



Vaadin Mail Manager

Migration von JAMM

aarboard AG

Egliweg 10
CH-2560 Nidau

Tel. +41 32 332 97 14
support@aarboard.ch



Inhaltsverzeichnis

1	MANAGEMENT SUMMARY	1
2	VERSIONIERUNG	2
3	AUFTRAG.....	3
3.1	AUSGANGSLAGE	3
3.2	ZIELE	3
3.3	PROJEKTUMFANG	4
3.3.1	<i>Eingeschlossene Aufgaben</i>	4
3.3.2	<i>Lieferobjekte</i>	4
3.3.3	<i>Abgrenzung</i>	4
3.4	ANFORDERUNGEN.....	5
3.4.1	<i>Funktionale Anforderungen</i>	5
3.4.2	<i>Technische Anforderungen</i>	5
3.4.3	<i>Qualitätsanforderungen</i>	5
3.5	ZEITPLAN UND MEILENSTEINE	6
3.6	RESSOURCEN UND BUDGET	6
3.6.1	<i>Personelle Ressourcen</i>	6
3.6.2	<i>Technische Ressourcen</i>	6
3.6.3	<i>Zeitliche Ressourcen</i>	6
3.7	STAKEHOLDER UND ERWARTUNGEN	7
3.8	RISIKEN UND HERAUSFORDERUNGEN	7
3.9	VORKENNTNISSE	7
3.10	NEUE LERNINHALTE	8
3.11	NACHTRÄGLICHE SCHWERPUNKTVERSCHIEBUNG	9
4	ZEITPLANUNG	10
5	ARBEITSRAPPORTE	11
5.1	TAG 1: 09.06.....	11
5.2	TAG 2: 10.06.....	12
5.3	ERSTER REALISIERUNGSTEIL 11.06 – 13.06.....	13
5.4	ZWEITER REALISIERUNGSTEIL 13.06 - 14.06.....	15
6	PROTOKOLLE.....	16
6.1	1. MEETING	16
6.2	2. MEETING	17
7	VORGEHEN	18
7.1	ANALYSE JAMM USER INTERFACE.....	18
7.1.1	<i>Site Admin</i>	18
7.1.2	<i>Domain Administration</i>	19
7.1.3	<i>Forms</i>	20
7.2	MOCKUPS NEUES USER INTERFACE.....	21
7.3	TECHNOLOGIE STACK	21
8	SYSTEMDOKUMENTATION.....	22
8.1	SYSTEMARCHITEKTUR	22

8.2	SYSTEMANFORDERUNGEN	22
8.2.1	<i>Funktionale Anforderungen</i>	22
8.3	NICHTFUNKTIONALE-ANFORDERUNGEN	23
8.4	ENTWICKLUNGSUMGEBUNG	23
8.5	KONZEPTION SOFTWARE	24
9	IMPLEMENTIERUNGSSTAND	25
10	TESTKONZEPT	26
10.1	TESTPHILOSOPHIE UND STRATEGISCHER ANSATZ	26
10.2	SCHICHTBASIERTE TESTARCHITEKTUR	27
10.2.1	<i>Entitätsschicht: Fundament der Datenintegrität</i>	27
10.2.2	<i>Repository-Schicht: LDAP-Integration und Datenzugriff</i>	28
10.2.3	<i>Service-Schicht: Geschäftslogik und Workflow-Orchestrierung</i>	29
10.3	QUALITÄTSSICHERUNG UND ERFOLGSMESSUNG	29
10.3.1	<i>Validierungsansatz</i>	29
10.3.2	<i>Erfolgskriterien</i>	29
11	FAZIT	30
12	AUSBLICK	30
13	VERWEISE	31
13.1	TABELLENVERZEICHNIS	31
13.2	ABBILDUNGSVERZEICHNIS	31
14	GLOSSAR	32
15	PROGRAMMCODE	34
15.1	APPLICATION.JAVA	34
15.2	ENTITÄTEN	35
15.2.1	<i>JammVirtualDomain.java</i>	35
15.2.2	<i>JammMailAccount.java</i>	40
15.2.3	<i>JammMailAlias.java</i>	45
15.2.4	<i>JammPostmaster.java</i>	50
16	ANHANG	56
16.1	TESTERGEBNISSE	56

1 Management Summary

Projektname: Vaadin Mail Manager

Die Aarboard AG nutzte bisher das veraltete JAMM-Tool für die E-Mail-Kontenverwaltung, dessen antiquierte Benutzeroberfläche den modernen Anforderungen nicht mehr entsprach. Das Vaadin Mail Manager Projekt zielte darauf ab, diese Anwendung durch eine zeitgemäße Vaadin-basierte Lösung zu ersetzen und dabei die bestehende LDAP-Integration zu erhalten.

Hauptvorteile

Die neue Anwendung bietet eine moderne, intuitive Weboberfläche mit Breadcrumb-Navigation und benutzerfreundlichen Verwaltungsdialogen. Die serviceorientierte Architektur mit klarer Schichtentrennung verbessert die Wartbarkeit erheblich und ermöglicht zukünftige Erweiterungen. Durch den Einsatz aktueller LTS-Versionen von Java 21, Spring Boot und Vaadin ist die technologische Zukunftssicherheit gewährleistet.

Wichtige Ergebnisse

Das Projekt erreichte einen funktionsfähigen Prototyp mit etwa 72 Prozent Implementierungsgrad. Die vollständige LDAP-Integration wurde erfolgreich umgesetzt, wobei alle JAMM-Datenstrukturen erhalten blieben. Eine umfassende Testabdeckung mit 112 systematischen Testfällen validiert alle Anwendungsschichten von der Datenintegration bis zur Geschäftslogik. Die Kernfunktionalitäten für Domain-, Account- und Alias-Verwaltung einschließlich Catch-All-Unterstützung sind vollständig implementiert.

Projektdurchführung

Das einmonatige Projekt mit 50 Lektionen wurde nach der IPERKA-Methodik durchgeführt, wobei die Parallelität zur regulären Arbeitszeit besondere Herausforderungen bei der Zeitplanung mit sich brachte. Eine strategische Anpassung während der Entwicklung verlagerte den Fokus von der ursprünglich geplanten UI-Modernisierung auf die Backend-Stabilität, nachdem sich die Komplexität der LDAP-Integration als größer erwies als erwartet. Diese bewusste Schwerpunktverschiebung ermöglichte trotz zeitlicher Einschränkungen eine robuste technische Grundlage mit umfangreicher Testabdeckung.

Herausforderungen

Die Integration der veralteten JAMM-LDAP-Strukturen in moderne Frameworks erwies sich als komplexer als ursprünglich kalkuliert. Die parallele Projektdurchführung neben der regulären Arbeitszeit stellte zusätzliche zeitliche Herausforderungen dar, die eine intensive Priorisierung gegen Projektende erforderlich machten. Die Transformation von string-basierten Architekturmustern in objektorientierte Patterns erforderte umfangreiche Modernisierungsarbeiten, wobei das erhebliche LDAP-Wissensdefizit zu Beginn des Projekts eine besondere Hürde darstellte.

2 Versionierung

Version	Autor	Datum	Änderung
1.0	Fokko Vos	09.06.25	Erstellung und Grundlegendes Layout
1.1	Fokko Vos	09.06.25	Schreiben des Projektauftrages, Grundlegende Dokumentationsinhalte
1.2	Fokko Vos	10.06.25	Dokumentieren der JAMM UI
1.3	Fokko Vos	10.06.25	Dokumentation Testkonzept
1.4	Fokko Vos	10.06.25	Erstellung des Systemarchitektur Abschnittes
1.5	Fokko Vos	10.06.25	Technologie Stack dokumentiert
1.6	Fokko Vos	10.06.25	Festhalten der ersten 2 Tagesrapporte
1.7	Fokko Vos	12.06.25	Schwerpunktverschiebung – siehe: 3.11
1.8	Fokko Vos	13.06.25	Nachführen sämtlicher Protokolle
1.9	Fokko Vos	13.06.25	Bericht für den ersten Teil der Realisierung
2.0	Fokko Vos	16.06.25	Dokumentation Test Protokoll
2.1	Fokko Vos	18.06.25	Dokumentation Fazit
2.2	Fokko Vos	19.06.25	Bericht für den zweiten Teil der Realisierung
2.3	Fokko Vos	19.06.25	Dokumentation des Programmcodes
2.4	Fokko Vos	20.06.25	Festhalten des Implementierungsstandes
2.5	Fokko Vos	20.06.25	Projektabschluss. Feinschliff

1 - Versionierung

3 Auftrag

3.1 Ausgangslage

Die Aarboard AG verwendet aktuell das veraltete Open-Source-Tool JAMM (Java Mail Manager) für die interne Verwaltung von E-Mail-Konten. Die Benutzeroberfläche dieses Tools ist jedoch stark veraltet und entspricht weder den heutigen Standards der Benutzerfreundlichkeit noch den Anforderungen an moderne Wartbarkeit. Die manuelle Verwaltung von Mailkonten über die antiquierte Oberfläche führt zu ineffizienten Arbeitsabläufen und erschwert die Administration erheblich.

Das Unternehmen setzt bei der Entwicklung moderner Webanwendungen konsequent auf das Vaadin-Framework als technologische Grundlage. Diese strategische Ausrichtung macht eine Migration der bestehenden JAMM-Funktionalitäten auf eine Vaadin-basierte Lösung zu einer logischen und notwendigen Modernisierungsmaßnahme. Das ursprüngliche JAMM-Projekt ist unter <https://jamm.sourceforge.net/> verfügbar und bildet die Grundlage für diese Migration

3.2 Ziele

Das Hauptziel des Projekts besteht in der Entwicklung einer funktionalen Migration von JAMM auf Basis des Vaadin-Frameworks. Die neue Anwendung soll über eine moderne, benutzerfreundliche Weboberfläche verfügen und die zentralen Funktionen der bestehenden Lösung, insbesondere die LDAP-Anbindung zur Benutzerverwaltung, vollständig übernehmen.

Als Ergebnis soll ein eigenständiger, lauffähiger Prototyp entstehen, der als technisch saubere und erweiterbare Basis für eine spätere vollständige Ablösung der alten Anwendung dient. Die neue Anwendung muss sich nahtlos in die bestehende Technologielandschaft des Unternehmens integrieren lassen.

3.3 Projektumfang

3.3.1 Eingeschlossene Aufgaben

Das Projekt umfasst die vollständige Migration der bestehenden JAMM-Funktionalität, einschließlich der LDAP-Logik, in eine eigenständige, auf Vaadin basierende Webanwendung. Konkret beinhaltet dies:

- Analyse und Dokumentation der bestehenden JAMM-Anwendung
- Definition und Priorisierung der zu migrierenden Kernfunktionalitäten
- Konzeption einer modernen Softwarearchitektur unter Verwendung von Vaadin und Spring
- Entwicklung der LDAP-Integration für die Benutzerverwaltung
- Implementierung der zentralen Verwaltungsfunktionen für E-Mail-Domänen, -Konten und - Aliase
- Erstellung einer benutzerfreundlichen Web-Oberfläche
- Durchführung von Tests zur Sicherstellung der Funktionalität
- Umfassende Projektdokumentation

3.3.2 Lieferobjekte

- Vollständiges Git-Repository mit dem Quellcode der Vaadin-Spring-Anwendung
- Docker-Compose-Konfiguration für die LDAP-Entwicklungsumgebung
- Projektdokumentation mit Reflexion der Lernerfahrungen
- Funktionsfähiger Prototyp der migrierten Anwendung

3.3.3 Abgrenzung

Das Projekt beschränkt sich ausschließlich auf die Migration der bestehenden JAMM-Kernfunktionalitäten ohne funktionale Erweiterungen über den ursprünglichen Umfang hinaus. Explizit nicht Bestandteil des Projektauftrags sind:

- Funktionale Erweiterungen über den bestehenden JAMM-Umfang hinaus
- Sicherheitstechnische Analysen oder Audits der entwickelten Anwendung
- Die produktive Bereitstellung im Live-Betrieb
- Vollständige Integration in die Unternehmensinfrastruktur
- Performance-Optimierungen über die Grundfunktionalität hinaus

3.4 Anforderungen

3.4.1 Funktionale Anforderungen

- Vollständige LDAP-Anbindung mit sicherer Authentifizierung
- Verwaltung von E-Mail-Domänen (Erstellen, Aktivieren, Deaktivieren)
- Verwaltung von E-Mail-Konten (Erstellen, Bearbeiten, Löschen, Passwort-Management)
- Verwaltung von E-Mail-Aliassen einschließlich Catch-All-Funktionalität
- Postmaster-Verwaltung für Domänen-Administration
- Benutzerfreundliche Web-Oberfläche mit moderner Navigation
- Grundlegende Rollen- und Berechtigungsverwaltung

3.4.2 Technische Anforderungen

- Verwendung des Vaadin-Frameworks für die Benutzeroberfläche
- Spring Boot als Anwendungs-Framework
- Kompatibilität mit bestehenden LDAP-Strukturen
- Moderne Softwarearchitektur mit klarer Trennung der Schichten
- Umfassende Testabdeckung für kritische Funktionen
- Dokumentierte und wartbare Codebasis

3.4.3 Qualitätsanforderungen

- Stabile und fehlerfreie Grundfunktionalität
- Intuitive Benutzerführung
- Erweiterbare Architektur für zukünftige Entwicklungen

3.5 Zeitplan und Meilensteine

Das Projekt folgt der IPERKA-Methodik und ist auf einen Zeitraum von einem Monat mit einem Aufwand von 50 Lektionen ausgelegt. Die Umsetzung gliedert sich in folgende Meilensteine:

- **Meilenstein 1 – Informieren**
Analyse der bestehenden JAMM-Anwendung und Definition der zu migrierenden Kernfunktionalitäten
- **Meilenstein 2 – Planen**
Erstellung eines detaillierten Projektplans mit Zeitrahmen und konkreter Ziel- sowie Architekturdefinition
- **Meilenstein 3 - Entscheiden**
Konzeption der neuen Softwarearchitektur mit Vaadin und Integration von LDAP
- **Meilenstein 4 – Realisieren**
Entwicklung eines ersten funktionsfähigen Prototyps mit den definierten Funktionen
- **Meilenstein 5 – Kontrollieren**
Durchführung von Tests und iterative Verbesserung der Anwendung
- **Meilenstein 6 – Auswerten**
Dokumentation der Projektergebnisse und Reflexion der Lernerfahrungen

3.6 Ressourcen und Budget

3.6.1 Personelle Ressourcen

Die Projektdurchführung erfolgt eigenverantwortlich durch Fokko Vos im Rahmen seiner Ausbildung zum Applikationsentwickler bei der Aarboard AG. Die Gesamtkoordination des Projekts, einschließlich Konzeption, Softwareentwicklung, Testing und Dokumentation, liegt in seiner alleinigen Verantwortung.

3.6.2 Technische Ressourcen

Das Projekt wird ausschließlich mit bereits vorhandenen Ressourcen durchgeführt, wodurch keine zusätzlichen Geldmittel erforderlich sind. Die Entwicklung erfolgt lokal auf der bestehenden Arbeitsplatz-Hardware unter Verwendung kostenfreier Open-Source-Software und Frameworks:

- Vaadin-Framework (Open Source)
- Spring Boot (Open Source)
- Docker für Containerisierung (Open Source)
- Standard-Entwicklungstools (IDE, Versionskontrollsystem)
- OpenLDAP für Testumgebung (Open Source)

3.6.3 Zeitliche Ressourcen

- Gesamtaufwand: 50 Lektionen
- Projektdauer: 1 Monat
- Durchführung parallel zur regulären Arbeitszeit im Rahmen des SOL-Konzepts (Selbstorganisiertes Lernen)

3.7 Stakeholder und Erwartungen

Aarboard AG (Auftraggeber)

- Erwartung: Moderne, wartbare Alternative zur bestehenden JAMM-Lösung
- Nutzen: Verbesserte Benutzerfreundlichkeit und Integration in die Technologielandschaft

Fokko Vos (Projektdurchführender)

- Erwartung: Praktische Vertiefung der Kenntnisse in Vaadin, Spring und LDAP-Integration
- Nutzen: Entwicklung einer professionellen Softwarearchitektur und Projektmanagement-Erfahrung

3.8 Risiken und Herausforderungen

Der zeitlich begrenzte Rahmen des Projekts lässt voraussichtlich nur die Migration der zentralen Kernfunktionen von JAMM zu. Eine vollständige Umsetzung aller Funktionalitäten ist im vorgesehenen Umfang nicht realistisch. Der Fokus liegt daher auf der Entwicklung einer lauffähigen Webanwendung, die möglichst viele der wichtigsten Funktionen abdeckt, auch wenn diese den vollständigen Funktionsumfang des ursprünglichen Tools noch nicht erreicht.

Weitere identifizierte Risiken umfassen:

- Komplexität der LDAP-Integration bei unvollständiger Dokumentation
- Unvorhergesehene Architektur-Herausforderungen bei der Migration
- Zeitliche Engpässe bei der Implementierung komplexer Funktionalitäten

Das Ziel besteht darin, eine stabile und erweiterbare Grundlage zu schaffen, auf der zukünftige Erweiterungen aufbauen können.

3.9 Vorkenntnisse

Vorhandene Grundkenntnisse in der Vaadin-Entwicklung bilden eine solide Basis für das Projekt, auch wenn diese bisher nur oberflächlich und nicht von Grund auf erworben wurden. Die bestehende Erfahrung in der Java-Programmierung sowie das Verständnis für Webanwendungen unterstützen die Projektdurchführung erheblich.

Zusätzlich sind Grundkenntnisse in der Softwarearchitektur und in objektorientierten Programmierkonzepten vorhanden. Das Verständnis für Datenbanken und deren Integration in Java-Anwendungen bildet eine weitere wichtige Grundlage für die erfolgreiche Projektumsetzung.

3.10 Neue Lerninhalte

Das Projekt ermöglicht eine umfassende Vertiefung in verschiedene moderne Technologien und Konzepte. Die intensive Auseinandersetzung mit Vaadin und LDAP steht dabei im Vordergrund.

Die Entwicklung einer professionellen Softwarearchitektur mit Services, Repositories und Komponenten stellt einen wesentlichen Lerninhalt dar. Dabei werden moderne Architekturprinzipien wie die Trennung von Geschäfts- und Präsentationslogik praktisch angewendet. Die Konfiguration von Security-Aspekten sowie die tiefgreifende Beschäftigung mit Vaadin-Komponenten und deren fortgeschrittenen Funktionalitäten erweitern das technische Verständnis erheblich.

Die LDAP-Integration bildet einen weiteren zentralen Lernbereich, der sowohl die theoretischen Grundlagen als auch die praktische Implementierung und Konfiguration von LDAP-Verbindungen umfasst. Das Setup und die Verwaltung von LDAP-Testumgebungen mittels Docker-Compose vervollständigen die neu zu erwerbenden Kompetenzen in diesem Bereich.

3.11 Nachträgliche Schwerpunktverschiebung

Während der Projektdurchführung wurde eine strategische Anpassung der ursprünglichen Zielsetzung vorgenommen, die eine bewusste Schwerpunktverschiebung von der UI-Modernisierung hin zur Backend-Stabilität und umfassenden Testabdeckung darstellte.

Die initiale Projektplanung sah eine primäre Fokussierung auf die Modernisierung der Benutzeroberfläche vor, wobei die LDAP-Integration als sekundärer Aspekt betrachtet wurde. Während der Entwicklungsphase stellte sich jedoch heraus, dass die vorhandenen LDAP-Kenntnisse unzureichend waren, um eine stabile und zuverlässige Backend-Integration zu gewährleisten.

Diese Erkenntnis führte zu einer bewussten strategischen Neuausrichtung des Projekts. Die Entscheidung wurde getroffen, zunächst eine solide technische Grundlage durch umfassende LDAP-Integration und systematische Testabdeckung zu schaffen, bevor die UI-Modernisierung vorangetrieben wird. Diese Priorisierung folgte dem Prinzip, dass ein funktionsfähiger Prototyp mit stabiler Backend-Funktionalität einen höheren Geschäftswert darstellt als eine modernisierte Benutzeroberfläche ohne zuverlässige Datenverarbeitung.

Die umfangreiche Testentwicklung mit 112 systematischen Testfällen entstand als direkte Konsequenz dieser strategischen Neuausrichtung. Die Tests dienen nicht nur der Qualitätssicherung, sondern fungieren als Lernwerkzeug und Validierungsmechanismus für die LDAP-Integration, wodurch das ursprünglich identifizierte Wissensdefizit systematisch adressiert wurde.

Das Testkonzept wurde nachträglich nachgebessert.

Die UI-Modernisierung wurde dennoch erfolgreich umgesetzt, allerdings mit reduziertem Zeitaufwand durch den Einsatz von KI-unterstützter Entwicklung. Mithilfe von Claude wurden die grundlegenden View-Komponenten basierend auf den ursprünglich erstellten Mockups generiert, was eine effiziente Umsetzung der visuellen Anforderungen ermöglichte. Die Integration zwischen UI und Backend-Services sowie die Implementierung der Geschäftslogik-Verknüpfungen wurden manuell durchgeführt, um die vollständige Kontrolle über die kritischen Anwendungskomponenten zu gewährleisten.

Diese Herangehensweise resultierte in einem ausgewogenen Ergebnis, das sowohl die technische Stabilität durch umfassende Backend-Validierung als auch die moderne Benutzerführung durch aktualisierte UI-Komponenten sicherstellt. Die strategische Anpassung demonstriert adaptive Projektführung und die Fähigkeit, Prioritäten basierend auf identifizierten Risiken und Kompetenzlücken neu zu bewerten.

4 Zeitplanung

Vaadin Mail Manager

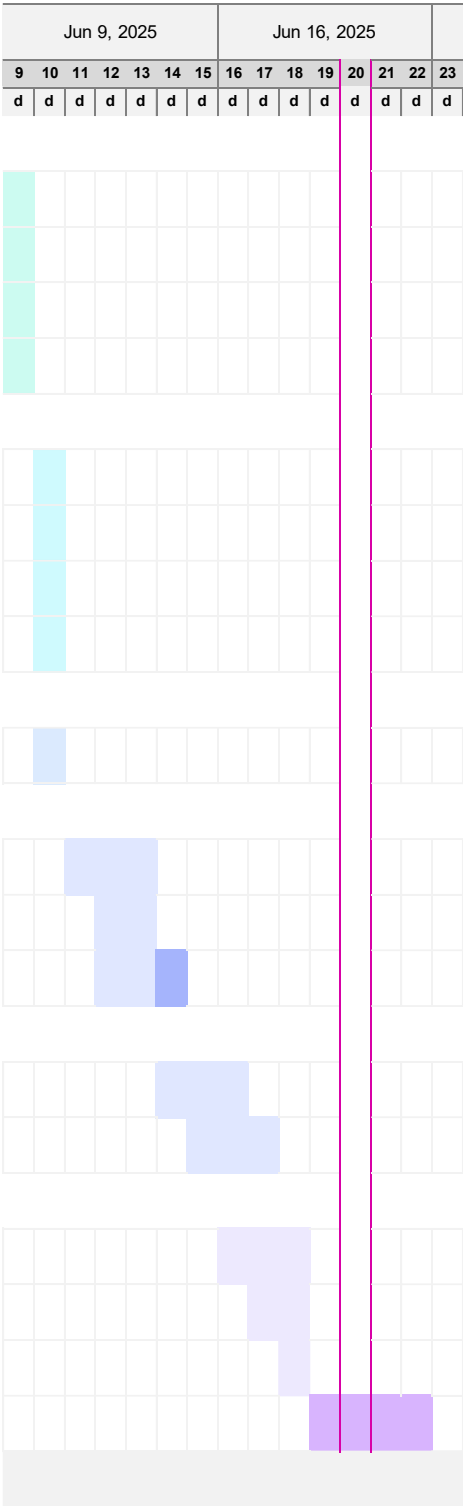
Prototyp

Projektleiter: Fokko Vos

TASK	SOLL	IST	PROGRESS	START	END
Informieren	8	8	100%	08.06.25	09.06.25
Analyse der bestehenden JAMM-Anwendung	4	3	100%	08.06.25	09.06.25
Analyse der zu migrierenden UI	1	1.5	100%	08.06.25	09.06.25
Erstellung der Dokumentation	1	2	100%	09.06.25	09.06.25
Anforderungsanalyse	2	1.5	100%	09.06.25	09.06.25
Planen	7.5	6	100%	10.06.25	10.06.25
Definition der zu migrierenden Kernfunktionen	3	2	100%	10.06.25	10.06.25
Detaillierten Zeitplan erstellen	1.5	2	100%	10.06.25	10.06.25
Ausarbeitung der Systemarchitektur und Mockups	2	1	100%	10.06.25	10.06.25
Testkonzept	1	1	100%	10.06.25	10.06.25
Entscheiden	1	0	100%	10.06.25	10.06.25
Technologiestack (Versionen)	1	0	100%	10.06.25	10.06.25
Realisieren	12	14	100%	11.06.25	14.06.25
Grundgerüst Vaadin UI erstellen	6	2	100%	11.06.25	13.06.25
LDAP-Anbindung implementieren	3	8	120%	12.06.25	13.06.25
Usability-Feinschliff und Styling	3	4	80%	12.06.25	14.06.25
Kontrollieren	5	2	100%	14.06.25	17.06.25
Anwenung Testkonzept	3	2	100%	14.06.25	16.06.25
ggf. Anpassungen Vornehmen	2	0	100%	15.06.25	17.06.25
Auswerten	10	11	75%	16.06.25	22.06.25
Abgleich mit den Zielen	1	2	100%	16.06.25	18.06.25
Reflektion und Fazit	1	1	100%	17.06.25	18.06.25
Finalisierung der Dokumentation	5	8	100%	18.06.25	18.06.25
Vorbereitung der Präsentation	3	0	0%	19.06.25	22.06.25
	43.5	41		6.8.25	6.22.25

Project start: Sonntag, 8. J

Display week: 1



5 Arbeitsrapporte

5.1 Tag 1: 09.06

Aktivität	Aufwand geplant (h)	Aufwand effektiv (h)
Informationen sammeln	8	8
Tagesablauf		
<ul style="list-style-type: none">- Detaillierte Analyse der bestehenden JAMM-Anwendung- Untersuchung der Kernfunktionalitäten und Systemarchitektur- Analyse der veralteten Benutzeroberfläche- Identifikation von Schwachstellen und Verbesserungspotentialen- Erstellung der Dokumentation der Analyseergebnisse- Durchführung der Anforderungsanalyse		
Hilfestellung		
<ul style="list-style-type: none">- Bestehende JAMM-Dokumentation- Jamm Repository- Screenshots der aktuellen Benutzeroberfläche- Vaadin-Framework-Dokumentation (online)		
Reflektion		
Die Anforderungsanalyse fiel mir schwer, da ich nicht wirklich abschätzen konnte, was ich zu analysieren hatte und habe daher auch etwas den Punkt verfehlt. Die Dokumentationserstellung nahm ebenfalls mehr Zeit in Anspruch, da ich mich um eine Richtig Vorlage bemühen musste.		
Nächste Schritte		
Übergang in die Planungsphase		

2 - Tag 1: 09.06

5.2 Tag 2: 10.06

Aktivität	Aufwand geplant (h)	Aufwand effektiv (h)
Planung des Projektes, Dokumentierung des Technologie Stacks	7.5	6
Tagesablauf		
<ul style="list-style-type: none">- Priorisierung der JAMM-Kernfunktionen basierend auf Benutzeranforderungen- Definition der MVP-Features für den Prototyp- Erstellung eines realistischen Projektplanes- Erste Skizzen der neuen Vaadin-basierten Benutzeroberfläche- Konzeption Anwendungsarchitektur- Grundlegendes Testkonzept für Unit- und Integrationstests- Dokumentation des zu verwendenden Technologie stacks		
Hilfestellung		
<ul style="list-style-type: none">- Vaadin Design System Dokumentation- Spring Boot Best Practices Diskussionen auf Reddit- Draw.io für Systemdiagramme- Balsamiq für UI-Mockups- Bestehende Vaadin-Projekte als Referenz		
Reflektion		
<p>Der Zeitplan benötigte mehr Detailarbeit als ursprünglich kalkuliert, insbesondere bei der Berücksichtigung von Abhängigkeiten zwischen den Entwicklungsphasen. Die Architekturplanung profitierte stark von der bestehenden Erfahrung mit Java und Objektorientierter Programmierung, wodurch bewährte Patterns direkt angewendet werden konnten. Das Mockup-Design gestaltete sich unkompliziert, da bereits ein klares Verständnis der gewünschten Benutzerführung vorhanden war.</p>		
Nächste Schritte		
Beginn der konkreten Implementierungsplanung.		

3 - Tag 2: 10.06

5.3 Erster Realisierungsteil 11.06 – 13.06

Aktivität	Aufwand geplant (h)	Aufwand effektiv (h)
Grundarchitektur der Vaadin Mail Manager Anwendung mit vollständiger LDAP-Integration, Implementierung der JAMM-Datenstrukturen, Repository-Pattern und umfassender Testabdeckung.	4.5	14
Tagesablauf		
<ul style="list-style-type: none">- Erstellung der grundlegenden Projektstruktur und LDAP-Konfiguration- Implementierung des LDAP-Authentifizierungssystems- Einrichtung der Docker-Compose-Entwicklungsumgebung mit JAMM-LDAP-Schemas- Modellierung der vier JAMM-Kernentitäten (JammVirtualDomain, JammMailAccount, JammMailAlias, JammPostmaster)- Entwicklung der entsprechenden Unit-Tests für alle Entitäten- Implementierung des Repository-Patterns mit Interfaces und konkreten LDAP-Implementierungen- Erstellung umfangreicher Integrationstests für die Repository-Schicht- Entwicklung der Service-Schicht mit Geschäftslogik für Domain-, Account- und Alias-Verwaltung- Erstellung der Service-Layer-Tests für vollständige Testabdeckung- Validierung der Services durch entsprechende Tests		
Hilfestellung		
<ul style="list-style-type: none">- Vaadin-Framework Documentation- Spring Boot Reference Documentation- Spring LDAP Documentation- JAMM-Dokumentation (Github-Repository)- Claude AI (Architekturentscheidungen, Code-Reviews, LDAP-Optimierung)- Stack Overflow- Vaadin Community Forums- Docker Hub- OpenLDAP-Dokumentation		
Reflektion		
<p>Die erste Realisierungsphase verlief größtenteils planmäßig und erreichte die gesetzten Ziele vollständig. Aufgrund der geringen Vorerfahrung mit LDAP-Technologien wurde für alle Hauptentitäten und Repository-Klassen eine umfangreiche Testabdeckung implementiert, um sicherzustellen, dass die grundlegende Logik den Erwartungen entspricht. Diese systematische Herangehensweise erwies sich als besonders wertvoll, da dadurch frühzeitig Inkonsistenzen in den LDAP-Mappings identifiziert und behoben werden konnten.</p> <p>Die Implementierung der Repository-Schicht gestaltete sich komplexer als ursprünglich antizipiert. Bei der Analyse der bestehenden JAMM-Implementierung wurde deutlich, dass die ursprüngliche Architektur stark auf String-Konkatenationen und unstrukturierte Filter-Mechanismen angewiesen war. Anstatt diese veralteten Patterns direkt zu übernehmen, wurde entschieden, die Struktur zu modernisieren und sauber abzubilden, was zu einem nachhaltigen und wartbaren Codebase führt. Positiv hervorzuheben ist die erfolgreiche Integration der Docker-Compose-Umgebung, die eine konsistente und reproduzierbare Entwicklungsumgebung schafft. Die umfangreiche Testabdeckung</p>		

bietet eine solide Grundlage für die weiteren Entwicklungsphasen und minimiert das Risiko von Regressionen.

Verbesserungspotential bestand in der initialen Zeitschätzung für die LDAP-Integration. Die ursprüngliche Planung hatte eine einfache Eins-zu-Eins-Migration vorgesehen, jedoch wurde die Modernisierungsnotwendigkeit der veralteten JAMM-Architektur unterschätzt. Dies führte zu einem erhöhten Zeitaufwand für die Repository-Implementierungen. Zukünftig sollten Legacy-Modernisierungsprojekte mit einem zusätzlichen Zeitpuffer für Architektur-Refactoring versehen werden.

Nächste Schritte

Implementierung der Vaadin-Benutzeroberfläche mit Navigation, Breadcrumbs, Verwaltungsansichten für alle Entitäten und Event-System für UI-Aktualisierungen.

4 - Erster Realisierungsteil 11.06 – 13.06

5.4 Zweiter Realisierungsteil 13.06 - 14.06

Aktivität	Aufwand geplant (h)	Aufwand effektiv (h)
Vollständige Implementierung der Vaadin-Benutzeroberfläche mit allen Verwaltungsansichten, dynamischer Navigation, Event-System und Breadcrumb-Funktionalität.		
Tagesablauf		
<ul style="list-style-type: none"> - Entwicklung aller Dialoge (CreateDomainDialog, CreateAccountDialog, CreateAliasDialog) - Implementierung der Breadcrumb-Navigation mit BreadcrumbLayout und BreadcrumbItem - Entwicklung der Hauptverwaltungsansichten (DomainsView, ManageDomainView, ManageMailAccountView, ManageMailAliasView) - Implementierung der AccountView für Benutzerkontoverwaltung - Integration des Event-Systems mit DomainContentChangedEvent für UI-Synchronisation - Konfiguration der dynamischen Navigation im MainLayout - Code-Refactoring und Strukturbereinigung - Theme-Reset zur Behebung von Layout-Problemen 		
Hilfestellung		
<ul style="list-style-type: none"> - Vaadin Documentation (Component APIs, Event Handling) - Spring Boot Testing Framework Documentation - Vaadin Community Forums - Stack Overflow - Claude AI (UI-Basis-Klassen, Event-System Design) - Browser Developer Tools für CSS-Debugging 		
Reflektion		
<p>Die zweite Realisierungsphase verlief erfolgreich und brachte die Anwendung zu einem funktionsfähigen Prototyp-Status. Die Implementierung der Vaadin-Benutzeroberfläche erwies sich als weniger komplex als die vorhergehende LDAP-Integration, da die Service-Layer bereits eine solide Grundlage bot. Das Event-System für die UI-Synchronisation funktionierte problemlos und ermöglicht eine konsistente Benutzererfahrung bei Datenänderungen.</p> <p>Herausfordernd gestaltete sich zunächst das Theme-Management, da das ursprünglich erstellte Custom-Theme zu Layout-Konflikten führte. Die Entscheidung, das Theme zurückzusetzen und auf Vaadin-Standards zu setzen, erwies sich als pragmatisch und zeiteffizient.</p> <p>Die umfangreiche Testabdeckung der Service-Layer zahlt sich bereits aus, da Änderungen an der Geschäftslogik sicher validiert werden können. Die dynamische Navigation im MainLayout bietet eine intuitive Benutzerführung und entspricht modernen Webanwendungsstandards. Ich bin äusserst zufrieden mit dem aktuellen stand des Projektes, auch wenn bei weitem nicht alle funktioniert, denke ich das ich für den Zeitrahmen ganz Ordentliche Arbeit abliefern konnte, speziell am Ende der Zweiten Realisierung Phase habe ich an einzelnen Elementen, welche nicht Systemkritisch sind, länger hantiert als ich hätte sollen.</p>		
Nächste Schritte		
Nachführen der Projektdokumentation, abschliessen des Projektes		

5 - Zweiter Realisierungsteil 13.06 - 14.06

6 Protokolle

6.1 1. Meeting

Ablauf:

- Vorstellung der Bewertungskriterien
- Besprechung der Projektanträge
- Kurze Vorstellung des eigenen Projektes
- Allgemeine Klärung von Offenen Fragen
- Statusbericht bis am Ende des Tages, spätestens morgen früh

Statusbericht: 27.05.2025

Projektauftrag

Der Projektauftrag wurde formuliert und genehmigt.

Projektverlauf

Eine detaillierte Projektplanung fehlt noch

Arbeitsrapport

Es wurde noch keine Zeit in das Projekt investiert

Aktueller Stand der Dokumentation

Die Projektdokumentation hat noch nicht begonnen.

6.2 2. Meeting

Ablauf:

- Besprechung des Aktuellen Projektes
 - o Mündliche Mitteilung des aktuellen Standes
 - o Ich werde meine «Wochen» berichte flexibel anpassen, damit es sinn macht, da mein Projekt zu etwas speziellen Zeitbedingungen umgesetzt wurde.
- Festlegung der Abgabe und Bedingungen.
 - o Pünktlichkeit ist alles
- Wieder 10-15min, 5min Demo, ca. 10min Fachgespräch
- Weitere Fragerunde zur Klärung von offenen Fragen

Statusbericht: 13.06.2025

Projektverlauf

Eine detaillierte Projektplanung wurde erstellt, die Architektur dokumentiert, Mockups erstellt & ein Grundgerüst der Realisierung ist bereits vorhanden. Alles in allem verläuft das Projekt soweit reibungslos

Arbeitsrapport

Die Arbeitsrapporte für die erledigten aufgaben wurden direkt in der Dokumentation erstellt, nicht Wochen sondern Phasen spezifisch. Der Projekt Start wurde auf den 8.06.2025 gelegt. Vorgestern wurde mit der Realisierung begonnen

Aktueller Stand der Dokumentation

Die Dokumentation ist etwas hinterher, vor allem bei allem nicht-Technischen.

Eigene Punkte

Bisher keine Probleme oder Fragen aufgekommen, die meisten habe ich in SOL 2 schon geklärt & notiert, ich schaue bei Unklarheiten in meine alten Unterlagen.

7 Vorgehen

7.1 Analyse JAMM User Interface

Die wichtigsten Interfaces von Jamm sind rudimentäre Formulare.

7.1.1 Site Admin

[Log out](#) : [Site Admin](#) :

Jamm 0.9.7-rc1
SITE ADMINISTRATION
MANAGE DOMAINS [Add Domain](#)


	DELETE DOMAIN	CAN EDIT ACCOUNTS	APPOINT POSTMASTERS	DOMAIN IS ACTIVE
> domain1	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain2	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain3	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain4	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain5	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain6	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain7	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain8	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain9	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain10	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain11	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
> domain12	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain13	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain14	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain15	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain16	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain17	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain18	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain19	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain20	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain21	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain22	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain23	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain24	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain25	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain26	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain27	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain28	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain29	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain30	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain31	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain32	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain33	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain34	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain35	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain36	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain37	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain38	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain39	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain40	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain41	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain42	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain43	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain44	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain45	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain46	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain47	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain48	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain49	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain50	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain51	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain52	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain53	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain54	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain55	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain56	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain57	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain58	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain59	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain60	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain61	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain62	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain63	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain64	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain65	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain66	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain67	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain68	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain69	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain70	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain71	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain72	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain73	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain74	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain75	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain76	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain77	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain78	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain79	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain80	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain81	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain82	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain83	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain84	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain85	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain86	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain87	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain88	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain89	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain90	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain91	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain92	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain93	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain94	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain95	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain96	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain97	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain98	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain99	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> domain100	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

2 - Jamm Site Admin

Die Benutzeroberfläche gliedert sich in zwei Hauptbereiche: Links in der Tabelle befinden sich die Namen der Domains, während rechts eine tabellarische Übersicht mit Verwaltungsoptionen dargestellt wird. Die Tabelle verfügt über mehrere Spalten mit Aktionsschaltflächen und Konfigurationseinstellungen für jede Domain. Bei einer Änderung muss ganz nach unten gescrollt werden, um die Aktion zu bestätigen.

7.1.2 Domain Administration

[Log out](#) : [Site Admin](#) : [Domain Admin](#) :

Jamm 0.9.7-rc1
DOMAIN ADMINISTRATION for 
[Change Postmaster Password](#)
Catch-All: **Inactive** [Edit Catch-All](#)
MANAGE ACCOUNTS [Add Account](#)

Edit Account (Postmaster)	>
Edit Account (Postmaster)	>
Edit Account (Postmaster)	>
Edit Account (Postmaster)	>

MANAGE ALIASES [Add Alias](#)

Edit Alias (Postmaster)	>
Edit Alias (Postmaster)	>
Edit Alias (Postmaster)	>

DELETE ACCOUNT	ACCOUNT IS ACTIVE	POSTMASTER
<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>

[Submit Changes](#) [Reset](#)

DELETE ALIAS	ALIAS IS ACTIVE	POSTMASTER
<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>

[Submit Changes](#) [Reset](#)

3 - Jamm Domain Admin

Oben auf der Seite kann man das Passwort für die Postmaster Festlegen, wie auch die Catch-All Adresse verwalten.

Im Bereich "MANAGE ACCOUNTS" werden vier E-Mail-Konten aufgelistet. Jedes Konto verfügt über standardisierte Verwaltungsoptionen durch entsprechende Aktionsschaltflächen in der rechten Spalte.

Der Abschnitt "MANAGE ALIASES" zeigt die Konfiguration von E-Mail-Weiterleitungen. Hier sind mehrere Alias-Adressen mit ihren jeweiligen Zielkonten dokumentiert, ebenfalls mit zugehörigen Verwaltungsfunktionen.

7.1.3 Forms

[Log out](#) :



ADD DOMAIN

Domain Name:

Postmaster Password: (Optional)

Retype Postmaster Password: (Optional)

Note: If the Postmaster password is not entered, only the site administrator can manage the new domain.

4 - Jamm Add Domain

[Log out](#) :



ADD ACCOUNT for domain

Account Name: @

Name:

Password:

Retype Password:

5 - Jamm Add Account

[Log out](#) :



ADD ALIAS for domain

Alias Name: @

Name:

Destinations:

Password: (Optional)

Retype Password: (Optional)

Note: If a password is not entered, only the domain or site administrator can manage the new alias.

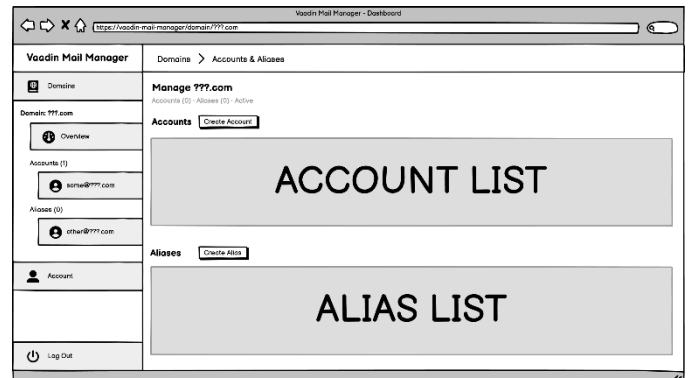
6 - Jamm Add Alias

7.2 Mockups neues User Interface

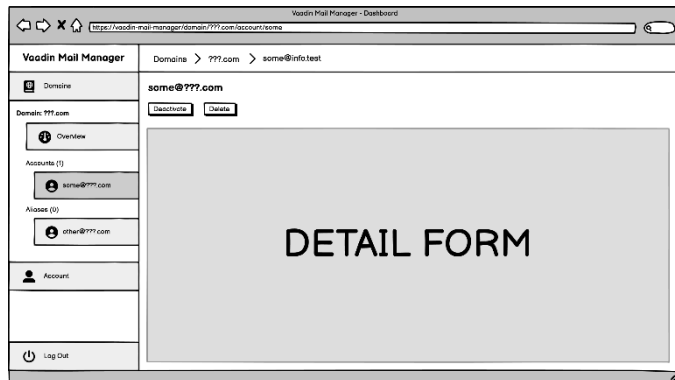
Das neue Interface konzentriert sich auf Etablierte Layouts für Admin Panels, damit die Applikation einfach zu verwalten ist, und sich für jeden seiner Benutzer von der ersten Sekunde an wie das richtige anfühlt.



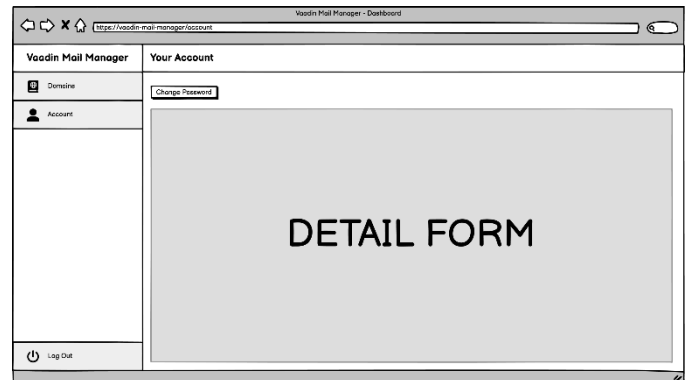
7 - Vamm Site Admin



8 - Vamm Domain Admin



9 - Vamm Manage Alias/Account



10 - Vamm User Account

7.3 Technologie Stack

Der Vaadin Mail Manager nutzt einen modernen Java-Enterprise-Stack basierend auf Spring Boot und Vaadin. Die Architektur ermöglicht eine effiziente Migration der JAMM-Funktionalitäten unter Beibehaltung der bestehenden LDAP-Integration

Komponente	Version	Zweck
Java	21	Laufzeitumgebung (LTS)
Spring Boot	3.4.6	Application Framework
Vaadin	24.7.6	Frontend Framework
Spring Security	-	Authentifizierung & Autorisierung
Spring LDAP	-	LDAP-Integration
Maven	-	Build Management

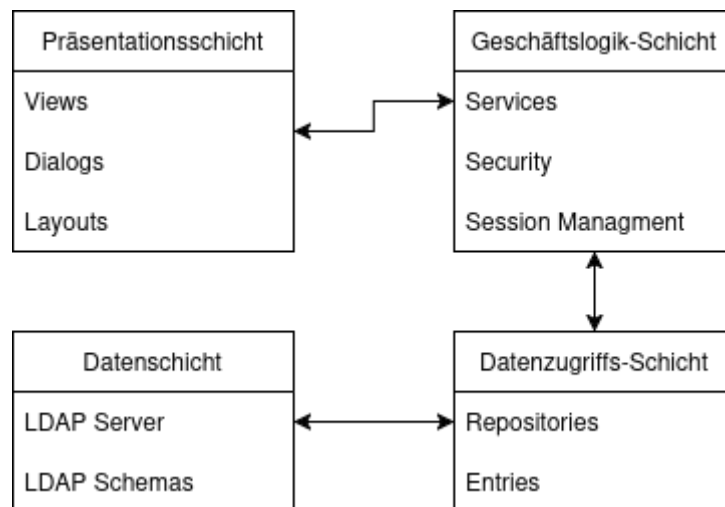
6 - Technologie Stack

8 Systemdokumentation

8.1 Systemarchitektur

Das Vaadin Mail Manager System basiert auf einer modernen, serviceorientierten Architektur, die eine klare Trennung zwischen Präsentationsschicht, Geschäftslogik und Datenschicht implementiert. Die Anwendung nutzt das Spring Framework als Fundament und integriert Vaadin für die Benutzeroberfläche sowie Spring LDAP für die Datenpersistierung.

Die Systemarchitektur folgt dem Model-View-Controller (MVC) Pattern in Kombination mit Domain Driven Design (DDD) Prinzipien. Die Anwendung ist in mehrere Schichten unterteilt, die jeweils spezifische Verantwortlichkeiten übernehmen und durch klar definierte Schnittstellen miteinander kommunizieren.



11 - Systemarchitektur

8.2 Systemanforderungen

8.2.1 Funktionale Anforderungen

- Domain-Verwaltung
 - o Erstellen, Bearbeiten und Löschen virtueller E-Mail-Domains
 - o Aktivierung und Deaktivierung von Domains mit sofortiger Wirkung auf alle zugehörigen Accounts und Aliase
- Account-Verwaltung
 - o Vollständige Benutzerkonto-Verwaltung einschließlich Erstellen, Bearbeiten, Löschen und Status-Management
 - o Sichere Passwort-Verwaltung mit SSHA-Hashing gemäß JAMM-Standards
- Alias-Verwaltung
 - o Erstellung und Verwaltung von E-Mail-Aliassen mit Unterstützung für mehrere Zieladressen
 - o Catch-All-Alias-Unterstützung für domainweite E-Mail-Weiterleitung
- Postmaster-Funktionalität
 - o Automatische Postmaster-Erstellung bei Domain-Einrichtung

- Role-Occupant-Management für die Zuweisung von Domain-Administratoren

8.3 Nichtfunktionale-Anforderungen

- Vollständige LDAP-Schema-Kompatibilität zu bestehenden JAMM-Installationen
- Erhaltung aller JAMM-Attribute und deren Semantik
- **Breadcrumb-Navigation** für bessere Orientierung in der Anwendungsstruktur
- Status-Indikatoren für sofortige Erkennung des Account-/Alias-Status
- Bestätigungsdialoge für kritische Operationen wie Löschvorgänge

8.4 Entwicklungsumgebung

Die Entwicklung des Vaadin Mail Manager erfolgt mit IntelliJ IDEA Ultimate als primäre Entwicklungsumgebung. Diese professionelle IDE bietet umfassende Unterstützung für Spring Boot und Vaadin-Entwicklung, einschließlich intelligenter Code-Vervollständigung, Refactoring-Tools und integrierter Debugging-Funktionen.

Das Projekt basiert auf dem offiziellen Vaadin Start Template, welches eine vorkonfigurierte Spring Boot Anwendung mit allen notwendigen Dependencies bereitstellt. Diese Vorlage beinhaltet bereits optimierte Konfigurationen für Maven, Vaadin Components und Spring Security, wodurch die Entwicklungszeit erheblich reduziert wird.

Für die LDAP-Integration wird ein lokaler OpenLDAP-Server verwendet, der mittels Docker-Container bereitgestellt wird. Die beiliegende Docker-Compose-Konfiguration startet automatisch einen OpenLDAP-Server mit dem erforderlichen JAMM-Schema und Beispieldaten. Diese Konfiguration ermöglicht eine konsistente Entwicklungsumgebung unabhängig vom verwendeten Betriebssystem.

Apache Directory Studio dient als grafisches Werkzeug zur Inspektion und Verwaltung der LDAP-Verzeichnisstruktur während der Entwicklung. Dieses Tool ermöglicht die Visualisierung der Directory-Hierarchie, das Durchsuchen von Einträgen und die Validierung der Schema-Implementierung.

Die Entwicklungsumgebung nutzt Maven als Build-Tool und Dependency-Management-System.

8.5 Konzeption Software

Die Softwarearchitektur des Vaadin Mail Manager folgt etablierten Enterprise-Patterns und modernen Entwicklungsprinzipien. Das Domain-Driven Design (DDD) bildet das konzeptionelle Fundament, wobei die fachlichen Domänen-Objekte (Virtual Domain, Mail Account, Mail Alias, Postmaster) als zentrale Entitäten modelliert werden.

Das Repository Pattern abstrahiert den Datenzugriff und ermöglicht eine einheitliche Schnittstelle zum LDAP-Verzeichnis. Jede Domänen-Entität verfügt über ein dediziertes Repository, welches CRUD-Operationen und domänenspezifische Abfragen bereitstellt. Diese Abstraktion erleichtert Wartung, Testing und potenzielle Migration zu anderen Datenquellen.

Die Service-Schicht implementiert die Geschäftslogik und orchestriert Operationen zwischen verschiedenen Repositories. Management-Services kapseln komplexe Workflows wie die Erstellung von Domänen oder die Validierung von E-Mail-Adressen vor der Account-Erstellung.

Spring Security integriert sich nahtlos in die Anwendungsarchitektur und stellt rollenbasierte Zugriffskontrolle bereit. Ein custom LDAP Authentication Provider ermöglicht die Authentifizierung gegen den JAMM-LDAP-Server, während der Security Service autorisierungsrelevante Funktionen zentral verwaltet.

Die Vaadin-basierte Benutzeroberfläche implementiert das Model-View-Presenter (MVP) Pattern. Views sind reine Präsentationskomponenten ohne Geschäftslogik, während Services die erforderlichen Daten bereitstellen. Event-basierte Kommunikation zwischen UI-Komponenten gewährleistet lose Kopplung und verbesserte Reaktivität.

Ein umfassendes Exception-Handling sorgt für robuste Fehlerbehandlung auf allen Architektur-Ebenen. Validation erfolgt sowohl client- als auch serverseitig, wobei Vaadin's integrierte Validierungsmechanismen mit custom Business Rules kombiniert werden.

9 Implementierungsstand

LDAP-Backend Integration Vollständige LDAP-Anbindung mit sicherer Authentifizierung, Session-Management und konfigurierbaren Verbindungsparametern. Unterstützt Benutzer-spezifische Credentials und automatisches DN-Management.	100%
Benutzerauthentifizierung LDAP-basierte Anmeldung mit Root-User-Support. <i>Fehlt: Domain-Admin und Standard-User Login vollständig implementiert.</i>	50%
Domain-Verwaltung Komplette Verwaltung virtueller E-Mail-Domains mit Status-Kontrolle, Aktivierung/Deaktivierung und automatischer Postmaster-Erstellung. <i>Fehlt: Edit-Dialog.</i>	95%
E-Mail Account Management Vollständige Account-Erstellung, -Bearbeitung und -Löschung mit sicherer Passwort-Verwaltung, Quota-Unterstützung und Status-Kontrolle. <i>Fehlt: Bulk-Operationen und erweiterte Suchfunktionen.</i>	90%
E-Mail Account Account-Detail-Management Detaillierte Account-Ansicht mit Status-Management, Quota-Verwaltung und vollständigen Account-Informationen. <i>Fehlt: Erweiterte Account-Eigenschaften, Editierbarkeit und Set Passwort</i>	85%
Alias-Verwaltung Umfassende Alias-Verwaltung inkl. Catch-All-Unterstützung, Multiple-Ziel und dynamische Ziel -Verwaltung. <i>Fehlt: Erweiterte Alias-Typen und Validierung von Alias-Ketten.</i>	75%
Alias-Detail-Management Umfassende Alias-Detailansicht mit Ziel-Management und Catch-All-Erkennung. Fehlt: Erweiterte Alias-Konfigurationen., Edit Dialog	70%
Rollenbasierte Zugriffskontrolle Grundlegende Sicherheitsarchitektur mit Site-Admin-Rolle implementiert. <i>Fehlt: Granulare Domain-Admin Überprüfung.</i>	70%
Postmaster-Verwaltung Backend vollständig implementiert. <i>Fehlt: Handling in AliasDetailView oder Dedizierte View</i>	75%
Status-Management Vollständige Status-Kontrolle für Domains, Accounts und Aliase mit visuellen Indikatoren und Toggle-Funktionalität	100%
Catch-All-Support Vollständige Unterstützung für Catch-All-Aliase mit spezieller Behandlung und Visualisierung. <i>Fehlt: Erstellung von Catch-All Adressen</i>	90%
GESAMT	Ca 72%

10 Testkonzept

10.1 Testphilosophie und strategischer Ansatz

Die Teststrategie für den Vaadin Mail Manager basiert auf dem Prinzip der systematischen Qualitätssicherung durch schichtweise Validierung. Der Ansatz folgt einer Bottom-Up-Architektur, die von der atomaren Funktionsprüfung einzelner Datenstrukturen bis zur vollständigen Validierung komplexer Geschäftsprozesse reicht.

Die Testphilosophie stellt die Zuverlässigkeit und Sicherheit der E-Mail-Verwaltung in den Mittelpunkt. Jede Schicht wird isoliert validiert, bevor ihre Integration mit anderen Komponenten getestet wird. Dies gewährleistet eine systematische Identifikation und Behebung von Problemen und schafft eine solide Grundlage für die produktive Nutzung des Systems.

Die Migration von JAMM zu einer modernen Vaadin-Anwendung erfordert besondere Aufmerksamkeit bei der Validierung der LDAP-Integration und der Geschäftslogik-Korrektheit. Die Teststrategie berücksichtigt diese Anforderungen durch umfassende Integrationstests gegen eine reale LDAP-Instanz und systematische Validierung aller E-Mail-Management-Workflows.

10.2 Schichtbasierte Testarchitektur

10.2.1 Entitätsschicht: Fundament der Datenintegrität

Die Entitätsschicht bildet das Fundament der Anwendung und wird durch 37 systematische Testfälle validiert. Diese Schicht gewährleistet die korrekte Abbildung der JAMM-Datenstrukturen in moderne Java-Objekte und die ordnungsgemäße Integration mit dem Spring LDAP Framework.

Die 10 kritischsten Validierungen der Entitätsschicht:

1. Sichere Passwort-Verschlüsselung: Validierung der SSHA-Hash-Generierung und Verifikation verschiedener Passwort-Schemas für maximale Sicherheit
2. LDAP-DN-Generierung: Automatische Erstellung korrekter Distinguished Names entsprechend der JAMM-Verzeichnisstruktur
3. E-Mail-Format-Validierung: Zuverlässige Extraktion von Benutzernamen und Domains aus E-Mail-Adressen für konsistente Datenverarbeitung
4. Boolean-String-Konvertierung: Korrekte Umwandlung zwischen Java-Boolean-Werten und LDAP-String-Repräsentationen
5. Zeitstempel-Management: Automatische Aktualisierung von Änderungszeitpunkten bei Datenmodifikationen
6. Account-Status-Verwaltung: Zuverlässige Aktivierung, Deaktivierung und Löschmarkierung von E-Mail-Accounts
7. Alias-Destination-Management: Sichere Verwaltung von E-Mail-Weiterleitungszielen mit Duplikatsprävention
8. Postmaster-Role-Occupant-Sicherheit: Gewährleistung mindestens eines Administrators pro Domain
9. Domain-Lifecycle-Management: Korrekte Behandlung von Domain-Status-Übergängen und Berechtigungen
10. Konstruktor-Validierung: Zuverlässige Objektinitialisierung mit korrekten Standardwerten

10.2.2 Repository-Schicht: LDAP-Integration und Datenzugriff

Die Repository-Schicht wird durch 47 Integrationstests gegen eine reale LDAP-Instanz validiert. Diese Schicht gewährleistet die zuverlässige Persistierung und den sicheren Abruf von E-Mail-Verwaltungsdaten aus dem LDAP-Verzeichnis.

Die 10 kritischsten Validierungen der Repository-Schicht:

1. CRUD-Operationen-Integrität: Vollständige Create-Read-Update-Delete-Zyklen für alle Entitätstypen mit korrekter LDAP-Persistierung
2. Domain-basierte Hierarchie-Suche: Effiziente Filterung von Accounts und Aliassen nach Domain-Zugehörigkeit
3. Existenz-Prüfung ohne Objektladung: Performance-optimierte Validierung der Entitäts-Existenz für Duplikatsprävention
4. Batch-Löschoperationen: Sichere Domain-weite Löschung aller abhängigen Entitäten
5. Exception-Propagation: Aussagekräftige Fehlerbehandlung bei LDAP-Operationsfehlern
6. LDAP-Injection-Schutz: Sichere Escaping-Mechanismen bei Suchoperationen mit Benutzereingaben
7. Transaktionale Konsistenz: Gewährleistung der Datenintegrität bei mehreren LDAP-Operationen
8. Complex-Query-Performance: Effiziente Ausführung von Wildcard- und Präfix-Suchoperationen
9. DN-basierte Direktzugriffe: Optimierte Objektladung über Distinguished Names
10. Concurrent-Access-Sicherheit: Thread-sichere Repository-Operationen bei parallelen Zugriffen

10.2.3 Service-Schicht: Geschäftslogik und Workflow-Orchestrierung

Die Service-Schicht wird durch 65 umfassende Testfälle validiert, die die komplette E-Mail-Management-Geschäftslogik und die Orchestrierung mehrerer Repository-Zugriffe gewährleisten.

Die 10 kritischsten Validierungen der Service-Schicht:

1. Account-Erstellung mit Konfliktprävention: Umfassende Validierung neuer E-Mail-Accounts mit Duplikats- und Alias-Konfliktprüfung
2. Cross-Entity-Validierung: Gewährleistung der referenziellen Integrität zwischen Accounts, Aliassen und Domains
3. Domain-Management mit Postmaster-Integration: Automatische Erstellung und Verwaltung von Postmaster-Accounts bei Domain-Operationen
4. Passwort-Management-Sicherheit: Sichere Passwort-Änderung mit Validierung und Hash-Aktualisierung
5. Status-Lifecycle-Management: Konsistente Verwaltung von Account- und Alias-Status-Übergängen
6. Cascade-Delete-Integrität: Vollständige Löschung von Domains mit allen abhängigen Entitäten
7. Geschäftsregel-Enforcement: Durchsetzung aller E-Mail-Management-Regeln wie Mindest-Destinations bei Aliassen
8. Service-Integration-Konsistenz: Korrekte Koordination zwischen Account-, Alias- und Domain-Management-Services
9. Error-Handling-Robustheit: Aussagekräftige Exception-Behandlung mit Kontext-spezifischen Fehlermeldungen
10. Statistik-Konsistenz: Zuverlässige Berechnung von Account- und Alias-Zählungen für Dashboard-Anzeigen

10.3 Qualitätssicherung und Erfolgsmessung

10.3.1 Validierungsansatz

Die Teststrategie implementiert eine mehrstufige Validierung, die sowohl positive als auch negative Testszenarien umfasst. Jede Funktionalität wird unter normalen Betriebsbedingungen sowie bei Fehlersituationen getestet. Dies gewährleistet die Robustheit der Anwendung unter verschiedenen Betriebsszenarien.

10.3.2 Erfolgskriterien

Der Erfolg der Teststrategie wird durch die vollständige Durchführung aller 112 Testfälle ohne kritische Fehler gemessen.

11Fazit

Das Projekt hat mir außerordentlich viel Freude bereitet. Seit Beginn meiner Ausbildung an dieser Schule hatte ich bei jedem Projekt ausreichend Zeit für eine gründliche Vorbereitung und konnte mir die notwendigen Freiräume schaffen. Der Arbeitsalltag stellte jedoch eine völlig neue Herausforderung dar. Plötzlich vergingen Wochen ohne nennenswerten Fortschritt, und die Abgabe rückte bedrohlich näher.

Meine ursprüngliche Zeitplanung erwies sich als unzureichend. Besonders gegen Projektende musste ich stark priorisieren, um das Vorhaben innerhalb der verfügbaren Zeit zu einem Abschluss zu bringen. Diese Erfahrung war ernüchternd, aber auch lehrreich für zukünftige Projekte.

Während der Entwicklung passte ich den Projektfokus an, da mir bewusst wurde, dass meine ursprüngliche Schwerpunktsetzung zu oberflächlich war. Statt oberflächlich um ein mir unbekanntes System zu arbeiten, konzentrierte ich mich intensiver auf das Verständnis der LDAP-Integration. Diese Anpassung habe ich in Abschnitt 3.11 dokumentiert.

Trotz der zeitlichen Herausforderungen bin ich mit dem Ergebnis sehr zufrieden. Das Projekt ist weit von einer vollständig funktionierenden Lösung entfernt, und nicht alle implementierten Funktionen sind fehlerfrei. Dennoch ist die Anwendung mit einem soliden Grundgerüst ausgestattet, das eine Bereinigung der verbleibenden Probleme mit ein bis zwei Arbeitstagen Aufwand ermöglichen wird.

Rückblickend kam diese Selbstständige Orientierungsarbeit einer IPA-Simulation am nächsten, da ich in relativ kurzer Zeit intensiv an einem komplexen Thema arbeiten musste. Ironischerweise habe ich deutlich weniger Zeit als ursprünglich geplant investiert, da die Deadline schneller näher rückte als meine eingeplante Projektzeit zur Verfügung stand. Die Zeit hat mich letztendlich eingeholt, aber die gewonnenen Erkenntnisse in Projektmanagement und LDAP-Integration waren die Herausforderungen wert.

12Ausblick

Das entwickelte Grundgerüst bietet eine solide Basis für die weitere Entwicklung des Vaadin Mail Managers. Die implementierten LDAP-Entitäten und die Teststruktur ermöglichen eine schrittweise Erweiterung der Funktionalität. Mit der gewonnenen Erfahrung in LDAP-Integration und den erkannten Verbesserungsmöglichkeiten wäre eine Fortsetzung des Projekts deutlich effizienter umsetzbar.

Ich hoffe und freue mich darauf dieses Projekt ausserhalb von SOL zuende zu bringen und für Aarboard AG einsetzbar zu machen.

13 Verweise

13.1 Tabellenverzeichnis

1 - VERSIONIERUNG.....	2
2 - TAG 1: 09.06	11
3 - TAG 2: 10.06	12
4 - ERSTER REALISIERUNGSTEIL 11.06 – 13.06	14
5 - ZWEITER REALISIERUNGSTEIL 13.06 - 14.06	15
6 - TECHNOLOGIE STACK	21
7 - IMPLEMENTIERUNGSSTAND.....	25

13.2 Abbildungsverzeichnis

1 - ZEITPLANUNG	10
2 - JAMM SITE ADMIN	18
3 - JAMM DOMAIN ADMIN.....	19
4 - JAMM ADD DOMAIN.....	20
5 - JAMM ADD ACCOUNT.....	20
6 - JAMM ADD ALIAS	20
7 - VAMM SITE ADMIN	21
8 - VAMM DOMAIN ADMIN	21
9 - VAMM MANAGE ALIAS/ACCOUNT	21
10 - VAMM USER ACCOUNT.....	21
11 - SYSTEMARCHITEKTUR.....	22

14Glossar

Begriff/Abkürzung	Beschreibung
Alias	Weiterleitung einer E-Mail-Adresse an eine oder mehrere Zieladressen. Ermöglicht es, E-Mails an verschiedene Konten ohne eigene Mailbox zu verteilen.
Artifact	Baustein in Maven-basierten Projekten, der eine spezifische Funktionalität oder Bibliothek repräsentiert und über Koordinaten eindeutig identifiziert wird.
Breadcrumb-Navigation	Navigationselement, das dem Benutzer seinen aktuellen Standort in der Anwendungsstruktur anzeigt und die Rückkehr zu übergeordneten Ebenen ermöglicht.
Catch-All-Alias	Spezielle E-Mail-Weiterleitung die alle an eine Domain gesendeten E-Mails abfängt, für die kein spezifisches Konto oder Alias existiert.
CRUD-Operationen	Create, Read, Update, Delete - Die vier grundlegenden Datenbankoperationen für die Verwaltung von Entitäten in persistenten Systemen.
Distinguished Name (DN)	Eindeutige Bezeichnung eines Objekts im LDAP-Verzeichnis, die den vollständigen Pfad vom Objekt zur Wurzel des Verzeichnisbaums darstellt.
Domain-Driven Design	Softwareentwicklungsansatz, der die Modellierung der Geschäftsdomäne in den Mittelpunkt stellt und fachliche Konzepte direkt in der Codestruktur abbildet.
Entry (LDAP)	Einzelnes Objekt im LDAP-Verzeichnis, das durch Attribute und Objektklassen definiert wird und über einen Distinguished Name eindeutig identifiziert wird.
Framework	Softwaregerüst, das eine wiederverwendbare Struktur für die Entwicklung von Anwendungen bereitstellt und gemeinsame Funktionalitäten abstrahiert.
IPERKA-Methodik	Projektmanagement-Ansatz mit den Phasen Informieren, Planen, Entscheiden, Realisieren, Kontrollieren und Auswerten.
JAMM (Java Mail Manager)	Legacy-System für E-Mail-Verwaltung, das als Grundlage für die Migration zum Vaadin Mail Manager dient.
LDAP (Lightweight Directory Access Protocol)	Netzwerkprotokoll für den Zugriff auf und die Verwaltung von Verzeichnisdiensten, das in diesem Projekt für die Persistierung von E-Mail-Verwaltungsdaten verwendet wird.
LDAP-Schema	Definition der Objektklassen und Attribute, die in einem LDAP-Verzeichnis verwendet werden können, vergleichbar mit einem Datenbankschema.
LTS (Long Term Support)	Langzeit-unterstützte Versionen von Software, die über einen erweiterten Zeitraum Sicherheitsupdates und Wartung erhalten.
Maven	Build-Management-Tool für Java-Projekte, das Abhängigkeiten verwaltet, den Erstellungsprozess automatisiert und Projektstrukturen standardisiert.

Model-View-Controller (MVC)	Weiterentwicklung des MVC-Patterns, bei dem der Presenter die gesamte UI-Logik übernimmt und die View von Geschäftslogik befreit.
MVP (Model-View-Presenter)	Die Methoden und Funktionen, die verwendet werden, um Eingabedaten von Sensoren oder Benutzeraktionen zu verarbeiten.
Objektklasse (LDAP)	Template im LDAP-Schema, das definiert, welche Attribute ein LDAP-Entry haben kann und welche davon erforderlich sind.
ODM (Object Document Mapping)	Technik zur Abbildung von objektorientierten Programmstrukturen auf dokumentenbasierte oder verzeichnisbasierte Datenstrukturen wie LDAP.
Postmaster	Administrativer E-Mail-Account einer Domain, der für die Verwaltung domänenspezifischer E-Mail-Konfigurationen und administrativer Aufgaben zuständig ist.
Repository-Pattern	Entwurfsmuster, das eine einheitliche Schnittstelle für den Datenzugriff bereitstellt und die Geschäftslogik von der spezifischen Datenquelle entkoppelt.
Role-Occupant	LDAP-Attribut, das definiert, welche Benutzer administrative Berechtigungen für bestimmte Domänen oder Ressourcen besitzen.
Service-Schicht	Architekturschicht, die Geschäftslogik kapselt und als Vermittler zwischen Präsentationsschicht und Datenzugriffsschicht fungiert.
Spring Boot	Framework für die Entwicklung von Java-Anwendungen, das Convention-over-Configuration-Prinzipien anwendet und die Konfiguration minimiert.
Spring LDAP	Spring-Framework-Erweiterung, die eine vereinfachte API für LDAP-Operationen bereitstellt und die Integration von LDAP in Spring-Anwendungen erleichtert.
SSHA (Salted SHA)	Passwort-Hashing-Verfahren, das SHA-1-Hashing mit einem zufälligen Salt kombiniert, um die Sicherheit gespeicherter Passwörter zu erhöhen.
Transient (JPA/LDAP)	Kennzeichnung für Objektattribute, die nicht in der persistenten Speicherung (Datenbank/LDAP) gespeichert werden, sondern nur zur Laufzeit existieren.
UI (User Interface)	Benutzeroberfläche, die die Interaktion zwischen Anwender und Anwendung ermöglicht und in diesem Projekt mit Vaadin-Komponenten realisiert wird.
Vaadin	Java-Framework für die Entwicklung moderner Webanwendungen, das serverseitige Programmierung mit komponentenbasierten UI-Elementen ermöglicht.
Virtual Domain	Logische E-Mail-Domain, die im System verwaltet wird, ohne zwingend eine eigenständige physische Server-Infrastruktur zu erfordern.
Workflow-Orchestrierung	Koordination und Steuerung komplexer Geschäftsprozesse durch systematische Abfolge von Service-Aufrufen und Datenverarbeitungsschritten.

15Programmcode

15.1 Application.java

Haupteinstiegspunkt der Vaadin Mail Manager Anwendung

```
@SpringBootApplication
@Theme(value = "vaadin-mail-manager")
public class Application implements AppShellConfigurator {

    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }

}
```

15.2 Entitäten

Kerngeschäftslogik und Datenmodell

15.2.1 JammVirtualDomain.java

Diese Entitätsklasse repräsentiert eine virtuelle E-Mail-Domain im LDAP-Verzeichnis und kapselt alle domänenspezifischen Funktionalitäten. Die Klasse verwaltet Domain-Zustände wie Aktivierung, Löschmarkierung und Bearbeitungsberechtigungen für Konten und Postmaster. Durch die Spring LDAP ODM-Integration erfolgt die direkte Abbildung auf die JammVirtualDomain-Objektklasse im LDAP-Schema.

```
@Entry(objectClasses = {"top", "JammVirtualDomain"})
public final class JammVirtualDomain {

    @Id
    private Name id;

    @Attribute(name = "jvd")
    private String jvd;

    // Boolean
    @Attribute(name = "accountActive")
    private String accountActive;

    // Long
    @Attribute(name = "lastChange")
    private String lastChange;

    // Boolean
    @Attribute(name = "delete")
    private String delete;

    // Boolean
    @Attribute(name = "editAccounts")
    private String editAccounts;

    // Boolean
    @Attribute(name = "editPostmasters")
    private String editPostmasters;

    @Attribute(name = "postfixTransport")
    private String postfixTransport;

    @Attribute(name = "description")
    private String description;

    @Transient
    private int accountCount = 0;
```

```

@Transient
private int aliasCount = 0;

// Constructors
public JammVirtualDomain() {
    this.accountActive = "TRUE";
    this.delete = "FALSE";
    this.editAccounts = "TRUE";
    this.editPostmasters = "TRUE";
    this.lastChange = String.valueOf(Instant.now().getEpochSecond());
    this.postfixTransport = "virtual:";
}

public JammVirtualDomain(String domainName) {
    this();
    this.jvd = domainName;
    this.id = LdapUtils.domainDN(domainName).build();
}

/**
 * Mark the domain for deletion.
 * This sets the delete flag to "TRUE" and deactivates the account.
 */
public void markForDeletion() {
    setMarkedForDeletion(true);
    setActive(false);
    updateLastChange();
}

/**
 * Activate the domain.
 * This sets the accountActive flag to "TRUE" and resets the delete flag.
 */
public void activate() {
    setMarkedForDeletion(false);
    setActive(true);
    updateLastChange();
}

/**
 * Deactivate the domain.
 * This sets the accountActive flag to "FALSE" and updates the last change timestamp.
 */
public void deactivate() {
    setActive(false);
    updateLastChange();
}

/**
 * Check if the domain is active.

```

```

* @return true if the domain is active, false otherwise.
*/
public boolean isActive() {
    return "TRUE".equals(accountActive);
}

/**
* Set the active status of the domain.
* @param active true to activate, false to deactivate.
*/
public void setActive(boolean active) {
    this.accountActive = active ? "TRUE" : "FALSE";
}

/**
* Check if the domain is marked for deletion.
* @return true if the domain is marked for deletion, false otherwise.
*/
public boolean isMarkedForDeletion() {
    return "TRUE".equals(delete);
}

/**
* Set the domain marked for deletion status.
* @param deleted true to mark the domain for deletion, false to unmark.
*/
public void setMarkedForDeletion(boolean deleted) {
    this.delete = deleted ? "TRUE" : "FALSE";
}

/**
* Check if the domain allows editing of accounts.
* @return true if accounts can be edited, false otherwise.
*/
public boolean canEditAccounts() {
    return "TRUE".equals(editAccounts);
}

/**
* Set whether the domain allows editing of accounts.
* @param canEdit true to allow editing, false to disallow.
*/
public void setCanEditAccounts(boolean canEdit) {
    this.editAccounts = canEdit ? "TRUE" : "FALSE";
}

/**
* Check if the domain allows editing of postmasters.
* @return true if postmasters can be edited, false otherwise.
*/

```



```

public boolean canEditPostmasters() {
    return "TRUE".equals(editPostmasters);
}

/**
 * Set whether the domain allows editing of postmasters.
 * @param canEdit true to allow editing, false to disallow.
 */
public void setCanEditPostmasters(boolean canEdit) {
    this.editPostmasters = canEdit ? "TRUE" : "FALSE";
}

/**
 * Update the last change timestamp to the current time.
 * This is typically called when any significant change is made to the domain.
 */
public void updateLastChange() {
    this.lastChange = String.valueOf(Instant.now().getEpochSecond());
}

/**
 * Convenience method to get the last change timestamp as a Long.
 * @return the last change timestamp as a Long, or null if it cannot be parsed.
 */
public Long getLastChangeAsLong() {
    try {
        return lastChange != null ? Long.valueOf(lastChange) : null;
    } catch (NumberFormatException e) {
        return null;
    }
}

/**
 * Convenience method to set the last change timestamp from a Long.
 * @param lastChange the last change timestamp as a Long.
 */
public void setLastChangeAsLong(Long lastChange) {
    this.lastChange = lastChange != null ? String.valueOf(lastChange) : null;
}

// Getters and Setters - all return/accept Strings to match LDAP format
public Name getId() { return id; }
public void setId(Name id) { this.id = id; }

public String getJvd() { return jvd; }
public void setJvd(String jvd) { this.jvd = jvd; }

public String getAccountActive() { return accountActive; }
public void setAccountActive(String accountActive) { this.accountActive = accountActive; }

```

```

public String getLastChange() { return lastChange; }
public void setLastChange(String lastChange) { this.lastChange = lastChange; }

public String getDelete() { return delete; }
public void setDelete(String delete) { this.delete = delete; }

public String getEditAccounts() { return editAccounts; }
public void setEditAccounts(String editAccounts) { this.editAccounts = editAccounts; }

public String getEditPostmasters() { return editPostmasters; }
public void setEditPostmasters(String editPostmasters) { this.editPostmasters = editPostmasters; }
}

public String getPostfixTransport() { return postfixTransport; }
public void setPostfixTransport(String postfixTransport) { this.postfixTransport = postfixTransport; }

public String getDescription() { return description; }
public void setDescription(String description) { this.description = description; }

// Transient fields
public int getAccountCount() { return accountCount; }
public void setAccountCount(int accountCount) { this.accountCount = accountCount; }

public int getAliasCount() { return aliasCount; }
public void setAliasCount(int aliasCount) { this.aliasCount = aliasCount; }

@Override
public String toString() {
    return "JammVirtualDomain{" +
        "jvd=\"" + jvd + "\" +
        ", accountActive=\"" + accountActive + "\" +
        ", delete=\"" + delete + "\" +
        ", editAccounts=\"" + editAccounts + "\" +
        ", editPostmasters=\"" + editPostmasters + "\" +
        ", lastChange=\"" + lastChange + "\" +
        ", postfixTransport=\"" + postfixTransport + "\" +
        ", description=\"" + description + "\" +
        ", accountCount=\"" + accountCount +
        ", aliasCount=\"" + aliasCount +
        '}';
}
}

```

15.2.2 JammMailAccount.java

Die zentrale Entität für die Verwaltung individueller E-Mail-Konten implementiert umfassende Kontofunktionalitäten einschließlich sicherer Passwort-Hashing-Mechanismen und Quota-Management. Die Klasse bietet verschiedene Passwort-Verschlüsselungsverfahren (SSHA, SHA, MD5) und verwaltet Konto-Metadaten wie Aktivierungsstatus und Löschmarkierungen. Die LDAP-Integration erfolgt über die JammMailAccount-Objektklasse.

```
/**
 * LDAP entity representing a Jamm mail account.
 * Maps to the JammMailAccount objectClass.
 */
@Entry(objectClasses = {"top", "JammMailAccount"})
public final class JammMailAccount {

    @Id
    private Name id;

    @Attribute(name = "mail")
    private String mail;

    @Attribute(name = "homeDirectory")
    private String homeDirectory;

    @Attribute(name = "mailbox")
    private String mailbox;

    // Boolean
    @Attribute(name = "accountActive")
    private String accountActive;

    // Long
    @Attribute(name = "lastChange")
    private String lastChange;

    // Boolean
    @Attribute(name = "delete")
    private String delete;

    @Attribute(name = "uidNumber")
    private Integer uidNumber;

    @Attribute(name = "gidNumber")
    private Integer gidNumber;

    @Attribute(name = "uid")
    private String uid;

    @Attribute(name = "cn")
```

```

private String commonName;

@Attribute(name = "description")
private String description;

@Attribute(name = "quota")
private String quota;

@Attribute(name = "userPassword")
private String userPassword;

@Attribute(name = "clearPassword")
private String clearPassword;

public JammMailAccount() {
    this.accountActive = "TRUE";
    this.delete = "FALSE";
    this.lastChange = String.valueOf(Instant.now().getEpochSecond());
}

public JammMailAccount(String mail, String homeDirectory, String mailbox) {
    this();
    this.mail = mail;
    this.homeDirectory = homeDirectory;
    this.mailbox = mailbox;
    this.id = LdapUtils.mailDN(mail).build();
}

/**
 * Set password securely using SSHA hashing
 * This method hashes the password and updates the last change timestamp.
 * @param plainPassword the plain text password to hash
 */
public void setPasswordSecure(String plainPassword) {
    if (plainPassword == null || plainPassword.trim().isEmpty()) {
        throw new IllegalArgumentException("Password cannot be empty");
    }

    this.clearPassword = plainPassword;
    this.userPassword = PasswordUtils.hashPasswordSsha(plainPassword);
    updateLastChange();
}

/**
 * Set password with a specific scheme
 * This method allows setting the password with different hashing schemes.
 * @param plainPassword the plain text password to hash
 * @param scheme the password hashing scheme to use
 */

```

```

public void setPasswordWithScheme(String plainPassword, PasswordUtils.PasswordScheme
scheme) {
    if (plainPassword == null || plainPassword.trim().isEmpty()) {
        throw new IllegalArgumentException("Password cannot be empty");
    }

    this.clearPassword = plainPassword;

    switch (scheme) {
        case PLAIN -> this.userPassword = PasswordUtils.PasswordScheme.PLAIN.getPrefix() +
plainPassword;
        case SSHA -> this.userPassword = PasswordUtils.hashPasswordSsha(plainPassword);
        case SHA -> this.userPassword = PasswordUtils.hashPasswordSha(plainPassword);
        case MD5 -> this.userPassword = PasswordUtils.hashPasswordMd5(plainPassword);
    }

    updateLastChange();
}

/**
 * Verify the provided plain password against the stored hashed password.
 * This method uses the PasswordUtils to verify the password.
 * @param plainPassword the plain text password to verify
 * @return true if the password matches, false otherwise
 */
public boolean verifyPassword(String plainPassword) {
    if (userPassword == null || plainPassword == null) {
        return false;
    }
    return PasswordUtils.verifyPassword(userPassword, plainPassword);
}

/**
 * Check if the account is active.
 * @return true if the account is active, false otherwise
 */
public boolean isActive() {
    return "TRUE".equals(accountActive);
}

/**
 * Set the account active status.
 * @param active true to activate the account, false to deactivate
 */
public void setActive(boolean active) {
    this.accountActive = active ? "TRUE" : "FALSE";
}

/**
 * Check if the account is marked for deletion.

```

```

* @return true if the account is marked for deletion, false otherwise
*/
public boolean isMarkedForDeletion() {
    return "TRUE".equals(delete);
}

/**
* Set the account marked for deletion status.
* @param deleted true to mark the account for deletion, false to unmark
*/
public void setMarkedForDeletion(boolean deleted) {
    this.delete = deleted ? "TRUE" : "FALSE";
}

/**
* Get the full path to the mailbox.
* This combines homeDirectory and mailbox into a single path.
* @return the full path to the mailbox, or null if either part is missing
*/
public String getFullPathToMailbox() {
    if (homeDirectory != null && mailbox != null) {
        return homeDirectory + "/" + mailbox;
    }
    return null;
}

/**
* Get the account name (local part of the email).
* This is a convenience method to extract the local part of the email address.
* @return the local part of the email address, or the full email if it cannot be parsed
*/
public String getAccountName() {
    if (mail != null && mail.contains("@")) {
        return mail.substring(0, mail.indexOf("@"));
    }
    return mail;
}

/**
* Get the domain part of the email address.
* This is a convenience method to extract the domain from the email address.
* @return the domain part of the email address, or an empty string if it cannot be parsed
*/
public String getDomain() {
    return MailUtils.extractDomainFromMail(mail);
}

/**
* Update the last change timestamp to the current time.
* This method sets the lastChange attribute to the current epoch second.

```

```

*/
public void updateLastChange() {
    this.lastChange = String.valueOf(Instant.now().getEpochSecond());
}

/**
 * Convenience method to get the last change timestamp as a Long.
 * This method parses the lastChange string and returns it as a Long.
 * @return the last change timestamp as a Long, or null if it cannot be parsed
 */
public Long getLastChangeAsLong() {
    try {
        return lastChange != null ? Long.valueOf(lastChange) : null;
    } catch (NumberFormatException e) {
        return null;
    }
}

/**
 * Set the last change timestamp from a Long value.
 * This method converts the Long to a String and sets it as the lastChange attribute.
 * @param lastChange the last change timestamp as a Long, or null to clear it
 */
public void setLastChangeAsLong(Long lastChange) {
    this.lastChange = lastChange != null ? String.valueOf(lastChange) : null;
}

// Getters and Setters - return/accept Strings for LDAP compatibility
public Name getId() { return id; }
public void setId(Name id) { this.id = id; }

public String getMail() { return mail; }
public void setMail(String mail) { this.mail = mail; }

public String getHomeDirectory() { return homeDirectory; }
public void setHomeDirectory(String homeDirectory) { this.homeDirectory = homeDirectory; }

public String getMailbox() { return mailbox; }
public void setMailbox(String mailbox) { this.mailbox = mailbox; }

public String getAccountActive() { return accountActive; }
public void setAccountActive(String accountActive) { this.accountActive = accountActive; }

public String getLastChange() { return lastChange; }
public void setLastChange(String lastChange) { this.lastChange = lastChange; }

public String getDelete() { return delete; }
public void setDelete(String delete) { this.delete = delete; }

public Integer getUidNumber() { return uidNumber; }

```

```

public void setUidNumber(Integer uidNumber) { this.uidNumber = uidNumber; }

public Integer getGidNumber() { return gidNumber; }
public void setGidNumber(Integer gidNumber) { this.gidNumber = gidNumber; }

public String getUid() { return uid; }
public void setUid(String uid) { this.uid = uid; }

public String getCommonName() { return commonName; }
public void setCommonName(String commonName) { this.commonName = commonName; }

public String getDescription() { return description; }
public void setDescription(String description) { this.description = description; }

public String getQuota() { return quota; }
public void setQuota(String quota) { this.quota = quota; }

public String getUserPassword() { return userPassword; }
public void setUserPassword(String userPassword) { this.userPassword = userPassword; }

public String getClearPassword() { return clearPassword; }
public void setClearPassword(String clearPassword) { this.clearPassword = clearPassword; }

@Override
public String toString() {
    return "JammMailAccount{" +
        "mail=\"" + mail + "\" +
        ", accountActive=\"" + accountActive + "\" +
        ", delete=\"" + delete + "\" +
        ", commonName=\"" + commonName + "\" +
        ", mailbox=\"" + mailbox + "\" +
        '}'";
}
}

```

15.2.3 JammMailAlias.java

Diese Klasse implementiert die Verwaltung von E-Mail-Weiterleitungen und Catch-All-Aliases mit Unterstützung für mehrere Zieladressen. Die Entität unterscheidet zwischen Standard-Aliases und Catch-All-Konfigurationen und bietet dynamische Destination-Verwaltung. Die Implementierung nutzt die JammMailAlias LDAP-Objektklasse für die persistente Speicherung

```

/**
 * LDAP entity representing a Jamm mail alias.
 * Maps to the JammMailAlias objectClass.
 */
@Entry(objectClasses = {"top", "JammMailAlias"})
public final class JammMailAlias {

```



```

@Id
private Name id;

@Attribute(name = "mail")
private String mail;

@Attribute(name = "maildrop")
private List<String> maildrop;

// Boolean
@Attribute(name = "accountActive")
private String accountActive;

// Long
@Attribute(name = "lastChange")
private String lastChange;

@Attribute(name = "mailsource")
private String mailsource;

@Attribute(name = "cn")
private String commonName;

@Attribute(name = "description")
private String description;

@Attribute(name = "userPassword")
private String userPassword;

public JammMailAlias() {
    this.accountActive = "TRUE";
    this.lastChange = String.valueOf(Instant.now().getEpochSecond());
    this.maildrop = new ArrayList<>();
}

public JammMailAlias(String mail, List<String> destinations, String commonName) {
    this();
    this.mail = mail;
    this.maildrop = new ArrayList<>(destinations);
    if (commonName != null && !commonName.isEmpty()) {
        this.commonName = commonName;
    } else {
        this.commonName = MailUtils.extractUserFromMail(mail);
    }
    this.id = LdapUtils.mailDN(mail).build();
}

public JammMailAlias(String mail, String... destinations) {
    this();

```

```

    this.mail = mail;
    this.maildrop = new ArrayList<>();
    Collections.addAll(this.maildrop, destinations);
    this.commonName = MailUtils.extractUserFromMail(mail);
    this.id = LdapUtils.mailDN(mail).build();
}

/**
 * Convenience method to check if the alias is active.
 * @return true if the alias is active, false otherwise.
 */
public boolean isActive() {
    return "TRUE".equals(accountActive);
}

/**
 * Convenience method to set the active status of the alias.
 * @param active true to activate, false to deactivate.
 */
public void setActive(boolean active) {
    this.accountActive = active ? "TRUE" : "FALSE";
}

/**
 * Convenience method to get the alias name from the mail.
 * @return the alias name extracted from the mail address.
 */
public String getAliasName() {
    return MailUtils.extractUserFromMail(mail);
}

/**
 * Convenience method to get the domain from the mail address.
 * @return the domain extracted from the mail address.
 */
public String getDomain() {
    return MailUtils.extractDomainFromMail(mail);
}

/**
 * Convenience method to check if the alias is a catch-all alias.
 * A catch-all alias starts with "@".
 * @return true if the alias is a catch-all, false otherwise.
 */
public boolean isCatchAll() {
    return mail != null && mail.startsWith("@");
}

/**
 * Convenience method to check if the alias is a postmaster alias.

```

```

* A postmaster alias typically has "postmaster" in the mail address.
* @return true if the alias is a postmaster alias, false otherwise.
*/
public void addDestination(String destination) {
    if (maildrop == null) {
        maildrop = new ArrayList<>();
    }
    if (!maildrop.contains(destination)) {
        maildrop.add(destination);
        updateLastChange();
    }
}

/**
* Method to remove a destination from the maildrop.
* @param destination the destination to remove.
* @return true if the destination was removed, false otherwise.
*/
public boolean removeDestination(String destination) {
    if (maildrop != null) {
        boolean removed = maildrop.remove(destination);
        if (removed) {
            updateLastChange();
        }
        return removed;
    }
    return false;
}

/**
* Get a read-only view of the maildrop destinations.
* @return an unmodifiable list of destinations.
*/
public List<String> getDestinations() {
    return maildrop != null ? Collections.unmodifiableList(maildrop) : Collections.emptyList();
}

/**
* Set the maildrop destinations.
* @param destinations the list of destinations to set.
*/
public void setDestinations(List<String> destinations) {
    this.maildrop = new ArrayList<>(destinations);
    updateLastChange();
}

/**
* Update the last change timestamp to the current time.
* This is typically called when the alias is modified.
*/

```

```

public void updateLastChange() {
    this.lastChange = String.valueOf(Instant.now().getEpochSecond());
}

/**
 * Convenience method to get the last change timestamp as a Long.
 * @return the last change timestamp as a Long, or null if it cannot be parsed.
 */
public Long getLastChangeAsLong() {
    try {
        return lastChange != null ? Long.valueOf(lastChange) : null;
    } catch (NumberFormatException e) {
        return null;
    }
}

/**
 * Convenience method to set the last change timestamp from a Long.
 * @param lastChange the last change timestamp as a Long.
 */
public void setLastChangeAsLong(Long lastChange) {
    this.lastChange = lastChange != null ? String.valueOf(lastChange) : null;
}

// Getters and Setters - return/accept Strings for LDAP compatibility
public Name getId() { return id; }
public void setId(Name id) { this.id = id; }

public String getMail() { return mail; }
public void setMail(String mail) { this.mail = mail; }

public List<String> getMaildrop() { return maildrop; }
public void setMaildrop(List<String> maildrop) { this.maildrop = maildrop; }

public String getAccountActive() { return accountActive; }
public void setAccountActive(String accountActive) { this.accountActive = accountActive; }

public String getLastChange() { return lastChange; }
public void setLastChange(String lastChange) { this.lastChange = lastChange; }

public String getMailsource() { return mailsource; }
public void setMailsource(String mailsource) { this.mailsource = mailsource; }

public String getCommonName() { return commonName; }
public void setCommonName(String commonName) { this.commonName = commonName; }

public String getDescription() { return description; }
public void setDescription(String description) { this.description = description; }

public String getUserPassword() { return userPassword; }

```

```

public void setUserPassword(String userPassword) { this.userPassword = userPassword; }

@Override
public String toString() {
    return "JammMailAlias{" +
        "mail='" + mail + '\'' +
        ", destinations=" + (maildrop != null ? maildrop.size() : 0) +
        ", accountActive='" + accountActive + '\'' +
        ", commonName='" + commonName + '\'' +
        '}';
}
}

```

15.2.4 JammPostmaster.java

Die Postmaster-Entität erweitert die Standard-Alias-Funktionalität um rollenbasierte Zugriffskontrolle und kombiniert die JammMailAlias- und JammPostmaster-Objektklassen. Die Klasse verwaltet Role-Occupant-Zuweisungen für administrative Berechtigungen und stellt sicher, dass mindestens ein Role-Occupant pro Postmaster-Instanz existiert, um die administrative Kontinuität zu gewährleisten.

```

/**
 * LDAP entity representing a Jamm postmaster.
 * Maps to the JammPostmaster auxiliary objectClass combined with JammMailAlias.
 * This represents a special alias that has postmaster privileges.
 */
@Entry(objectClasses = {"top", "JammMailAlias", "JammPostmaster"})
public final class JammPostmaster {

    @Id
    private Name id;

    // Inherited from JammMailAlias
    @Attribute(name = "mail")
    private String mail;

    @Attribute(name = "maildrop")
    private List<String> maildrop;

    // Boolean
    @Attribute(name = "accountActive")
    private String accountActive;

    // Long
    @Attribute(name = "lastChange")
    private String lastChange;

    @Attribute(name = "mailsource")

```

```

private String mailsource;

@Attribute(name = "cn")
private String commonName;

@Attribute(name = "description")
private String description;

@Attribute(name = "userPassword")
private String userPassword;

// JammPostmaster specific attribute - MUST have at least one value
@Attribute(name = "roleOccupant")
private List<String> roleOccupant;

public JammPostmaster() {
    this.accountActive = "TRUE";
    this.lastChange = String.valueOf(Instant.now().getEpochSecond());
    this.maildrop = new ArrayList<>();
    this.roleOccupant = new ArrayList<>();
    this.commonName = "postmaster";
}

public JammPostmaster(String domain) {
    this();
    this.mail = "postmaster@" + domain;
    this.maildrop.add("postmaster"); // Default destination

    // Add a default role occupant - use the domain admin DN or a default DN
    // Since roleOccupant is MUST attribute, we need at least one value
    String defaultRoleOccupant = "cn=admin,o=hosting,dc=example,dc=com"; // Default admin DN
    this.roleOccupant.add(defaultRoleOccupant);

    // Auto-generate DN for postmaster
    this.id = LdapUtils.postmasterDN(domain).build();
}

public JammPostmaster(String domain, List<String> destinations) {
    this(domain);
    this.maildrop = new ArrayList<>(destinations);
}

/**
 * Convenience method to check if the account is active.
 * @return true if the account is active, false otherwise.
 */
public boolean isActive() {
    return "TRUE".equals(accountActive);
}

```

```

/**
 * Convenience method to set the account active status.
 * @param active true to activate the account, false to deactivate.
 */
public void setActive(boolean active) {
    this.accountActive = active ? "TRUE" : "FALSE";
}

/**
 * Convenience method to get the domain from the postmaster email.
 * @return the domain part of the postmaster email.
 */
public String getDomain() {
    return MailUtils.extractDomainFromMail(mail);
}

/**
 * Convenience method to add a role occupant (user DN).
 * This method ensures that the roleOccupant list is initialized and
 * does not contain duplicates.
 * @param userDn the distinguished name (DN) of the user to add as a role occupant.
 */
public void addRoleOccupant(String userDn) {
    if (roleOccupant == null) {
        roleOccupant = new ArrayList<>();
    }
    if (!roleOccupant.contains(userDn)) {
        roleOccupant.add(userDn);
        updateLastChange();
    }
}

/**
 * Method to remove a role occupant (user DN).
 * This method ensures that at least one role occupant remains.
 * @param userDn the distinguished name (DN) of the user to remove as a role occupant.
 * @return true if the user was removed, false if it was the last occupant and could not be
removed.
 */
public boolean removeRoleOccupant(String userDn) {
    if (roleOccupant != null && roleOccupant.size() > 1) { // Keep at least one role occupant
        boolean removed = roleOccupant.remove(userDn);
        if (removed) {
            updateLastChange();
        }
        return removed;
    }
    return false; // Cannot remove the last role occupant
}

```

```

/**
 * Method to check if a user DN is a role occupant.
 * @param userDn the distinguished name (DN) of the user to check.
 * @return true if the user DN is a role occupant, false otherwise.
 */
public boolean isRoleOccupant(String userDn) {
    return roleOccupant != null && roleOccupant.contains(userDn);
}

/**
 * Method to get a read-only view of the role occupants.
 * This returns an unmodifiable list of role occupant DNs.
 * @return an unmodifiable list of role occupant DNs.
 */
public List<String> getRoleOccupants() {
    return roleOccupant != null ? Collections.unmodifiableList(roleOccupant) :
Collections.emptyList();
}

/**
 * Method to add a destination to the maildrop.
 * This method ensures that the maildrop is initialized and does not contain duplicates.
 * @param destination the destination email address to add.
 */
public void addDestination(String destination) {
    if (maildrop == null) {
        maildrop = new ArrayList<>();
    }
    if (!maildrop.contains(destination)) {
        maildrop.add(destination);
        updateLastChange();
    }
}

/**
 * Method to remove a destination from the maildrop.
 * @param destination the destination email address to remove.
 * @return true if the destination was removed, false otherwise.
 */
public boolean removeDestination(String destination) {
    if (maildrop != null) {
        boolean removed = maildrop.remove(destination);
        if (removed) {
            updateLastChange();
        }
        return removed;
    }
    return false;
}

```



```

/**
 * Method to get a read-only view of the maildrop destinations.
 * @return an unmodifiable list of maildrop destinations.
 */
public List<String> getDestinations() {
    return maildrop != null ? Collections.unmodifiableList(maildrop) : Collections.emptyList();
}

/**
 * Method to set the maildrop destinations.
 * This replaces the current maildrop with a new list of destinations.
 * @param destinations the new list of destination email addresses.
 */
public void setDestinations(List<String> destinations) {
    this.maildrop = new ArrayList<>(destinations);
    updateLastChange();
}

/**
 * Method to update the last change timestamp to the current time.
 * This sets the lastChange attribute to the current epoch second.
 */
public void updateLastChange() {
    this.lastChange = String.valueOf(Instant.now().getEpochSecond());
}

/**
 * Convenience method to get the last change timestamp as a Long.
 * This parses the lastChange string to a Long, handling potential format issues.
 * @return the last change timestamp as a Long, or null if it cannot be parsed.
 */
public Long getLastChangeAsLong() {
    try {
        return lastChange != null ? Long.valueOf(lastChange) : null;
    } catch (NumberFormatException e) {
        return null;
    }
}

/**
 * Convenience method to set the last change timestamp from a Long.
 * This converts the Long to a String and sets it as the lastChange attribute.
 * @param lastChange the last change timestamp as a Long.
 */
public void setLastChangeAsLong(Long lastChange) {
    this.lastChange = lastChange != null ? String.valueOf(lastChange) : null;
}

// Getters and Setters - return/accept Strings for LDAP compatibility
public Name getId() { return id; }

```

```

public void setId(Name id) { this.id = id; }

public String getMail() { return mail; }
public void setMail(String mail) { this.mail = mail; }

public List<String> getMaildrop() { return maildrop; }
public void setMaildrop(List<String> maildrop) { this.maildrop = maildrop; }

public String getAccountActive() { return accountActive; }
public void setAccountActive(String accountActive) { this.accountActive = accountActive; }

public String getLastChange() { return lastChange; }
public void setLastChange(String lastChange) { this.lastChange = lastChange; }

public String getMailsource() { return mailsource; }
public void setMailsource(String mailsource) { this.mailsource = mailsource; }

public String getCommonName() { return commonName; }
public void setCommonName(String commonName) { this.commonName = commonName; }

public String getDescription() { return description; }
public void setDescription(String description) { this.description = description; }

public String getUserPassword() { return userPassword; }
public void setUserPassword(String userPassword) { this.userPassword = userPassword; }

public List<String> getRoleOccupant() { return roleOccupant; }
public void setRoleOccupant(List<String> roleOccupant) {
    // Ensure at least one role occupant
    if (roleOccupant == null || roleOccupant.isEmpty()) {
        this.roleOccupant = new ArrayList<>();
        this.roleOccupant.add("cn=admin,o=hosting,dc=example,dc=com");
    } else {
        this.roleOccupant = roleOccupant;
    }
}

@Override
public String toString() {
    return "JammPostmaster{" +
        "mail=\"" + mail + "\"" +
        ", accountActive=\"" + accountActive + "\"" +
        ", roleOccupants=\"" + (roleOccupant != null ? roleOccupant.size() : 0) +
        ", destinations=\"" + (maildrop != null ? maildrop.size() : 0) +
        "\"";
}
}

```

16.1 Testergebnisse

[illegible]

```

2025-06-16T16:26:14.857+02:00 INFO 980 --- [Pool-1-worker-1] t.c.s.AnnotationConfigContextLoaderUtils : Could not detect default
configuration classes for test class [ch.aarboard.vamm.data.repositories.JammMailAliasRepositoryTest]:
JammMailAliasRepositoryTest does not declare any static, non-private, non-final, nested classes annotated with @Configuration.
2025-06-16T16:26:14.857+02:00 INFO 980 --- [Pool-1-worker-1] .b.t.c.SpringBootTestContextBootstrapper : Found
@SpringBootConfiguration ch.aarboard.vamm.Application for test class
ch.aarboard.vamm.data.repositories.JammMailAliasRepositoryTest
2025-06-16T16:26:14.941+02:00 ERROR 980 --- [Pool-1-worker-1] c.a.v.d.r.JammVirtualDomainRepository : Error finding domain
example.com: [LDAP: error code 32 - No Such Object]
2025-06-16T16:26:14.959+02:00 ERROR 980 --- [Pool-1-worker-1] c.a.v.d.r.JammVirtualDomainRepository : Error finding domain
example.com: [LDAP: error code 32 - No Such Object]
2025-06-16T16:26:15.007+02:00 ERROR 980 --- [Pool-1-worker-1] c.a.v.d.r.JammMailAliasRepository : Error finding alias by email
test.alias@example.com: [LDAP: error code 32 - No Such Object]
2025-06-16T16:26:15.376+02:00 ERROR 980 --- [Pool-1-worker-1] c.a.v.d.r.JammMailAliasRepository : Error finding alias by email
nonexistent@example.com: [LDAP: error code 32 - No Such Object]
2025-06-16T16:26:15.463+02:00 ERROR 980 --- [Pool-1-worker-1] c.a.v.d.r.JammMailAliasRepository : Error finding alias by email
test.alias@example.com: [LDAP: error code 32 - No Such Object]
2025-06-16T16:26:15.539+02:00 INFO 980 --- [Pool-1-worker-1] c.a.v.d.r.JammVirtualDomainRepository : Successfully deleted
domain: jvd=example.com,o=hosting
[INFO] Tests run: 5, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.698 s -- in
ch.aarboard.vamm.data.repositories.JammMailAliasRepositoryTest
[INFO] Running ch.aarboard.vamm.data.repositories.JammPostmasterRepositoryTest
2025-06-16T16:26:15.570+02:00 INFO 980 --- [Pool-1-worker-1] t.c.s.AnnotationConfigContextLoaderUtils : Could not detect default
configuration classes for test class [ch.aarboard.vamm.data.repositories.JammPostmasterRepositoryTest]:
JammPostmasterRepositoryTest does not declare any static, non-private, non-final, nested classes annotated with @Configuration.
2025-06-16T16:26:15.570+02:00 INFO 980 --- [Pool-1-worker-1] .b.t.c.SpringBootTestContextBootstrapper : Found
@SpringBootConfiguration ch.aarboard.vamm.Application for test class
ch.aarboard.vamm.data.repositories.JammPostmasterRepositoryTest
2025-06-16T16:26:15.640+02:00 ERROR 980 --- [Pool-1-worker-1] c.a.v.d.r.JammVirtualDomainRepository : Error finding domain
example.com: [LDAP: error code 32 - No Such Object]
2025-06-16T16:26:15.656+02:00 ERROR 980 --- [Pool-1-worker-1] c.a.v.d.r.JammVirtualDomainRepository : Error finding domain
example.com: [LDAP: error code 32 - No Such Object]
2025-06-16T16:26:15.702+02:00 ERROR 980 --- [Pool-1-worker-1] c.a.v.d.r.JammPostmasterRepository : Error finding accounts by
domain example.com: [LDAP: error code 32 - No Such Object]
2025-06-16T16:26:16.146+02:00 ERROR 980 --- [Pool-1-worker-1] c.a.v.d.r.JammPostmasterRepository : Error finding accounts by
domain example.com: [LDAP: error code 32 - No Such Object]
2025-06-16T16:26:16.192+02:00 INFO 980 --- [Pool-1-worker-1] c.a.v.d.r.JammVirtualDomainRepository : Successfully deleted
domain: jvd=example.com,o=hosting
[INFO] Tests run: 9, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.638 s -- in
ch.aarboard.vamm.data.repositories.JammPostmasterRepositoryTest
[INFO] Running ch.aarboard.vamm.data.repositories.JammVirtualDomainRepositoryTest
2025-06-16T16:26:16.208+02:00 INFO 980 --- [Pool-1-worker-1] t.c.s.AnnotationConfigContextLoaderUtils : Could not detect default
configuration classes for test class [ch.aarboard.vamm.data.repositories.JammVirtualDomainRepositoryTest]:
JammVirtualDomainRepositoryTest does not declare any static, non-private, non-final, nested classes annotated with
@Configuration.
2025-06-16T16:26:16.224+02:00 INFO 980 --- [Pool-1-worker-1] .b.t.c.SpringBootTestContextBootstrapper : Found
@SpringBootConfiguration ch.aarboard.vamm.Application for test class
ch.aarboard.vamm.data.repositories.JammVirtualDomainRepositoryTest
2025-06-16T16:26:16.255+02:00 ERROR 980 --- [Pool-1-worker-1] c.a.v.d.r.JammVirtualDomainRepository : Error finding domain
test-domain.com: [LDAP: error code 32 - No Such Object]
2025-06-16T16:26:16.334+02:00 ERROR 980 --- [Pool-1-worker-1] c.a.v.d.r.JammVirtualDomainRepository : Error finding domain
nonexistent-domain.com: [LDAP: error code 32 - No Such Object]
2025-06-16T16:26:16.349+02:00 ERROR 980 --- [Pool-1-worker-1] c.a.v.d.r.JammVirtualDomainRepository : Error finding domain
second-domain.com: [LDAP: error code 32 - No Such Object]
2025-06-16T16:26:16.714+02:00 INFO 980 --- [Pool-1-worker-1] c.a.v.d.r.JammVirtualDomainRepository : Successfully deleted
domain: jvd=test-domain.com,o=hosting
2025-06-16T16:26:16.730+02:00 ERROR 980 --- [Pool-1-worker-1] c.a.v.d.r.JammVirtualDomainRepository : Error finding domain
test-domain.com: [LDAP: error code 32 - No Such Object]
2025-06-16T16:26:16.774+02:00 INFO 980 --- [Pool-1-worker-1] c.a.v.d.r.JammVirtualDomainRepository : Successfully deleted
domain: jvd=second-domain.com,o=hosting
2025-06-16T16:26:16.790+02:00 ERROR 980 --- [Pool-1-worker-1] c.a.v.d.r.JammVirtualDomainRepository : Error finding domain
second-domain.com: [LDAP: error code 32 - No Such Object]
[INFO] Tests run: 12, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.582 s -- in
ch.aarboard.vamm.data.repositories.JammVirtualDomainRepositoryTest
[INFO] Running ch.aarboard.vamm.services.JammMailAccountManagementServiceTest
2025-06-16T16:26:16.806+02:00 INFO 980 --- [Pool-1-worker-1] t.c.s.AnnotationConfigContextLoaderUtils : Could not detect default
configuration classes for test class [ch.aarboard.vamm.services.JammMailAccountManagementServiceTest]:

```

```

JammMailAccountManagementServiceTest does not declare any static, non-private, non-final, nested classes annotated with
@Configuration.
2025-06-16T16:26:16.821+02:00 INFO 980 --- [Pool-1-worker-1] .b.t.c.SpringBootTestContextBootstrapper : Found
@SpringBootConfiguration ch.aarboard.vamm.Application for test class
ch.aarboard.vamm.services.JammMailAccountManagementServiceTest
2025-06-16T16:26:16.852+02:00 ERROR 980 --- [Pool-1-worker-1] c.a.v.d.r.JammVirtualDomainRepository : Error finding domain
test-service-domain.com: [LDAP: error code 32 - No Such Object]
2025-06-16T16:26:16.868+02:00 ERROR 980 --- [Pool-1-worker-1] c.a.v.d.r.JammVirtualDomainRepository : Error finding domain
test-service-domain.com: [LDAP: error code 32 - No Such Object]
2025-06-16T16:26:16.893+02:00 ERROR 980 --- [Pool-1-worker-1] c.a.v.d.r.JammPostmasterRepository : Error finding accounts by
domain test-service-domain.com: [LDAP: error code 32 - No Such Object]
2025-06-16T16:26:16.984+02:00 ERROR 980 --- [Pool-1-worker-1] c.a.v.d.r.JammVirtualDomainRepository : Error finding domain
nonexistent-domain.com: [LDAP: error code 32 - No Such Object]
2025-06-16T16:26:17.016+02:00 ERROR 980 --- [Pool-1-worker-1] c.a.v.d.r.JammVirtualDomainRepository : Error finding domain
second-service-domain.com: [LDAP: error code 32 - No Such Object]
2025-06-16T16:26:17.016+02:00 ERROR 980 --- [Pool-1-worker-1] c.a.v.d.r.JammVirtualDomainRepository : Error finding domain
second-service-domain.com: [LDAP: error code 32 - No Such Object]
2025-06-16T16:26:17.072+02:00 ERROR 980 --- [Pool-1-worker-1] c.a.v.d.r.JammPostmasterRepository : Error finding accounts by
domain second-service-domain.com: [LDAP: error code 32 - No Such Object]
2025-06-16T16:26:17.355+02:00 ERROR 980 --- [Pool-1-worker-1] c.a.v.d.r.JammPostmasterRepository : Error finding accounts by
domain test-service-domain.com: [LDAP: error code 32 - No Such Object]
2025-06-16T16:26:17.371+02:00 INFO 980 --- [Pool-1-worker-1] c.a.v.d.r.JammVirtualDomainRepository : Successfully deleted
domain: jvd=test-service-domain.com,o=hosting
2025-06-16T16:26:17.403+02:00 ERROR 980 --- [Pool-1-worker-1] c.a.v.d.r.JammVirtualDomainRepository : Error finding domain
test-service-domain.com: [LDAP: error code 32 - No Such Object]
[INFO] Tests run: 8, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.597 s -- in
ch.aarboard.vamm.services.JammMailAccountManagementServiceTest
[INFO] Running ch.aarboard.vamm.services.JammMailAliasManagementServiceTest
2025-06-16T16:26:17.403+02:00 INFO 980 --- [Pool-1-worker-1] t.c.s.AnnotationConfigContextLoaderUtils : Could not detect default
configuration classes for test class [ch.aarboard.vamm.services.JammMailAliasManagementServiceTest]:
JammMailAliasManagementServiceTest does not declare any static, non-private, non-final, nested classes annotated with
@Configuration.
2025-06-16T16:26:17.403+02:00 INFO 980 --- [Pool-1-worker-1] .b.t.c.SpringBootTestContextBootstrapper : Found
@SpringBootConfiguration ch.aarboard.vamm.Application for test class
ch.aarboard.vamm.services.JammMailAliasManagementServiceTest
2025-06-16T16:26:17.435+02:00 ERROR 980 --- [Pool-1-worker-1] c.a.v.d.r.JammVirtualDomainRepository : Error finding domain
test-alias-domain.com: [LDAP: error code 32 - No Such Object]
2025-06-16T16:26:17.451+02:00 ERROR 980 --- [Pool-1-worker-1] c.a.v.d.r.JammVirtualDomainRepository : Error finding domain
test-alias-domain.com: [LDAP: error code 32 - No Such Object]
2025-06-16T16:26:17.497+02:00 ERROR 980 --- [Pool-1-worker-1] c.a.v.d.r.JammPostmasterRepository : Error finding accounts by
domain test-alias-domain.com: [LDAP: error code 32 - No Such Object]
2025-06-16T16:26:17.528+02:00 ERROR 980 --- [Pool-1-worker-1] c.a.v.d.r.JammMailAliasRepository : Error finding alias by email
test.alias@test-alias-domain.com: [LDAP: error code 32 - No Such Object]
2025-06-16T16:26:17.544+02:00 ERROR 980 --- [Pool-1-worker-1] c.a.v.d.r.JammMailAccountRepository : Error finding account by
email test.alias@test-alias-domain.com: [LDAP: error code 32 - No Such Object]
2025-06-16T16:26:17.576+02:00 ERROR 980 --- [Pool-1-worker-1] c.a.v.d.r.JammMailAliasRepository : Error finding alias by email
test.alias@test-alias-domain.com: [LDAP: error code 32 - No Such Object]
2025-06-16T16:26:17.891+02:00 ERROR 980 --- [Pool-1-worker-1] c.a.v.d.r.JammMailAliasRepository : Error finding alias by email
@test-alias-domain.com: [LDAP: error code 32 - No Such Object]
2025-06-16T16:26:17.906+02:00 ERROR 980 --- [Pool-1-worker-1] c.a.v.d.r.JammMailAliasRepository : Error finding alias by email
@test-alias-domain.com: [LDAP: error code 32 - No Such Object]
2025-06-16T16:26:17.922+02:00 ERROR 980 --- [Pool-1-worker-1] c.a.v.d.r.JammMailAccountRepository : Error finding account by
email @test-alias-domain.com: [LDAP: error code 32 - No Such Object]
2025-06-16T16:26:17.937+02:00 ERROR 980 --- [Pool-1-worker-1] c.a.v.d.r.JammMailAliasRepository : Error finding alias by email
@test-alias-domain.com: [LDAP: error code 32 - No Such Object]
2025-06-16T16:26:18.053+02:00 ERROR 980 --- [Pool-1-worker-1] c.a.v.d.r.JammMailAliasRepository : Error finding alias by email
test.alias@test-alias-domain.com: [LDAP: error code 32 - No Such Object]
2025-06-16T16:26:18.178+02:00 ERROR 980 --- [Pool-1-worker-1] c.a.v.d.r.JammPostmasterRepository : Error finding accounts by
domain test-alias-domain.com: [LDAP: error code 32 - No Such Object]
2025-06-16T16:26:18.209+02:00 INFO 980 --- [Pool-1-worker-1] c.a.v.d.r.JammVirtualDomainRepository : Successfully deleted
domain: jvd=test-alias-domain.com,o=hosting
2025-06-16T16:26:18.225+02:00 ERROR 980 --- [Pool-1-worker-1] c.a.v.d.r.JammVirtualDomainRepository : Error finding domain
test-alias-domain.com: [LDAP: error code 32 - No Such Object]
[INFO] Tests run: 9, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.822 s -- in
ch.aarboard.vamm.services.JammMailAliasManagementServiceTest
[INFO] Running ch.aarboard.vamm.services.JammVirtualDomainManagementServiceTest

```

```

2025-06-16T16:26:18.240+02:00 INFO 980 --- [Pool-1-worker-1] t.c.s.AnnotationConfigContextLoaderUtils : Could not detect default
configuration classes for test class [ch.aarboard.vamm.services.JammVirtualDomainManagementServiceTest]:
JammVirtualDomainManagementServiceTest does not declare any static, non-private, non-final, nested classes annotated with
@Configuration.
2025-06-16T16:26:18.240+02:00 INFO 980 --- [Pool-1-worker-1] .b.t.c.SpringBootTestContextBootstrapper : Found
@SpringBootConfiguration ch.aarboard.vamm.Application for test class
ch.aarboard.vamm.services.JammVirtualDomainManagementServiceTest
2025-06-16T16:26:18.256+02:00 ERROR 980 --- [Pool-1-worker-1] c.a.v.d.r.JammVirtualDomainRepository : Error finding domain
test-vdomain-service.com: [LDAP: error code 32 - No Such Object]
2025-06-16T16:26:18.272+02:00 ERROR 980 --- [Pool-1-worker-1] c.a.v.d.r.JammVirtualDomainRepository : Error finding domain
test-vdomain-service.com: [LDAP: error code 32 - No Such Object]
2025-06-16T16:26:18.303+02:00 ERROR 980 --- [Pool-1-worker-1] c.a.v.d.r.JammPostmasterRepository : Error finding accounts by
domain test-vdomain-service.com: [LDAP: error code 32 - No Such Object]
2025-06-16T16:26:18.350+02:00 ERROR 980 --- [Pool-1-worker-1] c.a.v.d.r.JammVirtualDomainRepository : Error finding domain
nonexistent-domain.com: [LDAP: error code 32 - No Such Object]
2025-06-16T16:26:18.427+02:00 ERROR 980 --- [Pool-1-worker-1] c.a.v.d.r.JammVirtualDomainRepository : Error finding domain
second-vdomain-service.com: [LDAP: error code 32 - No Such Object]
2025-06-16T16:26:18.442+02:00 ERROR 980 --- [Pool-1-worker-1] c.a.v.d.r.JammVirtualDomainRepository : Error finding domain
second-vdomain-service.com: [LDAP: error code 32 - No Such Object]
2025-06-16T16:26:18.499+02:00 ERROR 980 --- [Pool-1-worker-1] c.a.v.d.r.JammPostmasterRepository : Error finding accounts by
domain second-vdomain-service.com: [LDAP: error code 32 - No Such Object]
2025-06-16T16:26:19.150+02:00 ERROR 980 --- [Pool-1-worker-1] c.a.v.d.r.JammPostmasterRepository : Error finding accounts by
domain second-vdomain-service.com: [LDAP: error code 32 - No Such Object]
2025-06-16T16:26:19.181+02:00 INFO 980 --- [Pool-1-worker-1] c.a.v.d.r.JammVirtualDomainRepository : Successfully deleted
domain: jvd=second-vdomain-service.com,o=hosting
2025-06-16T16:26:19.196+02:00 ERROR 980 --- [Pool-1-worker-1] c.a.v.d.r.JammVirtualDomainRepository : Error finding domain
second-vdomain-service.com: [LDAP: error code 32 - No Such Object]
2025-06-16T16:26:19.259+02:00 ERROR 980 --- [Pool-1-worker-1] c.a.v.d.r.JammPostmasterRepository : Error finding accounts by
domain test-vdomain-service.com: [LDAP: error code 32 - No Such Object]
2025-06-16T16:26:19.290+02:00 INFO 980 --- [Pool-1-worker-1] c.a.v.d.r.JammVirtualDomainRepository : Successfully deleted
domain: jvd=test-vdomain-service.com,o=hosting
2025-06-16T16:26:19.306+02:00 ERROR 980 --- [Pool-1-worker-1] c.a.v.d.r.JammVirtualDomainRepository : Error finding domain
test-vdomain-service.com: [LDAP: error code 32 - No Such Object]
[INFO] Tests run: 16, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 1.066 s -- in
ch.aarboard.vamm.services.JammVirtualDomainManagementServiceTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 112, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 42.479 s
[INFO] Finished at: 2025-06-16T16:26:19+02:00
[INFO] -----

```