Spam Solution

Bachelorarbeit

Studiengang Informatik
OST - Ostschweizer Fachhochschule

Frühjahrssemester 2022

Autoren:	Bojan Dakic, Fabian Tiri
Betreuer:	Prof. Frank Koch
Experte:	Prof. Hansjörg Huser
Gegenleser:	Prof. Mirko Stocker
Industriepartner:	Michael Güntensperger (AdaptIT)

Abstract

1978 wurde die erste Spam-Mail verschickt. Damals verschickte Gary Thurek eine Werbemail an fast 600 Personen und somit an 25% der damaligen Internetbenutzer. Beschwerden liessen nicht lange auf sich warten, jedoch war es Gary Thurek dank der E-Mail möglich, Verkäufe im Wert von zwölf Millionen Dollar abzuschliessen.[1] Seitdem ist Spam zu einem immer grösseren Problem geworden, dass es ständig zu bekämpfen gilt. Mittlerweile werden ausgefeilte Techniken für die Erkennung und Sortierung von Spam angewendet. Durch diese Techniken, wird das Problem für den einzelnen Benutzer zwar stark eingedämmt, jedoch kommt auch der Fall eines „False Positive“ vor. Somit werden legitime E-Mails als Spam kategorisiert und eventuell nicht vom Benutzer wahrgenommen.

Das Ziel dieser Arbeit ist es eine Lösung zu entwickeln, die dem Problem des „False Positive “ entgegenwirkt, indem sie den Ansatz einer Whitelist verfolgt. Hierfür soll in der Kopfzeile bei E-Mail von unbekannten Absendern ein Hinweis hinzugefügt werden, der es dem Benutzer ermöglicht, die E-Mail auf die Whitelist oder Blacklist zu setzen. Weiterhin soll eine Webseite entwickelt werden, mithilfe derer der Benutzer seine Listen verwalten kann.

Das Ergebnis unserer Arbeit ist ein Service, der eine Abhilfe gegen Spam schafft. Ein Banner zuoberst in der E-Mail gibt klar zu erkennen, ob ein Absender bekannt ist. Zudem ermöglicht der Banner es Benutzern schnell und einfach zu entscheiden, wie in Zukunft mit diesem Absender verfahren werden soll. E-Mails, die auf der Blacklist stehen, werden automatisch mit dem IMAP-Protokoll in den Spam-Ordner verschoben. Auf der dazugehörigen Webseite kann man sich für den Service registrieren und dort seine Listen verwalten. Weiterhin ist es möglich, auf der Webseite sogar ganze Domains auf eine Liste zu setzen.

Management Summary

Ausgangslage

Seit den 90er Jahren sind Spam-Mails ein andauerndes Problem. Im Jahr 2021 wurden durch sie in der Schweiz 75% der Sicherheitsprobleme im Internet verursacht.[2] Ihr Anteil am gesamten E-Mail Verkehr zwischen 2017 und 2021 lag weltweit bei rund 45%.[3] Algorithmen, die von E-Mail-Anbietern genutzt werden, um Spam zu erkennen und zu filtern, funktionieren bereits sehr gut. Dabei kommt es jedoch auch vor, dass eine legitime Nachricht vom Algorithmus als Spam erkannt und gefiltert wird.

In dieser Bachelorarbeit (BA) werden wir einen Service vorstellen, der dem Problem des „False Posiives“ entgegenwirkt.

Vorgehen

Im Vorfeld der Arbeit wurde mit dem Industriepartner AdaptIT GmbH der Scope für diese BA definiert. Nachdem der Projektplan fertiggestellt war, wurde die Architektur entworfen und die grundlegenden Technologien festgelegt. Daraufhin erstellten wir diverse Modelle und Use Cases (UCs), die in der Entwicklung als Hilfestellung dienen sollten. Zum Abschluss der Vorarbeit wurde mittels eines Prototypen die technische Machbarkeit sichergestellt und anschliessend die Lösung entwickelt.

Ergebnisse

Das Ergebnis dieser Arbeit ist ein Service, der eine Abhilfe gegen Spam verschafft und das Problem des „False Positives“ adressiert. Basierend auf dem Ansatz einer White- und Blacklist werden die Absender hereinkommender E-Mails analysiert. Dabei werden Nachrichten von Absendern auf der Blacklist mittels IMAP automatisch in den Spam-Ordner verschoben. Bei unbekannten Absendern wird oben in der E-Mail ein Banner hinzugefügt, der den Benutzer warnt und ihm die Möglichkeit bietet den Absender auf die White- oder Blacklist zu setzen. Die persönliche White- und Blacklist kann über eine dazugehörige Webseite verwaltet werden.

Ausblick

In dieser Arbeit wurde bereits eine funktionsfähige Lösung entwickelt. Allerdings dient sie nur als Grundstein für einen grösseren Service. Die Applikation sollte um die noch nicht implementierten UCs erweitert werden.

Folgende UCs sind noch offen:

- Den Inhalt der E-Mail analysieren und im Banner eine Empfehlung über die Vertrauenswürdigkeit dieser hinzufügen.
- Weiterleiten der E-Mails an einen Exchange-Server.
- Integration für Organisationen mit einer neuen Rolle eines Superusers. Dieser soll organisationsweite White- und Blacklisten erstellen können, die daraufhin von den Nutzern abonniert werden.
- Ein Outlook Plugin, welches Adressen, an die man in der Vergangenheit eine E-Mail geschickt hat, extrahiert und automatisch zu der Whitelist hinzufügt.

Danksagung

Wir möchten unserer Betreuungsperson Herr Prof. Frank Koch einen grossen Dank für den wertvollen Input während den Meetings aussprechen. Auch möchten wir uns beim Industriepartner Michael Güntensperger von der Firma AdaptIT für die gute Zusammenarbeit bedanken. Die schnellen Antworten und die unkomplizierte Kommunikation haben wir bei beiden sehr geschätzt.

Unser Dank gilt auch den beiden Korrekturleserinnen Dajana Dakic und Sophie Kunze. Zudem danken wir allen Personen, die uns bei unserer Bachelorarbeit in irgendeiner Art und Weise unterstützt haben.

Inhaltsverzeichnis

1 Einleitung

In diesem Abschnitt gehen wir auf die Problemstellung und die Ziele dieser Arbeit ein. Diese leiten sich aus der offiziellen Aufgabenstellung ab, die im Anhang zu finden ist. Weiterhin gehen wir auf Einschränkungen ein, die für dieses Projekt gelten.

1.1 Problemstellung

Eines der wichtigsten Kommunikationsmittel in der heutigen Zeit ist die E-Mail. Nahezu jeder, der das Internet verwendet, besitzt auch eine E-Mail-Adresse. Jedoch wird ihre Funktionalität oftmals missbraucht, um Spam zu verschicken. Um diesen zu bekämpfen, verwenden E-Mail Anbieter Algorithmen, die ihn erkennen und automatisch blockieren sollen. Die Algorithmen sind zwar in der Lage das Problem zu minimieren, allerdings führen sie zu einem anderen. Es kann passieren, dass eine legitime E-Mail fälschlicherweise als Spam erkannt wird und von somit auch nicht vom Benutzer wahrgenommen wird. Dieses Problem des „False Positives“ soll in dieser Arbeit adressiert werden.

1.2 Ziele

In dieser Arbeit soll eine Applikation entwickelt werden, die E-Mails anhand des Absenders filtert. Absender, die auf der Whitelist stehen, sollen im Posteingang landen, während jene auf der Blacklist in den Spam-Ordner verschoben werden. Bei unbekannten Absendern soll zunächst ein Banner hinzugefügt werden, mit dem er auf die White- oder Blacklist gesetzt werden kann. Zudem soll eine simple Webseite zum verwalten der Listen erstellt werden.

Als zusätzliches Ziel soll die Arbeit dokumentiert und ein Bericht erstellt werden.

1.3 Abgrenzungen

Diese Arbeit entsteht im Rahmen des Moduls „Bachelorarbeit“ der OST. Dadurch ist sie sowohl zeitlich als auch im Budget, hier Arbeitsstunden, begrenzt. Die BA beginnt am 21.02.2022 und endet am 17.06.2022 um 17:00 Uhr mit der Abgabe des Berichtes. Für das Modul sind 12 ECTS vorgesehen. Somit ergeben sich bei einem Zeitaufwand von 30 Stunden pro ECTS, 360 Arbeitsstunden für jeden Studenten.

In Anbetracht des Scopes gibt es ebenfalls Einschränkungen, da der volle Umfang des Projektes sowohl die Zeit als auch das Budget sprengen würde. Daher wurde der Funktionsumfang, zusammen mit dem Industriepartner und dem Betreuer, in verpflichtende und optionale Funktionen unterteilt. Die Erfüllung der verpflichtenden Funktionalität ist ausschlaggebend für den Projekterfolg. An den optionalen Funktionen wird gearbeitet, falls noch Zeit übrig bleibt.

2 Analyse

Im Teil Analyse beschäftigen wir uns mit Aufgaben, die vor dem Beginn der produktiven Tätigkeit abgeschlossen werden müssen. Hierzu gehen wir zunächst auf die Anforderungen ein, die zusammen mit dem Industriepartner erarbeitet wurden und letztendlich den Projekterfolg bestimmen. Das Kapitel schliessen wir danach mit unserem Projektplan ab.

2.1 Anforderungen

Dieser Abschnitt befasst sich mit den Anforderungen, welche die Applikation erfüllen muss. Die Anforderungen werden zunächst in funktionale und nicht funktionale unterteilt. Weiterhin unterscheiden wir bei den funktionalen Anforderungen zwischen verpflichtenden und optionalen, da der Umfang dieser zu gross ist, um alle in einer einzelnen BA zu behandeln.

2.1.1 Use Cases

Für die funktionalen Anforderungen, die der Kunde an die Applikation stellt, haben wir UCs erstellt. Abbildung 2.1 zeigt eine Übersicht dieser und welche Aktoren an diesen beteiligt sind.

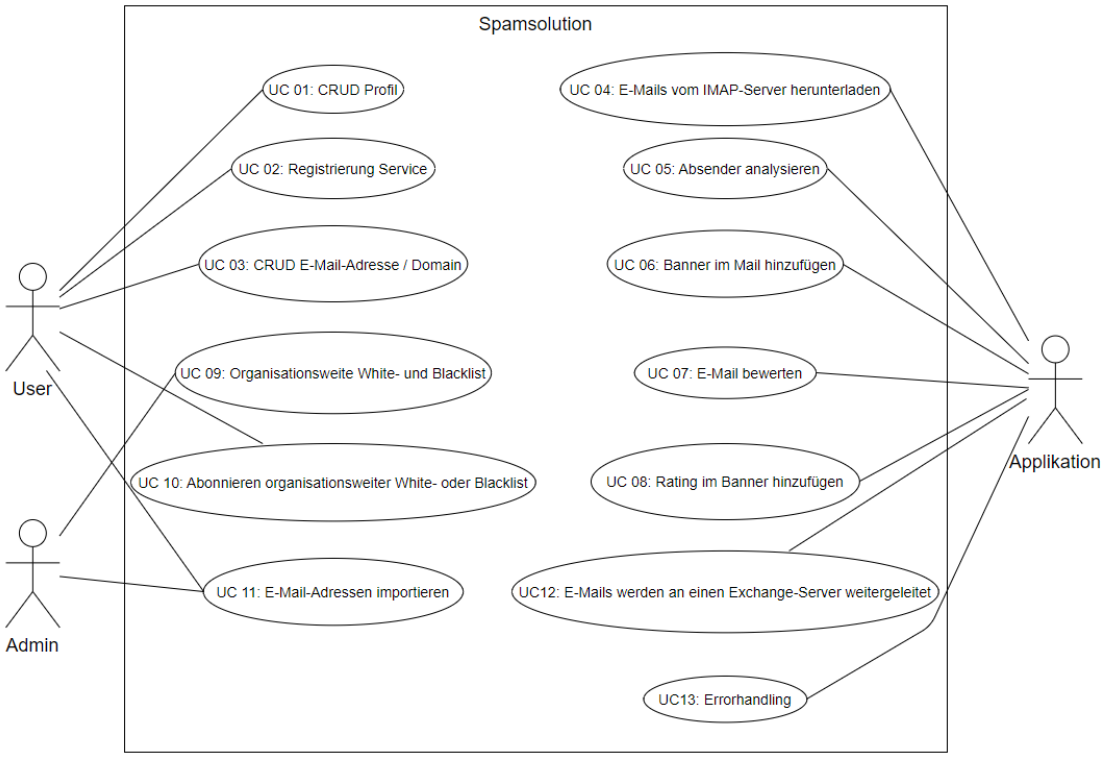


Abbildung 2.1: Übersicht der Use Cases

Rollen

Bevor wir die UCs auflisten geben wir noch einen kurzen Überblick über die Aktoren. In unserem System unterscheiden wir zwischen den drei Rollen Applikation, Admin und User.

Applikation

Viele Aufgaben werden im Hintergrund ausgeführt, ohne dass der User etwas davon merkt. Daher verwenden wir die Applikation als eigenständigen Aktor.

User

Der User ist eine Person, der auf Spamsolution registriert ist und seine White- und Blacklist dort verwaltet.

Admin

Der Admin verwaltet White- und Blacklisten, die User abonnieren können, um nicht selbstständig eine Liste verwalten oder ihre persönliche erweitern zu müssen.

UC 01: CRUD Profil

Main Actor	User
Main Success Scenario	Der User kann seine persönlichen Angaben wie Name, Vorname, E-Mail, IMAP-Adresse anpassen.

Tabelle 2.1: UC 01

UC 02: Registrierung Service

Main Actor	User
Main Success Scenario	Der User kann sich für den Service registrieren.

Tabelle 2.2: UC 02

UC 03: CRUD E-Mail-Adresse / Domain

Main Actor	User
Main Success Scenario	Der User kann eine E-Mail-Adresse bzw. eine komplette Domain zu seiner White- oder Blacklist hinzufügen. Weiterhin kann er diese wieder entfernen oder zwischen den Listen verschieben.

Tabelle 2.3: UC 03

UC 04: E-Mails vom IMAP-Server herunterladen

Main Actor	Applikation
Main Success Scenario	Die Applikation holt anhand der Angaben von den Usern E-Mails von IMAP-Servern.

Tabelle 2.4: UC 04

UC 05: Absender analysieren

Main Actor	Applikation
Main Success Scenario	Die Applikation prüft ob der Absender auf der White- oder Blacklist des Users ist und verschiebt sie dementsprechend.

Tabelle 2.5: UC 05

UC 06: Banner im Mail hinzufügen

Main Actor	Applikation
Preconditions	UC 05 ist abgeschlossen.
Main Success Scenario	Falls der Absender der E-Mail unbekannt ist, fügt die Applikation zuoberst einen Banner hinzu, der Buttons zur Verfügung stellt, um den Absender zur White- oder Blacklist hinzuzufügen.

Tabelle 2.6: UC 06

UC 07: E-Mail bewerten (Optional)

Main Actor	Applikation
Preconditions	UC 05 ist abgeschlossen.
Main Success Scenario	Bei einem unbekannten Absender wird der Inhalt der E-Mail geprüft und mit einem Rating versehen, das die Vertrauenswürdigkeit widerspiegelt (0 = nicht vertrauenswürdig / 100 = vertrauenswürdig).

Tabelle 2.7: UC 07

UC 08: Rating im Banner hinzufügen (Optional)

Main Actor	Applikation
Preconditions	UC 06 ist abgeschlossen.
Main Success Scenario	Die Applikation fügt zusätzlich zu den in UC 06 beschriebenen Buttons das Rating zum Banner hinzu.

Tabelle 2.8: UC 08

UC 09: Organisationsweite White- und Blacklist (Optional)

Main Actor	Admin
Main Success Scenario	Ein Admin kann eine organisationsweite White- und Blacklist verwalten und den Usern zur Verfügung stellen.

Tabelle 2.9: UC 09

UC 10: Abonnieren organisationsweiter White- oder Blacklist (Optional)

Main Actor	User
Preconditions	UC 09 ist abgeschlossen.
Main Success Scenario	Ein User kann sich bei einer Organisationsweiten White- oder Blacklist anmelden bzw. sich wieder abmelden.

Tabelle 2.10: UC 10

UC 11: E-Mail-Adressen importieren (Optional)

Main Actor	Admin & User
Main Success Scenario	Ein User kann mithilfe eines Outlook Plugins E-Mail-Adressen exportieren an die er Mails verschickt hat und diese dann importieren.

Tabelle 2.11: UC 11

UC 12: E-Mails werden an einen Exchange-Server weitergeleitet (Optional)

Main Actor	Applikation
Main Success Scenario	Die E-Mails werden von der Applikation an einen Exchange-Server weitergeleitet.

Tabelle 2.12: UC 12

UC 13: Errorhandling

Main Actor	Applikation
Main Success Scenario	Bei einem Error setzt die Applikation das System auf einen vorherigen funktionierenden Stand zurück und loggt den Error.

Tabelle 2.13: UC 13

2.1.2 Nichtfunktionale Anforderungen

Nachfolgend listen wir die nichtfunktionalen Anforderungen (NFAs) auf. Die NFAs wurden vom Industriepartner bereits im Voraus definiert und von uns akzeptiert. Für die bessere Lesbarkeit und Verständlichkeit haben wir die NFAs nach dem FURPS Modell kategorisiert. Die NFAs wurden aus der Aufgabenstellung übernommen.

Functionality

- Die Web-Applikation soll auf Firefox, Chrome und Safari laufen.
- Die Datenbank soll bis zu 10'000 Mail-Adressen managen können.
- Via Internet soll auf eine vom Kunden bereitgestellte Domain zugegriffen werden können.
- User-Passwörter werden nicht in plain-text in der Datenbank gespeichert.
- Wenn sich ein User in die Web-Applikation einloggt, werden ihm nur Daten auf die er Zugriff haben soll angezeigt.
- Jede Kommunikation zwischen Front- und Backend soll mit einem SSL-Zertifikat verschlüsselt werden.

Usability

- Drei von vier Testusern sollen das User Interface (UI) (Kategorien: Layout, Responsiveness, Colour, Content) der Applikation mit einer Note von mindestens 7 von 10 bewerten, wobei 10 das Beste ist.

Reliability

- Die Backend-API soll durch api-testing Tools getestet werden.
- Daten, welche in Eingabefelder abgefüllt werden, sollen zuerst validiert werden, bevor diese durch das System verarbeitet werden. SQL Injection Test der Eingabefelder sollte keine Verletzlichkeiten zeigen.

Performance

- Das Backend (Mail-Adressen-Verwaltung) soll 100 Requests pro Minute handeln können.
- Das Scannen einer Mail soll maximal zwei Sekunden dauern.
- Jede Seite sollte nicht länger als 200ms für das Laden benötigen.

Supportability

- Businesslogik im Backend soll modular aufgebaut werden, sodass sie erweitert werden kann.

2.2 Projektplan

In diesem Abschnitt definieren wir zunächst einen Zeitplan, anhand dem wir unsere Arbeiten ausrichten. Ausserdem machen wir eine erste Risikoanalyse, um uns auf eventuelle Verzögerungen vorbereiten zu können. Den Abschnitt beenden wir schliesslich damit, dass wir Qualitätsmassnahmen definieren, um ein hochwertiges Produkt entwickeln zu können.

2.2.1 Zeitplan

Das Projekt wird gemäss dem Scrum+ organisiert. Hierfür unterteilen wir zunächst das Projekt in Phasen und legen Meilensteine fest.

Phasen

Das Projekt ist gemäss RUP in vier Phasen unterteilt, die jeweils abgeschlossen werden müssen, bevor man zu der nächsten übergehen kann.

Inception

Die Inception Phase läuft vom 23.02.2022 bis zum 08.03.2022. In dieser Phase machen wir uns mit der Aufgabenstellung vertraut und planen das weitere Vorgehen im Projekt. Weiterhin richten wir unsere Arbeitsumgebung ein, damit beim Start der Construction Phase keine Zeit verloren geht.

Elaboration

Die Elaboration Phase dauert vom 09.03.2022 bis zum 05.04.2022. In dieser Phase konkretisieren wir die Anforderungsspezifikation, indem wir UCs und Wireframes erstellen. Als Vorbereitung für die Construction Phase wird die Architektur, sowie das Domain- und Datenmodell erarbeitet.

Construction

Die Construction Phase läuft vom 06.04.2022 bis zum 31.05.2022. Während der Construction Phase wird die Applikation der BA entwickelt.

Transition

Die Transition Phase dauert vom 01.06.2022 bis zum 17.06.2022. In dieser Phase werden keine neuen Funktionen zur Applikation hinzugefügt. Es wird lediglich ein Feinschliff durchgeführt, indem letzte Bugs beseitigt werden und ein letztes Code Refactoring durchgeführt wird, damit das Produkt sauber an den Kunden übergeben werden kann. Ausserdem soll die Dokumentation fertiggestellt und für die Abgabe vorbereitet werden.

Meilensteine

Innerhalb der Phasen wird mit Iterationen von einer Dauer von zwei Wochen gearbeitet. Meilensteine werden nur zum Ende einer Iteration terminiert und stellen einen wichtigen Schritt im Projekt dar.

M1 End of Inception

Datum: 08.03.2022

Das Ziel dieses Meilensteins ist es die Aufgabenstellung verstanden zu haben und den Projektplan zu erstellen.

Arbeitspakete

- Projektplan

M2 End of Elaboration

Datum: 05.04.2022

Zur End of Elaboration sollen die Requirements klar definiert und ein Prototyp der Architektur erstellt sein.

Arbeitspakete

- Use Cases
- Wireframes
- Architekturentwurf
- Domain- und Datenmodell
- Prototyp der Architektur

M3 Alpha Release (MVP)

Datum: 03.05.2022

Mit dem Alpha Release wollen wir das Minimal Viable Product (MVP) fertigstellen und demonstrieren.

Arbeitspakete

- MVP
- Demonstration der Funktionalität an einem UC

M4 End of Construction

Datum: 31.05.2022

Mit diesem Meilenstein sollen die Arbeiten an neuen Funktionalitäten eingestellt werden.

Arbeitspakete

- Fertige Applikation gemäss Anforderungsspezifikation

M5 Abstract Submission

Datum: 15.06.2022

Mit diesem Meilenstein wird das Abstract im Abgabetool abgegeben.

Arbeitspakete

- Abstract

M6 Final Submission

Datum: 17.06.2022

Mit Abgabe des Projektberichtes wird das Projekt abgeschlossen.

Arbeitspakete

- Projektbericht

2.2.2 Risikoanalyse

Im folgendem Abschnitt werden von uns evaluierte Risiken, die im Verlauf des Projektes auftreten könnten, aufgelistet. Die Risiken werden über die ganze Projektdauer immer wieder neu bewertet. Die Neuauswertungen sind im Anhang zu finden.

R01 - Zeitmanagement

Der Zeitaufwand wurde unterschätzt.

max. Schaden [h]: 40
Eintrittswahrscheinlichkeit: 20%
Gewichteter Schaden: 8
Vorbeugung: Einen sauberen Projektplan erstellen.
Verhalten beim Eintreten: Überstunden machen, Abstriche bei der Funktionalität.

R02 - Komplexität

Das Projekt wird komplexer als gedacht. Während der Entwicklung tauchen unerwartet Herausforderungen auf.

max. Schaden [h]: 50
Eintrittswahrscheinlichkeit: 15%
Gewichteter Schaden: 7.5
Vorbeugung: Richtige Architektur wählen und mit einer gründlichen Vorbereitung alle Eventualitäten ausmerzen.
Verhalten beim Eintreten: Weiteres Vorgehen mit dem Betreuer und Industriepartner besprechen. Hier kommt eine Scope Reduktion in Frage.

R03 - Fehlendes Know-How

Mit der Programmiersprache Python und der Datenbank MySQL haben wir bis zu diesem Zeitpunkt nie an grösseren Projekten gearbeitet. Hier könnte bei komplexeren Arbeiten mehr Zeit erforderlich sein, um das Problem zu lösen.

max. Schaden [h]: 8
Eintrittswahrscheinlichkeit: 5%
Gewichteter Schaden: 0.4
Vorbeugung: Die Konzepte von Programmiersprachen und relationalen Datenbanken sind bekannt. Mit einer Einarbeitung in den Technologien lässt sich das Risiko stark minimieren.
Verhalten beim Eintreten: Hilfe bei Experten holen.

R04 - Ausfall eines Teammitglied

Ein Teammitglied fällt krankheitsbedingt aus.

max. Schaden [h]: 32
Eintrittswahrscheinlichkeit: 10%
Gewichteter Schaden: 3.2
Vorbeugung: Gute Kommunikation innerhalb vom Team
Verhalten beim Eintreten: Die Teammitglieder tauschen sich untereinander aus und arbeiten an dem weiter, was Priorität hat. Bei einer längeren Abwesenheit kommt eine Scope Reduktion in Frage.

R05 - Ausfall der Infrastruktur

Die Infrastruktur fällt aus und behindert das Weiterarbeiten am Projekt.

max. Schaden [h]: 4
Eintrittswahrscheinlichkeit: 1%
Gewichteter Schaden: 0.04
Vorbeugung: Keine Vorbeugung möglich
Verhalten beim Eintreten: Offline weiterarbeiten

R06 - Probleme beim Deployment

Es wird uns eine Infrastruktur zur Verfügung gestellt, auf die wir keinen Einfluss haben. Dadurch kann es zu Problemen kommen.

max. Schaden [h]: 8
Eintrittswahrscheinlichkeit: 1%
Gewichteter Schaden: 0.08
Vorbeugung: Der Industriepartner wird uns zeitnah die Infrastruktur zur Verfügung stellen, damit wir uns frühzeitig mit der Umgebung vertraut machen können.
Verhalten beim Eintreten: Hilfe beim Industriepartner anfragen.

R07 - Komplikationen Mail Abfragen

Unerwartete Komplikationen, Mails von gewissen Domains abzurufen, weil z.B ein Token / eine Authentifzierung nötig ist.

max. Schaden [h]: 16
Eintrittswahrscheinlichkeit: 15%
Gewichteter Schaden: 2.4
Vorbeugung: Die gängigsten E-Mail Anbieter darauf prüfen, ob ein extra Schritt benötigt wird.
Verhalten beim Eintreten: Abwägen, wie wichtig dieser E-Mail Anbieter für den Industriepartner ist und fragen, ob Zusatzaufwand gewünscht ist.

R08 - Ausfall von der Hardware

Die Hardware eines Teammitglieds fällt aus.

max. Schaden [h]: 7
Eintrittswahrscheinlichkeit: 5%
Gewichteter Schaden: 0.35
Vorbeugung: Keine Vorbeugung möglich.
Verhalten beim Eintreten: Ein neues Gerät besorgen und die Tools beim Neuen einrichten.

R09 - Missverständnis in der Kommunikation

Die Anforderungen werden vom Team falsch interpretiert, was eine Abweichung des Endprodukts bedeuten würde und ein nicht zufriedenstellendes Produkt entsteht.

max. Schaden [h]: 80
Eintrittswahrscheinlichkeit: 5%
Gewichteter Schaden: 4
Vorbeugung: Wöchentliche Meetings und Statusupdates über den Fortschritt mit dem Industriepartner und klare Abholung der Anforderungen.
Verhalten beim Eintreten: Lagebestimmung in einer Sondersitzung mit dem Industriepartner und das weitere Vorgehen besprechen. Eventuell eine Reduktion vom Scope.

Erkenntnisse

Der gewichtete Schaden beträgt ca. 26 Stunden. Das bedeutet, dass wir mit einer Verzögerung des Projekts von 26 Stunden rechnen können. Durch vorbeugende Massnahmen unsererseits sind wir zuversichtlich, dass der Schaden weiter reduziert wird und sich nicht negativ auf den Projekterfolg auswirken wird.

2.2.3 Qualitätsmassnahmen

Um die Qualität der Software, als auch die der Dokumentation, hoch zu halten, werden entsprechende Massnahmen ergriffen. Einen Überblick dieser liefert Tabelle 2.14.

Massnahme	**Ziel**
Unit Tests	Mit Unit Tests wollen wir sicherstellen, dass Methoden, die Logik enthalten, richtig funktionieren.
Code Reviews	Bevor eine Story abgeschlossen wird soll ein Code Review vom anderen Teammitglied durchgeführt werden, um sicherzustellen, dass der Code leicht verständlich ist.
Refactoring	Refactorings dienen dazu Probleme, die bei einem Code Review festgestellt wurden zu beheben.
System Tests	Bei den System Tests werden die UCs per Hand durchgespielt um sicherzustellen, dass die Software wie Vorgesehen funktioniert.
Continuous Integration	Continuous Integration sorgt dafür, dass Tests nicht übersprungen werden können, wodurch wir gewährleisten können, dass die Applikation immer in einem funktionierenden Zustand ist.
Usability Test	Ein Usability Test soll sicherstellen, dass die Applikation intuitiv verwendbar ist und den Wünschen der Endbenutzer entspricht.
Meetings	Regelmässige Meetings mit dem Industriepartner und dem Betreuer dienen dazu den aktuellen Stand des Projektes zu besprechen, damit sich alle versichern können, dass das Projekt auf dem richtigen Weg ist.
Dokumentationsreview	Bevor Änderungen der Dokumentation in den Master-Branch gemerged werden, werden diese in einem Meeting gemeinsam besprochen.

Tabelle 2.14: Qualitätsmassnahmen

Projektmanagement

Für das Projektmanagement verwenden wir Jira. In diesem Tool werden die Arbeitspakete je nach Art als Story oder Task erfasst und der zeitliche Aufwand geschätzt. Die tatsächlich gearbeitete Zeit wird dann direkt auf dem Arbeitspaket erfasst. Gehören mehrere Tasks und Storys logisch zusammen, werden diese in einem Epic zusammengefasst. Ein Scrum-Board in der Jira-Applikation ermöglicht es uns den Überblick über den Status der aktuellen Arbeitspakete zu behalten.

Dokumentation

Für die Dokumentation wurde ein eigenes Repository auf GitLab erstellt. Um die Qualität hoch zu halten, führen wir vor einem Update einen gemeinsamen Review der geplanten Änderungen durch. Ausserdem verwenden wir LateX, wodurch wir die einzelnen Kapitel in gesonderten Dateien abspeichern können, was uns die Reviews erleichtert. Weiterhin lassen sich so Kapitel leicht im Bericht verschieben und es wird eine einheitliche Formatierung gewährleistet.

Entwicklung

Der Code wird, wie die Dokumentation, auf einem eigenen Repository in GitLab versioniert. Hierfür wird eine Pipeline erstellt, die unsere Testcases ausführt. Weiterhin deployed die Pipeline bei Änderung der Master-Branch die Applikation automatisch auf den Test-Server.

Workflow

Unseren standardmässigen Workflow kann man der Abbildung 2.2 entnehmen. Dabei ist zu beachten, dass es möglich ist, von jedem Status ein Arbeitspaket auf den Status „Done“ zu setzen. Dies ist nötig, da wir auch Meetings als Arbeitspaket erfassen und diese keinen Review erfordern.

Abbildung 2.2: Workflow

Branches

Da wir eine völlig neue Software entwickeln und sie noch nirgendwo in Betrieb ist, werden wir mit einem Master-Branch arbeiten und für die einzelnen Storys bzw. Tasks eigene Branches erstellen. Sobald dann ein Arbeitspaket abgeschlossen ist, wird es mittels eines Merge Requests in den Master-Branch integriert. Wenn die Applikation später in Betrieb genommen wird, ist es zu empfehlen für die weitere Entwicklung einen Development-Branch einzurichten.

Namenskonvention Branch
Ein Branch wird anhand des Vorgangs in Jira benannt.
Das Muster ist **„BS-<Vorgangsnummer>-<Vorgangsbeschreibung>“** z.B. **BS-12-Qualitätsmassnahmen-definieren**.

Definition of Done

Damit ein Arbeitspaket als abgeschlossen betrachtet wird, müssen folgende Kriterien erfüllt sein:

- Funktionalität gemäss Beschreibung des Arbeitspaketes ist vollständig umgesetzt.
- Alle Tests sind erfolgreich (Pipeline läuft durch).
- NFAs sind erfüllt.
- Der Code wurde vom anderen Teammitglied reviewed.

3 Design

Im Kapitel Design geht es um die Arbeiten, die während der Elaboration Phase erledigt werden müssen. Zu Beginn stellen wir die Architektur des Produktes vor und gehen danach über zu den verwendeten Technologien. Daraufhin widmen wir uns Modellen, die wir als Vorbereitung für die Entwicklung erstellt haben. Zum Schluss gehen wir noch auf Wireframes ein, die visualisieren, wie wir uns das Endprodukt vorstellen.

3.1 Architektur

Bei der Architektur haben wir basierend auf dem Vorschlag des Industriepartners ein Deployment-Diagramm entworfen (Abbildung 3.1). Der ursprüngliche Vorschlag ist im Anhang zu finden.

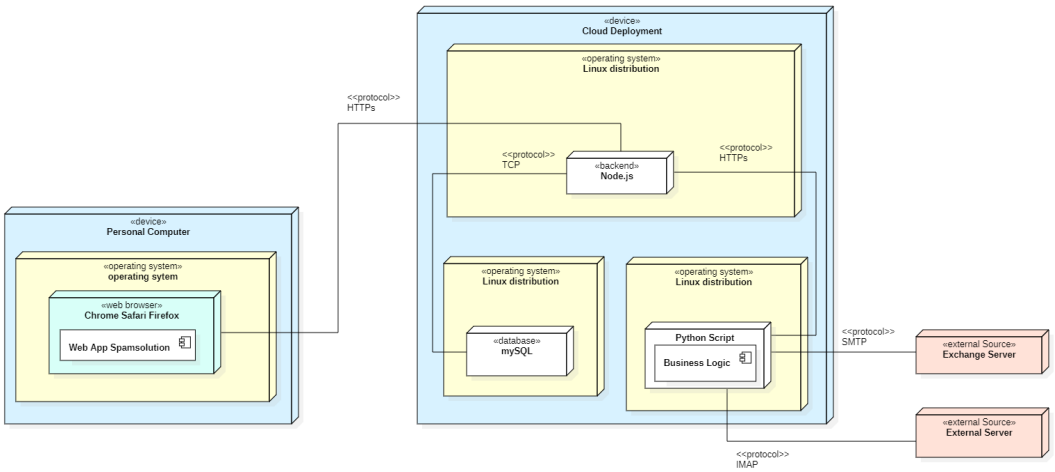


Abbildung 3.1: Architekturentwurf

3.1.1 Komponenten

Zum besseren Verständnis des Diagramms werden die Komponenten noch kurz in der Tabelle 3.1 näher beschrieben.

Komponente	Beschreibung
Web-App Spamsolution	Ist die Schnittstelle des Users zum System. Über die Web-App kann sich ein User für den Service registrieren und seine White- und Blacklist verwalten. Ausserdem kann er bei Bedarf seine Einstellungen bezüglich der IMAP-Daten ändern.
Node.js	Node.js[4] ist das Backend der Applikation. Es ist das Verbindungsglied für die anderen Komponenten und jegliche Kommunikation läuft über dieses. Dem User stellt es die Web-App zur Verfügung und dem Python-Script liefert es die benötigten Daten um seine Funktionalität ausführen zu können.
MySQL	MySQL[5] ist die verwendete Datenbank und zuständig für das Persistieren der Daten.
Python-Script	Das Python-Script ist der eigentliche Service den Spam Solution anbietet. Es verbindet sich über IMAP mit den Mail-Servern der User und bearbeitet die neuen E-Mails gemäss der persönlichen Einstellungen.

Tabelle 3.1: Komponenten

3.2 Technologien

In diesem Abschnitt gehen wir auf verwendete Libraries und Frameworks ein. Dabei werden nur die wichtigsten näher erläutert. Eine Auflistung aller verwendeten Libraries und Module ist im Anhang zu finden.

3.2.1 Frontend

Für die Erstellung des Frontends waren wir auf der Suche nach einer auf JavaScript basierenden Library. Die Bibliothek soll simpel sein und eine hohe Performance aufweisen. Weiterhin soll sie weitverbreitet sein, da wir nur das Grundgerüst für die Anwendung legen und diese später von einem anderen Team weiterentwickelt wird. Deswegen fiel unsere Entscheidung auf React.[6]

React

React ist eine Javascript Bibliothek, welche von Meta Platforms entwickelt wird. Die Bibliothek ermöglicht es 3rd Party Libraries zu verwenden, wodurch wir nicht unabhängig von React bleiben. Ein auf Komponenten basierendes System erlaubt es auch komplexere UIs zu erstellen und gewährleistet eine hohe Skalierbarkeit der Seite. Client-side Rendering nimmt Last vom Server und macht es dadurch möglich mehr Anfragen zu bearbeiten.

Bootstrap

Für die Gestaltung der Elemente auf der Webseite haben wir uns für das CSS-Framework Bootstrap[7] entschieden. Bootstrap ist ein weitverbreitetes Framework und uns bereits aus vorherigen Projekten bekannt, wodurch wir keine Einarbeitungszeit benötigen. Es ist eine unkomplizierte Lösung, um Elemente schnell und effizient zu gestalten.

3.2.2 Backend

Das Backend bildet die Schnittstelle zwischen den anderen Komponenten. Es stellt für den User das Frontend zur Verfügung und für die anderen Komponenten eine api, womit sie alle benötigten Daten aufrufen können. Dadurch nimmt die Komplexität der Software ab und die Erweiterbarkeit wird erhöht, da Änderungen auf den Datenbankzugriff nur an einer Stelle gemacht werden müssen. Um das Backend zu realisieren haben wir als Grundgerüst Node.js genommen und darauf aufbauend Express.js[8] als Server-Framework. Da wir bisher alle Projekte an der OST im Backend so gehandhabt haben, sind wir bestens damit vertraut und können somit den Kern der Anwendung schneller entwickeln, um mehr Zeit für die anderen Komponenten zu haben.

Node.js

Node.js ist eine Open Source Runtime Umgebung, die es einem ermöglicht serverseitigen Code in JavaScript zu schreiben. Dadurch wird unsere Arbeit vereinfacht, da wir uns nicht noch mit einer dritten Programmiersprache auseinandersetzen müssen. Weiterhin gibt es sehr viele Open Source Packages, die wir verwenden können.[9]

Express.js

Express.js erlaubt uns einfach und schnell einen lauffähigen Server zu erstellen. Durch die Verwendung von Routern und Middleware ist es möglich strukturierten und gut leserlichen Code zu schreiben.

3.2.3 Datenbank

Die Daten werden mittels einer MySQL-Datenbank persistiert. Wir haben uns für eine MySQL Datenbank entschieden, weil der Industriepartner bei der Vorstellung der Arbeit diese vorgeschlagen hat. Wir haben zwar noch nie mit MySQL gearbeitet, jedoch sind wir mit anderen relationalen Datenbanken bewandert, wodurch die Einarbeitungszeit gering ausfällt.

3.2.4 Python-Script

Der eigentliche Service von der Applikation wird von einem Python-Script ausgeführt. Wie MySQL war Python eine Empfehlung des Industriepartners. Da auch wir Python für eine gute Wahl halten und wir keine negativen Aspekte in der Verwendung sahen, haben wir uns für diese Programmiersprache entschieden.

3.3 Domain-Model

Um die Problem-Domain verstehen zu können, wurde ein erstes Domain-Model erstellt, das alle benötigten Geschäftsobjekte darstellt. Dabei stellte sich heraus, dass die Problem-Domain eigentlich aus zwei Sub-Domains besteht. Deshalb haben wir uns dazu entschlossen, die Problem-Domain in die zwei Sub-Domains Web und App zu untergliedern. Nachfolgend sind die beiden Modelle zu finden (Abbildung 3.2 und 3.3), wobei wir ausgewählte Objekte näher erläutern.

3.3.1 Web-Domain

Die Web-Domain bildet die Interaktion der User mit dem System ab.

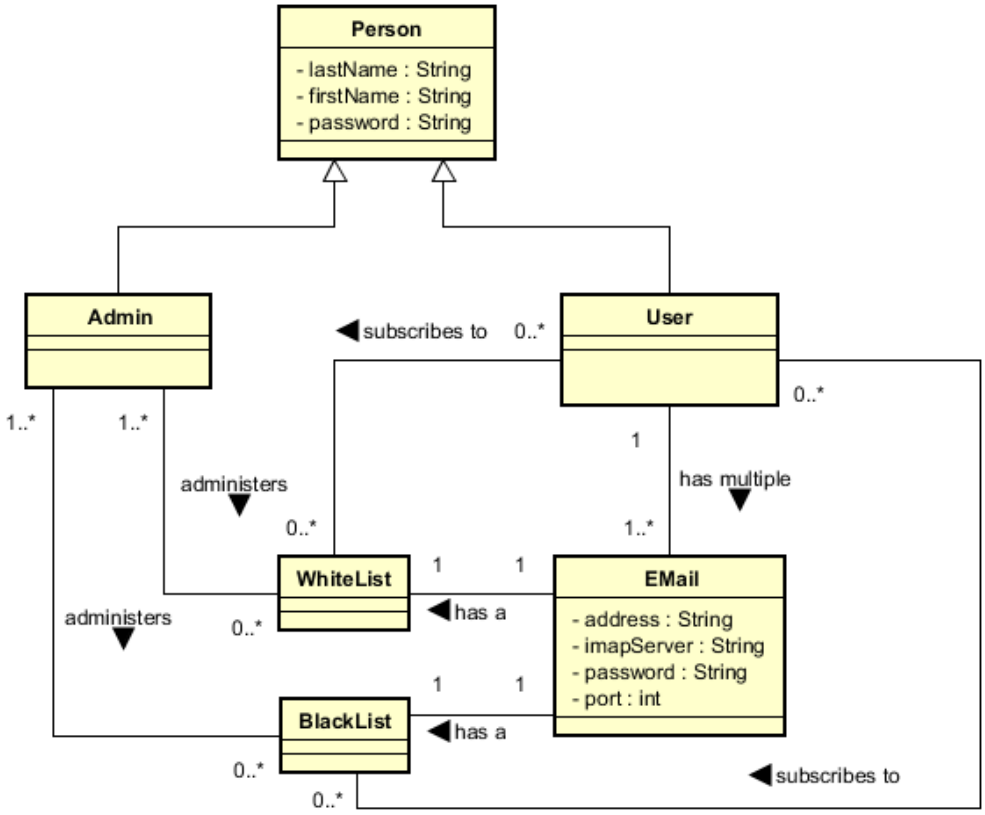


Abbildung 3.2: Web-Domain

Admin / User

Wir unterscheiden zwischen zwei unterschiedlichen Anwendergruppen. Die User sind diejenigen, die den Service für sich beanspruchen und ihre eigenen Listen verwalten. Admins nutzen den Service nicht, sondern verwalten lediglich organisationsweite Listen, die User abonnieren können. Abgesehen von der Funktionalität unterscheiden sich die Accounts darin, dass User zusätzlich ihre IMAP-Daten angeben müssen.

E-Mail

Bei der E-Mail ist zu beachten, dass das Passwort nicht dasselbe ist, wie das Login-Passwort. Um den Service nutzen zu können wird hier das Passwort für den E-Mail Account bzw. ein App-Passwort benötigt.

Whitelist / Blacklist

Bei den beiden Listen wird zwischen zwei verschiedenen Arten unterschieden. Jeder User hat eine persönliche, die nur für ihn gilt und auch nur von ihm verwaltet werden kann. Ausserdem gibt es organisationsweite Listen, die jederzeit von einem User abonniert bzw. deabonniert werden können.

3.3.2 App-Domain

Die App-Domain bildet ab, was im Hintergrund stattfindet und für den User nicht ersichtlich ist.

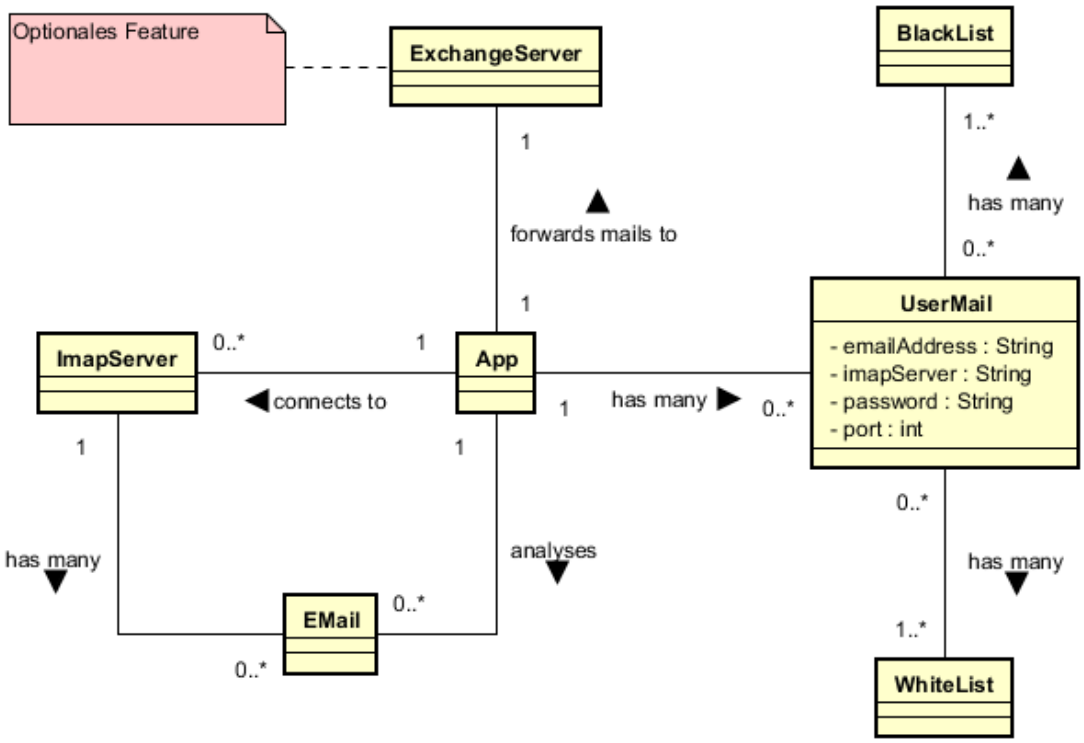


Abbildung 3.3: App-Domain

IMAP-Server / Exchange-Server

Bei den beiden Servern ist zu beachten, dass sie nicht von uns verwaltet werden. Die Applikation stellt lediglich eine Verbindung zu diesen her, um E-Mails herunterladen bzw. E-Mails weiterleiten zu können.

3.4 Data-Model

Für die Persistierung der Daten ist eine Datenbank im Einsatz. Dabei richten wir uns nach dem Vorschlag des Industriepartners und verwenden MySQL. Das Data-Model kann der Abbildung 3.4 entnommen werden.

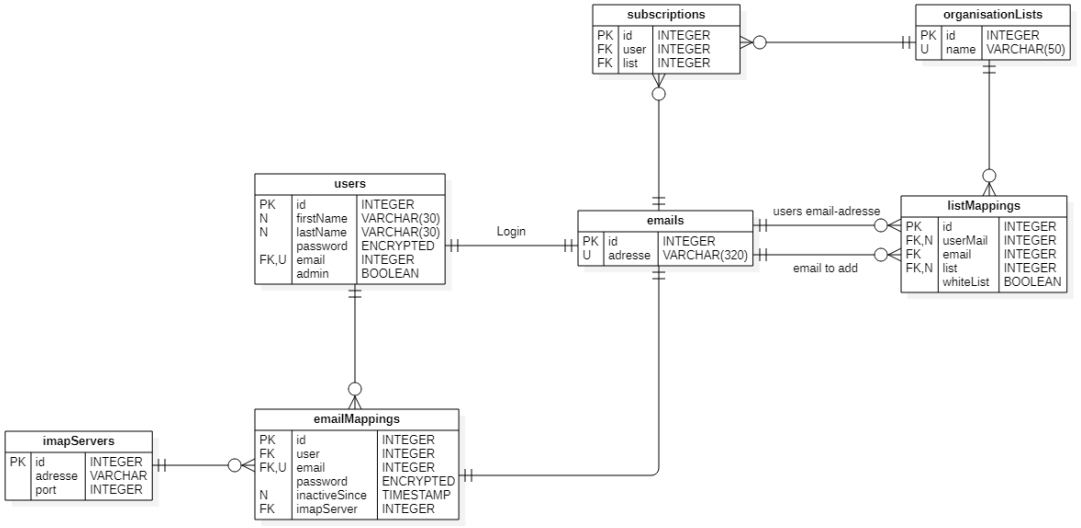


Abbildung 3.4: Data-Model

Das Modell ist weitestgehend selbsterklärend. Wir gehen hier nur noch kurz auf die angeschriebenen Beziehungen ein, da diese unklar sein könnten.

Login

Der User verwendet zum einloggen seine E-Mail-Adresse, mit der er sich für den Service registriert hat. Daher wird hier zusätzlich zu den emailMappings noch eine weitere Beziehung benötigt.

Users Email-Adresse / Email to add

Hier gibt es zwei Beziehungen zwischen den Tabellen emails und listMappings. Eine Beziehung ist für die E-Mails, welche geblockt oder zugelassen werden sollen. Die andere Beziehung ist nötig, da die Listen an eine E-Mail-Adresse des Users gekoppelt sind, um es ihm zu ermöglichen, unterschiedliche Listen für z.B. Geschäfts-E-Mail und private E-Mail zu verwenden.

3.5 Wireframes

Ein wichtiger Aspekt der Arbeit ist es ein Frontend mit einem ansprechenden UI zu erstellen. Um bereits frühzeitig sicherzugehen, dass unsere Vision die festgelegten Anforderungen abbildet und den Erwartungen des Industriepartners entspricht, haben wir Wireframes angefertigt. Diese stellen einen ersten groben Entwurf da, an dem wir uns in der weiteren Entwicklung orientieren können. Es ist wichtig zu beachten, dass sich das Endprodukt von den Wireframes unterscheiden kann.

Die wichtigsten Wireframes und deren Funktionen werden nachfolgend erläutert. Eine vollständige Liste der Wireframes ist im Anhang zu finden.

3.5.1 Login

Das Anmeldefenster wird schlicht gehalten. Der Benutzer meldet sich mit seiner E-Mail-Adresse und dem Passwort von Spam Solution an. Will ein neuer Benutzer einen Account für sich anlegen, kann er das machen indem er auf den Knopf „Registrieren“ klickt. Dadurch wird er zu der Registrierungsseite weitergeleitet.

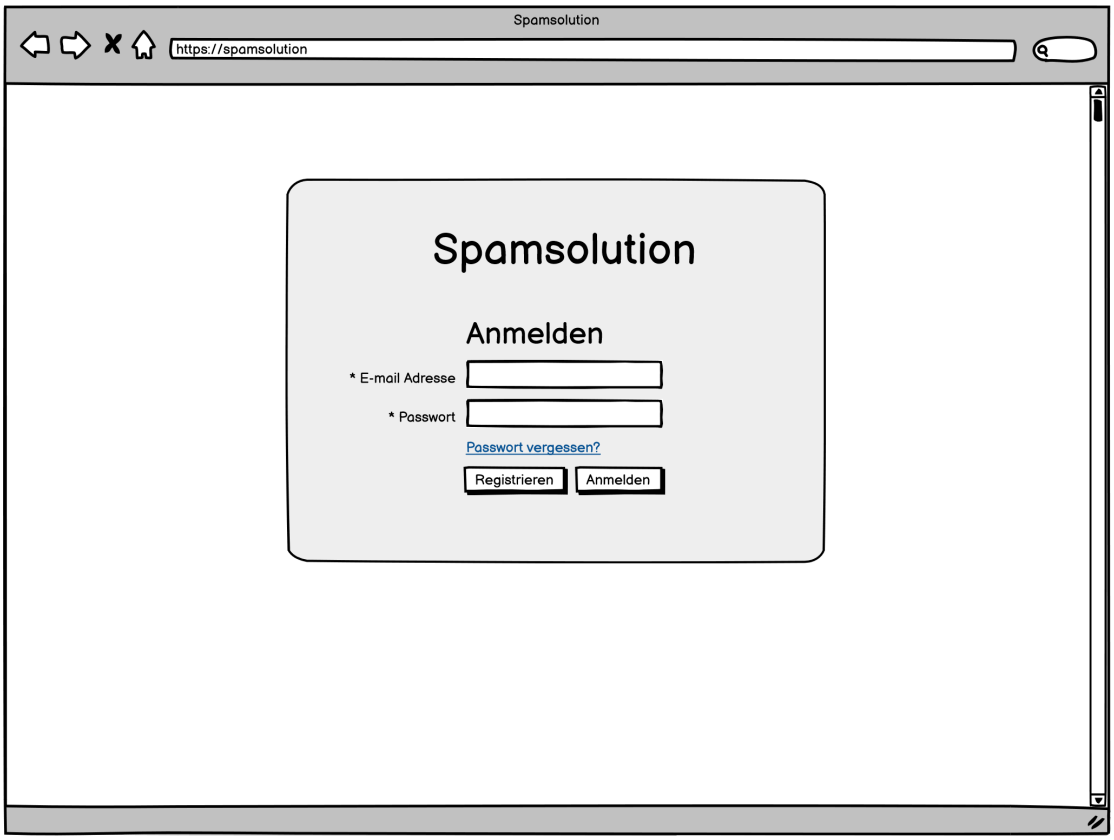


Abbildung 3.5: Anmeldefenster

3.5.2 Registrierung

Auf dieser Seite werden die benötigten Daten für einen neuen Benutzer erfasst und ein Account für diesen erstellt. Die meisten Felder sind selbsterklärend, jedoch gibt es bei Feldern technischer Natur eine Hilfestellung für den User. Sobald alle Daten erfasst sind und man auf „Registrieren “ klickt, werden die Daten geprüft und, falls die E-Mail-Adresse nicht bereits verwendet wird, ein neuer Account angelegt. Daraufhin wird der Benutzer automatisch angemeldet und auf seinen „Homescreen“ weitergeleitet.

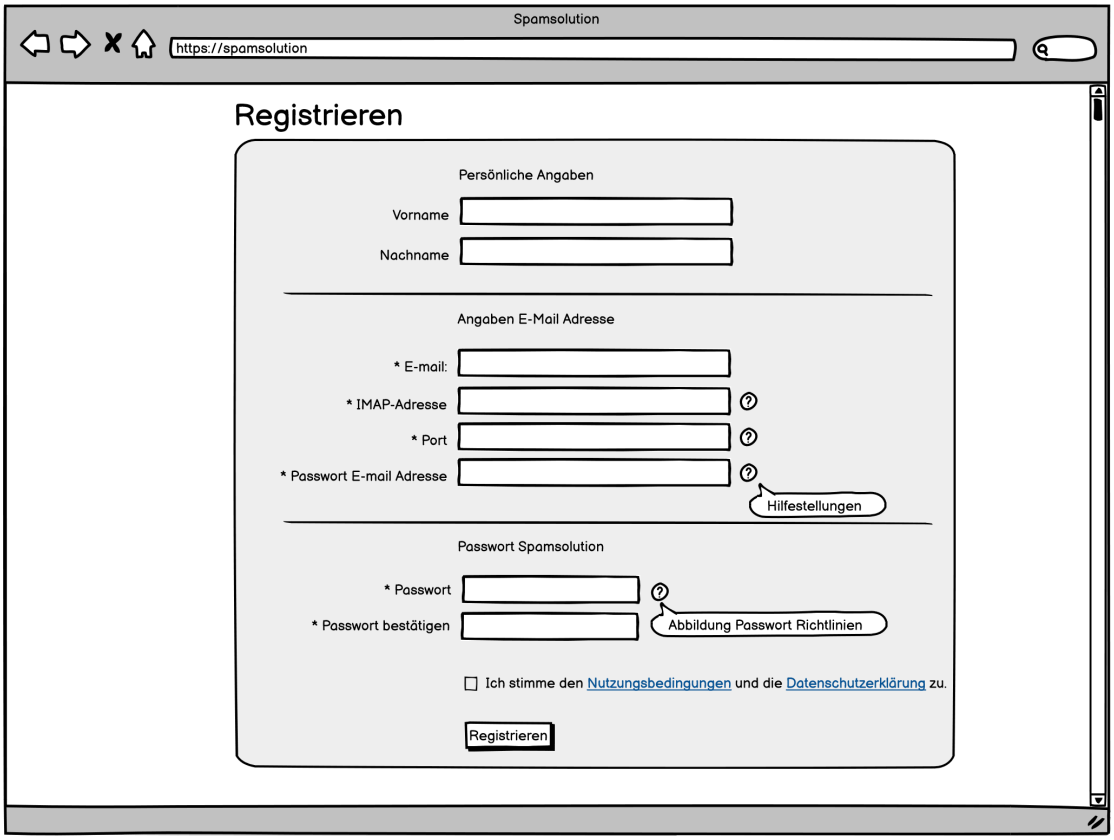


Abbildung 3.6: Registrierung neuer Benutzer

3.5.3 Home-Screen

Diese Seite dient als Startseite für den Benutzer, von der aus er Zugriff auf alle Funktionen hat. Oben rechts kann der User, über das Kontextmenü seine persönlichen Daten ändern und sich von der Plattform abmelden. In der Mitte sieht der User zwei Kacheln, die ihn zum einen zu der White- und Blacklist weiterleiten und zum anderen zu der Ansicht, um vordefinierte Listen zu abonnieren.

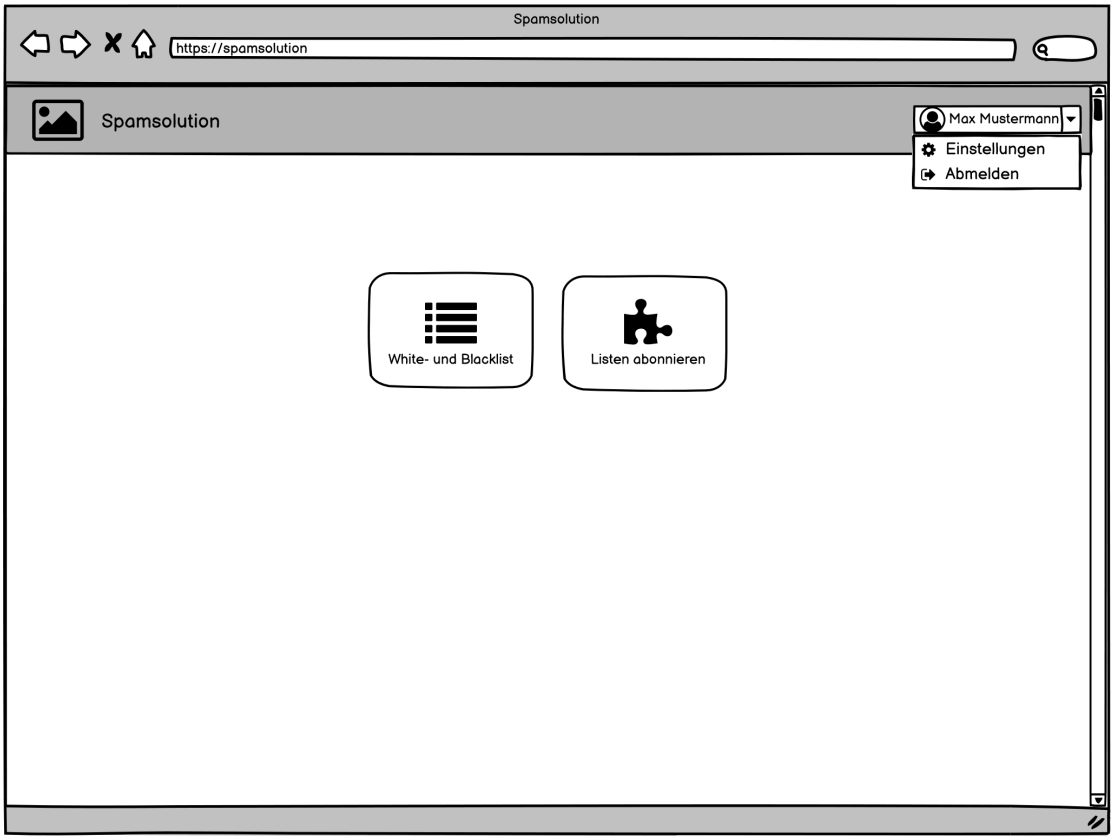


Abbildung 3.7: Startseite nach der Anmeldung

3.5.4 White- und Blacklist

Diese Ansicht liefert einen Überblick über die E-Mail-Adressen, welche sich bei dem User auf der White- bzw. auf der Blacklist befinden. Klickt der User eine E-Mail-Adresse an, kann er diese bearbeiten, in die andere Liste verschieben oder auch löschen. Unten kann der Benutzer neue E-Mail-Adressen hinzufügen.

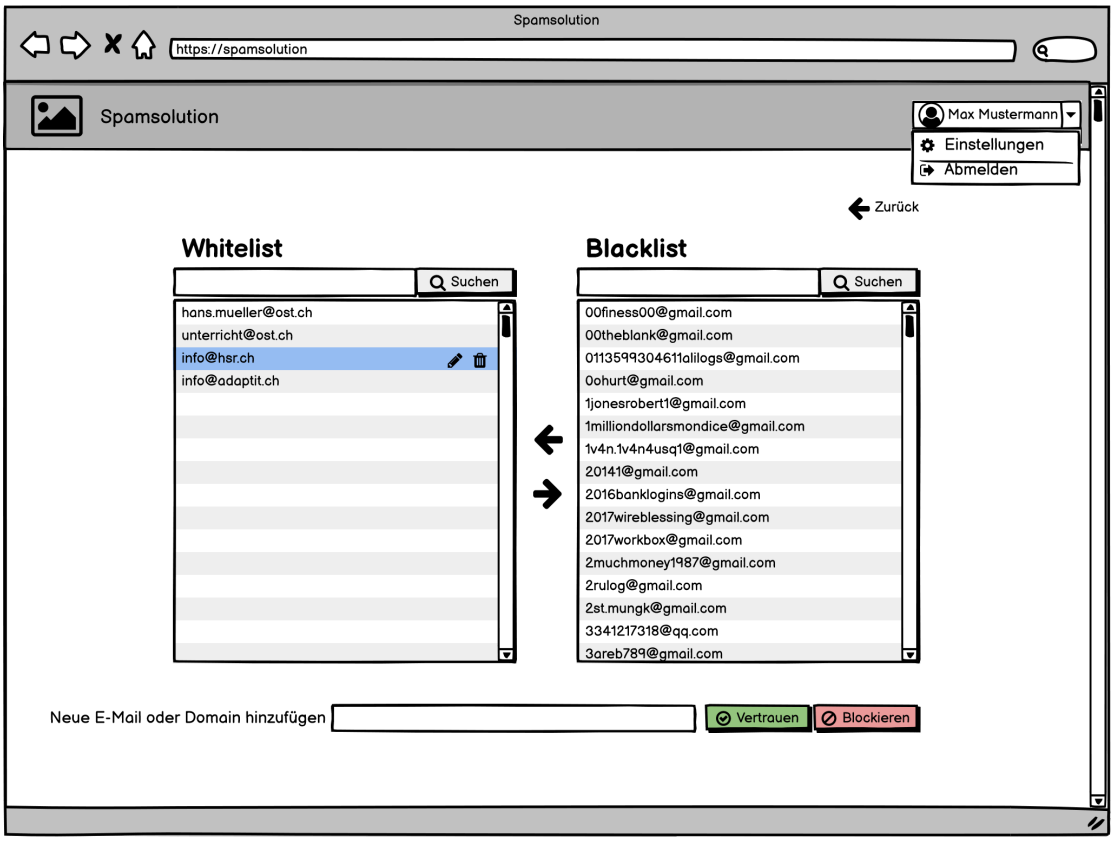


Abbildung 3.8: Ansicht White- und Blacklist

3.5.5 Abonnierbare Listen

In dieser Ansicht sieht der Benutzer Listen, die von einem Admin erstellt wurden und nun abonniert werden können. Um nachzusehen, welche E-Mail-Adressen in der Liste enthalten sind, kann der User die Liste aufklappen. Mit einem Klick auf „Abonnieren“ wird die Liste zu dem User hinzugefügt. Ist der Benutzer nicht mehr zufrieden mit einer Liste, kann er sich von dieser wieder abmelden, indem er auf „Deabonniern“ klickt.

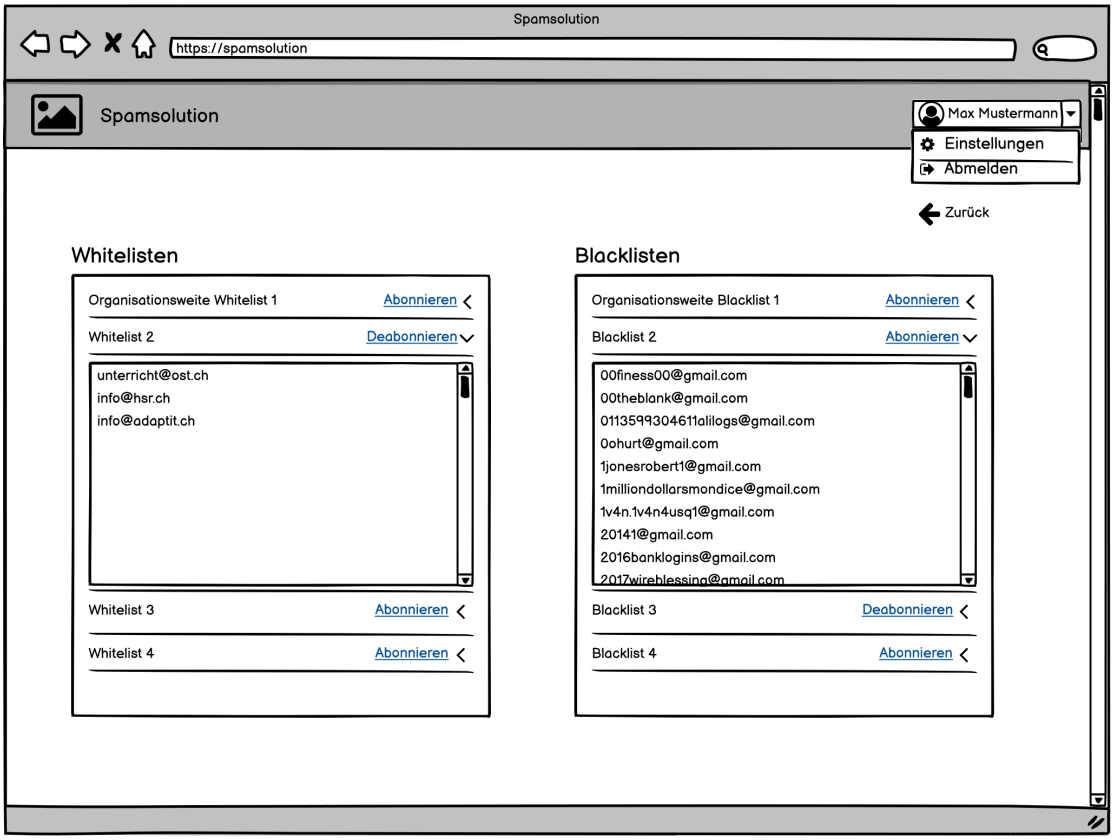


Abbildung 3.9: Ansicht Abonnierbare Listen

3.5.6 Listen verwalten Admin

Der Admin hat für die White- und Blacklist jeweils separate Ansichten, wobei sich diese optisch nicht von einander unterscheiden. In der obersten Leiste kann der Admin eine vorhandene Liste auswählen. Die Liste wird daraufhin angezeigt und er kann diese bearbeiten. Die ausgewählte Liste kann er löschen, indem er auf „ausgewählte Liste löschen“ klickt. Um eine neue Liste zu erstellen, muss der Admin auf „Neue Liste hinzufügen“ klicken.

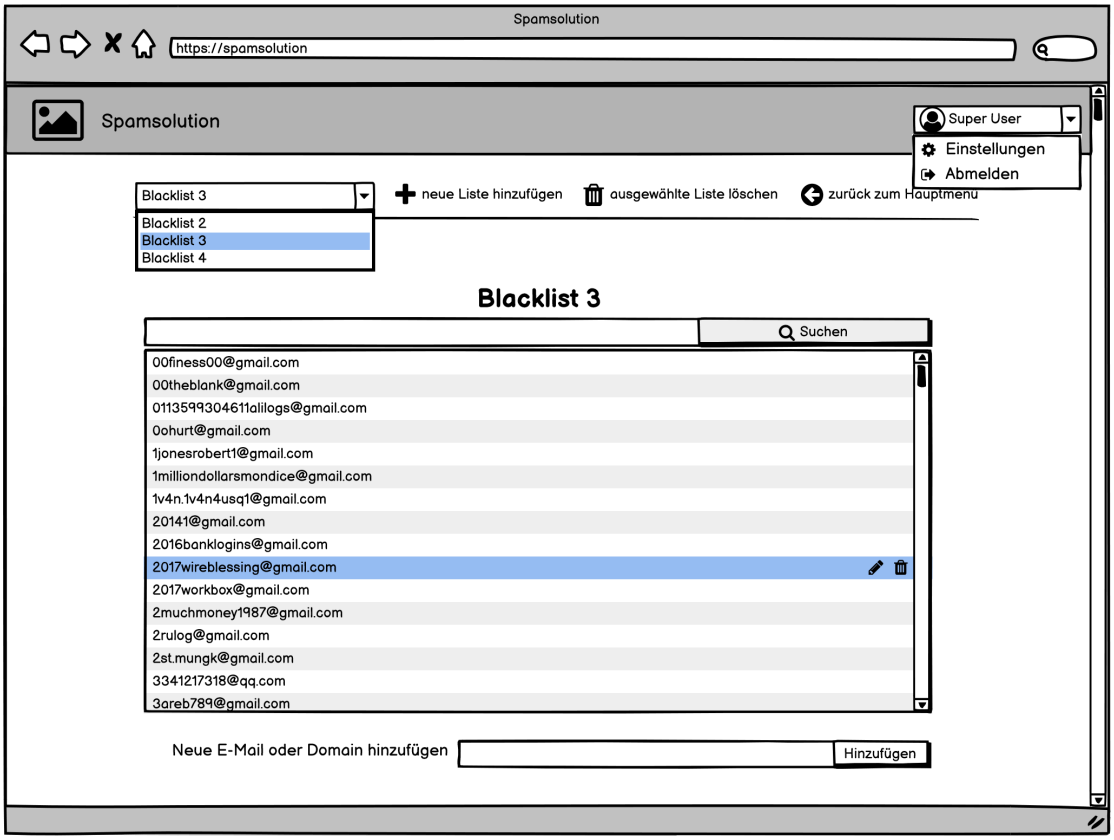


Abbildung 3.10: Admin Ansicht Blacklist

3.5.7 Banner in E-Mail

Bei jeder hereinkommenden E-Mail wird der Absender analysiert und sollte dieser weder auf der Whitelist noch auf der Blacklist sein, wird ein Banner zuoberst eingefügt. Je nachdem wie vertrauenswürdig die Applikation die E-Mail einschätzt, wird der Banner rot, gelb oder grün eingefärbt. Dadurch ist auf den ersten Blick erkennbar, ob man dieser vertrauen kann. Der User hat die Möglichkeit, den Absender zu der White- oder Blacklist hinzuzufügen, indem er auf „Vertrauen“ bzw. „Blockieren“ klickt.

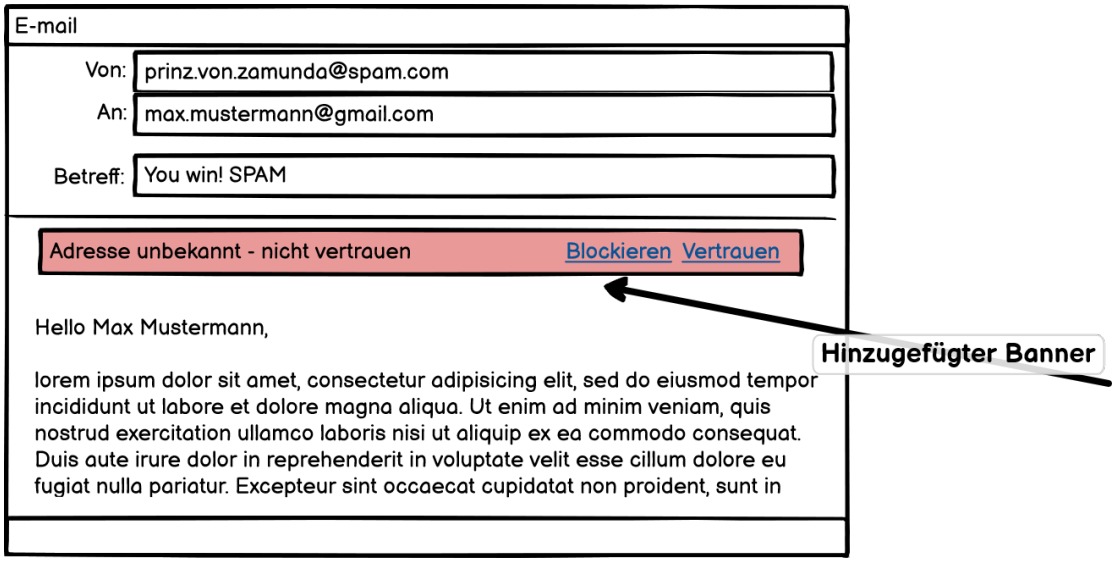


Abbildung 3.11: Beispiel einer nicht vertrauenswürdigen E-Mail

4 Implementierung

Das Kapitel Implementierung beschäftigt sich mit der technischen Seite des Projektes. Daher werden zunächst die wichtigsten Aspekte des Codes und das Testing erläutert. Als Abschluss wird noch auf Schwierigkeiten, die während der Entwicklung eintraten, eingegangen.

4.1 Codedokumentation

Im folgenden Abschnitt werden wir den Code näher betrachten. Wir werden die wichtigsten Klassen mit ihren Aufgaben beschreiben und auf das Zusammenspiel zwischen diesen eingehen.

4.1.1 Frontend

Das Frontend setzt sich aus mehreren JavaScript Dateien zusammen. Jede Seite in der Applikation wird durch eine eigene JavaScript Datei generiert. Wiederverwendbare Komponenten, wie z.B. die Navigationsleiste, sind in separaten JavaScript Dateien ausgelagert, um Code-Duplikationen zu vermeiden. Hinzu kommen Dateien, die für die Koordination zwischen den einzelnen Seiten zuständig sind. Alle diese Dateien werden kompiliert und übrig bleibt eine JavaScript Datei, eine CSS Datei und eine HTML Datei. An den User wird letztendlich nur die HTML-Datei übermittelt.

Index.js und App.js

Index.js ist der Einstiegspunkt in das Frontend. Es importiert App.js, was als Router fungiert und damit für die Navigation zwischen den Seiten zuständig ist, und aktiviert den Strict Mode für die Applikation. Dadurch wird man während der Entwicklung besser auf potenzielle Probleme aufmerksam gemacht, wobei es keinerlei Auswirkungen auf die Produktion hat (vgl. Abbildung 4.1 und 4.2).[10]

```
ReactDOM.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
```

Abbildung 4.1: Ausschnitt index.js

```
function App() {
  return (
    <Router>
      <Routes>
        <Route path="/logoff" element={<LogOff />} />
        <Route path="/settings" element={<Settings />} />
        <Route path="/registration" element={<Registration />} />
        <Route path="/homescreen" element={<HomeScreen />} />
        <Route path="*" element={<Login />} />
      </Routes>
    </Router>
  );
}
```

Abbildung 4.2: Ausschnitt app.js

Aufbau der Seite

Der Aufbau der JavaScript Dateien für die einzelnen Seiten folgt einem grundlegendem Schema. Es wird eine Funktion deklariert, die die HTML Seite zurückgibt. Variablen, deren Wert sich ändern kann, werden mithilfe des Hooks useState zwischengespeichert.[11] Dadurch wird sichergestellt, dass die Seite neu gerendert wird, sobald sich die Daten aktualisieren. Die Kommunikation mit dem Backend erfolgt mittels der Fetch-API (vgl. Abbildung 4.3).

```
import react, {useEffect, useState} from 'react';

function Example(){
    const variablen = "Beispiel";
    const host = window.location.origin;

    const [hook, setHooks] = useState();

    useEffect( effect: ()=>{
        fetch( input: host + "/apicall") Promise<Response>
            .then( r => r.json()) Promise<any>
            .then(//Daten bearbeiten)
            .catch((error) =>{
                //error.message
            })
    });

    function foo (){ }
    //... weitere Funktionen, die benötigt werden.

    return(
        <div onclick={foo}>Html Elemente, die gerendert werden sollen.</div>
    )
}

export default Example;
```

Abbildung 4.3: Pseudo Code einer Seite

4.1.2 Backend

Das Backend bildet, wie bereits in Kapitel 3.2 erwähnt, die Schnittstelle zwischen den anderen Komponenten. Das Backend verwendet einen standardmässigen Aufbau für eine Express.js Anwendung. Den Kern bilden ein Index, ein Routes und ein Controller File. Hinzu kommen noch eine Klasse für den Datenbankzugriff und diverse Hilfsklassen.

Index.js

Index.js ist der Einstiegspunkt für das Backend und startet den Server. Es bindet die benötigten Middlewares ein und sorgt dafür, dass HTTP-Anfragen an HTTPs weitergeleitet werden (vgl. Abbildung 4.4).

```
api.use(cookieParser());
api.use(bodyParser.json());
api.use(express.static(__dirname + '/../frontend/build'));
api.use(routes.router);

http.get('*', (request, response) => {
    response.redirect('https://' + request.headers.host + request.url);
});
```

Abbildung 4.4: Ausschnitt Index.js

Controller.js

Der Controller ist für die Logik des Backends zuständig. Alle möglichen Routen werden auf eine der Funktionen des Controllers gemapped und von dieser behandelt.

Bei Anfragen von Usern prüft es zunächst mithilfe von der Sessionmanager Klasse, ob ein gültiges Cookie übermittelt wurde. Ist dies der Fall, wird die Anfrage ausgeführt, ansonsten wird entweder eine Fehlernachricht zurückgegeben oder man wird auf den Loginscreen weitergeleitet (vgl. Abbildung 4.5).

```
checkCookie(request, response) {
    if(sessionmanager.sessions[request.cookies.AC]) {
        return response.status(200).sendFile(path.resolve(__dirname + '../../frontend/build/index.html'));
    }
    return response.redirect('https://' + request.headers.host);
}
```

Abbildung 4.5: Funktion checkCookie

Passwörter werden gemäss Anforderung verschlüsselt in der Datenbank gespeichert. Für die Verschlüsselung bzw. Entschlüsselung verwendet der Controller die Hilfsklasse Crypto. Abbildung 4.6 zeigt wie das Passwort beim Login überprüft wird.

```
async login(request, response) {
    const result = await dataservice.getUserPassword(request.body.email);
    if(result.length > 0) {
        let password;
        try {
            password = crypto.decrypt(result[0].password);
        } catch(error) {
            return jsonResponse(response, 500, default_message);
        }
        if(request.body.password === password) {
```

Abbildung 4.6: Überprüfung des Passworts

Durch die Cookies ist gewährleistet, das ein User nur auf seine Daten zugreifen kann. Das Python-Script hingegen muss die Daten von allen Usern abrufen. Um sicherzustellen, dass nur das Python-Script die jeweiligen API-Calls machen kann, authentifiziert es sich mithilfe eines Tokens (vgl. Abbildung 4.7).

```
async allLists(request, response) {
    if(request.params.accessToken !== environment.pythonToken) {
        return jsonResponse(response, 401, access_denied);
    }
    const results = await dataservice.getLists();
    if(results.length > 0) {
        return response.status(200).json({listMapping: results});
    }
    return jsonResponse(response, 500, default_message);
}
```

Abbildung 4.7: Funktion um die Listen von allen Benutzern abzurufen

Dataservice.js

Der Dataservice hält eine Verbindung zu der Datenbank aufrecht und führt alle Queries durch. Zusätzlich zu der Validierung im Frontend werden bei jeder Funktion die übergebenen Variablen auf ihren Typen geprüft. Sollte dieser nicht mit dem erwarteten übereinstimmen, wird eine Fehlermeldung erzeugt. Die eigentlichen Queries sind in separaten Klassen, getrennt nach Art der Query (selectQueries.js, insertQueries.js, updateQueries.js, deleteQueries.js), gespeichert. Abbildung 4.8 zeigt die Funktion, welche die persönliche White- und Blacklist eines Benutzers zurückgibt.

```
async getUserList(userID) {
    checkType(userID, 'UserID', 'number');
    return await this.connection.promise()
        .query(selectQueries.getUserList, userID)
        .then(([rows]) => rows)
        .catch(error => {
            errorhandler.writeError(error);
        });
}
```

Abbildung 4.8: Rückgabe einer persönlichen White- und Blacklist

4.1.3 Python Script

Das Python-Script ist ein Algorithmus, der in einer Endlosschleife ausgeführt wird. Es setzt sich aus den drei Modulen „api_handler.py“, „email_handler.py“ und „connector.py“ zusammen.

Main

Main.py ist der Einstiegspunkt in das Script. Es führt in einer Endlosschleife den Algorithmus zum Analysieren der Mails durch. Der Algorithmus sieht wie folgt aus (die Variablen „api“, „email“ und „imap“ repräsentieren die drei Module „api_handler.py“, „email_handler.py“ und „connector.py“):

1. Mithilfe des Api_Handlers holt sich das Script die IMAP-Zugangsdaten, sowie die White- und Blacklisten (Abbildung 4.9).

```
try:
    api.get_imap_data()
    api.get_list_mappings()
```

Abbildung 4.9: Download aktueller Zugangsdaten und Listen

2. Sobald die Daten im Script aktuell sind werden die folgenden Schritte für jeden Benutzer durchgeführt.

   a) Es wird überprüft, ob der Account inaktiv ist (siehe nächster Schritt). Sollte dies der Fall sein, wird mit dem nächsten User weitergemacht (Abbildung 4.10).

```
for user in api.imap_data:
    if user["inactiveSince"]:
        continue
```

Abbildung 4.10: Prüfung, ob der Account aktiv ist

   b) Es wird versucht eine Verbindung per IMAP herzustellen. Falls dies in drei aufeinander folgenden Iterationen des Algorithmus fehlschlägt, wird der Account auf inaktiv gesetzt, bis der User seine Daten auf der Webseite aktualisiert (Abbildung 4.11).

```
try:
    imap.connect(username, password, address, port)
except Exception as e:
    error_handler.write_error(e, username, address, port)
    if email_id not in inactive:
        inactive[email_id] = 1
    elif inactive[email_id] == 2:
        try:
            api.set_inactive(email_id)
            inactive.pop(email_id, None)
        except Exception as e:
            error_handler.write_error(e, username, address, port)
            continue
    else:
        inactive[email_id] = 2
    continue
```

Abbildung 4.11: Verbindungsversuch zum IMAP-Server

c) Es werden die ungelesenen E-Mails aus dem Posteingang gefiltert (Abbildung 4.12). Daraufhin werden alle gefilterten E-Mails nach folgendem Schema analysiert:

```
ids = imap.search()
```

Abbildung 4.12: Filtern des Posteingangs nach ungelesenen E-Mails

i. Die E-Mail wird heruntergeladen und die Nachricht extrahiert (Abbildung 4.13).

```
for id in ids:
    message_raw = imap.fetch(id)
    message = email.get_message(message_raw)
```

Abbildung 4.13: Herunterladen von einer E-Mail

ii. Es wird überprüft, ob der Absender der E-Mail auf der White- oder Blacklist steht. Ist der Absender auf der Blacklist, wird die E-Mail in den Spamordner verschoben. Für den Fall, dass der Absender unbekannt ist, wird ein Banner eingefügt (Abbildung 4.14).

```
sender = email.get_sender(message)
list_type = email.analyze_sender(sender, list_mapping)
if list_type == "Blacklist":
    imap.move_to_spam(id)
elif list_type == "Unknown":
    if email.add_banner(message, username, sender, environment["host"]) == "Duplicate":
        continue
    imap.add_banner(id, message, date)
```

Abbildung 4.14: Analyse der E-Mail

d) Zum Schluss meldet sich das Script von dem IMAP-Server ab (Abbildung 4.15).

```
imap.disconnect()
```

Abbildung 4.15: Abmelden von dem IMAP-Server

Api_Handler

Der Api_Handler ist für die Verbindung zu dem Backend zuständig. Er macht Requests an das Backend und speichert die Rückgabe in Instanzvariablen, damit Main darauf zugreifen kann (Abbildung 4.16).

```
def get_imap_data(self):
    imap_request = requests.get(self.host + "/imap/" + self.backend_token)
    self.imap_data = json.loads(imap_request.text)
    self.imap_data = self.imap_data["imapServers"]
```

Abbildung 4.16: Funktion get_imap_data

Connector

Der Connector stellt die Verbindung zum IMAP-Server her und führt jegliche Operationen auf diesem durch. Er bedient sich des „imaplib“ Moduls und vereinfacht prinzipiell die Aufrufe im Algorithmus, indem er Funktionsaufrufe von „imaplib“ zusammenfasst (Abbildung 4.17).

```
def connect(self, username, password, address, port=993):
    self.imap = imaplib.IMAP4_SSL(host=address, port=port)
    self.imap.login(username, password)
    self.update_destination(username)
    self.imap.select("INBOX", readonly=False)
```

Abbildung 4.17: Funktion connect

Email_Handler

Der Email_Handler analysiert und bearbeitet heruntergeladene E-Mails. Er verwendet hierfür das „email“ Modul. Abbildung 4.18 zeigt, wie ein Banner in eine E-Mail eingefügt wird. Die restlichen Funktionen sind simpel und werden daher nicht weiter behandelt.

```
def adjust_payload(self, message, username, sender, host):
    encoding = message["Content-Transfer-Encoding"]
    if encoding is None or encoding == "7bit":
        encoding = "quoted-printable"
        message["Content-Transfer-Encoding"] = encoding

    charset = message.get_content_charset()
    part_payload = message.get_payload()

    body = self.decode_payload(part_payload, encoding)
    body = body.decode(charset)

    if body.find(self.id) != -1:
        return "Duplicate"

    new_payload = self.build_html(body, username, sender, host)
    new_payload = self.encode_payload(new_payload.encode(charset), encoding)
    message.set_payload(new_payload)
```

Abbildung 4.18: Funktion adjust_payload

4.1.4 Datenbank

Die Datenbank und der Datenbankbenutzer werden durch das Skript „create-database.sql“ erstellt. Dabei verwendet es für den Benutzernamen, das Passwort und den Namen der Datenbank Variablen, die beim Aufruf des Skriptes übergeben werden müssen (vgl. Abbildung 4.19).

```
SET @query = CONCAT('CREATE USER ', QUOTE(@username), ' IDENTIFIED BY ', QUOTE(@password));
PREPARE stmt FROM @query;
EXECUTE stmt;
DEALLOCATE PREPARE stmt;
```

Abbildung 4.19: Ausschnitt create-database.sql

„Populate-database.sql“ erzeugt schliesslich die Tabellen mit den dazugehörigen Constraints. Hierfür wird simples SQL, ohne grosse Besonderheiten, verwendet (vgl. Abbildung 4.20 und 4.21).

```
CREATE TABLE listMappings (
    id INTEGER NOT NULL AUTO_INCREMENT,
    userMail INTEGER,
    email INTEGER NOT NULL,
    list INTEGER,
    whiteList BOOLEAN NOT NULL,
    PRIMARY KEY (id)
);
```

Abbildung 4.20: Erzeugung der Tabelle listMappings

```
ALTER TABLE listMappings
    ADD CONSTRAINT uc_userMail_email
        UNIQUE(userMail, email);
```

Abbildung 4.21: Hinzufügen eines Constraints

4.2 Testing

Im Folgenden gehen wir näher darauf ein, wie wir die Applikation getestet haben. Dabei richten wir die Reihenfolge dieses Abschnittes nach der Häufigkeit der Tests aus.

4.2.1 Unit Tests

Die Unit Tests sind in der Pipeline enthalten und werden somit bei jedem Push durchgeführt. Der Grossteil der Logik befindet sich im Backend, wodurch dieses auch am ausführlichsten getestet wurde. Im Backend verwenden wir für die Unit Tests Jest[12] und im Python-Script unittest[13]. Insgesamt prüfen 137 Tests den Code auf Fehler bei einer Änderung.

4.2.2 System Tests

Die System Test wurden nach jedem Meilenstein in der Construction Phase durchgeführt. Während eines Testes wurde jeder UC manuell von einem Team Mitglied durchgespielt und der Fortschritt des jeweiligen UCs dokumentiert. Mithilfe der Tests können wir frühzeitig mögliche Softwarefehler erkennen, die nicht von den Unit Tests erkannt wurden, und diese beheben. Zudem gibt der System Test einen guten Überblick darüber, was der aktuelle Stand der Entwicklung ist. Nachfolgend führen wir den letzten System Test auf, da dieser aufzeigt, welche UCs umgesetzt wurden und welche noch offen sind. Die anderen beiden System Tests sind im Anhang zu finden.

System Test M4

Datum	01.06.2022
Tester	Fabian Tiri
Meilenstein	M4 End of Construction

Use Case	Implementiert	Fehler / Unschönheiten	Status
UC 01	Ja	-	100 %
UC 02	Ja	-	100 %
UC 03	Ja	-	100 %
UC 04	Ja	-	100 %
UC 05	Ja	-	100 %
UC 06	Ja	-	100 %
UC 07*	Nein	-	0 %
UC 08*	Nein	-	0 %
UC 09*	Nein	-	0 %
UC 10*	Nein	-	0 %
UC 11*	Nein	-	0 %
UC 12*	Nein	-	0 %
UC 13	Ja	-	100 %

** optionale UCs*

Tabelle 4.1: Systemtest 3

4.2.3 Usability Tests

Eine der Anforderungen vom Industriepartner ist, dass mindestens 3 von 4 Personen das UI in den Kategorien Layout, Responsiveness, Farbgestaltung und Inhalt mindestens mit 7 von 10 Punkten bewerten, wobei 10 das Beste ist. Um die Anforderung überprüfen zu können, haben wir diese ausgeweitet und aus der UI-Bewertung einen Usability Test gemacht.

Ziel

Das Ziel des Usability Tests ist es, die Applikation in einem realen Umfeld zu testen und ein Feedback von unabhängigen Benutzern zu erhalten. Mit dem Usability Test möchten wir vor allem zwei Sachen prüfen. Zum einem möchten wir wissen, ob die Applikation intuitiv zu bedienen ist und zum anderen ob das Design ansprechend ist.

Teilnehmer

Der Service ist so aufgebaut, dass sich rein theoretisch jeder mit einer E-Mail-Adresse für diesen registrieren kann. Da jeder seine eigene Arbeitsweise hat und nicht alle IT-affin sind, war es uns wichtig, dass wir die Tests mit Personen aus verschiedenen Altersgruppen und verschiedenen Berufsfeldern durchführen, um ein breites Feedback zu erhalten.

Vorgehen

Das Testszenario besteht aus zwei Teilen. Zuerst wird die Applikation auf ihre Funktionalität geprüft. Dafür haben wir ein Testkonzept erstellt, bei dem die Testperson verschiedene Szenarien durcharbeiten muss. Die Testszenarien basieren auf die von uns definierten UCs. Im zweiten Teil füllt die Test-Person ein Online-Formular aus, um das UI zu bewerten.

Bei den Tests war jeweils ein Teammitglied physisch anwesend, um diese dokumentieren zu können. Bei jedem Test wurde die Testperson gebeten laut mitzudenken, damit wir zusätzlich die Gedankengänge nachvollziehen und daraus Schlüsse ziehen können.

Resultate

Der Usability Test lieferte uns wertvolle Erkenntnisse und wir erhielten guten Input seitens der Testpersonen. Nachfolgend gehen wir auf die wichtigsten Ergebnisse des Usability Test ein und die Änderungen, welche wir anhand vom Feedback umgesetzt haben. Der vollständige Usability Test mit unseren Notizen ist Anhang zu finden.

Szenario – Registrierung
Die Testperson wurde gebeten, ein neues Konto auf der Plattform anzulegen. Die Aufgabe wurde gut gelöst, jedoch gab es Missverständnisse bei den Buttons auf der Anmeldeseite. Drei Personen trugen zunächst die E-Mail-Adresse und das Passwort ein und versuchten einen Account anzulegen, indem sie auf Login klickten. Aufgrund der Missverständnisse haben wir die Anmeldeseite überarbeitet und den Button für die Erstellung eines neuen Kontos umbenannt, als auch klarer von der Anmeldemaske abgegrenzt (Abbildung 4.22).

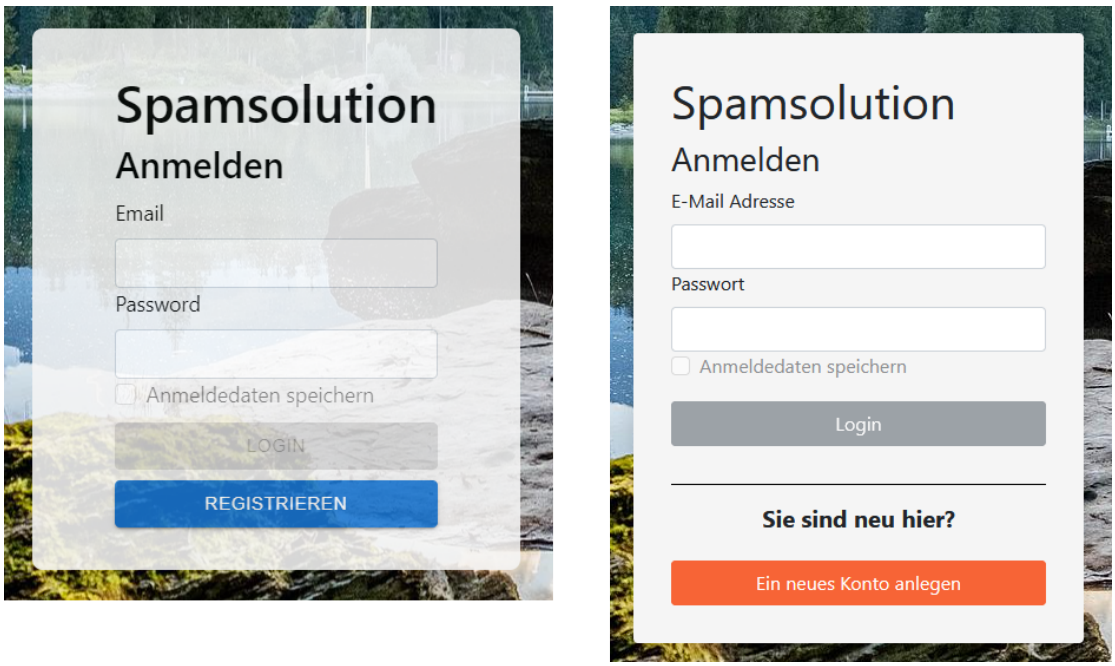


Abbildung 4.22: Der Vergleich zwischen der alten (links) und der neuen Anmeldemaske (rechts)

.

Auf der Registrierungsseite hatten alle Testpersonen Schwierigkeiten, die Angaben von der E-Mail-Adresse anzugeben. Vor allem bei der IMAP-Adresse und dem Port wussten viele nicht, was das überhaupt für Angaben sind. Um Abhilfe zu schaffen, haben wir bei den kritischen Feldern einen Placeholder mit einem Beispiel eingefügt. Zusätzlich gibt es bei einigen Feldern eine Infobox mit einer kurzen Beschreibung (Abbildung 4.23).

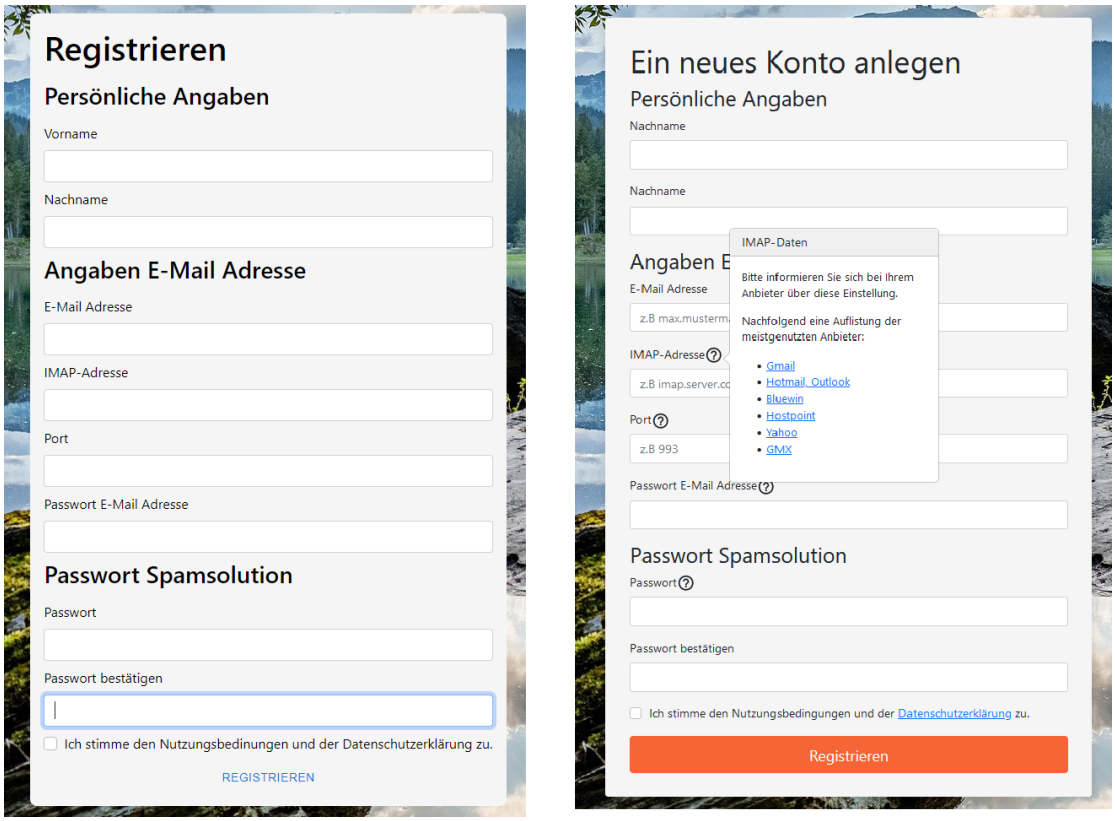


Abbildung 4.23: Der Vergleich zwischen der alten (links) und der neuen Registrierungsmaske (rechts)

.

Szenario - E-Mail hinzufügen
In diesem Szenario musste die Testperson eine E-Mail zu einer Liste hinzufügen. Einige Testpersonen haben zuerst auf die mittleren Buttons geklickt, da sie anhand von den Begriffen davon ausgegangen sind, dass man mit diesen Buttons eine neue E-Mail-Adresse hinzufügt. Dabei haben Sie auch nicht bemerkt, dass die Buttons nicht aktiv sind. Hier haben wir die Buttons umbenannt und den Unterschied zwischen einem inaktiven und einem aktiven Button stärker hervorgehoben (Abbildung 4.24).

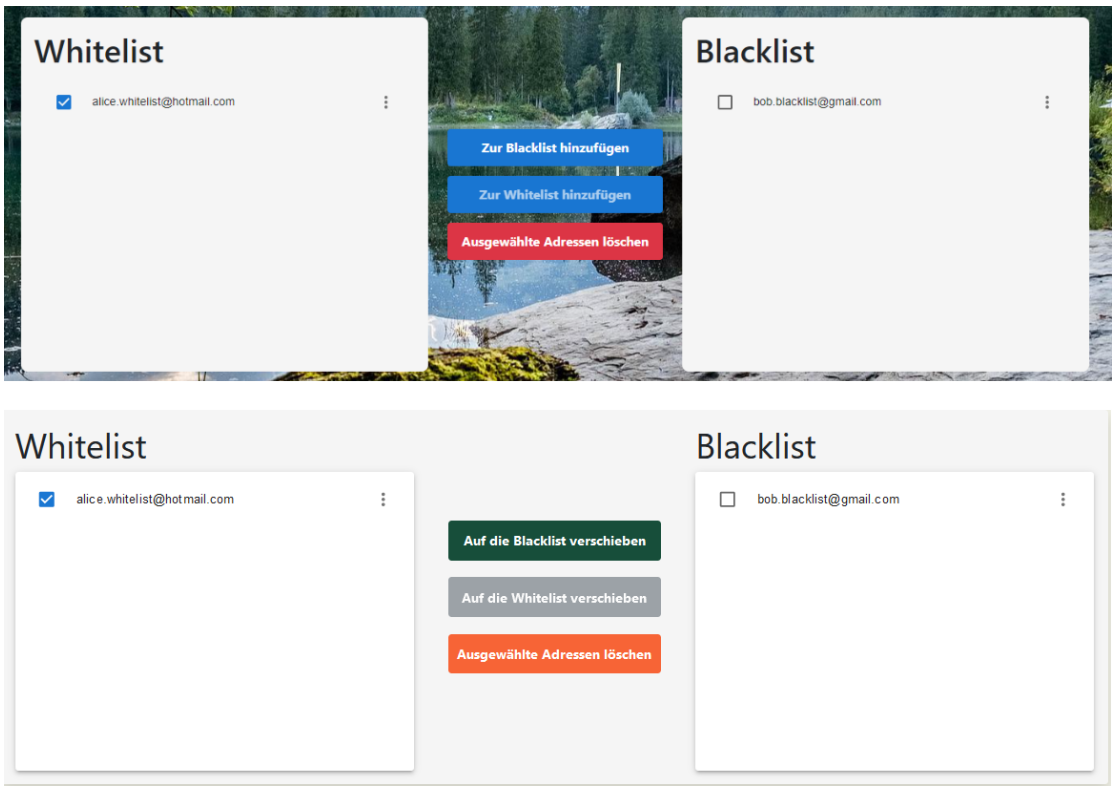


Abbildung 4.24: Der Vergleich zwischen der alten (oben) und neuen Version (unten)

.

Nachinterview

Nachdem die Szenarien durchgespielt wurden, haben wir mit jeder Testperson noch ein Nachinterview geführt. In diesem Interview wollten wir den allgemeinen Eindruck der Webseite und des Services erhalten. Grundsätzlich waren die Testpersonen mit der Webseite und dem Service zufrieden. Die Seite ist einfach aufgebaut, übersichtlich und man findet sich schnell zurecht. Es gab jedoch einige Verbesserungsvorschläge. Die Änderungsvorschläge, die wir als sinnvoll erachtet haben, wurden im Nachhinein umgesetzt. Nachfolgend ist eine Liste mit einigen Vorschlägen die umgesetzt wurden:

- Die Option eine E-Mail-Adresse zu einer Liste hinzuzufügen soll oberhalb der Listen stehen
- Das Hintergrundbild passt nicht zum Inhalt der Hauptseite.
- Mehr Informationen über den Service geben.

Nachdem wir die Änderungen vorgenommen hatten, waren wir jedoch selbst nicht mehr ganz zufrieden mit dem Design. Durch das Entfernen des Hintergrundbildes waren wir der Meinung, dass die Webseite zu viel Grau beinhaltet und somit leer und langweilig aussieht. Wir haben daraufhin entschieden, die Farbgestaltung neu zu machen, damit die Seite nicht zu monoton wirkt. Die Entwicklung der Webseite sieht man in Abbildung 4.25.

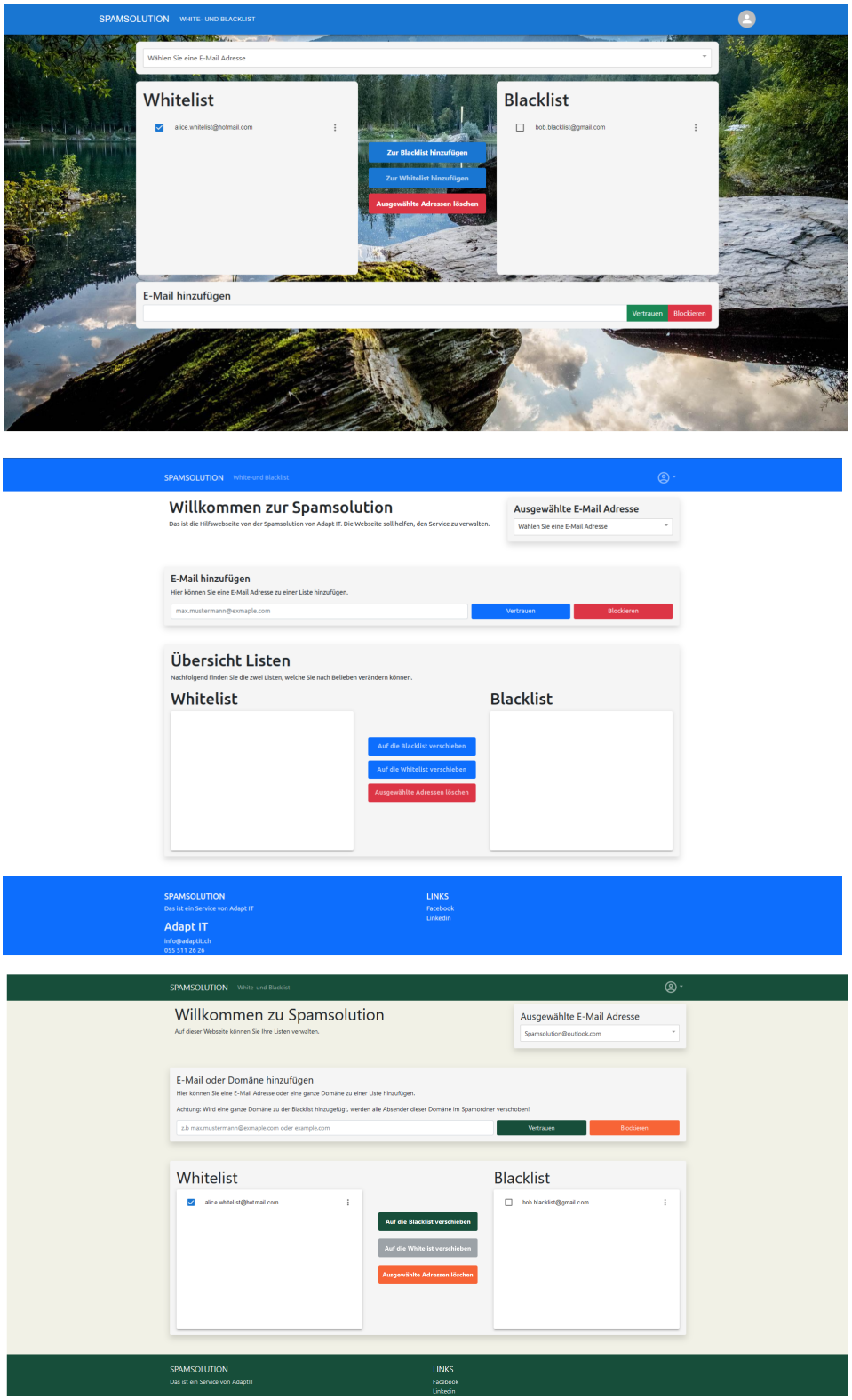


Abbildung 4.25: Der Vergleich zwischen der alten (oben), der Version nach dem Usability Test (mitte) und der finalen Version der Startseite(unten)

.

Ergebnisse Umfrage UI

Nachfolgend werden die Ergebnisse der UI Umfrage zusammengefasst. Die Resultate werden in den Kategorien Layout, Responsiveness, Farbgestaltung und Inhalt aufgeteilt.

Layout

Mit einer Durchschnittsbewertung von 8 haben wir in der Kategorie Layout die schlechteste Bewertung erhalten. Die Bewertung zeichnet sich auch aus dem Feedback ab, da wir hier am meisten Verbesserungsvorschläge erhalten haben (Abbildung 4.26).

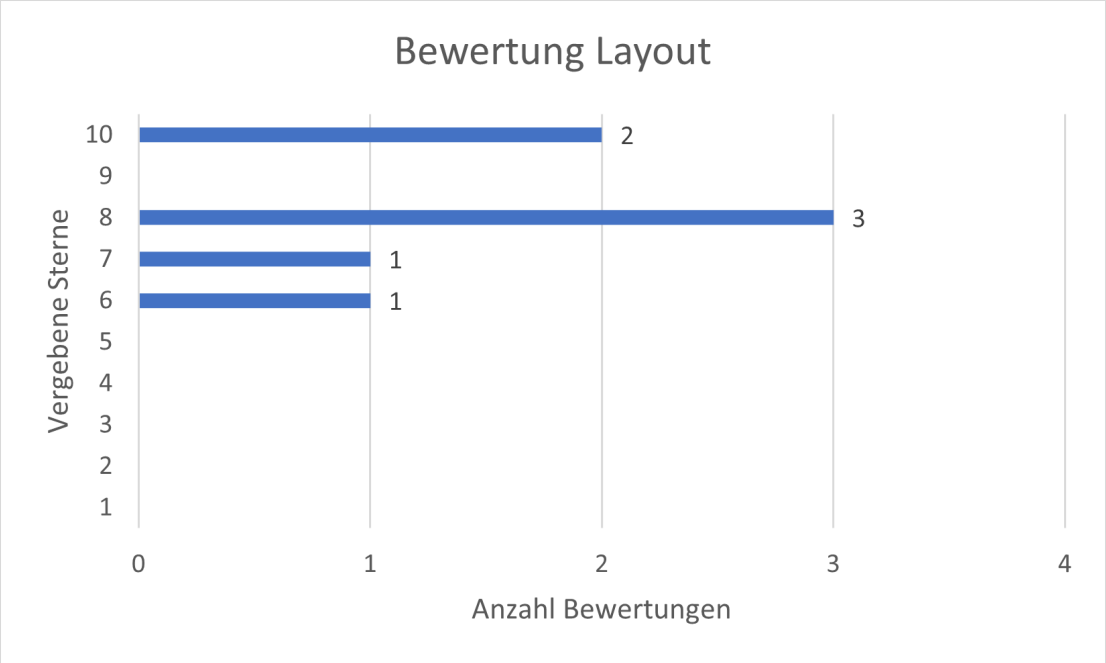


Abbildung 4.26: Bewertung Layout

Responsivness

Um diese Frage zu beantworten, haben die Testpersonen das Browserfenster auf verschiedenen Grössen angepasst. Sie waren mit der Anordnung und Darstellung der einzelnen Elemente bei jeder Ansicht sehr zufrieden. Dies spiegelt sich auch in der Bewertung wider. Die Testpersonen haben hier eine Durchschnittsbewertung von 9.42 gegeben (Abbildung 4.27).

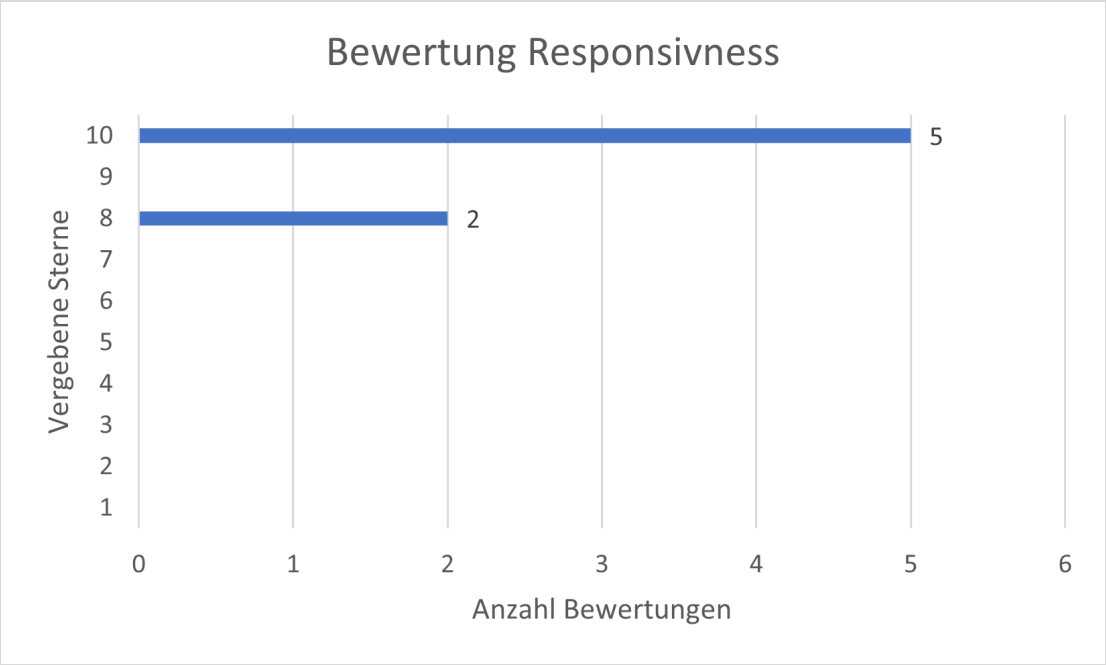


Abbildung 4.27: Bewertung Responsivness

Farbgestaltung

Für die Farbgestaltung haben wir eine Durchschnittsbewertung von 8.14 erhalten (Abbildung 4.28).

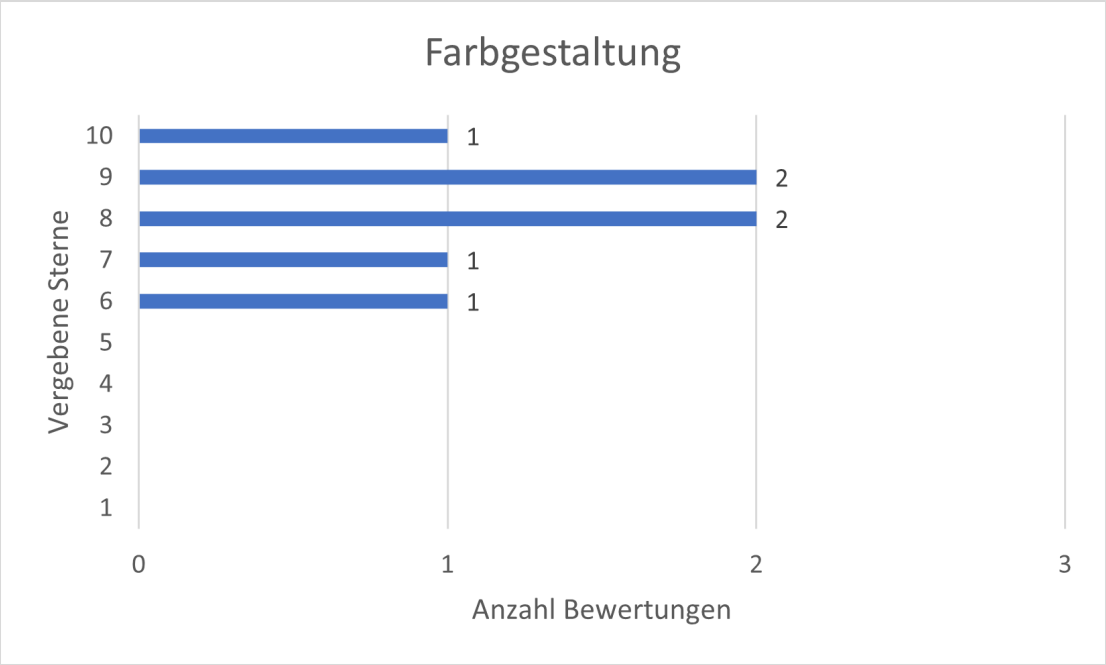


Abbildung 4.28: Bewertung Layout

Inhalt

Mit dem Inhalt der Webseite waren alle zufrieden. Die Webseite ist einfach aufgebaut und die Testpersonen haben sich schnell zurechtgefunden. Die Durchschnittsbewertung liegt hier bei 8.42 (Abbildung 4.29).

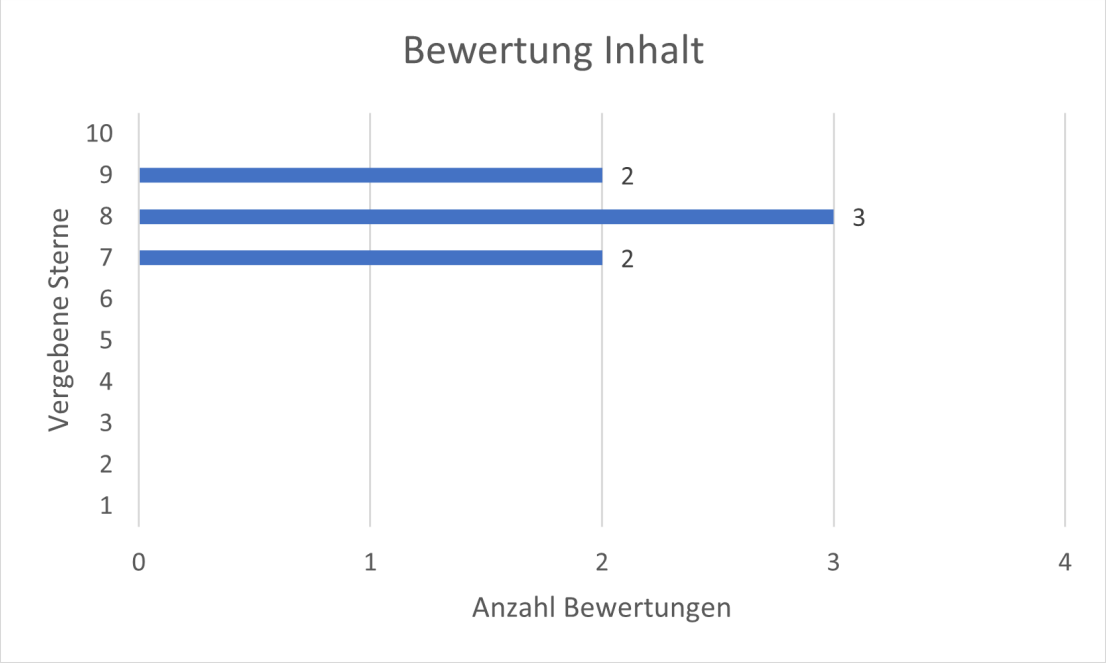


Abbildung 4.29: Bewertung Inhalt

Allgemeiner Eindruck und Fazit

Im Generellen kommt die Webseite gut bei den Testpersonen an. Die meisten Bewertungen liegen zwischen 8 und 9, was uns sehr erfreut, da wir viel Zeit in die Entwicklung der Applikation investiert haben.

Der Usability Test zeigt, dass die Anforderung des Industriepartners im Bezug auf das UI erfüllt wurde.

4.3 Herausforderungen

Im Laufe eines Projektes kann es vorkommen, dass Probleme auftreten, die den Fortschritt verzögern. Dieser Abschnitt widmet sich einigen Schwierigkeiten, die im Verlaufe dieser Arbeit auftraten, sowie deren Lösungen.

4.3.1 Verschlüsselung

Für die Funktionalität der Applikation ist es notwendig, Daten im Python-Script zu ver- und später im Backend zu entschlüsseln. Mittels eines symmetrischen Verschlüsselungsverfahren, sollte es ausreichen, wenn man beiden Komponenten denselben Schlüssel zur Verfügung stellt. Während dem Testen stellten wir allerdings fest, dass der entschlüsselte Text nicht mit dem Original übereinstimmte. Da wir für beide Komponenten denselben Algorithmus verwendeten, vermuten wir, dass das Padding anders gehandhabt wird. Um dies testen zu können, wurde nach einer Bibliothek gesucht, die man sowohl in Node.js als auch in Python verwenden kann. Unsere Wahl fiel auf die Bibliothek NaCl.[14] Diese hat Ports sowohl für JavaScript als auch für Python. Damit war es möglich, die Daten korrekt und unkompliziert zu entschlüsseln.

4.3.2 Authentifizierung

Um sicherzustellen, dass die Benutzer nur auf ihre persönlichen Daten zugreifen können, ist es notwendig, einen Authentifizierungsprozess einzubauen. Dieser funktioniert, indem sie sich zunächst mit einem Passwort anmelden, woraufhin bei ihnen ein Cookie gespeichert wird, dass künftig bei jeder Anfrage übermittelt wird. Das Backend kann dann den User anhand des Cookies eindeutig identifizieren. Die Authentifizierung beim Python-Script ist sogar noch wichtiger, da dieses auf die Daten aller Benutzer Zugriff hat. Besonders, da diesem über einen API-Call die E-Mail-Adressen mit den dazugehörigen Passwörtern übermittelt werden. Beim Python-Script verfolgen wir einen anderen Ansatz, da die Implementation mittels einem Cookie zusätzlichen Aufwand bedeuten würde. Das Python-Script verwendet ein Access-Token, welches es über die URL an das Backend übermittelt. Das Token wird zunächst überprüft und die Daten nur übertragen, wenn es korrekt ist.

5 Ergebnisse

Im letzten Teil schliessen wir den Bericht ab, indem zunächst ein Vergleich zwischen dem Endprodukt und dem ursprünglich geplanten gezogen wird. Anschliessend gehen wir noch auf Verbesserungsmöglichkeiten ein und runden das Kapitel mit einem Fazit und Ausblick ab.

5.1 Vergleich Wireframes mit dem Endprodukt

Während der Elaboration Phase entwarfen wir Wireframes, die das Endprodukt skizzieren sollten. Dieser Entwurf wurde in Kapitel 3.5 vorgestellt. In diesem Abschnitt wird nun auf Abweichungen des Endproduktes zu dem ursprünglichen Entwurf eingegangen.

5.1.1 Login

Bei der Anmeldemaske wurde der Button für das Registrieren von dem Rest der Maske getrennt, um den Unterschied in der Funktionalität stärker hervor zu heben. Eine Funktion, das Passwort automatisch zurückzusetzen, wurde nicht umgesetzt. Dies hätte einen beträchtlichen Mehraufwand bedeutet, ausserhalb des Scopes dieser Arbeit lag. Stattdessen wurde eine Checkbox hinzugefügt, die es ermöglicht, auch beim Schliessen des Browsers angemeldet zu bleiben (vgl. Abbildung 5.1).

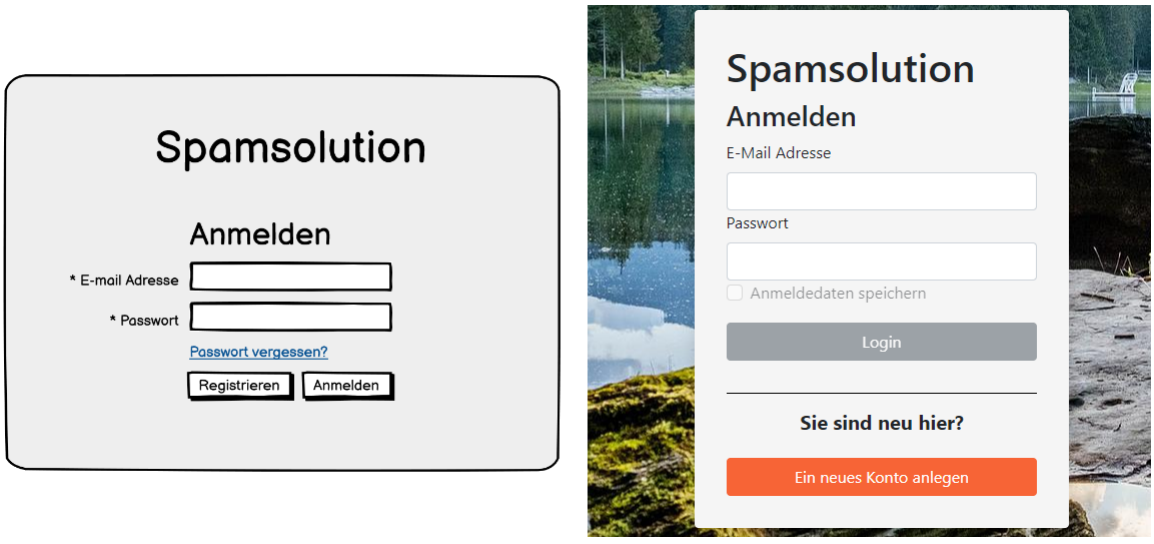


Abbildung 5.1: Vergleich des Logins des Wireframes (links) mit dem Login des Endproduktes (rechts).

5.1.2 Registrierung

Die Registrierungsseite wurde fast vollständig wie geplant umgesetzt. Die einzige Abweichung ist, dass das Label für die Inputbox oberhalb dieser angezeigt wird und nicht daneben (vgl. Abbildung 5.2).

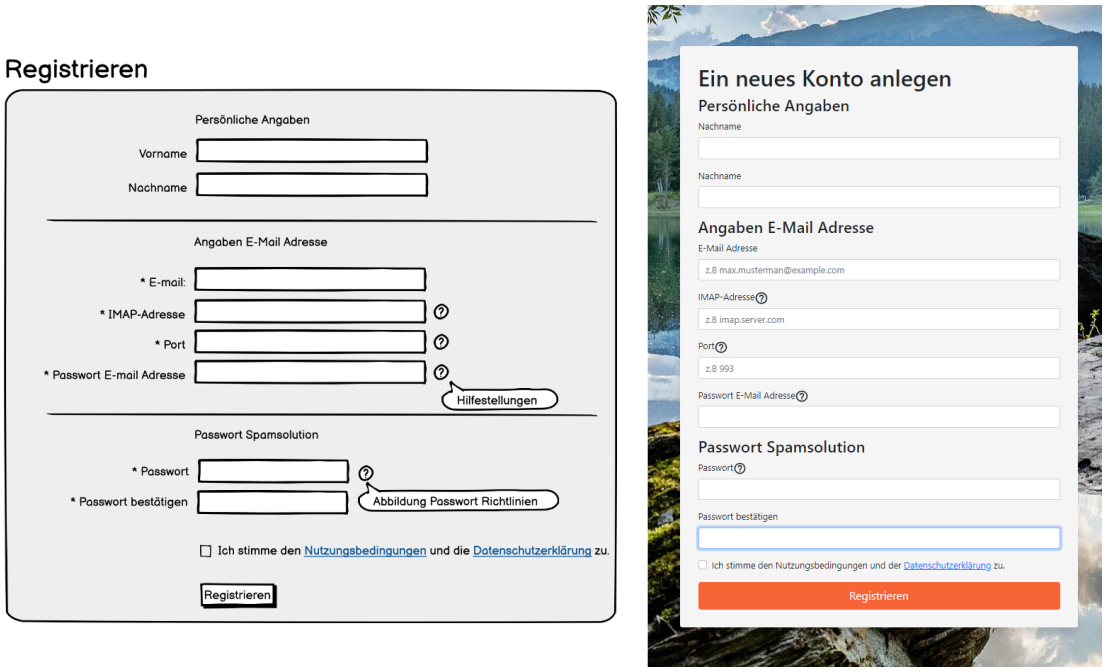


Abbildung 5.2: Vergleich der Registrierung des Wireframes (links) mit der Registrierung des Endproduktes (rechts).

5.1.3 Startseite

Ursprünglich war angedacht, dass nach dem Login zunächst eine Willkommensseite angezeigt wird. Auf dieser sollten Kacheln dazu dienen, zwischen den unterschiedlichen Aspekten des Services zu navigieren. Im Verlauf des Projektes stellte sich heraus, dass eine Navigationsleiste im oberen Teil der Seite sinnvoller und benutzerfreundlicher ist, als eine dedizierte Ansicht für die Navigation zu haben. Daher wurde dieses Wireframe nicht umgesetzt.

5.1.4 White- und Blacklist

Die Ansicht zum Anzeigen der White- und Blacklist unterscheidet sich am meisten von den Wireframes. Aufgrund des Feedbacks des Usability Tests wurden die Elemente komplett neu angeordnet und die Seite mit mehr Text befüllt. Die Pfeile in der Mitte wurden für das bessere Verständnis durch Buttons ersetzt. Zusätzlich gibt es ein Feld, bei dem der Benutzer seine E-Mail-Adresse sieht. Wegen der grossen Veränderungen nach dem Usability Test war es uns nicht möglich, die Suchfunktion umzusetzen (vgl. Abbildung 5.3).

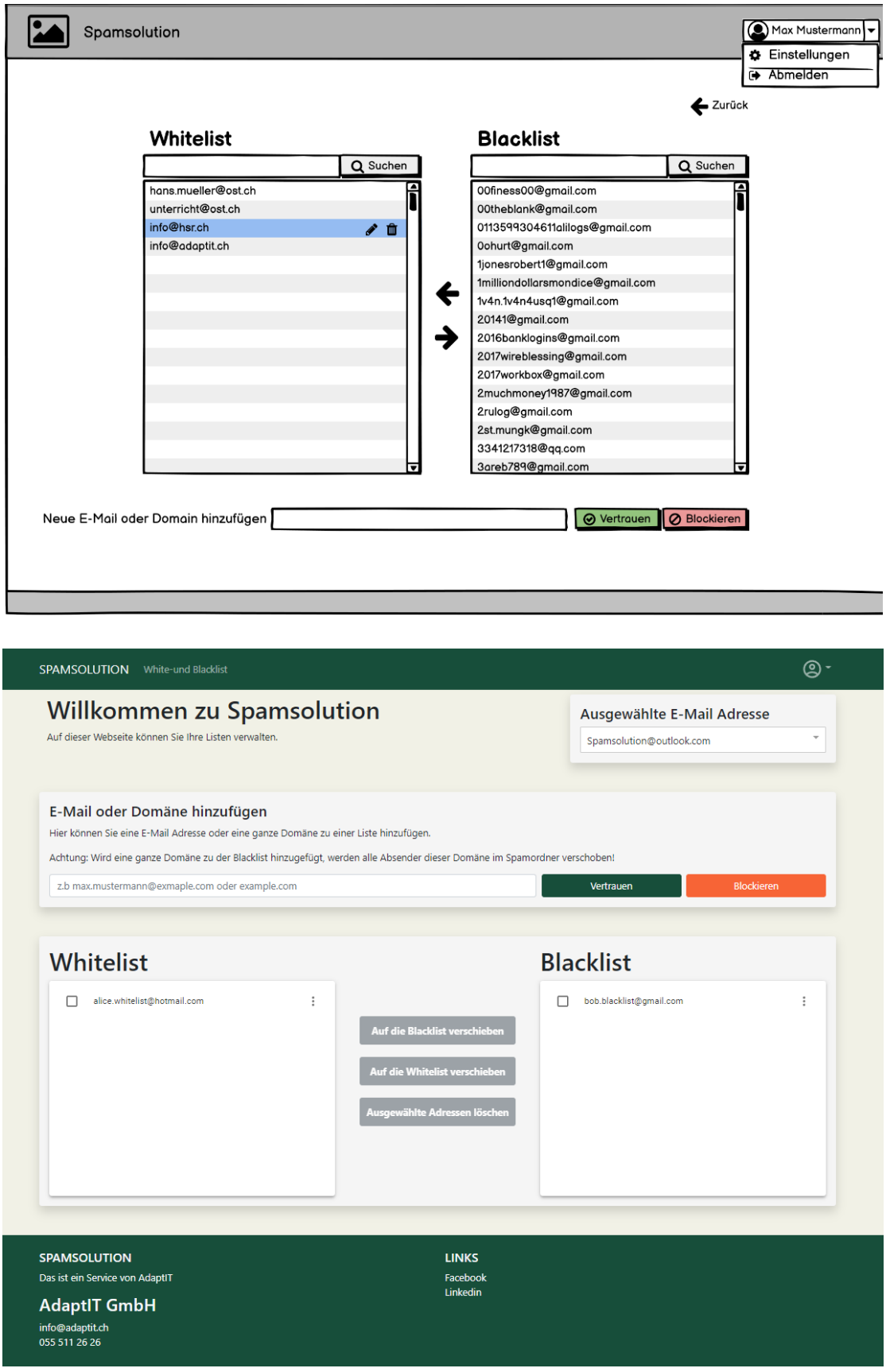


Abbildung 5.3: Vergleich der Ansicht der Listen des Wireframes (oben) mit der Ansicht der Listen des Endproduktes (unten).

5.1.5 Abonnierbare Listen und Listenverwaltung Admin

Die Ansichten „Abonnierbare Listen“ und „Listenverwaltung Admin“ sind im Endprodukt nicht enthalten, da dies optionale UCs sind und zeitlich nicht mehr umzusetzen waren.

5.1.6 E-Mail-Banner

Beim Banner gab es im Vergleich zum Wireframe mehrere Änderungen. Der farbige Hintergrund wurde durch eine Umrandung ersetzt, um die Lesbarkeit des Textes zu erhöhen. Der Banner ist simpel gestaltet, damit er bei möglichst vielen Mail-Anwendung identisch dargestellt wird (vgl. Abblidung 5.4).

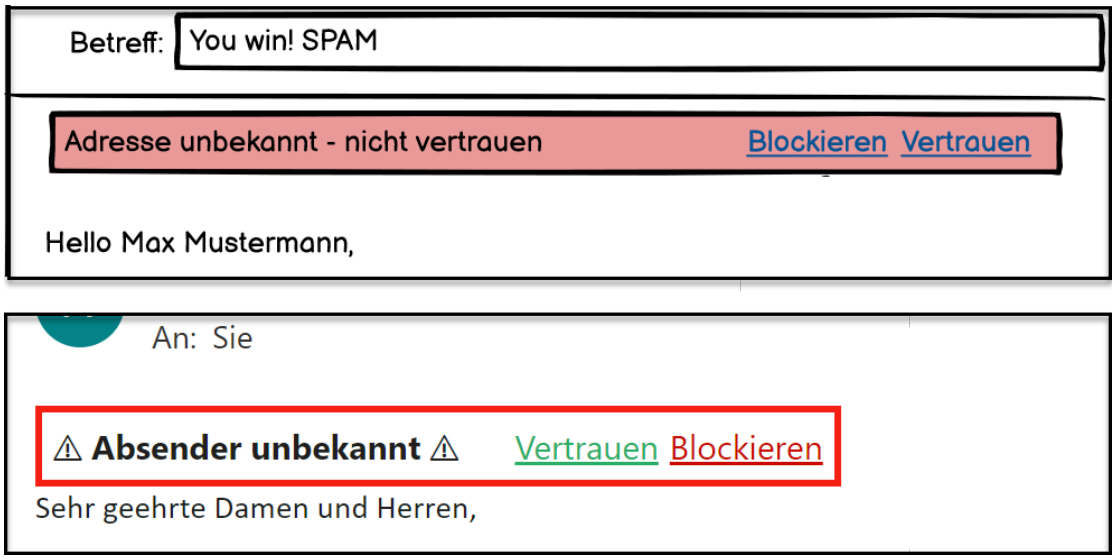


Abbildung 5.4: Vergleich des Banners des Wireframes (oben) mit dem Banner des Endproduktes (unten).

5.2 Verbesserungsmöglichkeiten

Eine Applikation wird niemals perfekt sein und bietet deshalb immer Raum für Optimierungen. Daher gehen wir im folgendem Abschnitt auf Verbesserungsvorschläge ein, die von uns zeitlich nicht mehr umgesetzt werden konnten. Weiterhin wird ein Bug erläutert, der nicht mehr rechtzeitig zu beheben war.

5.2.1 Vereinfachtes Deployment

Das initiale Deployment der Applikation ist nicht trivial. Bevor die Applikation gestartet wird, muss sichergestellt sein, dass benötigte Abhängigkeiten auf dem Server installiert sind. Es wäre sinnvoll für die Datenbank, das Python-Script und das Backend jeweils einen Docker-Container zu erstellen. Dies würde das Deployment stark vereinfachen.

5.2.2 Darstellungsbug

Für das Feld, das neue Adressen zu der White- oder Blacklist hinzufügt, sollten die beiden Buttons „Vertrauen“ und „Blockieren“ gemeinsam unterhalb des Inputfeldes angezeigt werden. Allerdings kommt es vor, dass bei gewissen Grössen des Bildschirms nur Letzterer unter dem Inputfeld angezeigt wird (vgl. Abbildung 5.5). Der Bug manifestiert sich jedoch nur bei spezifischen Formaten, weshalb er zu spät bemerkt wurde.

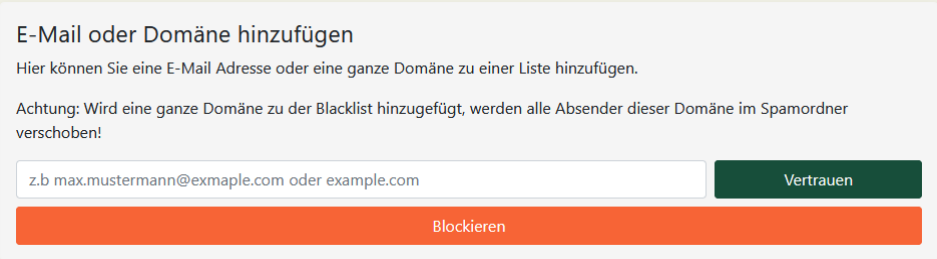


Abbildung 5.5: Darstellungsbug

5.2.3 Parallelisierung Python-Script

Das Bottleneck der Applikation ist das Python-Script. Dieses muss sich mit den Zugangsdaten der User an den jeweiligen IMAP-Servern anmelden und die E-Mails herunterladen. Dieser Prozess ist zeitaufwendig und kann von uns nicht beschleunigt werden. Daher empfehlen wird das Python-Script zu parallelisieren oder sogar auf mehrere Server zu verteilen. Ansonsten würde es bei einer hohen Anzahl von Benutzern zu Verzögerungen zwischen dem Empfang und der Bearbeitung von E-Mails kommen.

5.3 Zeitrapport

Gemäss dem Leitfaden der OST liegt der Aufwand für die BA bei 360 Stunden pro Student. Dementsprechend ergibt sich für das Projekt ein Gesamtaufwand von 720 Stunden.

Der tatsächliche Aufwand der Arbeit liegt bei 730 Stunden. Somit wurde die geplante Arbeitszeit leicht überschritten.

Die folgenden Diagramme zeigen, wie sich die Arbeitszeit aufgeteilt hat.

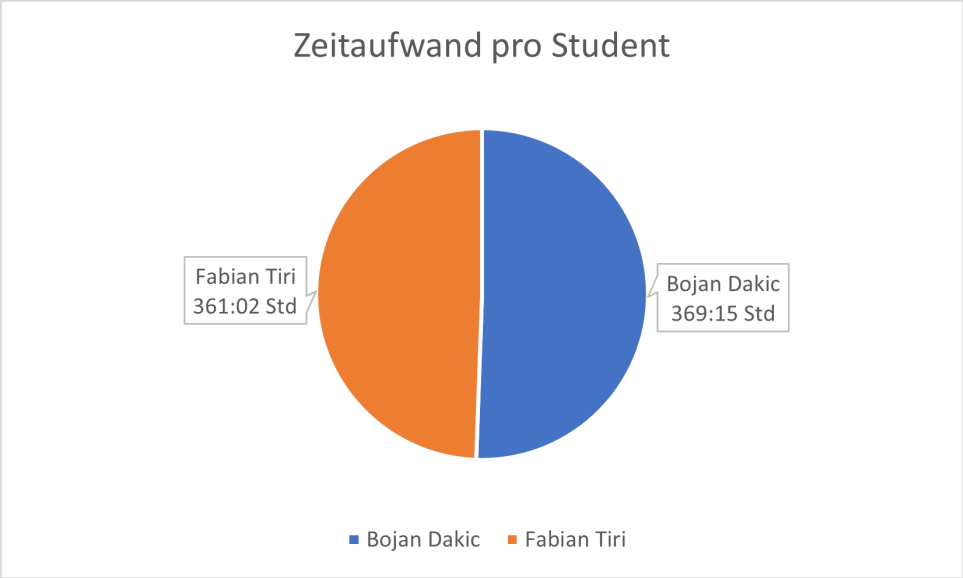


Abbildung 5.6: Zeitaufwand pro Student

.

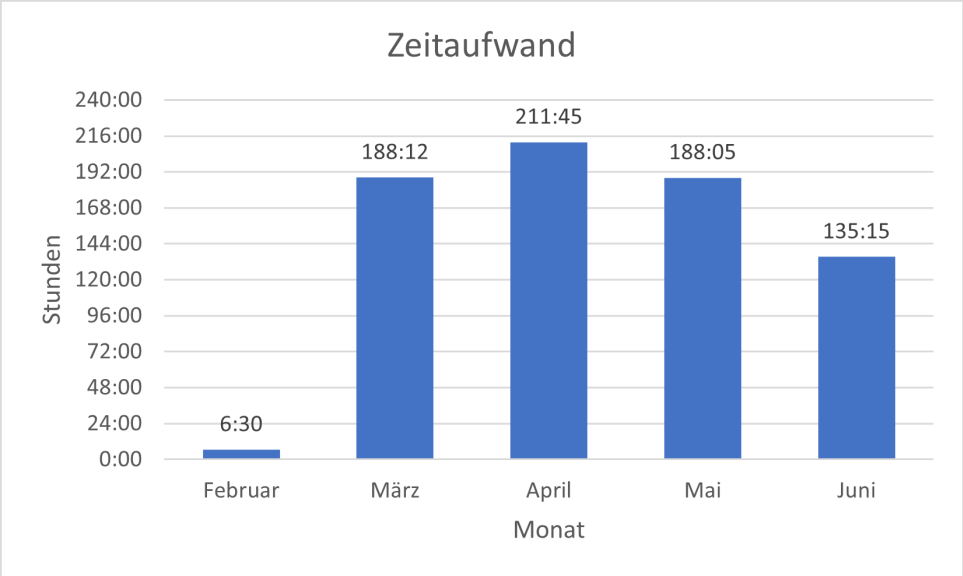


Abbildung 5.7: Zeitaufwand pro Monat insgesamt

.

Bemerkung Im Februar musste die finale Aufgabenstellung ausgebessert werden, weshalb der Aufwand in diesem Monat geringer ausfiel, als er sollte.

5.4 Fazit

In dieser Arbeit wurde eine Applikation entwickelt, die das Problem von Spam mit einem alternativen Ansatz löst. Es war zeitlich nicht möglich alle gewünschten Funktionen umzusetzen. Allerdings kann man bereits jetzt die Applikation mit den grundlegendsten Funktionen verwenden.

Die verpflichtenden UCs, die gemeinsam mit dem Kunden vereinbart wurden, sind alle umgesetzt worden. Ein Usability Test half dabei die Applikation unter möglichst realen Bedingungen zu testen und lieferte viele Inputs mit denen die Erfahrung für Benutzer verbessert werden konnte.

Zusammenfassend lässt sich sagen, dass das Projekt erfolgreich abgeschlossen werden konnte.

5.5 Ausblick

Wie bereits oben erwähnt konnte nicht die gesamte Vision des Kunden umgesetzt werden. Die noch offenen UCs kann man dem dritten Systemtest entnehmen. Wir empfehlen, sich bei der weiteren Entwicklung zunächst dem UC 12 zu widmen. Sobald die Weiterleitung an den Exchange-Server implementiert ist, kann die Software bereits produktiv verwenden werden.

UC 07 würde Benutzern einen hohen Mehrwert liefern und sollte deswegen auch priorisiert werden. Allerdings ist die Entwicklung eines Algorithmus hierfür schwierig und zeitaufwendig. Daher empfehlen wir eine passende Open Source Lösung zu integrieren und diese gemäss den persönlichen Anforderungen anzupassen.

6 Anhang

6.1 Auflistung verwendeter Libraries

6.1.1 Frontend

- React
- Bootstrap

6.1.2 Backend

- Node.js
- Express.js
- body-parser
- cookie-parser
- jest
- mysql2
- tweetnacl
- tweetnacl-util

6.1.3 Python-Script

- pynacl
- email
- imaplib
- requests
- unittest

6.2 Aufgabenstellung

OST
Ostschweizer
Fachhochschule

Aufgabenstellung Bachelor Thesis „Spam Solution“

Autor: **Frank Koch**
Version: **2.0**

Anpasst am: **26.02.22**

1. Beteiligte Personen

- Diplomanden: Bojan Dakic und Fabian Tiri
- Industriepartner: AdaptIT GmbH, Michael Güntensperger
- Experte: Hansjörg Huser
- Betreuer: Frank Koch

2. Problembeschrieb

Spam ist ein Dauerproblem. Obwohl die Algorithmen für die Sortierung bereits sehr gut sind, kommt der Fall «false positive» häufig vor, also dass ein legitimes Mail im Spam landet.

Diese Arbeit soll das Problem wie folgt adressieren:

- Grundsätzlich werden nur Mails von Adressen/Absendern empfangen, welche auf einer Whitelist stehen.
- Mails unbekannter Absender werden geprüft und mit einem Rating versehen (mögliche Parameter: Links im Mail, Rechtschreibung, Dokumente im Anhang, etc).
- Bevor das Mail in den Posteingang des Empfängers verschoben wird, werden das Rating und ein Button «Zu Whitelist hinzufügen», zuoberst am Mail angefügt.
- Zusätzlich soll eine simple Website erstellt werden, welche es ermöglicht, die Whitelist anzusehen und Mail-Adressen wieder daraus zu löschen.

Wir erhoffen uns dadurch ungewünschte Mails als Spam zu kategorisieren sowie Phishing-Versuche durch z.B. falsche Post / Swisscom-Rechnungen unterbinden zu können.

3. Aufgabenstellung

Grundlage dieser Aufgabenstellung ist der für das FS22 gültige Leitfaden für Bachelor- und Studienarbeiten[1].

Ziel ist zu zeigen, dass eine Spam-Lösung wie oben beschrieben umsetzbar ist und bereits erste Teile zu implementieren. Die implementierten Teile sollen deployed werden.

Technische Umgebung

- Die AT Provider AG wird die Cloud-Umgebung im eigenen Rechencenter für das Testing / Deployment zur Verfügung stellen oder in der Cloud
- Im Web-Bereich wird mit JavaScript (Node.js/React) gearbeitet, falls weitere Automatisierungsscripts zu entwickeln sind, wird Python eingesetzt.

Funktionale Anforderungen

- Mails per IMAP von Mailserver abfragen
- Inhalt von Mails scannen (in einem ersten Schritt geht es um den Absender)

[1] https://teams.microsoft.com/l/file/A7522E3F-6931-46F7-A965-B2A9081E71EF?tenantId=a6e70fa3-1c7a-4aa2-a25e-836eea52ca22&fileType=pdf&objectUrl=https%3A%2F%2Fostch.sharepoint.com%2Fteams%2FTS-StudiengangInformatik%2FFreigegebene%20Dokumente%2FStudieninformationen%2FLeitfaden%20BA%20SA%20v1.0.pdf&baseUrl=https%3A%2F%2Fostch.sharepoint.com%2Fteams%2FTS-StudiengangInformatik&serviceName=teams&threadId=19:88ac24bbee8e4682a73e81839cd2103c@thread.tacv2&groupId=4ad37fbb-4c36-4982-8bb5-971b8a539d2d

- Datenbank mit Mail-Adressen (White- und Blacklist) aufbauen
- Web-Interface zur Verwaltung von White- und Blacklist erstellen
    - Registrierung neuer Benutzer
    - Erfassen und Löschen von Mail-Adressen
    - Managen der IMAP Zugangsdaten
- Mails z.B. mit Python-Script verschieben (abhängig von White- /Blacklist der Absender-Mail-Adresse)
- Informationen in Mails ergänzen (Link zum White- / Blacklisten)
- Deployment in die Cloud / auf Server

Optionale Anforderungen

- Organisationsaufbau (mehrere Benutzer gehören einer Organisation an und Teilen die White- und Blacklist), inklusive Superuser für das verwalten der Liste.
- Neue User können sich optional einer organisationweiten White- / Blacklist angliedern (Subscriben), [Schwarmintelligenz]
- Inhalt von Mails analysieren (Links, Anhänge, Text,…) -> Rating abgeben.
  Virenschutz einbauen
- Outlook Plugin für den automatischen Export von Mail-Adressen mit denen man bereits Kontakt hatte

Nicht-Funktionale Anforderungen

- Das Entwicklerteam implementiert die Features gemäss der abgesprochenen Priorität mit dem Kunden
- Das Scannen eines Mail soll maximal zwei Sekunden dauern
- Das Backend (Mail-Adressen-Verwaltung) sollte 100 Requests pro Minute handeln können
- Jede Seite sollte nicht länger als 200ms für das Laden benötigen
- Die Web-Applikation sollte auf Firefox, Chrome und ~~Safari~~ laufen.
- Via Internet sollte auf eine vom Kunden bereit gestellte Domain zugegriffen werden können.
- Drei von vier Test-user sollten das UI (Kategorien: layout, responsiveness, colour, content) der Applikation mit einem Tablet mit einer Note von mindestens 7 von 10 bewerten, wobei 10 das Beste ist.
- Die Datenbank soll bis zu 10'000 Mail-Adressen managen können.
- Errors sollen keine Systemfehler erzeugen, aber eine Error Nachricht Zeigen und das System auf den vorherigen Zustand zurücksetzen**.**
- Jeder Error soll im System geloggt werden
- Jede Kommunikation zwischen Front- und Backend soll mit einem SSL-Zertifikat verschlüsselt werden.
- Daten welche in Eingabefelder abgefüllt werden, sollen zuerst validiert werden, bevor diese durch das System verarbeitet werden. SQL Injection test der Eingabefelder sollte keine Verletzlichkeiten zeigen.
- User-Passwörter werden nicht in plain-text in der Datenbank gespeichert.
- Wenn sich ein User in die Web-Applikation einloggt, werden ihm auch nur seine Daten / auf Daten die er Zugriff haben soll, angezeigt.
- Businesslogik im Backend soll modular aufgebaut werden, so dass sie erweitert werden kann.
- Die Backend-API soll durch API-testing Tools getestet werden.
- Implementierte Funktionalität (Datenbank, Backend, Frontend,…) sollen auf Digital Ocean oder dem RZ der AT Provider AG deployed werden.

4. Zur Durchführung

Mit dem Betreuer finden Besprechungen gemäss Absprache statt. Die Besprechungen sind von den Studierenden mit einer Traktandenliste vorzubereiten und die Ergebnisse in einem Protokoll zu dokumentieren, das dem Betreuer per E-Mail zugestellt wird.

Für die Durchführung der Arbeit ist ein Projektplan zu erstellen. Dabei ist auf einen kontinuierlichen und sichtbaren Arbeitsfortschritt zu achten. An Meilensteinen gemäss Projektplan sind einzelne Arbeitsresultate in vorläufigen Versionen abzugeben.

5. Dokumentation und Abgabe

Siehe Leitfaden Abschnitt 5.5 "Umfang und Form der Abgabe".

6. Termine

Es gelten folgende Termine:

- 21.02.2022 Beginn der Bachelorarbeit, Ausgabe der Aufgabenstellung durch den Betreuer/die Betreuerin.
- bis 13.06.2022 Die Studierenden erfassen den Abstract im Online Tool http://abstract.rj.ost.ch/ und geben den Abstract zur Kontrolle an ihren Betreuer/ihre Betreuerin frei.
- Vorlagen sowie eine ausführliche Anleitung betreffend Dokumentation stehen auf Teams zur Verfügung.
- 15.06.2022 Der Betreuer/die Betreuerin gibt das Dokument mit dem korrekten und vollständigen Abstract der Broschüre zur Weiterverarbeitung an das Studiengangsekretariat frei.
- 17.06.2022 Abgabe des Berichts an den Betreuer/die Betreuerin bis 17.00 Uhr
- 17.06.2022 Hochladen aller verlangten Dokumente auf https://avt.i.ost.ch/ .
- Abgabe des Berichts an den Betreuer/die Betreuerin bis 17.00 Uhr.
- 17.06.2022 Präsentation und Ausstellung der Bachelorarbeiten
- bis 31.08.2022 Mündliche BA-Prüfung

7. Bewertung

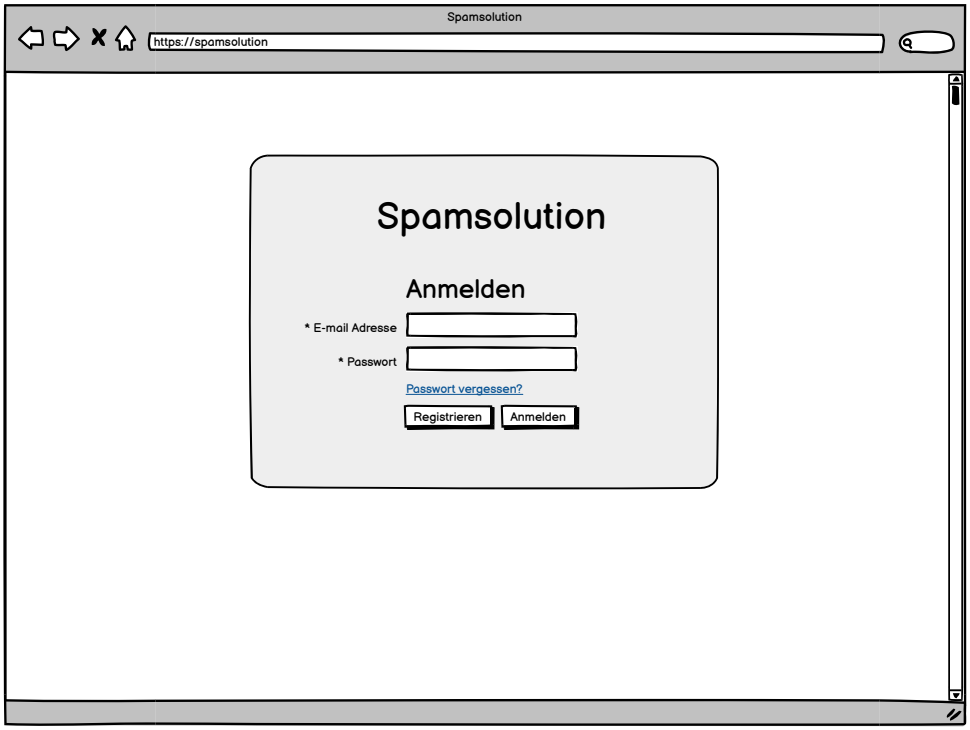
Siehe Leitfaden Abschnitt 6 "Bewertung", insbesondere 6.5.

Rapperswil, den 26.02.22

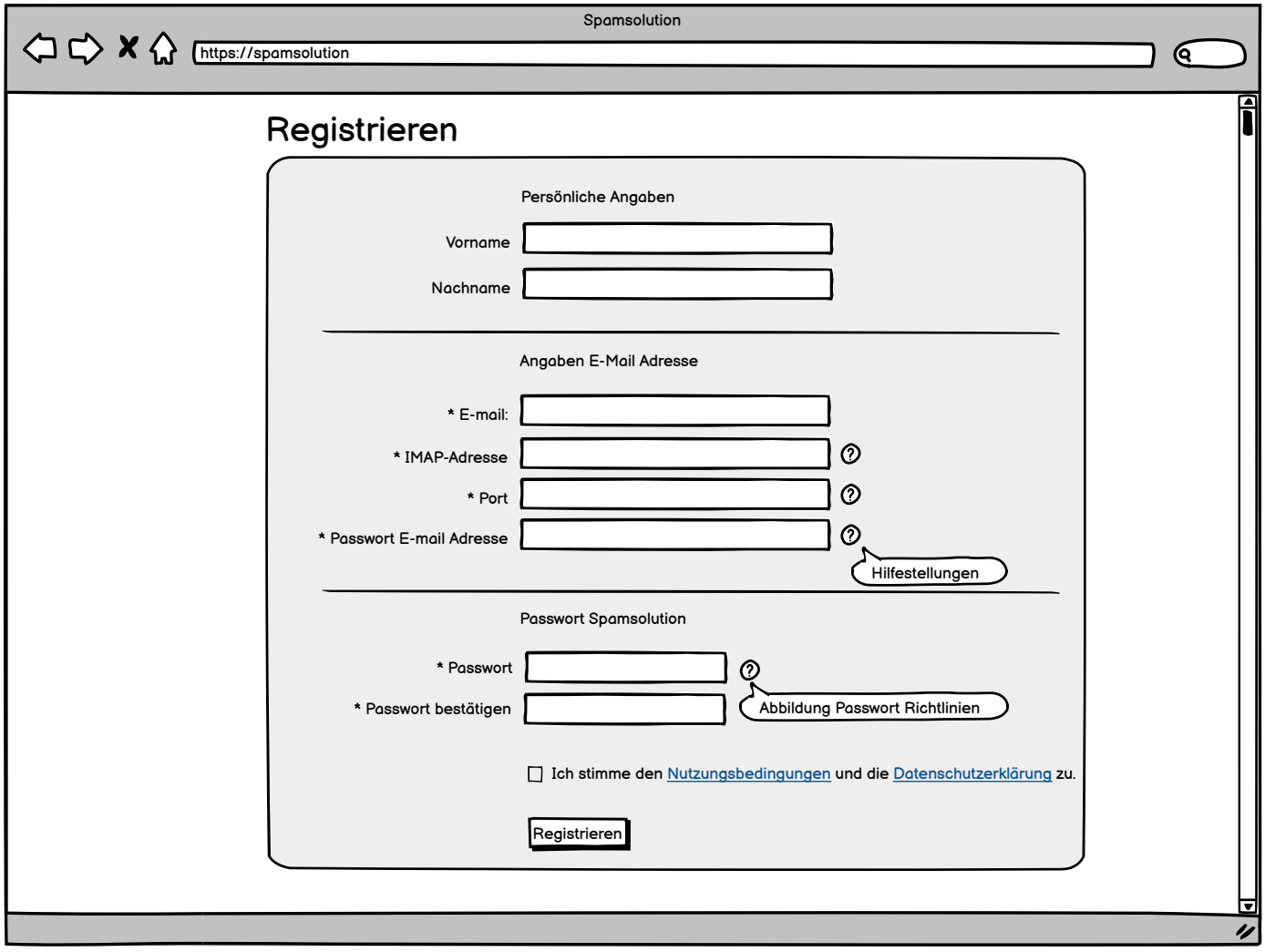
Frank Koch

6.3 Wireframes

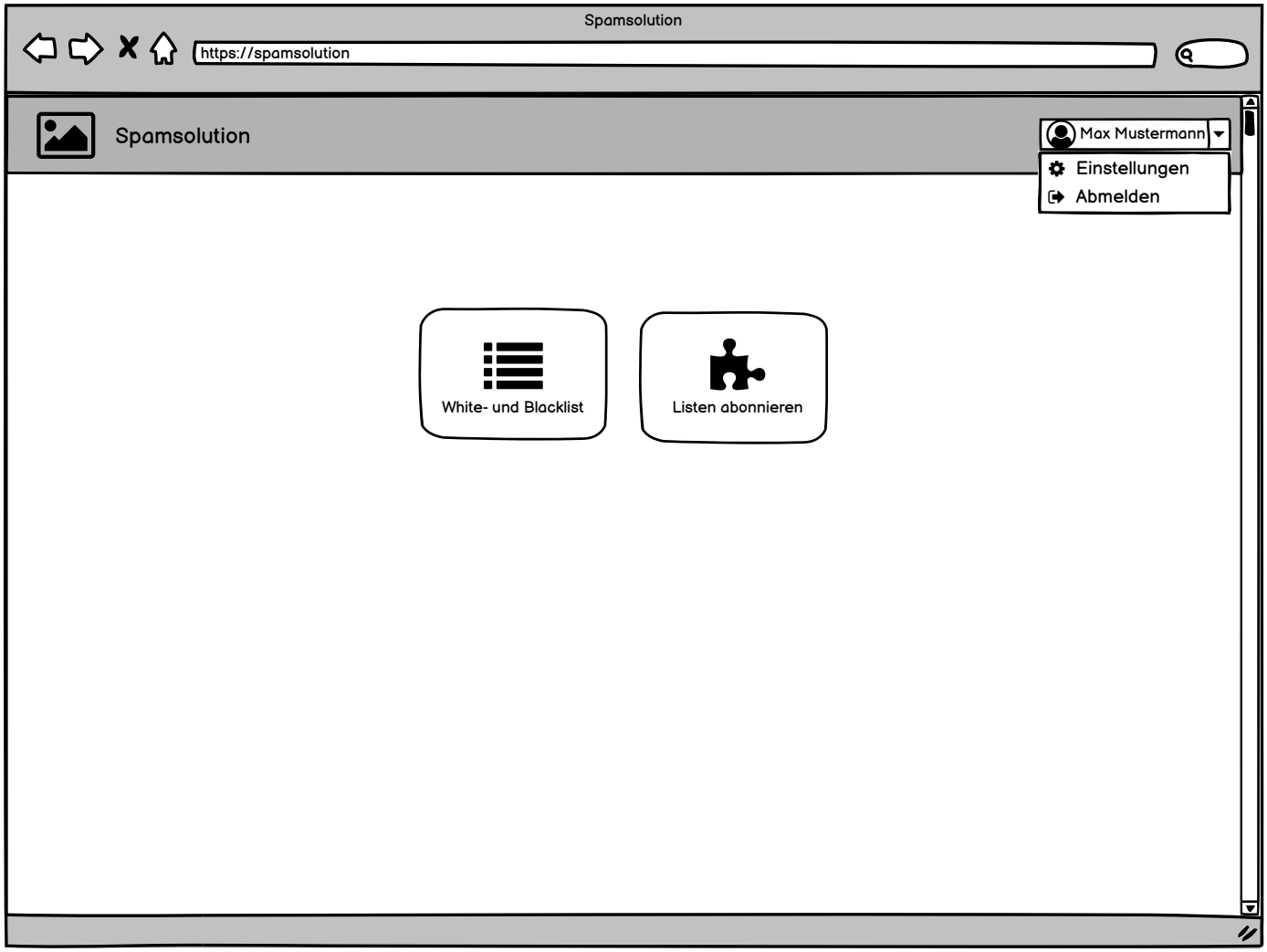
Anmeldefenster



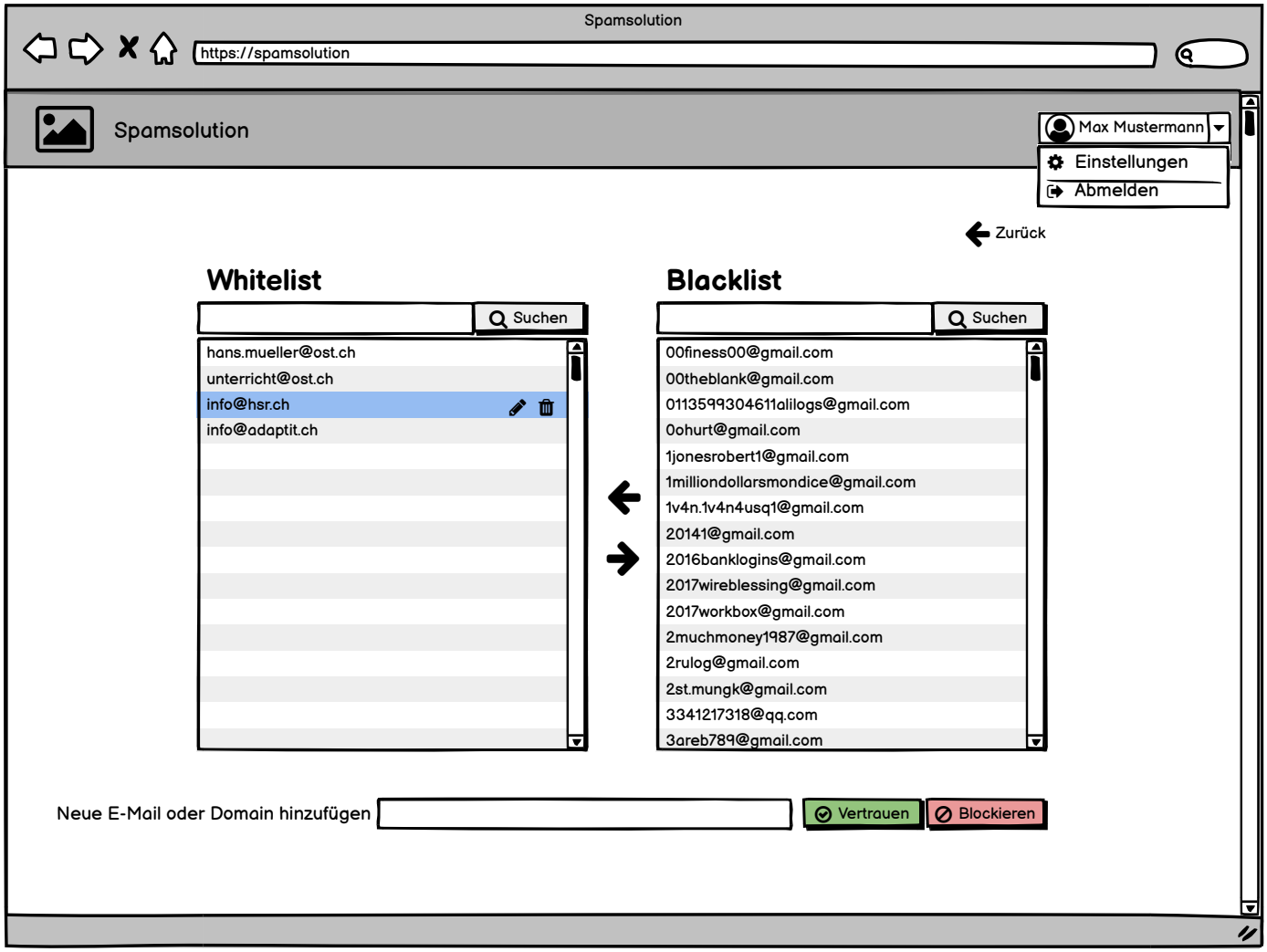
Registration



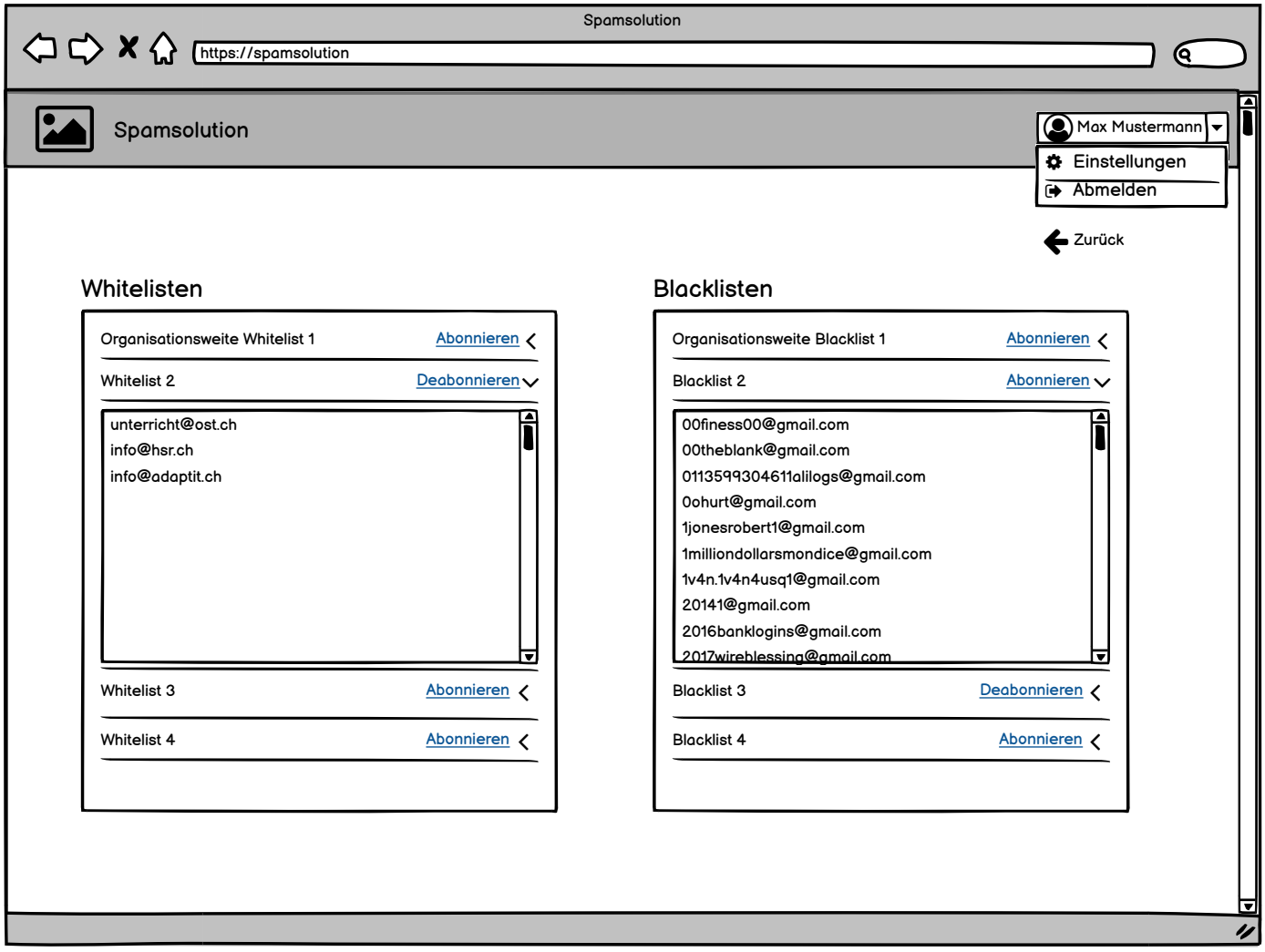
Startseite


Spamsolution
https://spamsolution
Spamsolution
Max Mustermann
Einstellungen
Abmelden
White- und Blacklist
Listen abonnieren

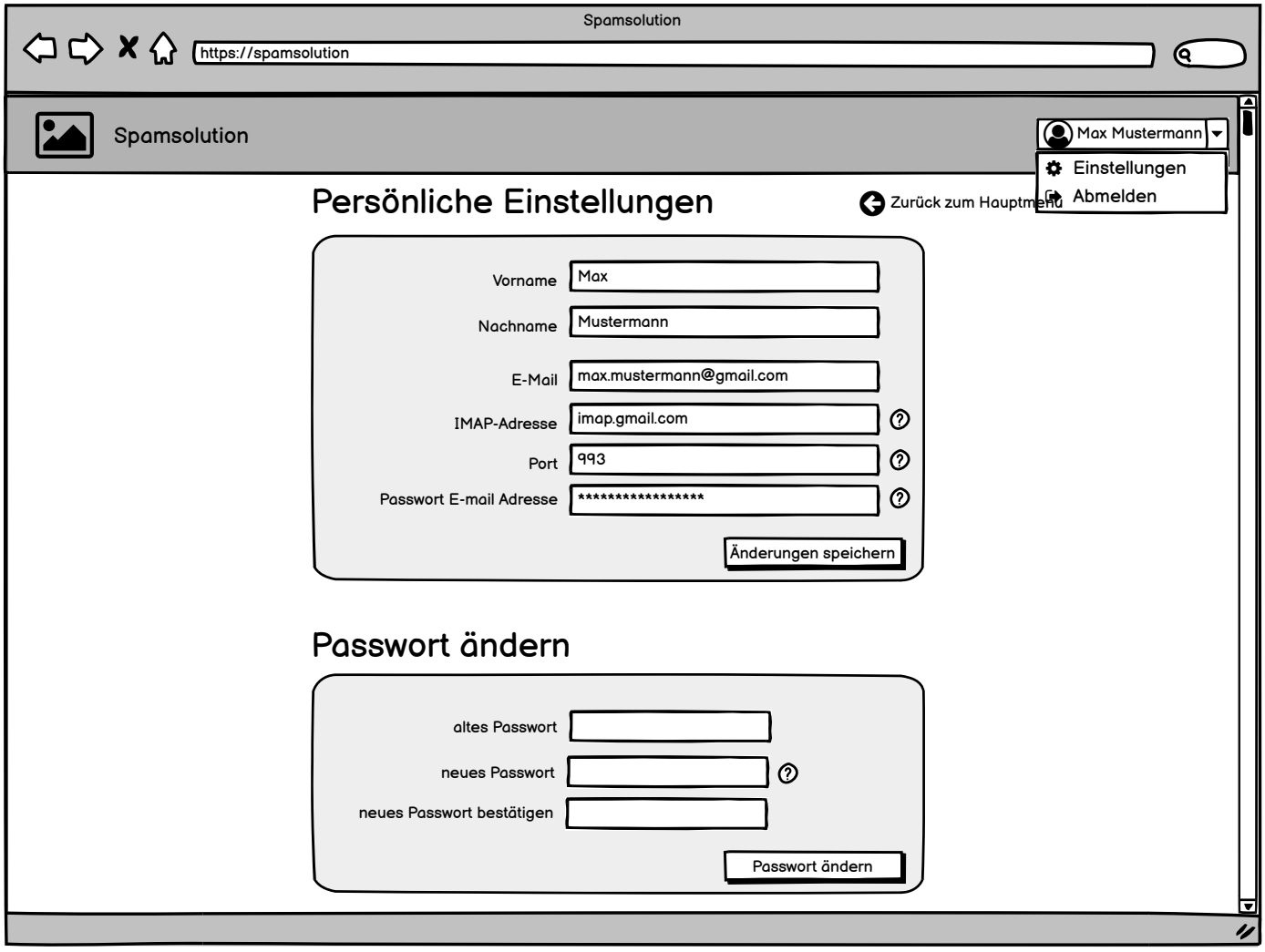
White- und Blacklist



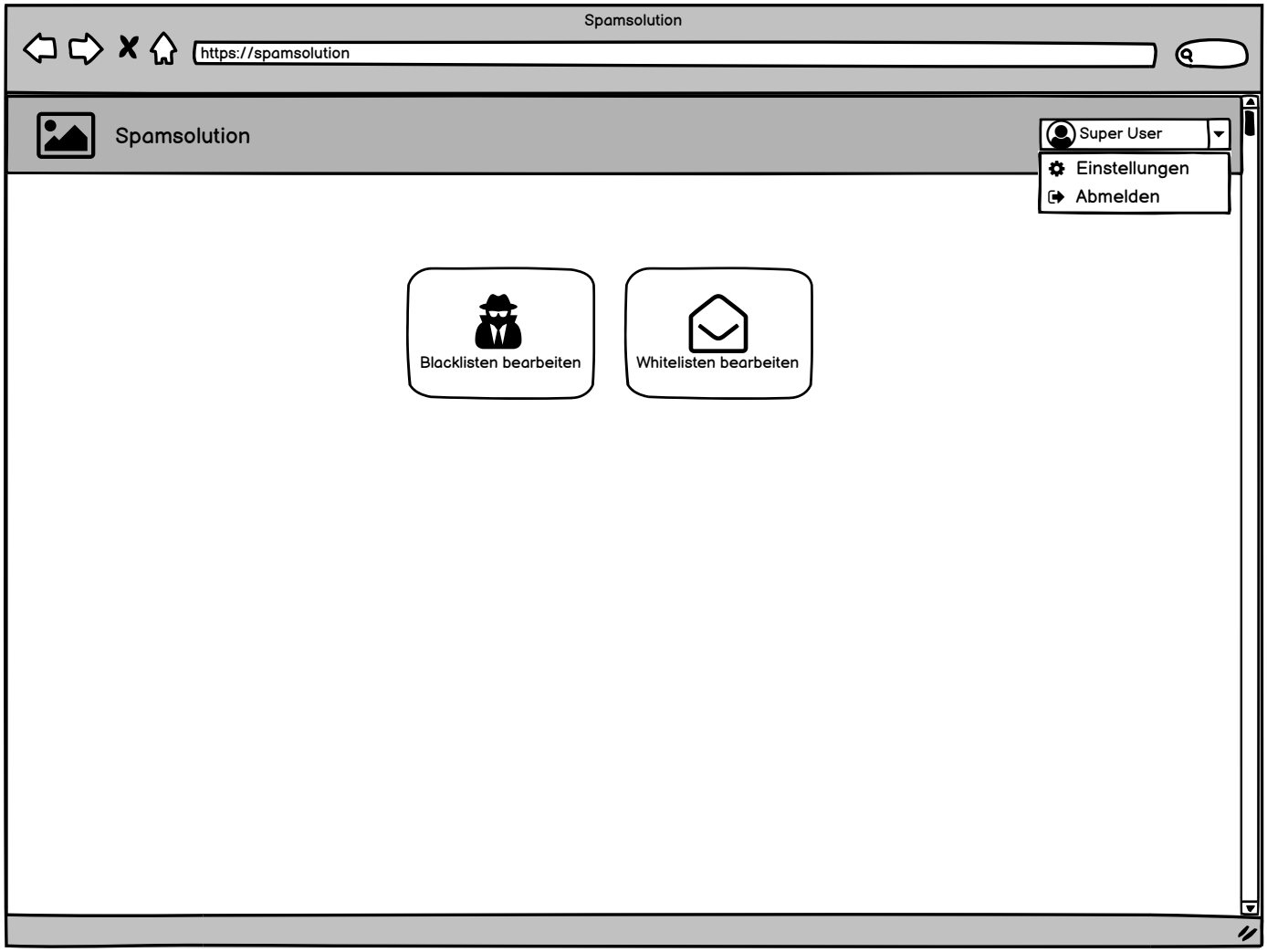
Ansicht Abonierbare Listen



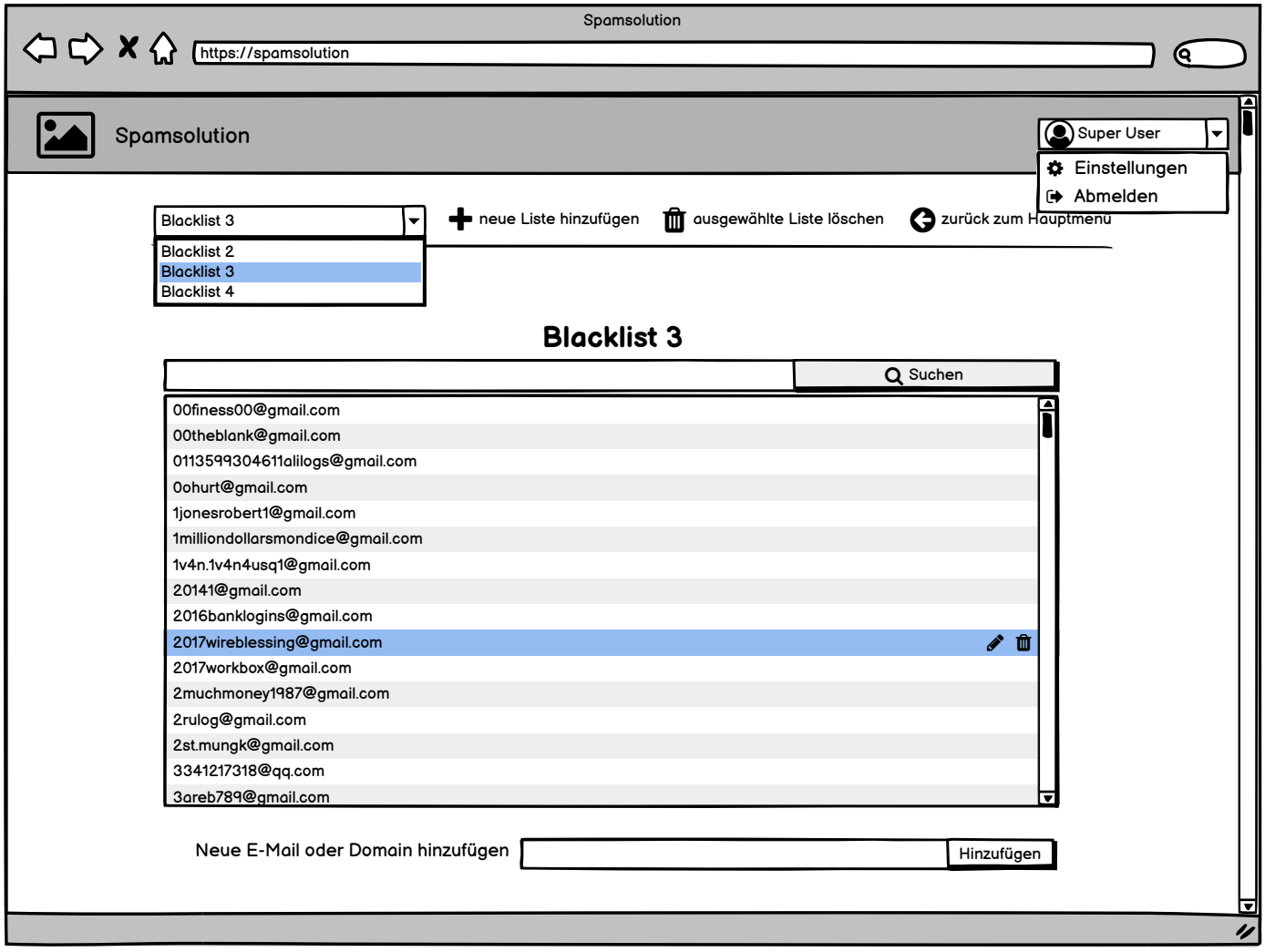
Einstellungen



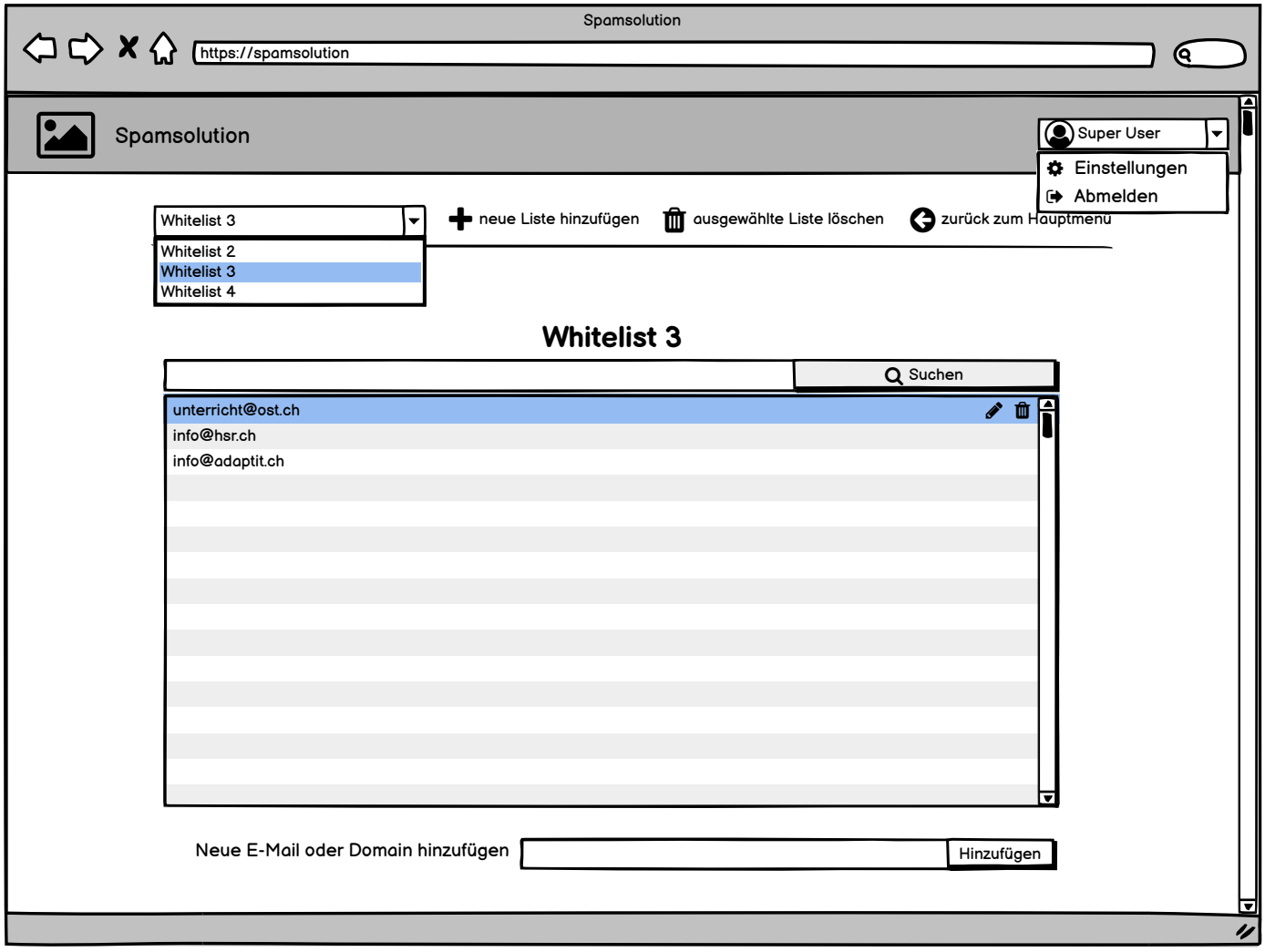
Admin_Startseite



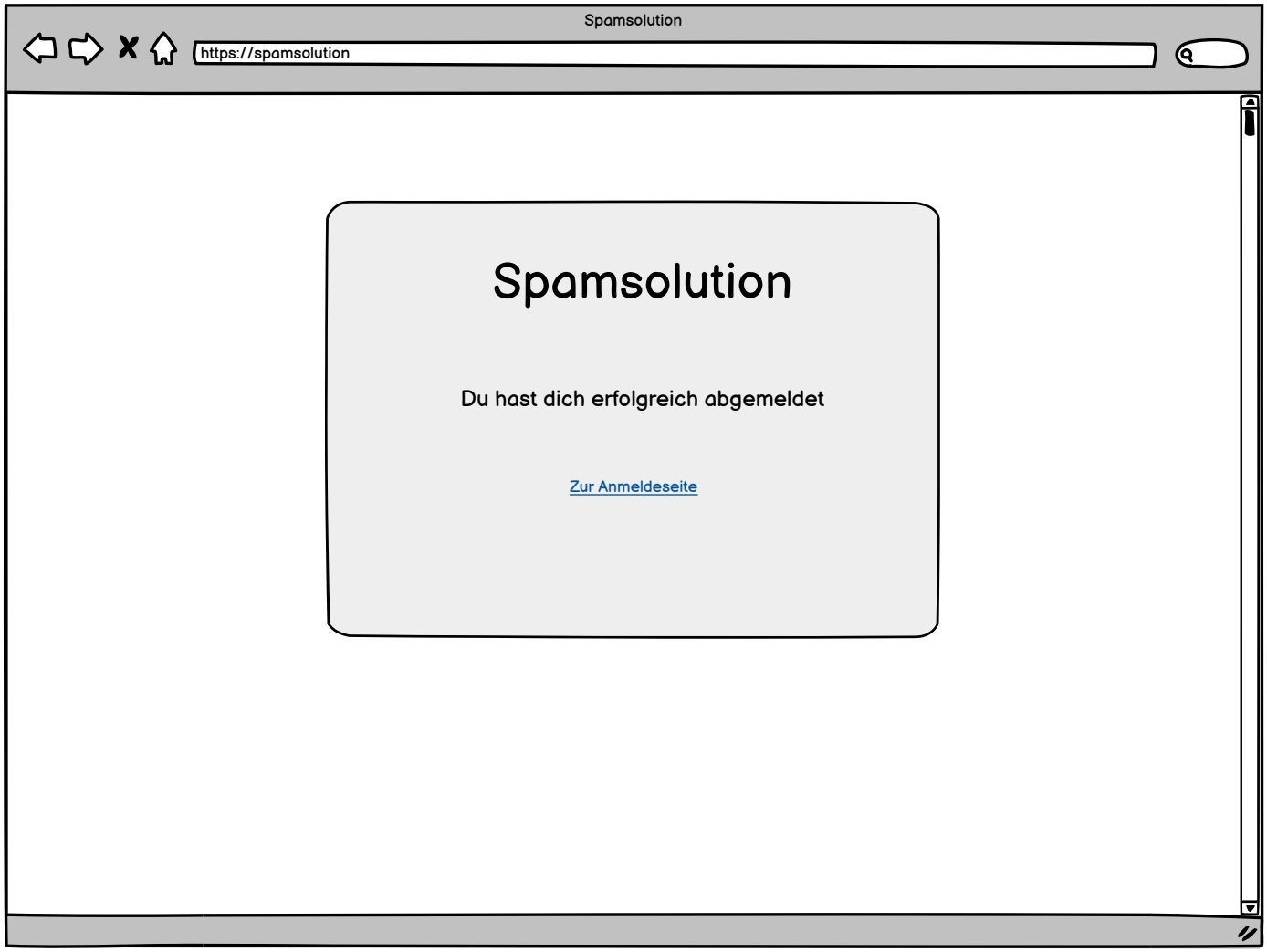
Admin_Blacklist



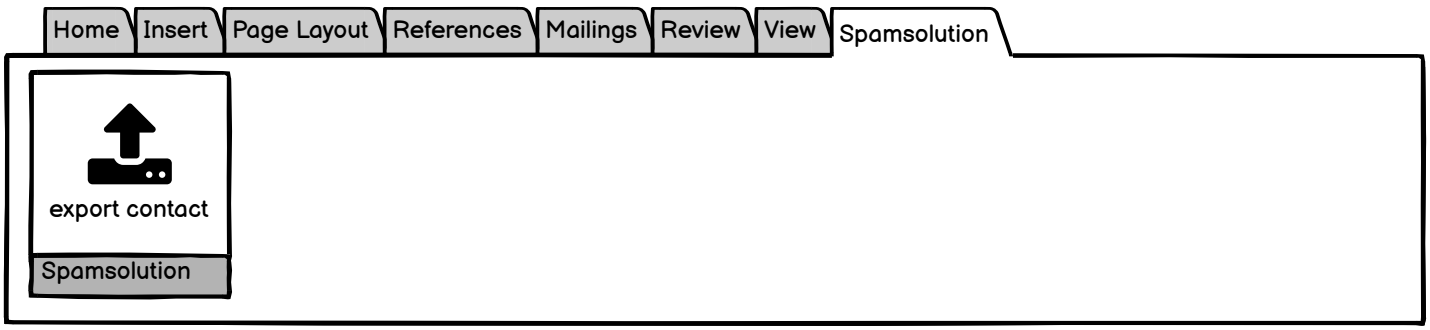
Admin_Whitelist



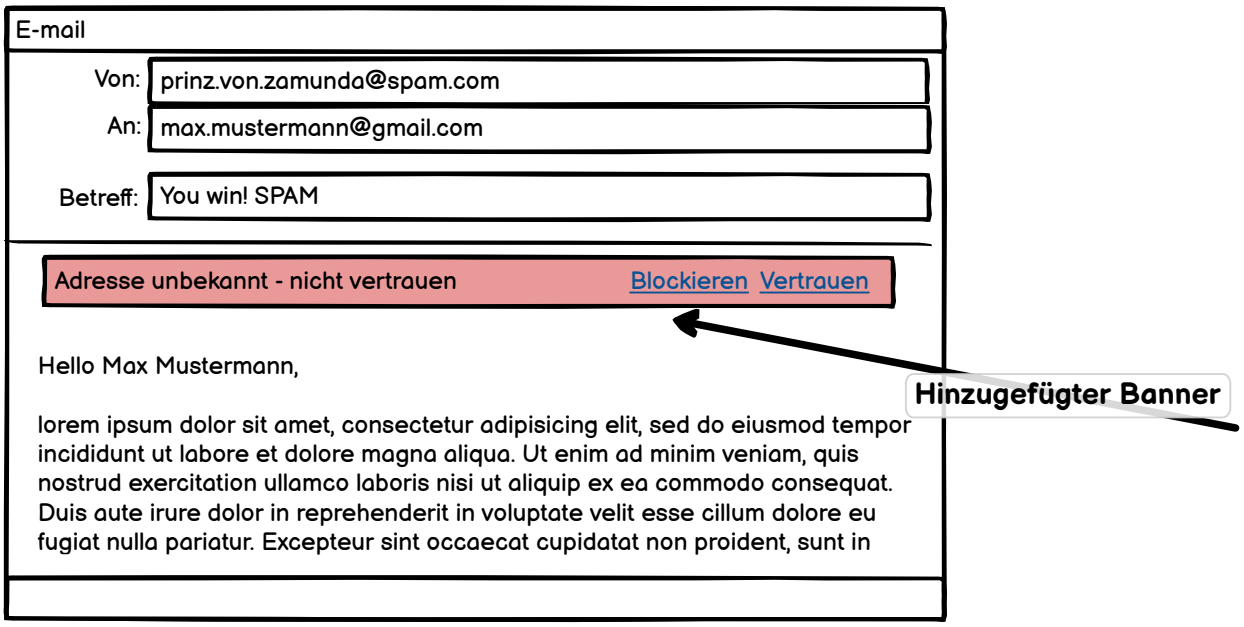
Abmeldeansicht



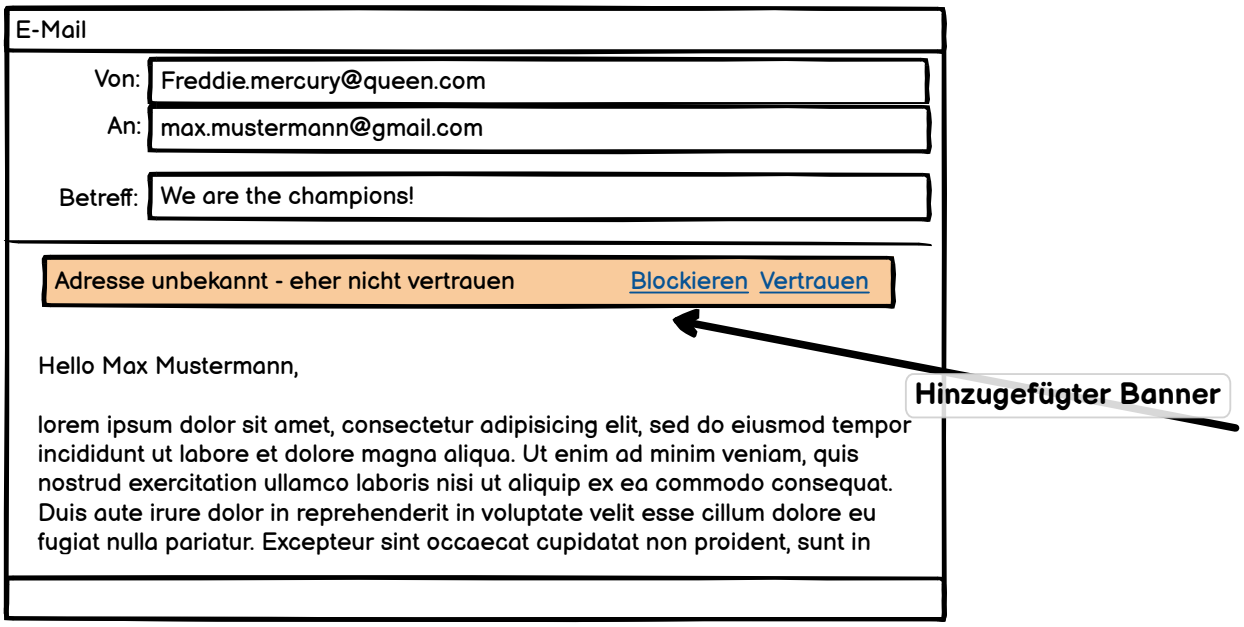
Outlook Plugin



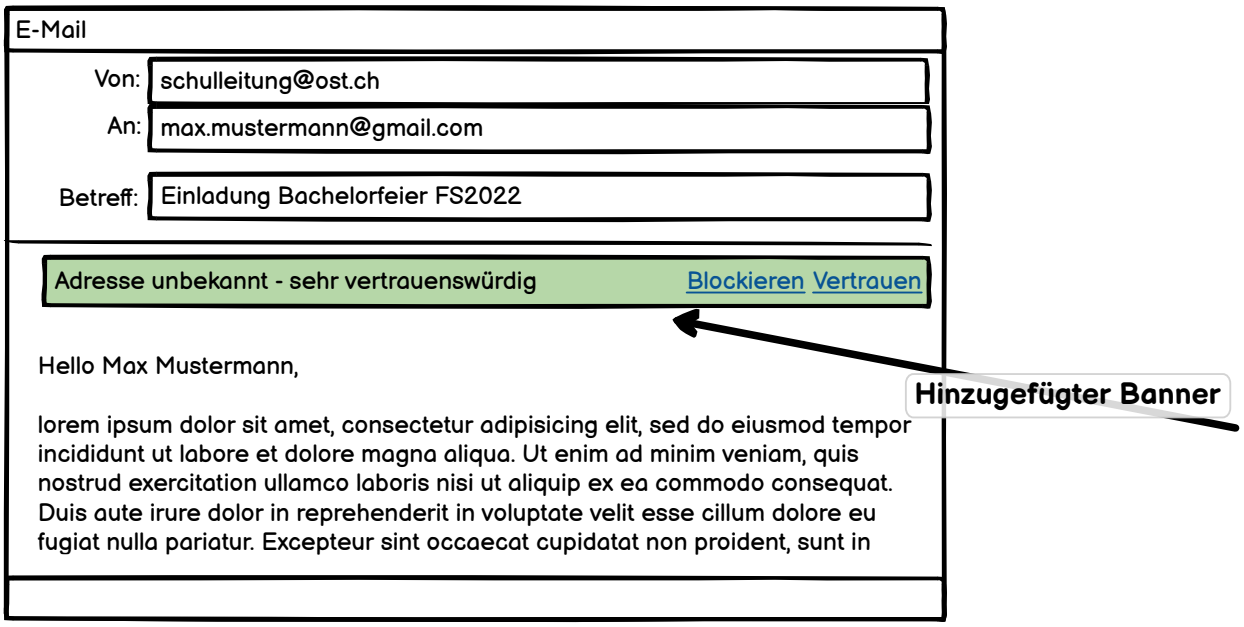
E-Mail Banner SPAM



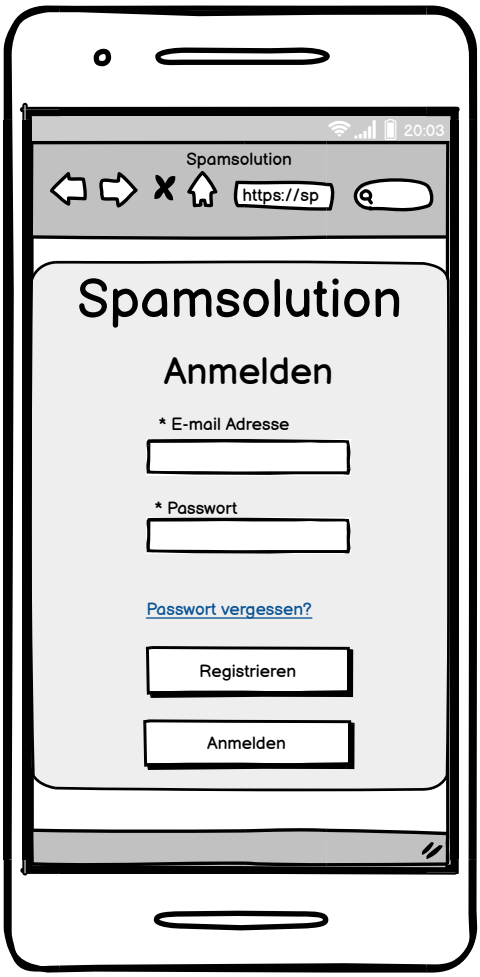
E-Mail Banner Mittel



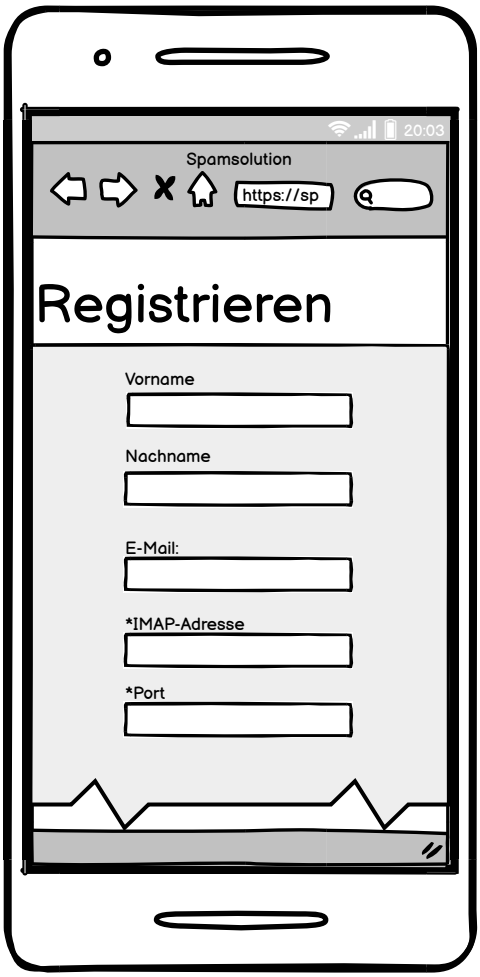
E-Mail Banner Vertrauenswürdig



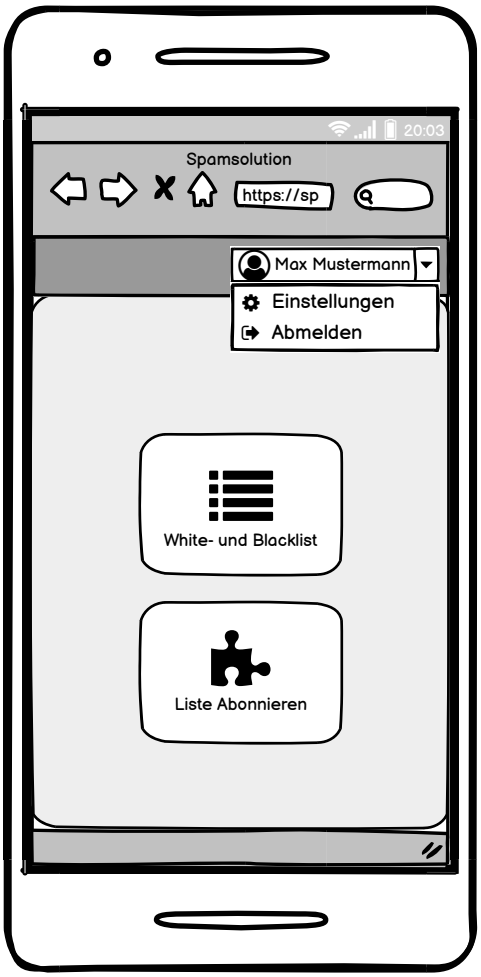
Responsive_Login



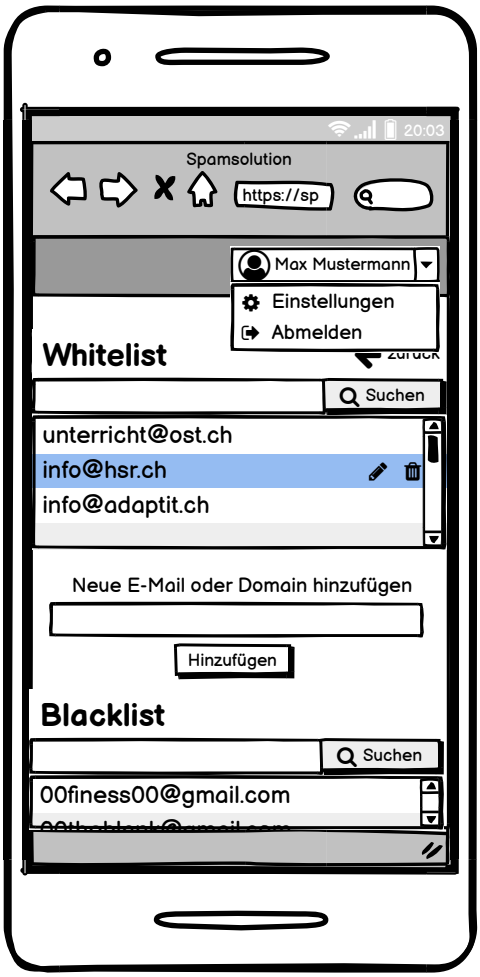
Responsive_Registration



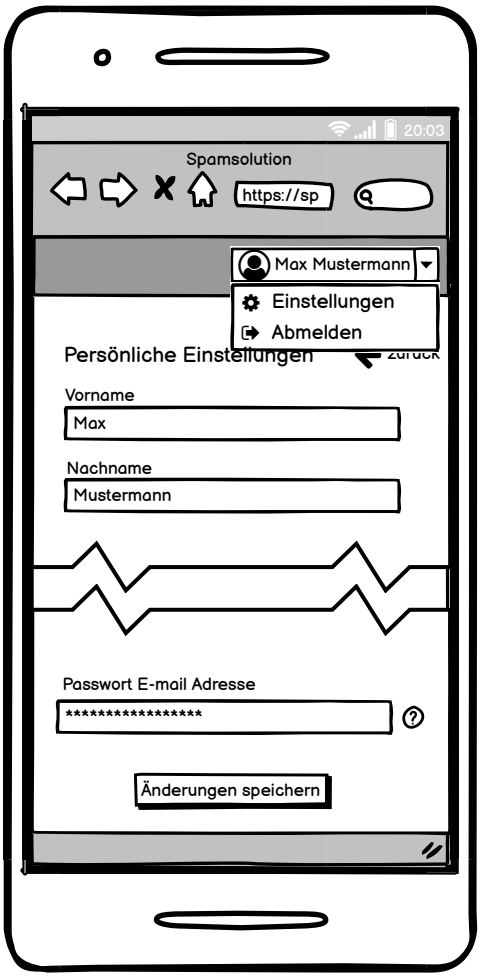
Responsive_Startseite



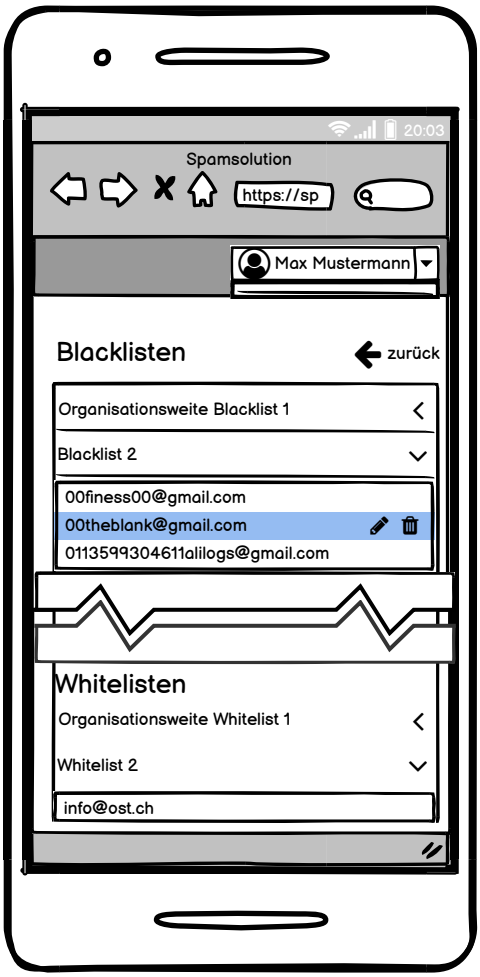
Responsive_White- und Blacklist



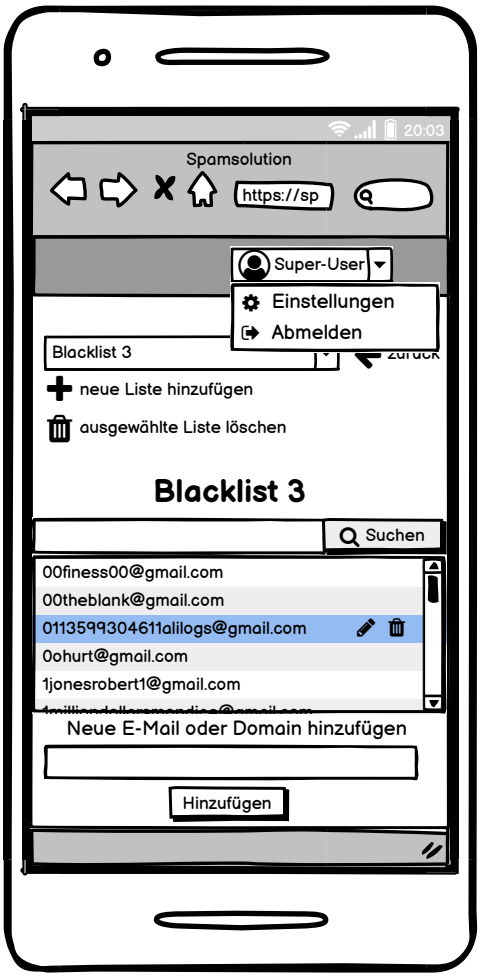
Responsive_Einstellungen



Responsive_Listen Abonnieren

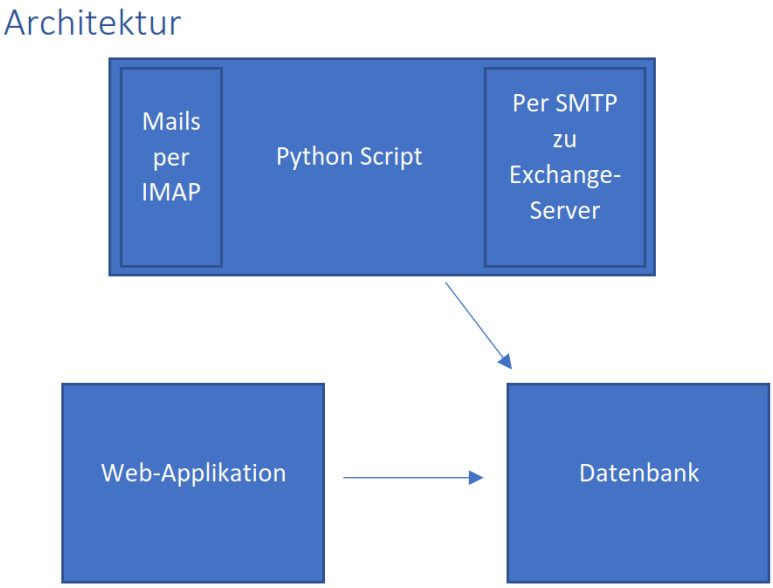


Responsive_Admin_Verwaltung Listen



Responsive_Abmelden

6.4 Vorgeschlagene Architektur



6.5 Risikoanalyse

Risiko	gewichteter Schaden
R01 - Zeitmanagement	6
R02 - Komplexität	5
R03 - Fehlendes Know-How	0.4
R04 - Ausfall eines Teammitglied	3
R05 - Ausfall der Infrastruktur	0.04
R06 - Probleme beim Deployment	0.08
R07 - Komplikationen Mail Abfragen	2
R08 - Ausfall von der Hardware	0.35
R09 - Missverständnis in der Kommunikation	3
Summe gewichteter Schaden:	19.87

Tabelle 6.1: Risikoanalyse 2

Risiko	gewichteter Schaden
R01 - Zeitmanagement	5
R02 - Komplexität	2
R03 - Fehlendes Know-How	0.2
R04 - Ausfall eines Teammitglied	3
R05 - Ausfall der Infrastruktur	0.04
R06 - Probleme beim Deployment	0.04
R07 - Komplikationen Mail Abfragen	0
R08 - Ausfall von der Hardware	0.35
R09 - Missverständnis in der Kommunikation	2
Summe gewichteter Schaden:	12.63

Tabelle 6.2: Risikoanalyse 3

Risiko	gewichteter Schaden
R01 - Zeitmanagement	4
R02 - Komplexität	0
R03 - Fehlendes Know-How	0
R04 - Ausfall eines Teammitglied	2
R05 - Ausfall der Infrastruktur	0.04
R06 - Probleme beim Deployment	0
R07 - Komplikationen Mail Abfragen	0
R08 - Ausfall von der Hardware	0.35
R09 - Missverständnis in der Kommunikation	1
Summe gewichteter Schaden:	7.39

Tabelle 6.3: Risikoanalyse 4

6.6 System Tests

System Test M2

Datum	06.04.2022
Tester	Fabian Tiri
Meilenstein	M2 End of Elaboration

Use Case	Implementiert	Fehler / Unschönheiten	Status
UC 01	Nein	-	0 %
UC 02	Nein	-	0 %
UC 03	Nein	-	0 %
UC 04	Nein	• Momentan wird nur User 1 vom Backend geliefert	50 %
UC 05	Nein	• Momentan wird nur User 1 vom Backend geliefert	50 %
UC 06	Nein	-	0 %
UC 07*	Nein	-	0 %
UC 08*	Nein	-	0 %
UC 09*	Nein	-	0 %
UC 10*	Nein	-	0 %
UC 11*	Nein	-	0 %
UC 12*	Nein	-	0 %
UC 13	Nein	-	0 %

** optionale UCs*

Tabelle 6.4: Systemtest 1

System Test M3

Datum	29.04.2022
Tester	Fabian Tiri
Meilenstein	M3 Alpha Release (MVP)

Use Case	**Implementiert**	**Fehler / Unschönheiten**	**Status**
UC 01	Nein	-	0 %
UC 02	Ja	-	100 %
UC 03	Nein	• Komplette Domain hinzufügen funktioniert noch nicht • Hinzufügen, verschieben und löschen von E-Mail-Adressen funktioniert nur lokal	40 %
UC 04	Ja	-	100 %
UC 05	Ja	-	100 %
UC 06	Nein	-	0 %
UC 07*	Nein	-	0 %
UC 08*	Nein	-	0 %
UC 09*	Nein	-	0 %
UC 10*	Nein	-	0 %
UC 11*	Nein	-	0 %
UC 12*	Nein	-	0 %
UC 13	Nein	-	0 %

** optionale UCs*

Tabelle 6.5: Systemtest 2

6.7 Usability Test Zusammenfassung

Testkonzept Usability Test

Wissensziele, Fragestellungen und Hypothesen

- Ist die App gut bedienbar?
- Ist die Optik der App zufriedenstellend?
- Ist eine Hilfestellung nötig?
- Sind die Begriffe verständlich?
- Was für Probleme treten auf?

Testform und Inhalt

- Durchspielen verschiedener Use-Cases

Geeignete Testpersonen

- Keine speziellen Anforderungen an die Testpersonen, aber sie sollten im Idealfall nicht aus der IT stammen.
- Demografie ab 18 Jahren

Anleitung für den Test-Durchführer

- Usability Tests sind qualitative Tests. Es geht um die Qualität und um die Funktionen der App.
- Testsetting: Am besten physisch zusammen mit der Testperson.
- Möglichst viele Beobachtungen und Bemerkungen notieren um möglichst viele Erkenntnisse zu gewinnen.
- Der Testperson soll nicht geholfen werden.

Testdurchführung

Vorbereitung

Der Testperson wird ein Link für die Webseite zur Verfügung gestellt. Weiterhin sollte die Testperson eine gültige E-Mail-Adresse besitzen, die für IMAP aktiviert ist. Möchte der User keine E-Mail-Adresse für den Test preisgeben wird von uns eine zur Verfügung gestellt.

Einführung

In diesem Test möchten wir die Applikation auf ihre Benutzerfreundlichkeit testen. Während du die App bedienst, artikuliere deine Gedanken, damit ich diese nachvollziehen kann. Ich werde dir jeweils eine Aufgabe vorlesen, die du daraufhin selbständig lösen sollst. Am Schluss muss noch ein kurzer Fragebogen ausgefüllt werden.

Szenario Registrierung

Du hast von der Spamsolution gehört und möchtest dich für den Service registrieren. Du navigierst auf die Seite um einen Account anzulegen.

Wissensziele

- Ist die Registrierung verständlich?
- Gab es Schwierigkeiten bei den Begrifflichkeiten (vor allem beim Passwort)
- Ist die Bedienung / Navigation zu umständlich?

Bemerkungen

- 7 x IMAP unverständlich
- 1 x Tippt zunächst in Login Feld
- 1 x Verlinkung zu Datenschutzerklärung fehlt.

Szenario An- und Abmeldung

Du hast dich nun erfolgreich registriert. Melde dich nun ab und anschliessend melde dich neu an.

Wissensziele

- Wurde der Abmeldebutton gefunden?
- Konnte der Benutzer sich erfolgreich wieder anmelden?

Bemerkungen

- 7 x konnten sich erfolgreich abmelden.
- 1 x Passwort vergessen

Szenario E-Mail hinzufügen

Nach der erfolgreichen Anmeldung möchtest du nun eine E-Mail Adresse deiner Wahl auf die White- und eine andere E-Mail Adresse auf die Blacklist hinzufügen.

Wissensziele

- Konnten die E-Mail Adressen erfolgreich hinzugefügt werden?
- Sind die Begrifflichkeiten verständlich?

Bemerkungen

- 7 x Mail erfolgreich hinzugefügt.
- 1 xZuerst auf den mittleren Buttons geklickt (Begrifflichkeit unklar, nicht erkannt, dass disabled)
- 1 x Anmerkung, dass die Begriffe White- und Blacklist für ältere Personen nicht verständlich sein könnte

Szenario E-Mail bearbeiten

Füge eine neue E-Mail Adresse hinzu und baue extra einen Schreibfehler ein. Korrigiere den Fehler nachdem die E-Mail Adresse hinzugefügt wurde.

Wissensziele

- Wurde die Option gefunden?
- Ist diese Funktion überhaupt gewünscht? Oder wird die falsch eingetragene E-Mail Adresse zuerst gelöscht und neu eingetragen?

Bemerkungen

- 6 x haben die Funktion gefunden und konnten die Email bearbeiten
- 1 x zuerst neue hinzugefügt und dann alte gelöscht

Szenario E-Mail löschen

Lösche nun eine E-Mail Adresse aus der Liste.

Wissensziele

- Wurde die Option gefunden?

Bemerkungen

- 7 x konnten die Email ohne Probleme löschen

Szenario E-Mail Adresse verschieben

Verschiebe nun eine E-Mail Adresse von einer Liste in die andere

Wissensziele

- War es verständlich, was zu tun ist?

Bemerkungen

- 7 x Die Email erfolgreich verschoben.

Szenario Einstellungen ändern

Du wechselst den E-Mail Anbieter und erhältst eine neue E-Mail Adresse. Den Service möchtest du weiterhin mit der neuen Adresse verwenden. Da du deine alte E-Mail Adresse nicht mehr nutzen wirst, möchtest du die Daten ändern.

Für dieses Beispiel nimm bitte folgende E-Mail Adresse: test@gmail.com, IMAP: imap.gmail.com, Port: 993, E-Mail Passwort: Sommer2022

Wissensziele

- Was würde der Benutzer als erstes tun? Neu registrieren oder nach Einstellungen suchen?

Bemerkungen

- 7 x Alle haben die Option gefunden

Szenario E-mail Banner

Prärequisite:

Der Tester schickt, sobald die Registrierung vollendet ist eine Mail an die Testperson bzw. an spamsolution@yahoo.com, damit die Testperson eine E-Mail mit einem Banner im Posteingang hat.

Szenario

Melde dich auf https://login.yahoo.com/ mit folgenden Anmeldedaten an:
Spamsoultion@yahoo.com / Passwort: Ba2022!!

Nun klicke bitte die neue Nachricht vom Tester an.

Wissensziele:

- Wird der Banner wahrgenommen?
- Gibt es Bemerkungen?

Bemerkungen

- 7 x Wird der Banner wahrgenommen, keine Probleme

Szenario E-Mail über den Banner hinzufügen

Die E-Mail ist nun offen und der Banner wird angezeigt. Füge nun den Absender zur Whitelist hinzu.

Wissensziele:

- Klickt der Benutzer auf «Zulassen» oder navigiert er wieder auf die Webseite?
- Ist der Bestätigungsscreen, welcher im Browser geöffnet wird störend?

Bemerkungen

- 7 x wurde eins der Buttons getätigt, keine Probleme
- 7 x Das Aufploppen eines neuen Fenster beim betätigen eines der beiden Buttons wird nicht als störend empfunden.

Nachinterview

- Was hat dir gut gefallen?
  - Übersichtlich
  - Die Webseite ist einfach aufgebaut, man findet sofort, was man sucht.
- Was hat dir weniger gut gefallen?
  - Begriffe teils unklar
- Hast du Verbesserungsvorschlägen?
  - Email hinzufügen nach oben
  - Bei Email hinzufügen Buttons zu White- und Blacklist umbenennen statt vertrauen/blockieren
  - Die momentane Email in die Navigationsleiste verlegen
  - Mehr Informationen über den Service
  - Das Hintegrundbild auf der Hauptseite passt nicht zum Inhalt der Seite.
  - Den Unterschied zwischen den deaktivierten Buttons deutlicher machen
  - Verspielte Schriftart

- Vermisst du eine Funktion?
    - Ein Dashboard auf der Startseite mit Fakten (z.B Wie viele Emails wurden bereits blockiert, wie viele Adressen sind auf der White- und Blacklist etc.)
- Eventuell weitere Fragen, falls Probleme auftraten
    - Bei allen Test traten keine weiteren Probleme auf.

Fragebogen der UI Umfrage

Wie vertrauenswürdig schätzt du die Webseite ein?

Wie sehr gefällt dir die farbliche Gestaltung?

Wie sehr gefällt dir das Design der Webseite?

Passt sich die Webseite sinnvoll an, wenn du die Größe des Browserfensters änderst?

Sind alle notwendigen Daten übersichtlich dargestellt?

Ist die Bedienung der Webseite intuitiv?

Wie bewertest du die Seite allgemein?

6.8 Literaturverzeichnis

[1] URL: https://www.sueddeutsche.de/digital/spam-jubilaeum-40-jahre-unerwuenschte-mails-1.3965584. Zugegriffen am: 11.06.2022.

[2] URL: https://www.bfs.admin.ch/bfs/de/home/statistiken/kultur-medien-informationsgesellschaft-sport/informationsgesellschaft/gesamtindikatoren/haushalte-bevoelkerung/sicherheit-internet.html. Zugegriffen am 13.06.2022.

[3] URL: https://de.statista.com/statistik/daten/studie/872986/umfrage/anteil-der-spam-mails-am-gesamten-e-mail-verkehr-weltweit/. Zugegriffen am 13.06.2022.

[4] URL: https://nodejs.org/en/. Zugegriffen am: 01.04.2022.

[5] URL: https://www.mysql.com/. Zugegriffen am: 28.03.2022.

[6] URL: https://reactjs.org/. Zugegriffen am: 05.06.2022.

[7] URL: https://getbootstrap.com/. Zugegriffen am: 05.06.2022.

[8] URL: https://expressjs.com/. Zugegriffen am: 06.06.2022.

[9] URL: https://nodejs.dev/learn. Zugegriffen am: 06.06.2022.

[10] URL: https://reactjs.org/docs/strict-mode.html. Zugegriffen am 09.06.2022.

[11] URL: https://reactjs.org/docs/hooks-state.html. Zugegriffen am 09.06.2022.

[12] URL: https://jestjs.io/. Zugegriffen am: 10.06.2022.

[13] URL: https://docs.python.org/3/library/unittest.html. Zugegriffen am: 10.06.2022.

[14] URL: http://nacl.cr.yp.to/. Zugegriffen am: 05.06.2022.

6.9 Abbildungsverzeichnis

6.10 Tabellenverzeichnis

6.11 Glossar

Bootstrap	Bootstrap ist ein freies Frontend-CSS-Framework. Es enthält Gestaltungsvorlagen für Oberflächengestaltungselemente für z.B Formulare, Buttons oder Tabellen.
Express.js	Express.js ist ein Backend Framework für Node.js.
FURPS	FURPS ist ein Akronym aus dem Bereich der Softwarequalität. Die Buchstaben stehen für Functionality, Usability, Reliability, Performance und Supportability
Meta Platforms	Meta Platforms ist ein US-amerikanisches Technologieunternehmen, früher bekannt als Facebook.
React	React ist eine Javascript Bibliothek, die die Entwicklung des UIs von Webseiten erleichtert.
RUP	RUP ist ein Projektmodell für die Entwicklung von Software. Mehr Informationen sind hier zu finden
Scrum+	Scrum+ vereint die agilen aspekte von Scrum mit der Planung von RUP, indem man zuerst eine grobe Planung mit Meilensteinen erstellt, aber trotzdem flexibel mit Iterationen arbeitet