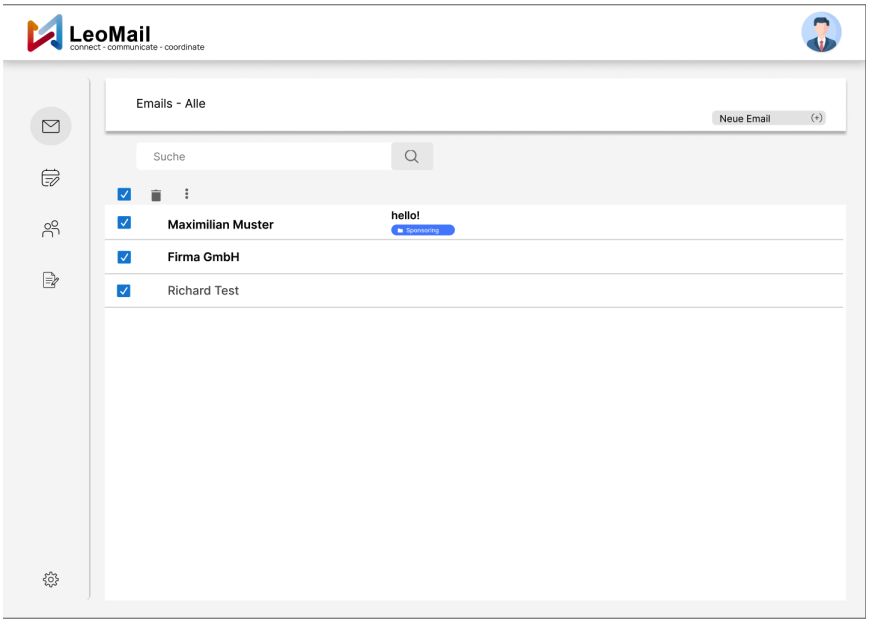

Abbildung 38: Erster Entwurf des Mail-Layouts

Die Mail-Seite enthält im Header-Bereich eine weiße Box mit dem Logo im linken Eck und dem Profilbild des Nutzers im rechten Eck.

Im Content-Bereich wurde auf der linken Sidebar die Navigationsleiste gestaltet, die auf die verschiedenen Unterseiten verweist. Das Icon der aktuell ausgewählten Unterseite wird durch einen dunkelgrauen Hintergrund hervorgehoben. Ein Trennstrich auf der rechten Seite der Navigationsleiste grenzt diese vom eigentlichen Content ab. Der Hauptinhalt wird in einer optisch abgesetzten, jedoch nicht umrandeten Box dargestellt.

Ganz oben befindet sich eine Überschrift in einer weißen Box sowie eine Schaltfläche mit dem Label „Neue Mail“ im rechten Eck. Darunter ist eine Fläche für die Suche nach Mails platziert. Direkt unter der Suchleiste sind drei verschiedene Icons waagerecht angeordnet: eine Checkbox, ein Mülleimer und die drei senkrechten Punkte, die als Menüfunktion dienen.

Unterhalb dieser Icons wird eine weiße Box mit waagrecht aufgelisteten Mails dargestellt. Jede Mail enthält links eine Checkbox, daneben den Empfänger, den Betreff und darunter ein Label, das angibt, zu welcher Veranstaltungsgruppe die Mail gehört, wie etwa Sponsoring, Finanzen oder Geschäftsführung.

Neue Mail

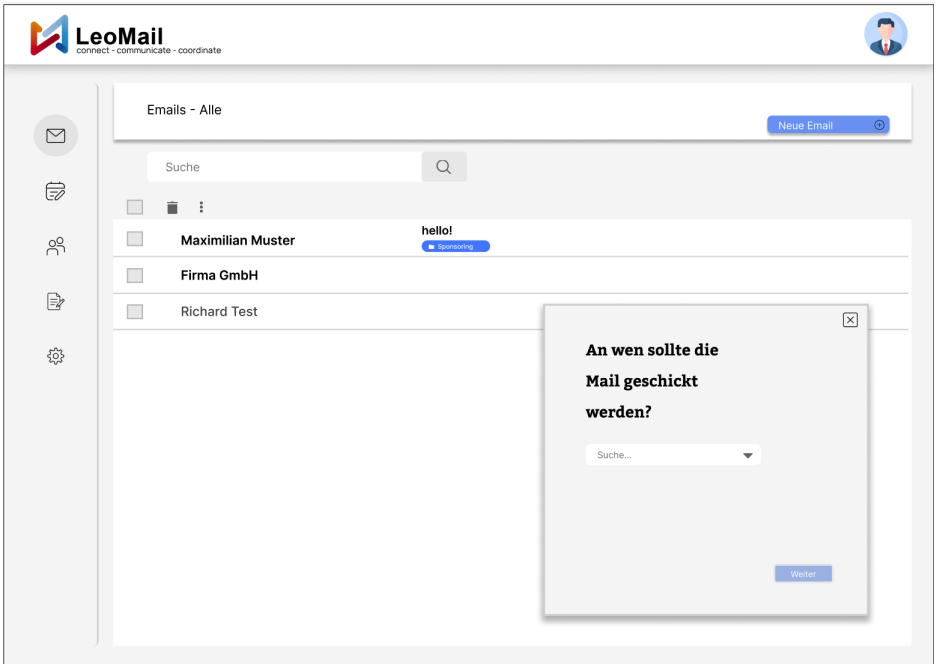

Abbildung 39: Erster Entwurf der „Neue Mail“-Ansicht

Beim Klick auf die Schaltfläche „Neue Mail“ öffnet sich ein Dialog mit mehreren Schritten. Zunächst wird nach dem Empfänger gefragt, anschließend nach dem gewünschten Versandzeitpunkt. Danach wird abgefragt, welche Vorlage verwendet werden soll und ob diese personalisiert werden soll oder ob eine neue Vorlage erstellt werden soll.

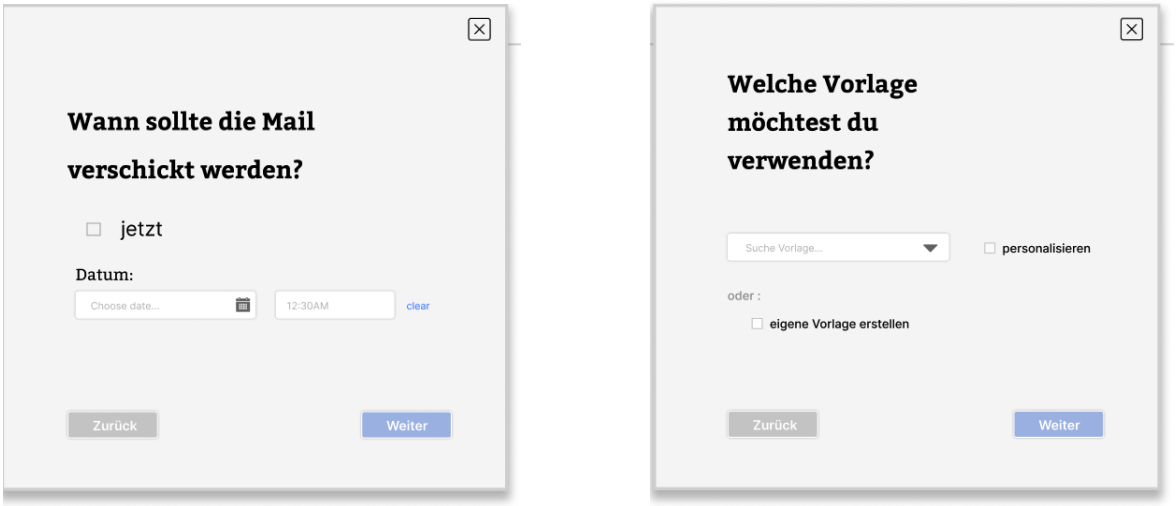

Abbildung 40: Erster Entwurf der Dialoge zur Festlegung des Versandzeitpunkts und zur Vorlagenwahl

Neue Mail – vordefinierte Vorlage

Falls die gewünschte Vorlage bereits existiert, muss sie nicht erneut definiert werden. Über ein Dropdown-Menü werden die Vorlagen aufgelistet und zur Auswahl bereitgestellt. Nach dem Klick auf die Schaltfläche „Weiter“ gelangt man zur nächsten Frage: dem Betreff der Mail. Im letzten Schritt werden alle ausgewählten Daten zusammengefasst, sodass diese überprüft werden können. Zudem wird eine Vorschau der Mail angezeigt, wie sie beim Empfänger aussehen wird. Um die Mail zu versenden, klickt man auf die Schaltfläche „Absenden“. Anschließend wird eine Bestätigung angezeigt, dass die Mail erfolgreich versendet wurde.

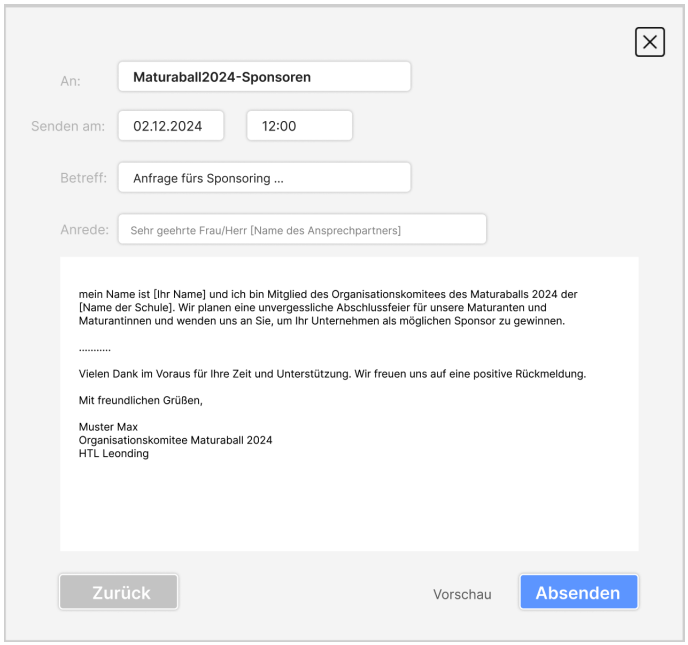

Abbildung 41: Erster Entwurf der Dialogansicht zur Zusammenfassung aller Daten

Neue Mail – neue Vorlage

Falls noch keine vordefinierte Vorlage existiert, kann eine neue Vorlage direkt im Dialog erstellt werden. Ähnlich wie bei vordefinierten Vorlagen wird zuerst nach dem Betreff gefragt. Danach öffnet sich ein Dialog, in dem ein Textfeld bearbeitet werden kann. Unterhalb dieses Feldes besteht die Möglichkeit, den Text als neue Vorlage zu speichern, um ihn bei späteren Mails erneut zu verwenden.

Wenn man sich entscheidet, die Vorlage zu speichern, öffnet sich nach dem Klick auf „Absenden“ ein Zwischendialog, in dem der Vorlage ein Name zugewiesen werden kann. Nach dem Klick auf „Speichern“ wird eine Bestätigung angezeigt, dass die E-Mail erfolgreich verschickt wurde.

Kalender

Der erste Entwurf beinhaltet außerdem einen Kalender, der als zentrale Komponente des Systems dient. Eine eigene Unterseite mit einer übersichtlichen Kalenderansicht wurde geplant, die es den Nutzern ermöglicht, einzelne Mails direkt anzuklicken, um die entsprechenden Mail-Details einzusehen.

Durch die Integration des Kalenders soll es ebenfalls möglich sein, geplante Mails zu verwalten. Nutzer können geplante Mails im Voraus ansehen und diese vor dem Absenden bei Bedarf bearbeiten oder löschen. Der Kalender bietet eine Monatsansicht an, die eine klare Übersicht über alle Mails eines Monats gewährleistet. Zur besseren Unterscheidung werden versendete und geplante Mails durch unterschiedliche Farben gekennzeichnet, was die Übersichtlichkeit erhöht und den Arbeitsfluss erleichtert.

Bei einem Klick auf eine Mail öffnet sich ein Dialogfenster, das die detaillierten Informationen zu der entsprechenden Mail anzeigt. Falls eine Mail noch nicht verschickt wurde, können die Nutzer in diesem Dialog Änderungen vornehmen oder die Mail komplett löschen. Auf diese Weise wird eine hohe Flexibilität und Kontrolle über die Mail-Kommunikation sichergestellt.

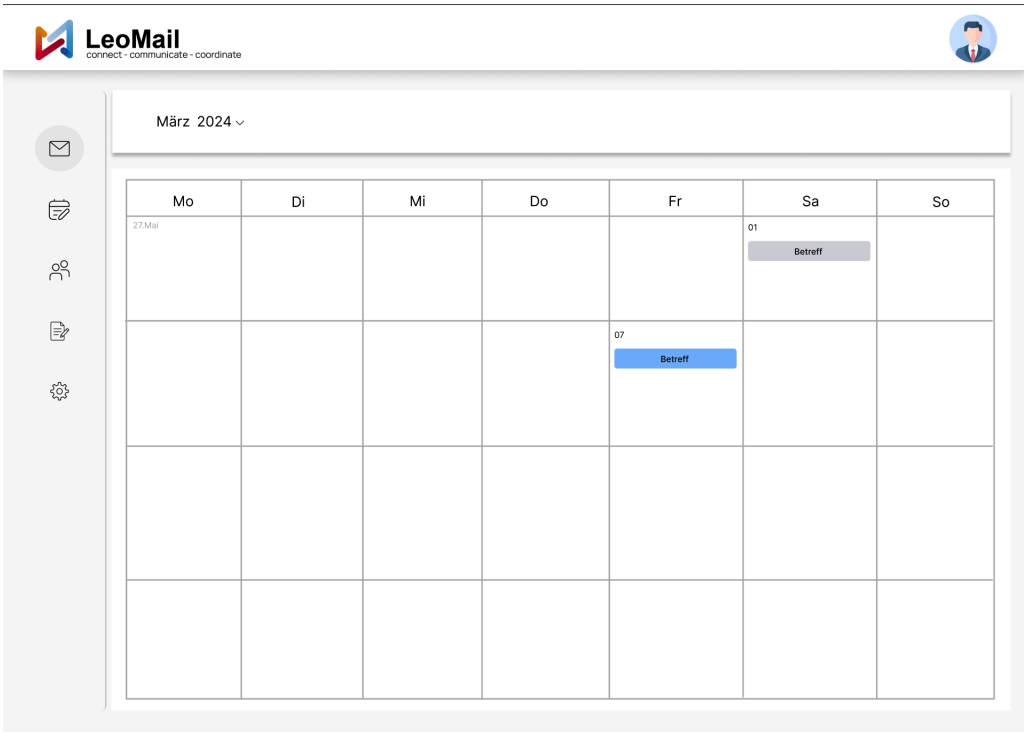

Abbildung 42: Erster Entwurf der Kalenderansicht

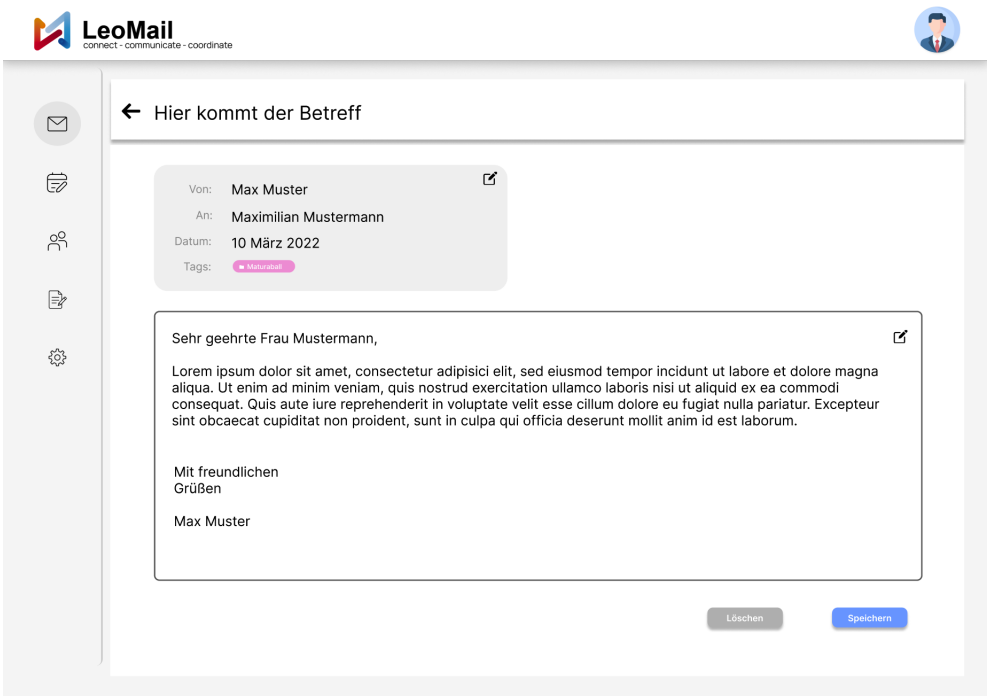

Abbildung 43: Erster Entwurf der Kalenderansicht für die Mailverwaltung

Gruppen und Vorlagen

Die Gruppen- und Vorlagenverwaltung aus dem ersten Entwurf wurde in der finalen Version vollständig übernommen und überzeugt durch eine übersichtliche Struktur. Beide Unterseiten verwenden die gleiche Grundvorlage, wobei lediglich die Formularinhalte angepasst wurden. Das Design ist einheitlich gestaltet und unterteilt sich in zwei Hauptbereiche:

- **Linke Seite:** Eine kompakte Box listet die bereits erstellten Gruppen oder Vorlagen auf und bietet eine Suchfunktion. Zusätzlich befindet sich dort eine Schaltfläche zum Erstellen neuer Gruppen oder Vorlagen.
- **Rechte Seite:** Hier befindet sich das Formular zum Anlegen oder Bearbeiten von Gruppen bzw. Vorlagen. Sobald eine bestehende Gruppe oder Vorlage ausgewählt wird, werden die Felder im Formular automatisch befüllt und können direkt bearbeitet sowie gespeichert werden. Außerdem ist an dieser Stelle auch das Löschen der ausgewählten Gruppe oder Vorlage möglich.

Für die Gruppenzuordnung gibt es eine Suchfunktion, um Mitglieder schnell zu finden und hinzuzufügen. Bei der Erstellung einer Vorlage können in einem Dropdown-Menü personalisierte Anreden ausgewählt werden. Der Vorlagen-Text selbst wird in einem Rich-Text-Editor erstellt, der mit zahlreichen Bearbeitungswerkzeugen ausgestattet

ist. Trotz der Vielzahl an Funktionen und Daten bleibt die Seitenstruktur klar und benutzerfreundlich.

Profil

Die Profilseite wurde so gestaltet, dass die persönlichen Daten sowie ein Profilbild der aktuell angemeldeten Person gut sichtbar sind. Hier kann man einen schnellen Überblick über sämtliche eigene Informationen gewinnen. Das Profil dient dabei vor allem der persönlichen Übersicht und informiert über die eigene Identität innerhalb des Systems. Auf diese Weise lassen sich die eigenen Daten leicht überprüfen und, falls nötig, anpassen, um stets aktuell zu bleiben.

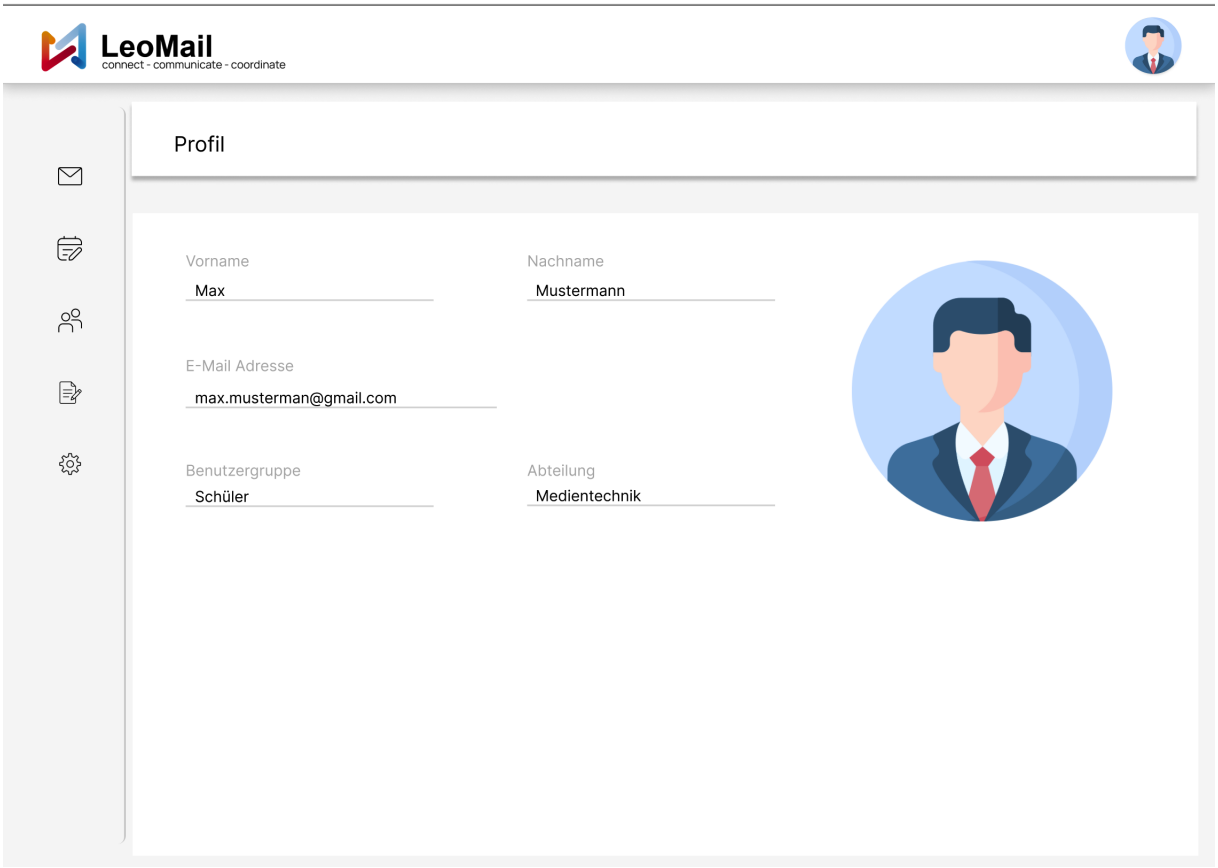

Abbildung 44: Erster Entwurf der Profilansicht

6.6 Finaler Design-Entwurf

Nach zahlreichen Meetings und ausführlichen Designbesprechungen wurden verschiedene Anpassungen vorgenommen, wobei das Grundlayout vieler Unterseiten weitgehend erhalten blieb. Einige Funktionen wurden entfernt und neue hinzugefügt. Zu den

wichtigsten Neuerungen gehören die Login-Funktion sowie eine E-Mail-Authentifizierung, die beim erstmaligen Anmelden zum Einsatz kommt.

Login

Die Login-Seite wurde bewusst schlicht gestaltet, um eine klare und übersichtliche Benutzerführung zu gewährleisten. Mittig befindet sich eine Box mit den erforderlichen Eingabefeldern für Benutzername und Passwort. Ein markanter, blauer Button dient zum Absenden der eingegebenen Daten und zum Einleiten des Login-Prozesses.

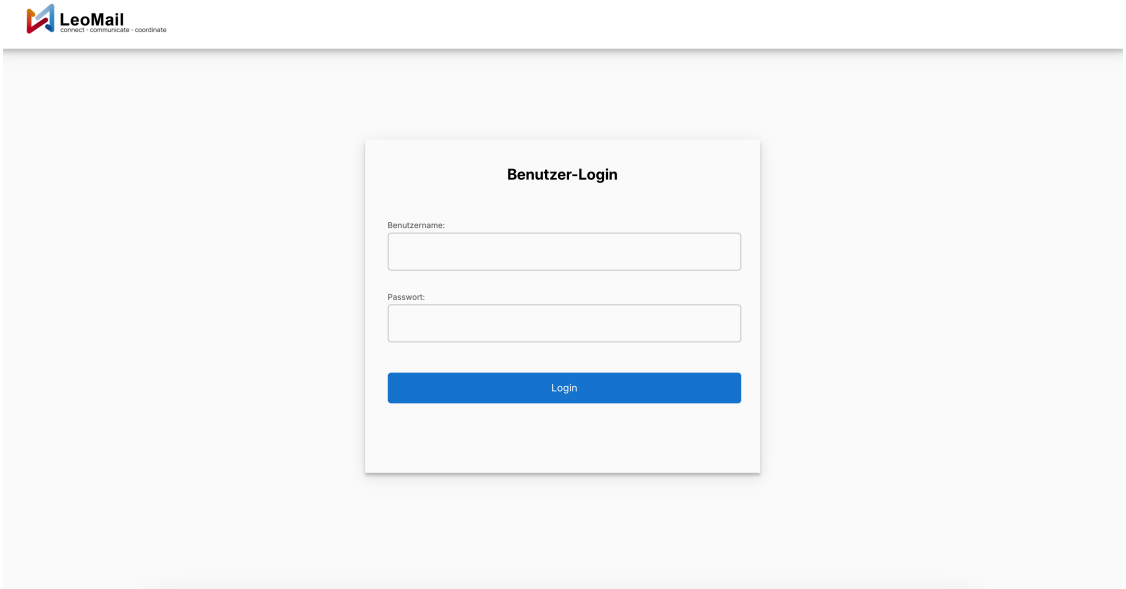

Abbildung 45: Finales Design der Login-Seite

E-Mail-Authentifizierung

Um sowohl die Sicherheit zu erhöhen als auch den späteren Versand von E-Mails zu ermöglichen, wird eine E-Mail-Authentifizierung angeboten. Hierbei gibt der Nutzer seine E-Mail-Adresse und sein Passwort ein, woraufhin die Daten überprüft werden. Das Design dieser Seite ist an die Login-Seite angelehnt und erscheint ausschließlich beim erstmaligen Anmelden, um den Zugang ordnungsgemäß abzusichern.

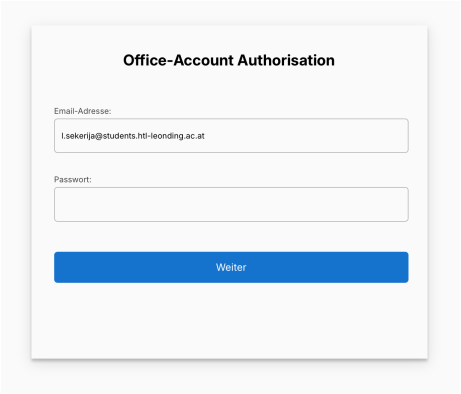

Abbildung 46: Finales Design für die Office-Account-Authentifizierung

Personen-Seite

Mit der neu hinzugefügten Personen-Unterseite wird eine übersichtliche Verwaltung aller beteiligten Personen ermöglicht. In der linken Spalte befindet sich eine Liste bereits angelegter Personen, die mithilfe einer Suchfunktion schnell durchforstet werden kann. Auf der rechten Seite finden sich die Eingabefelder, um neue Personen – zum Beispiel externe Mitarbeiter verschiedener Unternehmen – anzulegen oder bestehende Daten zu bearbeiten. Hierbei wird zudem zwischen juristischen und natürlichen Personen unterschieden, was die Verwaltung übersichtlicher gestaltet. Sobald Personen hinzugefügt wurden, können sie bei Bedarf als Empfänger für den späteren Mailversand ausgewählt werden.

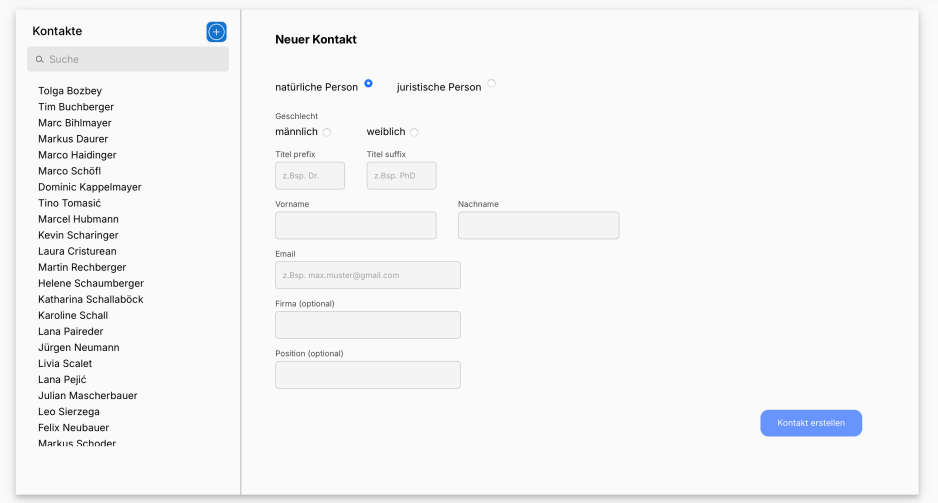

Abbildung 47: Finales Design der Personenübersicht

Projekt-Ansicht

Im Rahmen des finalen Designs wurde außerdem eine separate Projekt-Ansicht eingeführt, die eine strukturierte Verwaltung einzelner Projekte ermöglicht. In der linken Spalte sind alle angelegten Projekte aufgelistet; eine integrierte Suchleiste hilft dabei, schnell das gewünschte Projekt zu finden. Auf der rechten Seite werden grundlegende Informationen sowie eine kurze Beschreibung der Software und ihrer Funktionen angezeigt. Über einen Plus-Button kann ein Formular zum Anlegen neuer Projekte aufgerufen werden, das alle erforderlichen Felder beinhaltet. Zudem wird eine Überprüfung der angegebenen Projekt-E-Mail-Adresse durchgeführt, um mögliche Fehler frühzeitig zu erkennen.

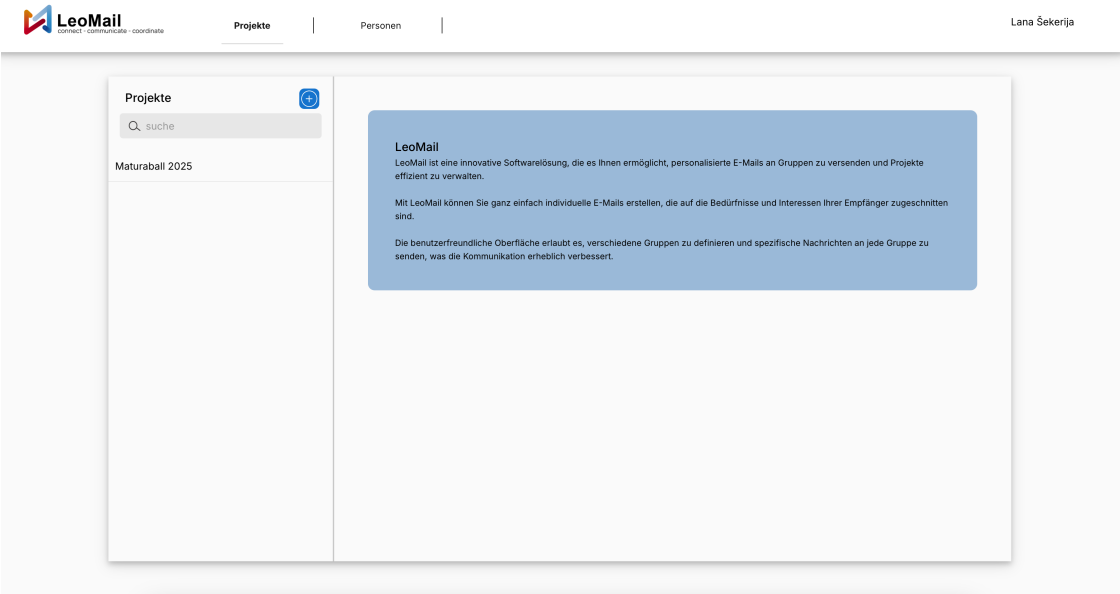

Abbildung 48: Finales Design der Projektansicht

Projekt-Verwaltung

Mail

Nach Auswahl eines Projekts erscheint in der rechten Spalte eine neue Navigationsleiste, die an das ursprüngliche Design des ersten Entwurfs angelehnt ist. Die Mail-Seite selbst wurde allerdings in einigen Punkten überarbeitet: Die Mails werden nun übersichtlich in der Reihenfolge Empfänger, Betreff und Datum dargestellt, um einen schnellen Überblick zu gewährleisten. Zusätzlich hebt sich die Schaltfläche „Neue Mail“ farblich vom Rest der Seite ab, was die Benutzerführung weiter verbessert und das Erstellen neuer Mails erleichtert.

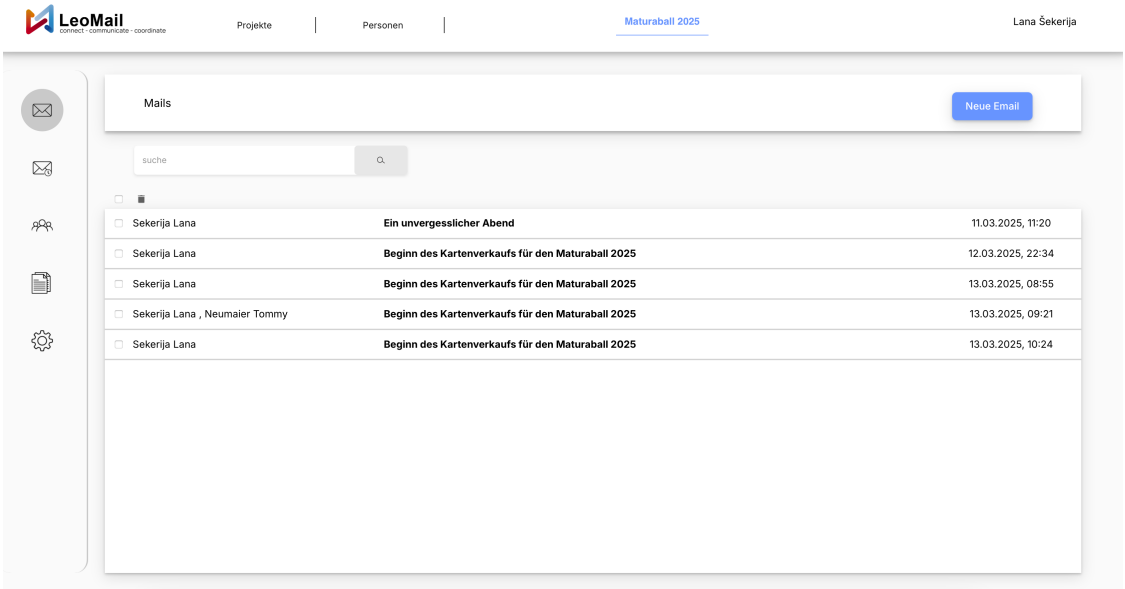

Abbildung 49: Finales Design der Mail-Übersicht

Neue-Mail

Im finalen Design wurde der Prozess zum Erstellen und Versenden einer E-Mail grundlegend überarbeitet. Anstelle von Pop-up-Dialogen gibt es nun eine eigene Unterseite, auf der sämtliche benötigten Daten erfasst werden können. Dabei lässt sich die Absender-Adresse entweder auf die persönliche E-Mail oder die Projekt-E-Mail setzen. Darüber hinaus können Empfänger bequem über eine integrierte Suchfunktion hinzugefügt, Vorlagen genutzt und bei Bedarf Dateien angehängt werden. Nach Eingabe aller Informationen führt ein Klick auf die Schaltfläche „Weiter“ zu einer Vorschau, in der alle Daten noch einmal überprüft werden können. Passt alles, erfolgt der Versand der E-Mail durch Bestätigung dieser Vorschau.

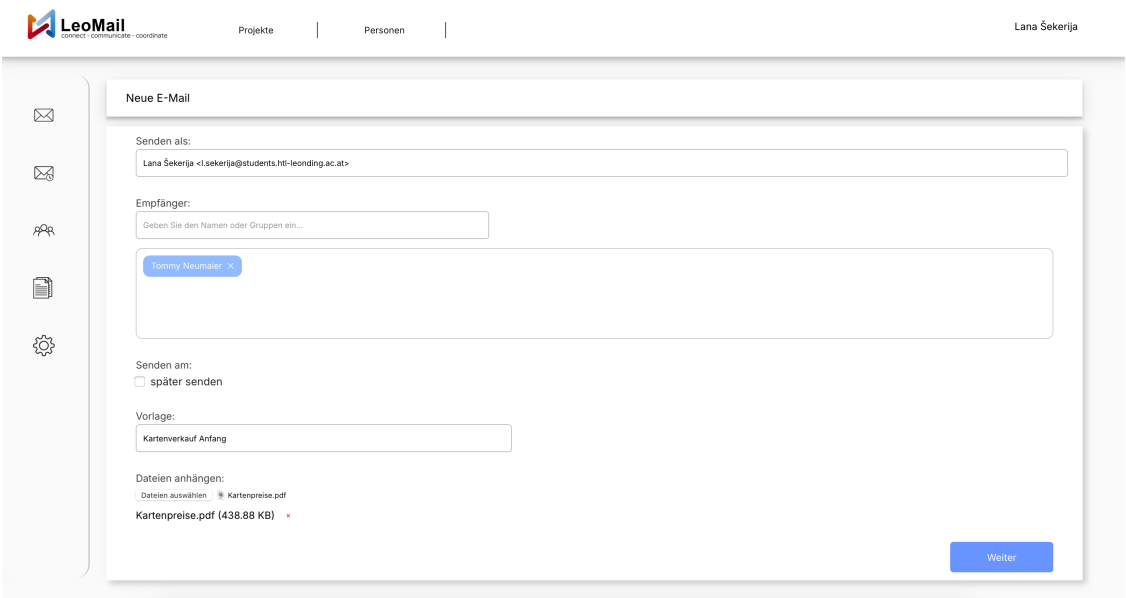

Abbildung 50: Finales Design der „Neue Mail“-Ansicht

Auch nach dem Versenden bleibt die Möglichkeit bestehen, versendete E-Mails zu betrachten. Bei einem Klick auf eine bereits gesendete Mail werden sämtliche eingegebenen Informationen übersichtlich dargestellt.

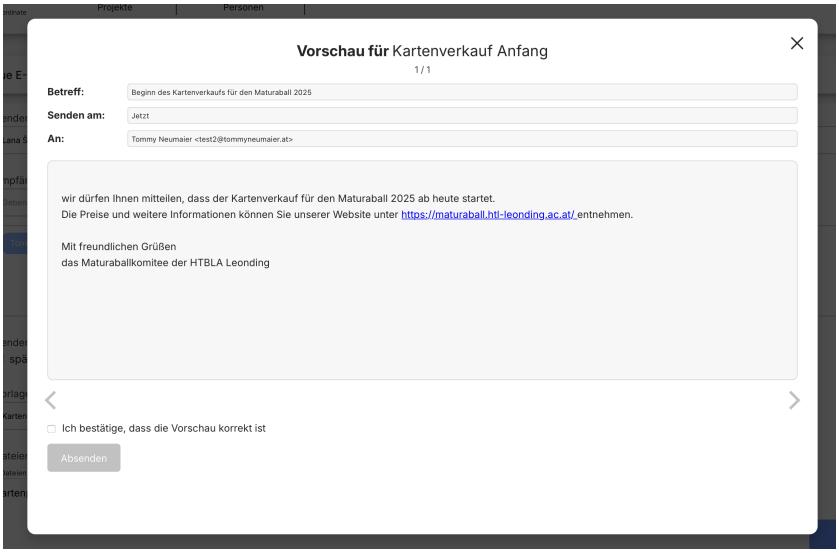

Abbildung 51: Finales Design der Mailvorschau

Geplante Mails

Die im ersten Entwurf vorgesehene Kalenderansicht wurde entfernt. Stattdessen gibt es nun eine Unterseite, auf der alle geplanten Mails aufgelistet sind. Diese können vor dem definierten Sendezeitpunkt gelöscht werden. Mit einem Klick auf eine geplante Mail lassen sich die entsprechenden Informationen anzeigen.

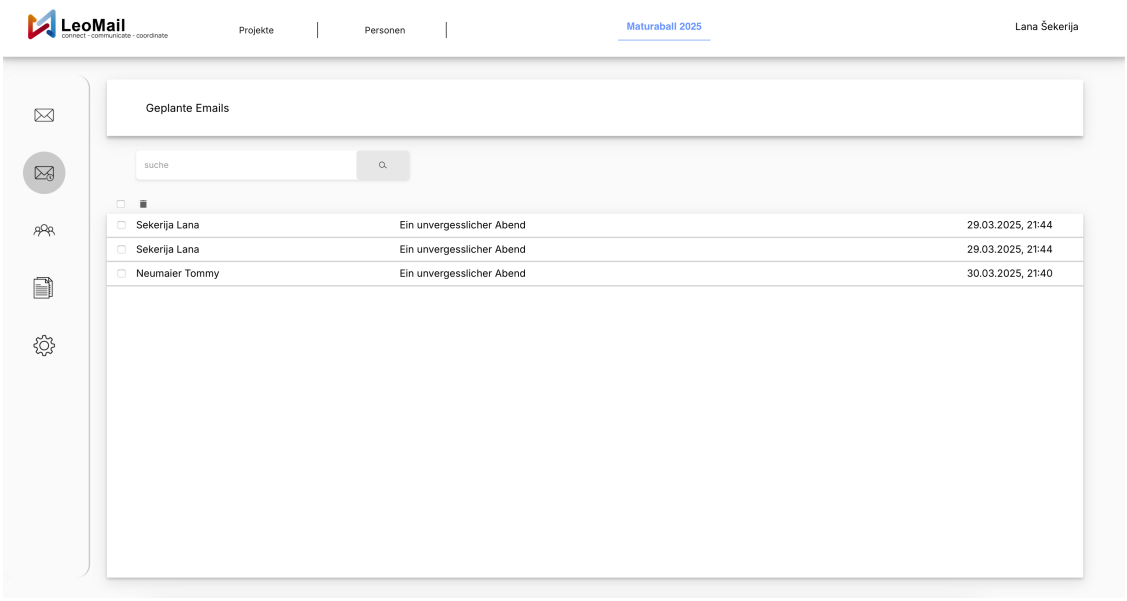

Abbildung 52: Finales Design der Übersicht „Geplante Mails“

Gruppen und Vorlagen

Die Ansichten für Gruppen und Vorlagen wurden – wie bereits im ersten Entwurf beschrieben – vollständig übernommen. Somit bleiben alle bewährten Funktionen erhalten.

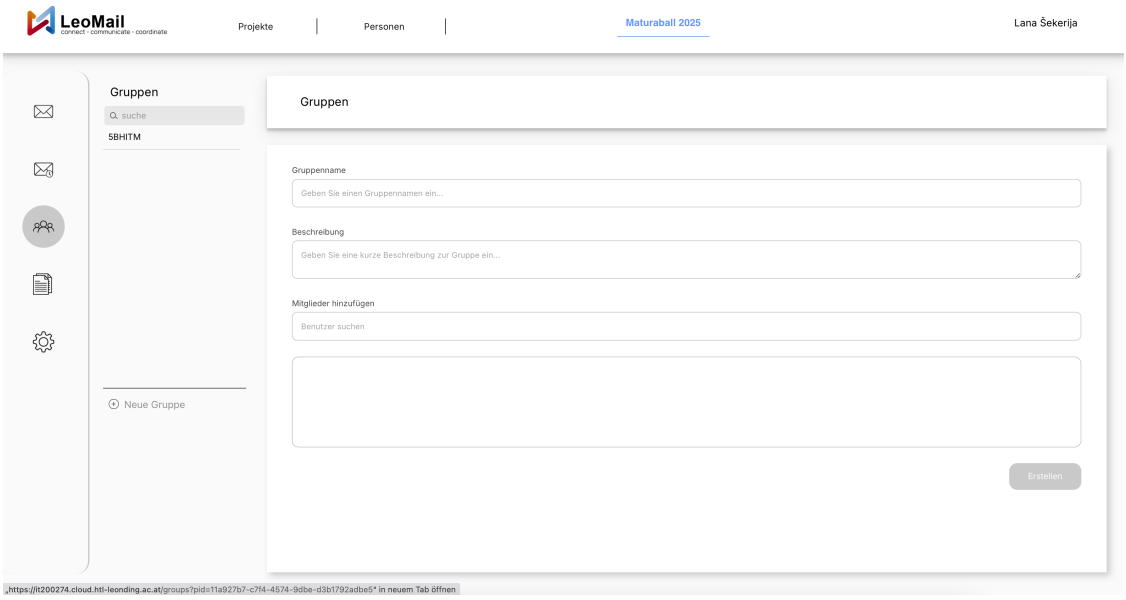

Abbildung 53: Finales Design der Gruppenübersicht

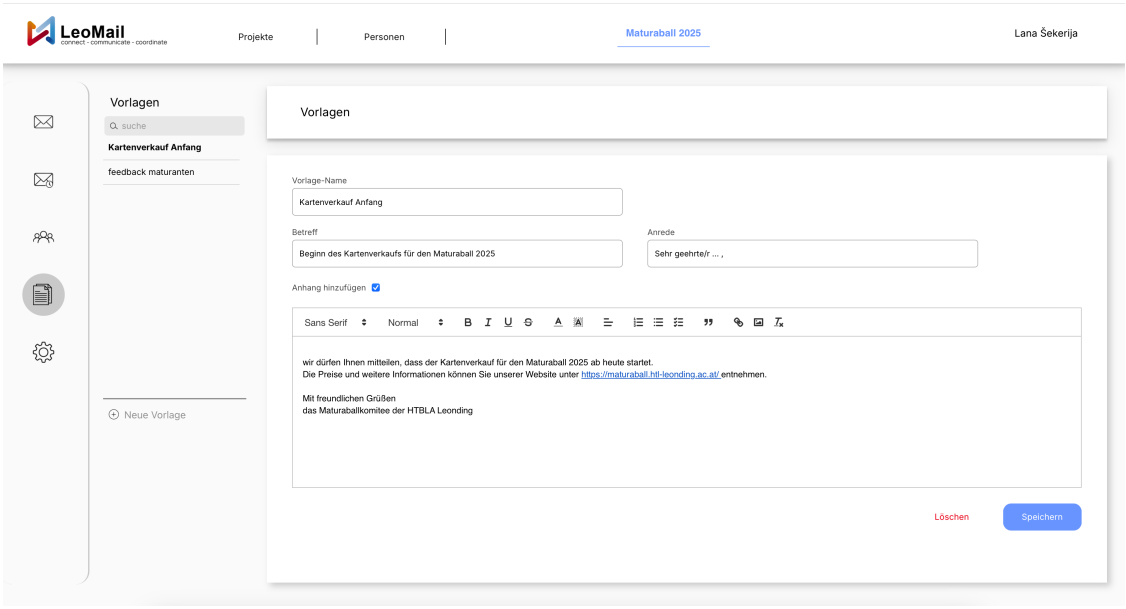

Abbildung 54: Finales Design der Vorlagenverwaltung

Projekteinstellungen

In der finalen Version wurde die Unterseite für Projekteinstellungen noch präziser ausgearbeitet. Ein Formular ermöglicht die Bearbeitung und Speicherung sämtlicher projektrelevanter Daten, die bereits bei der Erstellung des Projekts erfasst wurden. Darüber hinaus sorgt diese Seite für eine klare Übersicht, sodass alle wichtigen Informationen schnell zugänglich sind.

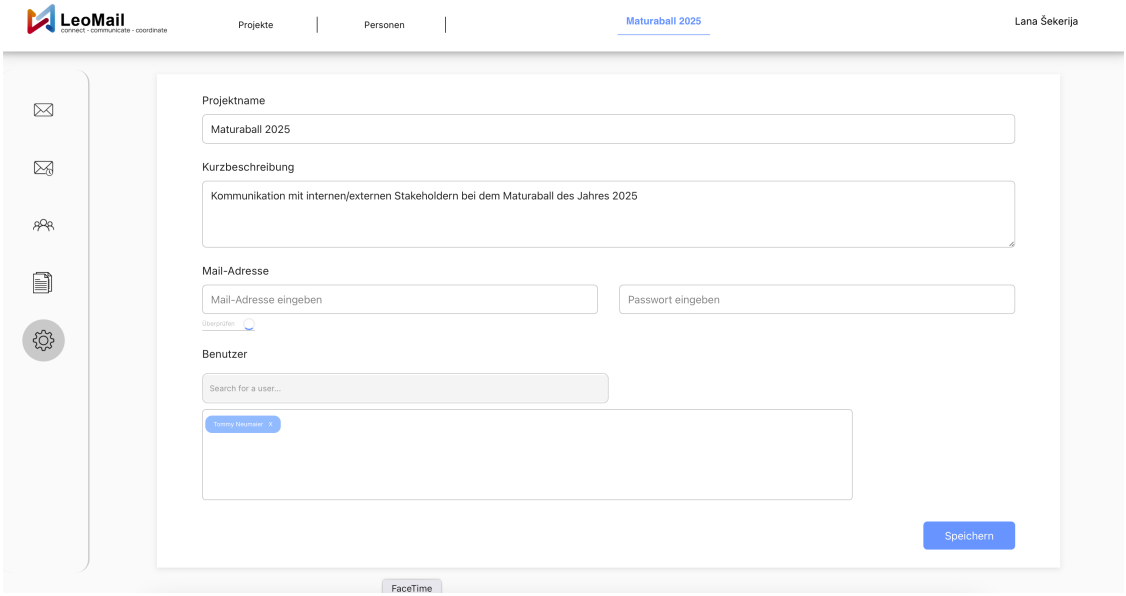

Abbildung 55: Finales Design der Projekteinstellungen

Profil

Auch die Profilseite wurde in der finalen Version weiter optimiert. Sie zeigt sämtliche wichtigen Daten der aktuell angemeldeten Person an und bietet zusätzlich eine Schaltfläche zum Ausloggen. Da in dieser Software keine Notwendigkeit für ein Profilbild besteht, wurde bewusst darauf verzichtet, um die Seite schlank und übersichtlich zu halten.

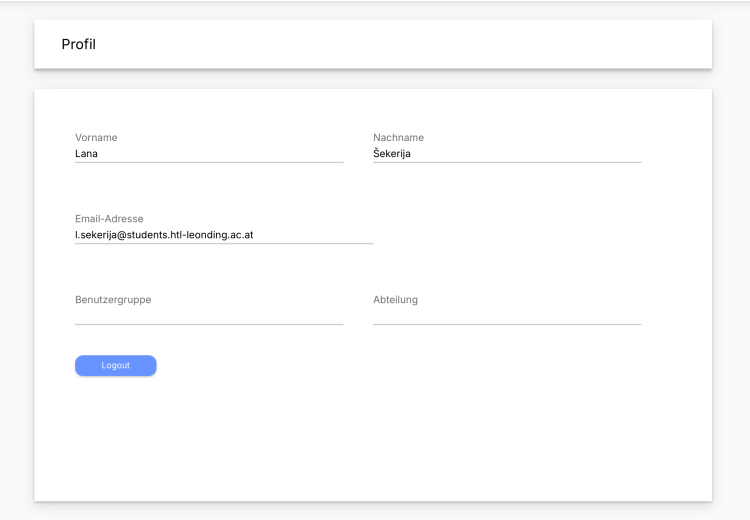

Abbildung 56: Finales Design der Profilansicht

7 Zusammenfassung

7.1 Schwierigkeiten und Lösungsansätze

Während der Entwicklung der Anwendung kam es immer wieder zu Problemen, welche die Entwicklung erschwerten oder gar einen gänzlich anderen Zugang nötig machten.

Ausfall des Identity Providers

Zu Entwicklungsstart kam es zu einem Ausfall des schuleigenen Keycloak-Identity Providers. Dieser Ausfall geschah exakt während der Entwicklung der Authentifizierung, was diese Situation umso drängender machte. Dieses Ereignis dauerte 2 Tage.

Da zur Installation des Mailservers bereits eine virtuelle Maschine angeschafft wurde, konnte hier eine eigene Keycloak-Instanz zur Überbrückung des Ausfalls installiert werden. Dies war, aufgrund der zuvor fehlenden Erfahrung im Umgang mit Keycloak, nicht einfach umzusetzen; Keycloak ist für Personen ohne vorherige Schulung oder umfassenden Umgang damit schwierig zu bedienen. Weiters wurde, vorsorglich eines möglichen Produktiveinsatzes der eigens installierten Instanz, auch auf Sicherheitsaspekte geachtet: was die Einarbeitung in Themen wie SSL-Zertifikate, sichere Proxy-Konfiguration und Absicherung der Anwendung notwendig machte.

Durch die Unterstützung von Hr. Prof. Christian Aberger und Hr. Prof. Thomas Stütz konnte dieses Problem, auch in den Ferien, gelöst werden. Anschließend konnte wieder die Instanz der HTBLA Leonding verwendet werden.

Fehlender Support von Vue.js

Die Nutzung von Vue.js hat das Projekt, gerade in Bezug auf die Geschwindigkeit der Anwendung, massiv bereichert. Trotzdem hat sich der Entwicklungsprozess schwierig gestaltet, da es bei weitem nicht den Community-Support erhält wie beispielsweise seine Konkurrenten Angular oder React. So musste bei Problemen mit Vue.js immer länger

nach Lösungen recherchiert werden, als es beispielsweise bei den größeren Frameworks der Fall gewesen wäre.

Weiters war es immer schwierig, für die jeweils aktuelle Vue-Version entsprechende Component-Libraries zu finden. Für Komponenten, bei denen der Aufwand einer Eigenentwicklung den Nutzen überstiegen hätte, konnten funktional geeignete Bibliotheken nicht verwendet werden, da sie nicht kompatibel waren.

Unklare Umsetzung des Dateiuploads

Mitten in der Entwicklungsphase stellte sich die Frage nach der Methodik, wie die Dateianhänge hochgeladen werden sollten. Das wurde notwendig, da es für die Übersichtlichkeit der Anwendung und der versendeten E-Mails notwendig war, auch nach dem Versand noch innerhalb der Web-Applikation einsehen zu können, ob und welche Dateianhänge versendet wurden. Dadurch mussten diese in einem Dateispeicher gespeichert werden.

Da die HTL Leonding über einen kostengünstigen Zugang zur Microsoft Azure Cloud verfügt, war der erste Plan, den Dateispeicher mittels einem *Azure Blob Storage* umzusetzen, da innerhalb des Projektteams hier bereits erste Erfahrungswerte gesammelt wurden. Durch organisatorische Schwierigkeiten kam es dazu, dass diese Option leider nicht möglich war und daher musste eine Alternative gefunden werden, welche in MinIO gefunden wurde.

Synchronisierung der Nutzerdaten

Die Synchronisierung der Nutzerdaten erwies sich als nächstes Problem: aufgrund von Keycloak und dessen Schnittstelle und weiterführend ohnehin aufgrund der veralteten Schnittstelle des LDAP-Servers konnten die IDP-User nicht laufend in der Anwendung nachsynchronisiert werden.

Diese Problematik ergab sich bei bisherigen (internen) Anwendungen der HTBLA Leonding noch nicht: denn meist war hier eine reine Authentifizierung oder ein sehr minimales Nutzen der hinterlegten Metadaten wie der Vorname, der Familienname oder der E-Mail-Adresse notwendig. Bei dieser Diplomarbeit allerdings werden teilweise die Daten von Hunderten Nutzern gleichzeitig benötigt, beispielsweise beim personalisierten Versand der E-Mails oder auch bei der Ansicht der gesendeten E-Mails, bei der die Empfänger aufgelistet werden.

Würde in diesen Anwendungsfällen jeweils immer eine GET-Anfrage an die Keycloak-Schnittstelle gesendet werden, würde das die Anwendung stark verlangsamen und beinahe unbenutzbar machen; eben gerade beim Versand der E-Mails, der ohnehin schon seine Zeit in Anspruch nimmt.

Momentan werden also alle relevanten Daten aus dem Identity Provider importiert. Das sind vor allem:

- Vor- und Nachname
- E-Mail-Adresse
- Klasse (oder etwaiger Lehrer-Status)

Diese werden mit dem *sub*, also der von Keycloak generierten ID, als Primärschlüssel in der Datenbank der Anwendung gespeichert. Das führt weitgehend zu dem Problem, dass dieser eindeutige Schlüssel vom Keycloak vergeben wird: bei etwaigen Resets der Keycloak-Datenbank würden alle Nutzerdaten verloren gehen bzw. ein riesiger Migrationsprozess entstehen.

Verwendung von Keycloak im Produktiveinsatz

Beim Deployment der Anwendung kam es zu dem Problem, dass seitens des Backend-Pods keine Anfragen an den Identity Provider der Schule gestellt werden konnten.

Dies hatte die Ursache einer Fehlkonfiguration des Routings innerhalb des Schulnetzwerkes: die Pods der LeoCloud versuchten, extern die Domain der Instanz *auth.htl-leonding.ac.at* aufzulösen. Da diese allerdings im selben Netzwerk lagen, kam es hier zu Problemen, da eigentlich intern innerhalb des Netzwerkes aufgelöst werden musste. Somit konnte das Backend den Provider nicht erreichen.

Dies wurde gelöst, indem beim Docker-Image des Backends in der */etc/hosts*-Datei eine Zeile mit der richtigen IP-Konfiguration des Authentifizierungsservices hinzugefügt wurde. Somit funktioniert nun die Authentifizierung, solange sich nicht die IP-Adresse des Servers, auf dem die Keycloak-Instanz liegt, ändert. Eine langfristige Lösung wäre es, diese Routing-Fehler im Schulnetzwerk zu beheben - dies ist allerdings nicht durch das Projektteam möglich und wurde auch entsprechend damals bei der zuständigen Person gemeldet.

Fehlende Daten

Zur zweckgemäßen Verwendung ist neben der Anwendung an sich auch essenziell, dass die Daten vorhanden sind. Benötigt werden dabei der vollständige Name, die E-Mail-Adresse und das Geschlecht, essenziell auch für die korrekte Personalisierung.

Der LDAP-Service der Schule weist hier Lücken auf. Während beispielsweise neuere Benutzerkonten über alle Daten, bis auf das Geschlecht, welches bis dato nirgends enthalten ist, verfügen, fehlen bei älteren Konten die Vor- und Nachnamen. Eine Behebung dieses Problems ist nur durch eine aufwendige Korrektur beziehungsweise Erweiterung der Daten möglich. Dieses Problem wurde ebenso an die zuständigen Personen eskaliert.

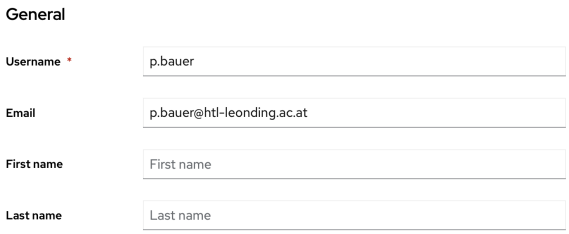

Abbildung 57: Unvollständiger Datensatz von AV Peter Bauer

7.2 Erweiterungsmöglichkeiten

Umstieg auf reaktive Programmierung

Zur besseren und geeigneteren Umsetzung der Service-Architektur wäre es vernünftig, von den klassischen Services, welche teilweise blockierend agieren, auf gänzlich reaktive Software umzusteigen.

Reaktive Anwendungen sind geeigneter, um viele Anfragen, vor allem auch bei solchen, die hohen I/O-Verkehr haben (bspw. Dateien, große Rückgabewerte, Service-Requests), zu verarbeiten. Reaktiv bedeutet, dass jede Anfrage separat in einen Thread ausgelagert wird und es keine Blockierungen in der Software gibt [70]. Quarkus bietet dafür umfassende Unterstützung über die Mutiny-Bibliothek [69].

Momentan bedeutet der klassische Zugang beispielsweise, dass beim Versenden einer E-Mail an viele Empfänger die Anwendung zeitweise für andere Personen unbrauchbar wäre, da solange diese Anfrage nicht endgültig abgeschlossen wird, keine weiteren Anfragen vom Server bearbeitet werden können. Dies würde sich in einem Einfrieren der Applikation äußern.

Für eine vollständige reaktive Umgebung müssen neben den klassischen REST-Schnittstellen allerdings auch die Datenbankzugriffe reaktiv verwendet werden: im Falle von PostgreSQL mit der Nutzung über Hibernate in Quarkus gibt es diese Möglichkeit [63]. Problematisch ist hierbei nur die Suche nach einem geeigneten Mail-Client: denn der Quarkus-Mail-Client, den es auch reaktiv gäbe, unterstützt keine Multiuser-Anwendungen [68]. Dies schränkt den praktischen Nutzen reaktiver Mail-Komponenten – insbesondere bei Multi-Tenant-Anwendungen – noch stark ein.

Listener für Nutzersynchronisation

Eine inkrementelle und dauerhafte Synchronisation der Nutzerdaten wäre speziell für die grundlegende Funktionalität und auch die Datenkonsistenz der Anwendung stark von Bedeutung. Leider bietet Keycloak hier grundlegend nicht allzu viele Möglichkeiten – momentan bestünde nur die Möglichkeit, mit einer GET-Anfrage sogenannte *Admin Events* abzufragen und daraufhin alle jeweils betroffenen Nutzer erneut zu fetchen.

Keycloak selbst bietet allerdings eine EventListenerProvider-Schnittstelle, über welche man eigene Event Listener schreiben und in die bestehende Instanz einbetten kann [93]. Diese kann auf spezifische Ereignisse – etwa Benutzeränderungen – reagieren und in Echtzeit eine Synchronisation anstoßen. Ein solcher Listener müsste als Keycloak-Extension implementiert und auf dem Server deployed werden [91]. Für die Implementierung sowie Integration in die laufende Keycloak-Instanz wäre jedoch die Unterstützung von Hr. Prof. Christian Aberger bzw. Hr. Prof. Thomas Stütz erforderlich. Diese Möglichkeit sollte – auch zwecks Wartbarkeit – genauer evaluiert werden.

Globale Gruppen

Globale, vordefinierte Gruppen würden Verwaltungsaufwand einsparen. Beispielsweise könnten Benutzergruppen wie Klassengruppen, Abteilungsgruppen, Maturanten oder Lehrpersonal als Verteilergruppen in jedem Projekt automatisch hinterlegt werden, da diese häufig gebraucht werden.

Die Zuweisung von Rollen und Rechten über Gruppen ist eine etablierte Praxis im Identity Management, die in Keycloak und Active Directory gleichermaßen empfohlen wird [53, 77]. Dabei werden Berechtigungen einmalig auf Gruppenebene definiert und automatisch an alle zugewiesenen Benutzer weitergegeben.

E-Mail-Benachrichtigungen

E-Mail-Benachrichtigungen über bevorstehende Events, wie beispielsweise dem baldigen Versenden einer geplanten E-Mail, oder größeren Aktivitäten wie dem Erstellen eines Projekts, sind eine weitere Möglichkeit, LeoMail zu erweitern. Somit können beispielsweise kurz vor dem Versand noch geplante E-Mails auf ihre Vollständigkeit bzw. Richtigkeit überprüft werden.

Eine Alternative dazu wären auch In-App-Benachrichtigungen, im Header der Anwendung eingebettet. Hier würde man über Aktivitäten anderer Projektmitglieder innerhalb des Projekts informiert werden. Beispiele hierzu wären das Erstellen einer neuen Vorlage oder der Versand von E-Mail-Vorlagen.

Die Kombination beider Kanäle – In-App und E-Mail – gilt als Best Practice in modernen Webanwendungen und ist beispielsweise bei Plattformen wie Facebook oder Trello üblich [60]. In Vue.js können mit Hilfe von WebSockets oder Server-Sent Events Echtzeit-Benachrichtigungen realisiert werden. Für E-Mail-Versand bieten sich sowohl klassische SMTP-Lösungen wie Quarkus Mailer als auch externe Anbieter wie SendGrid oder Novu an [79].

Kalenderansicht bei geplanten E-Mails

Zusätzlich zur Listenansicht wäre eine Kalenderansicht als zweite Anzeigemöglichkeit der geplanten E-Mails sinnvoll. Diese würde eine noch bessere Übersicht über geplante E-Mails bieten, vor allem in Hinsicht auf deren zeitliche Einordnung.

Durch UI-Komponenten wie *FullCalendar* lässt sich eine intuitive Kalenderansicht in Vue.js-Anwendungen integrieren. Sie erleichtert die zeitliche Planung und Übersicht geplanter Events deutlich.

7.3 Zielerreichung

Ziel dieser Untersuchung war es, zu evaluieren, inwiefern eine Software, die die Kommunikationsabläufe – insbesondere im Kontext zahlreicher Stakeholder – vereinfachen und zeiteffizienter gestalten soll, mit den ausgewählten Technologien umgesetzt werden kann. Die Analyse und der Prototypenbau haben ergeben, dass die gewählten Technologien eine solide Basis bieten, um diese Anforderungen zu erfüllen.

Die Entscheidung für Vue.js als Frontend-Framework ermöglichte eine schnelle und flexible Entwicklung der Benutzeroberfläche. Dank der komponentenbasierten Architektur und des Einsatzes von Pinia als State-Management-Lösung konnten Benutzerinteraktionen und dynamische Datenflüsse effizient umgesetzt werden, was gerade bei der Kommunikation mit vielen Stakeholdern von Vorteil ist.

Auf der Backend-Seite sorgt Quarkus in Kombination mit PostgreSQL für eine performante und skalierbare Infrastruktur. Diese ermöglicht es, große Mengen an Daten – wie beispielsweise personalisierte E-Mail-Informationen – zügig zu verarbeiten. Die Integration eines robusten Identity Managements via Keycloak stellt zudem sicher, dass Authentifizierung und Autorisierung sicher und nutzerfreundlich abgewickelt werden, was bei der Zusammenarbeit verschiedener Nutzergruppen unerlässlich ist.

Weitere wichtige Komponenten, wie MinIO für den Dateispeicher und der Einsatz eines Mailservers zur Unterstützung des E-Mail-Versands, tragen dazu bei, Kommunikationsprozesse zu automatisieren und zu personalisieren. Besonders hervorzuheben ist hierbei der Scheduler, der das zeitgesteuerte Versenden von E-Mails ermöglicht, sowie das Template Rendering mittels Qute, welches eine hohe Flexibilität in der Personalisierung der Nachrichten sicherstellt.

Insgesamt zeigt sich, dass die ausgewählten Technologien nicht nur die Machbarkeit, sondern auch die Effizienz der geplanten Kommunikationssoftware unter Beweis stellen. Durch die nahtlose Integration moderner Frontend- und Backend-Lösungen, ergänzt durch ein durchdachtes Identitätsmanagement und automatisierte Prozesse, wird das Ziel, die Kommunikationsabläufe zu vereinfachen und zeiteffizienter zu gestalten, weitestgehend erreicht.

Literatur

[1] adobe. *Lerne die neuen Funktionen von Adobe Illustrator kennen.* URL: https://www.adobe.com/at/products/illustrator/features.html.

[2] *Adobe Illustrator Logo.* URL: https://de.wikipedia.org/wiki/Adobe_Illustrator.

[3] Amazon Web Services. *What is SDLC (Software Development Lifecycle)?* https://aws.amazon.com/what-is/sdlc/. 2023.

[4] Ambient. *Vue.js.* 2024. URL: https://ambient.digital/wissen/ambipedia/vue-js/.

[5] *Angular Logo.* URL: https://en.wikipedia.org/wiki/File:Angular_full_color_logo.svg.

[6] Atlassian. *Git Hooks.* https://www.atlassian.com/git/tutorials/git-hooks. Atlassian Git Tutorials. 2024.

[7] Atlassian. *The Different Types of Testing in Software.* https://www.atlassian.com/continuous-delivery/software-testing/types-of-software-testing. Atlassian Continuous Delivery tutorial (n.d.)

[8] Atlassian. *Understanding the Project Management Phases.* https://www.atlassian.com/work-management/project-management/phases. Atlassian Work Life Blog (n.d.)

[9] Atlassian. *What is Agile Project Management? How to Start.* https://www.atlassian.com/agile/project-management. Atlassian Agile Coach guide (n.d.)

[10] Atlassian. *What is SDLC? Software Development Life Cycle Explained.* https://www.atlassian.com/agile/software-development/sdlc. Atlassian Agile Coach article (n.d.)

[11] Atlassian. *What is Version Control? (Atlassian Git Tutorial).* https://www.atlassian.com/git/tutorials/what-is-version-control. Atlassian DevOps Guide (n.d.)

[12] Axios. *Getting Started (Axios Documentation).* https://axios-http.com/docs/intro. 2023.

[13] Alexander Bergmann. *Mail Merge [Thunderbird-Add-on]*. https://addons.thunderbird.net/de/thunderbird/addon/mail-merge/. Version 11.0.0, aktualisiert am 5.2.2025. 2025.

[14] Conceptboard. *US Cloud Act: Bedrohung des europäischen Datenschutzes*. Blogartikel, Zugriff am 27.03.2025. 2021.

[15] *Die 10 wichtigsten Grundregeln für typografisches Gestalten.* 2017. URL: https://www.viaprinto.de/blog/10-grundregeln-typografie/.

[16] Docker. *Pre-seeding database with schema and data at startup for development environment*. https://docs.docker.com/guides/pre-seeding/. Docker Documentation. 2024.

[17] Alexandra Eger. *Den perfekten Firmennamen finden: Ein kreativer Leitfaden mit 24 Tipps*. 2022. URL: https://de.wix.com/blog/beitrag/firmennamen-finden?utm_source=google&utm_medium=cpc&utm_campaign=12446591242%5E122042633441%5Esearch%20-%20dsa&experiment_id=%5E%5E501759857012%5E&gad_source=1&gbraid=0AAAADwEfwXMMnm-WBgB35i9voxmt-b87&gclid=CjwKCAiAtYy9BhBcEiwANWQQL_pFY7JWMk39mHlvIri5GwSSxQjijFkqVTpGpgCBwE.

[18] eM Client Inc. *How to send an email to multiple recipients individually*. Zugriff am 27.03.2025. n.d.

[19] Empatic. *Was ist eigentlich UX-Design?* 2025. URL: https://empatic-ux.com/blog/was-ist-ux/.

[20] Wolf-Dieter Fiege. *8 Gründe, warum Sie das quelloffene Django Webframework nutzen sollten*. 2024. URL: https://www.hosteurope.de/blog/8-gruende-warum-sie-das-quelloffene-django-webframework-nutzen-sollten/.

[21] *Figma Design*. URL: https://www.figma.com/de-de/design/.

[22] *Figma Logo*. URL: https://commons.wikimedia.org/wiki/File:Figma-logo.svg.

[23] Eclipse Foundation. *Jakarta Mail API – SMTP Transport and Authentication*. https://jakarta.ee/specifications/mail/1.6/apidocs/com/sun/mail/smtp/package-summary. 2020.

[24] GitHub. *Connecting to GitHub with SSH*. https://docs.github.com/en/authentication/connecting-to-github-with-ssh. 2023.

[25] GitHub. *Deploying to Google Kubernetes Engine (GitHub Actions Guide)*. https://docs.github.com/en/actions/deploying/deploying-to-google-kubernetes-engine. 2023.

[26] GitHub. *Introduction to GitHub Packages (Container Registry)*. https://docs.github.com/en/packages/learn-github-packages/introduction-to-github-packages. 2023.

[27] GitHub. *Signing commits*. https://docs.github.com/en/authentication/managing-commit-signature-verification/signing-commits. 2023.

[28] GitHub. *Using secrets in GitHub Actions*. https://docs.github.com/en/actions/security-guides/encrypted-secrets. 2023.

[29] Martin Hahn. *Der Leitfaden für ein modernes UI Design*. URL: https://www.webdesign-journal.de/user-interface-design/.

[30] Martin Hahn. *Designprinzipien – warum und wie ein Design wirkt*. 2025. URL: https://www.webdesign-journal.de/designprinzipien.

[31] Red Hat. *Keycloak Server Administration Guide (Clients and Roles)*. 2023.

[32] Red Hat. *Secure Applications and Services with OpenID Connect (Keycloak Documentation)*. 2023.

[33] Red Hat. *Was ist Quarkus? Vorteile und Einsatz erklärt*. 2023. URL: https://www.redhat.com/de/topics/cloud-native-apps/what-is-quarkus.

[34] IBM. *Was ist Django?* 2024. URL: https://www.ibm.com/de-de/topics/django.

[35] JetBrains. *Quarkus (IntelliJ IDEA 2024.3 Documentation)*. 2024.

[36] JetBrains. *Vue.js (WebStorm 2024.3 Documentation)*. 2024.

[37] Keycloak. *Roles and Role Mappings in Keycloak*. 2023. URL: https://www.keycloak.org/docs/latest/server_admin/#roles.

[38] Keycloak. *Securing Applications: Direct Grant API*. 2023. URL: https://www.keycloak.org/docs/latest/securing_apps/#direct-grant.

[39] Keycloak. *Single Sign-On (SSO) Architecture*. 2023. URL: https://www.keycloak.org/docs/latest/server_admin/#single-sign-on-architecture.

[40] Keycloak. *Token Validation in Keycloak*. 2023. URL: https://www.keycloak.org/docs/latest/securing_apps/#_token_validation.

[41] Yunus Kimyonok. *Was ist Angular?* 2024. URL: https://platri.de/tech-wiki/was-ist-angular/.

[42] Dirk Knop. *Microsoft krallt sich Zugangsdaten: Achtung vor dem neuen Outlook*. Nov. 2023.

[43] Jessy Kösterke. *Logo-Design: Diese 5 einfachen Grundregeln solltest du berücksichtigen*. 2016. URL: https://t3n.de/news/logo-design-grundregeln-772395/.

[44] Landesbeauftragte für Datenschutz und Informationsfreiheit NRW. *Outlook-Update synchronisiert E-Mails in die Cloud*. https://www.ldi.nrw.de/outlook-update-synchronisiert-e-mails-die-cloud. Pressemitteilung. Dezember 2023.

[45] LogicMonitor. *Java vs. Python: Which Language is Faster?* 2022. URL: https://www.logicmonitor.com/blog/java-vs-python-which-language-is-faster.

[46] *Logo von Keycloak*. URL: https://commons.wikimedia.org/wiki/File:Keycloak_Logo.png.

[47] *Logo von Microsoft Outlook*. URL: https://upload.wikimedia.org/wikipedia/commons/thumb/d/df/Microsoft_Office_Outlook_%282018%E2%80%93present%29.svg/2203px-Microsoft_Office_Outlook_%282018%E2%80%93present%29.svg.png.

[48] *Logo von MinIO*. URL: https://panzura.com/wp-content/uploads/2021/04/minio.png.

[49] *Logo von Mozilla Thunderbird*. URL: https://www.chip.de/ii/3/8/6/8/6/9/thunderbird.gif-58984b5beeda9307.jpg.

[50] *Logo von PostgreSQL*. URL: https://upload.wikimedia.org/wikipedia/commons/thumb/2/29/Postgresql_elephant.svg/1200px-Postgresql_elephant.svg.png.

[51] *Logo von Python Django*. URL: https://www.svgrepo.com/svg/353657/django-icon.

[52] *Logo von Quarkus*. URL: https://gedoplan.de/wp-content/uploads/2024/03/quarkus_icon_ohne_text-e1710152142832.png.

[53] Microsoft Learn. *Gruppenbasierte Zugriffssteuerung – Übersicht*. https://learn.microsoft.com/de-de/entra/identity/users/groups-overview. Microsoft Learn, Zugriff am 27.03.2025. 2025.

[54] Microsoft Support. *Use mail merge in Word to send bulk email messages*. https://support.microsoft.com/en-us/office/use-mail-merge-in-word-to-send-bulk-email-messages-0f123521-20ce-4aa8-8b62-ac211dedefa4. Zugriff am 27.03.2025. n.d.

[55] Inc. MinIO. *Java Quickstart Guide – MinIO Java SDK for Amazon S3*. https://min.io/docs/minio/linux/developers/java/minio-java.html. 2023.

[56] Inc. MinIO. *MinIO Admin Console Complete Guide*. 2023. URL: https://docs.min.io/docs/minio-admin-complete-guide.html.

[57] Inc. MinIO. *MinIO Blog*. 2023. URL: https://blog.min.io/.

[58] Inc. MinIO. *MinIO: High Performance Object Storage*. 2023. URL: https://min.io/.

[59] Mozilla Foundation. *OpenPGP in Thunderbird - HOWTO and FAQ*. https://support.mozilla.org/en-US/kb/openpgp-thunderbird-howto-and-faq. Thunderbird Support-Artikel, zuletzt aktualisiert am 20.6.2022. Juni 2022.

[60] Dani Nevo. *Email vs In-App Notifications: When and Why to Use Each*. https://www.insiderapps.com/blog/email-vs-in-app-notifications. InsiderApps Blog. 2022.

[61] Nadine Noack. *React, Angular und Vue.js: Eine Gegenüberstellung von Frontend-Frameworks*. 2024. URL: https://www.dotnetpro.de/frontend/user-interface/react-angular-vuejs-gegenueberstellung-frontend-frameworks-2910346.html.

[62] Vera P. *Was ist React: Funktionen verstehen und wie man es für die moderne Webentwicklung einsetzt*. 2024. URL: https://www.hostinger.de/tutorials/was-ist-react.

[63] Abhishek Palriwal. *How to connect PostgreSQL in Quarkus using reactive programming*. https://medium.com/@abhishek.palriwal/quarkus-postgresql-reactive-guide. Medium Blogartikel. 2024.

[64] Conny Pflanz. *Website erstellen: 7 Regeln, die beim UI Design wichtig sind*. 2022. URL: https://www.lpsp.de/blog/website-erstellen-regeln-die-beim-ui-design-wichtig-sind.

[65] Mailcow Project. *DNS setup (mailcow: dockerized documentation)*. https://docs.mailcow.email/getstarted/prerequisite-dns/. 2023.

[66] Quarkus. *Quarkus Extensions – Search for MinIO*. 2023. URL: https://quarkus.io/extensions/io.quarkiverse.minio/quarkus-minio/.

[67] Quarkus. *Securing Applications with Keycloak and Quarkus*. 2023. URL: https://quarkus.io/guides/security-openid-connect.

[68] Quarkus Project. *Mailer Reference Guide*. https://quarkus.io/guides/mailer-reference. Quarkus Official Documentation (Email sending in Quarkus). 2023.

[69] Quarkus Project. *Mutiny – Async for mere mortals*. https://quarkus.io/guides/mutiny-primer. Quarkus Official Documentation (Mutiny guide). 2023.

[70] Quarkus Project. *Quarkus Reactive Architecture*. https://quarkus.io/guides/quarkus-reactive-architecture. Offizielle Quarkus-Dokumentation. 2023.

[71] Quarkus.io. *Configure data sources in Quarkus*. https://quarkus.io/guides/datasource. 2023.

[72] Quarkus.io. *Cross-Origin Resource Sharing (CORS)*. https://quarkus.io/guides/security-cors. 2023.

[73] Quarkus.io. *Mailer Reference Guide*. https://quarkus.io/guides/mailer-reference. 2023.

[74] Quarkus.io. *Qute Templating Engine*. https://quarkus.io/guides/qute. 2023.

[75] Quarkus.io. *Scheduler Reference Guide*. https://quarkus.io/guides/scheduler-reference. 2023.

[76] *React Logo*. URL: https://commons.wikimedia.org/wiki/File:React_Logo_SVG.svg.

[77] Red Hat. *Keycloak User Groups and Role Mapping*. https://www.keycloak.org/docs/latest/server_admin/#user-groups. Keycloak Admin Guide. 2023.

[78] Lukas Rudrof. *Die richtige Farbwahl im Webdesign*. 2023. URL: https://www.lukas-rudrof.de/blog/richtige-farbwahl-im-webdesign.

[79] Rogerio Santos. *Top 5 open-source notification systems (Novu, Postal, etc.)* https://dev.to/rogervs/top-5-open-source-notification-systems-novu-postal-etc-3gj5. DEV.to Blog. 2023.

[80] Patrick C. Shih, Gina Venolia und Gary M. Olson. "Brainstorming Under Constraints: Why Software Developers Brainstorm in Groups". In: *Proceedings of the 25th BCS Conference on Human-Computer Interaction (HCI 2011)*. British Computer Society, 2011, S. 74–83. DOI: 10.14236/ewic/HCI2011.30.

[81] Saurabh Singh. *Scan Git repositories for sensitive information and security issues by using git-secrets*. https://docs.aws.amazon.com/prescriptive-guidance/latest/patterns/scan-git-repositories-for-sensitive-information-and-security-issues-by-using-git-secrets.html. AWS Prescriptive Guidance. 2023.

[82] Christopher Brettingham Smith. *USER-INTERFACE-DESIGN, 7 REGELN, DIE SIE KENNEN SOLLTEN*. 2022. URL: https://www.brettingham.de/insights/user-interface-design-7-regeln-die-sie-kennen-sollten/.

[83] Christopher Brettingham Smith. *USER-INTERFACE-DESIGN, 7 REGELN, DIE SIE KENNEN SOLLTEN*. 2023. URL: https://www.brettingham.de/insights/user-interface-design-7-regeln-die-sie-kennen-sollten/.

[84] Stack Overflow. *Beyond Git: The other version control systems developers use*. https://stackoverflow.blog/2023/01/09/beyond-git-the-other-version-control-systems-developers-use/. Stack Overflow Developer Survey 2022–2023 insights. 2023.

[85] Philipp Steubel. *Die Nutzwertanalyse: Definition, Anwendung und Beispiel!* 2024. URL: https://asana.com/de/resources/utility-analysis.

[86] Tobias Surmann und Phillip Conrad. *Keycloak - Ein Überblick*. URL: https://www.smf.de/keycloak-ein-ueberblick/.

[87] Alistair G. Sutcliffe. *Requirements Engineering*. https://www.interaction-design.org/literature/book/the-encyclopedia-of-human-computer-interaction-2nd-ed/requirements-engineering. Retrieved 2025, from Interaction-Design.org (n.d.)

[88] Astera Analytics Team. *PostgreSQL vs. Oracle: 8 Differences to Know About*. URL: https://www.astera.com/knowledge-center/postgresql-vs-oracle/.

[89] Quarkus Team. *Quarkus Performance: Building Native Java Applications*. 2023. URL: https://quarkus.io/blog/quarkus-performance/.

[90] Vue.js Core Team. *Pinia – The intuitive store for Vue.js (Documentation)*. https://pinia.vuejs.org/. 2023.

[91] Ashwin Thattey. *Creating a Custom Event Listener in Keycloak*. https://thattey.medium.com/keycloak-custom-event-listener-55d78590b8cd. Medium Artikel. 2019.

[92] *The Inter typeface family*. URL: https://rsms.me/inter/#features.

[93] Stian Thorgersen. *Keycloak Event Listener SPI*. https://www.keycloak.org/docs/latest/server_development/#eventlistener-spi. Keycloak Developer Documentation. 2023.

[94] *Vue Logo*. URL: https://de.m.wikipedia.org/wiki/Datei:Vue.js_Logo_2.svg.

Abbildungsverzeichnis

Tabellenverzeichnis

Quellcodeverzeichnis