HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

Comparis Smtp Mailer

Bachelorarbeit Frühjahrssemester 2009

HSR Hochschule für Technik Rapperswil

Informatik – http://i.hsr.ch

Studenten:

Danny Meier
dmeier@hsr.ch

Micha Boller
mboller@hsr.ch

Betreuer:

Prof. Dr. Markus Stolze, HSR
mstolze@hsr.ch

Industriepartner:

Benedikt Unold, comparis.ch AG,
benedikt.unold@comparis.ch

Rapperswil, 10. Juni 2009

Versionsgeschichte

Version	Datum	Änderung	Autor
0.1	**03.06.2009**	Dokument erstellt	mb
0.2	**09.06.2009**	Dokument fertiggestellt	dm, mb
1.0	**10.06.2009**	Abgabeversion	dm, mb

Dokumentreview

Version	Datum	Bemerkung	Autor
1.0	**10.06.2009**	Abgabeversion	dm, mb

Impressum

Kontaktdaten der Autoren:

Danny Meier
Rietgrabenstrasse 10
CH – 8152 Opfikon

dmeier@danflash.com

Micha Boller
Austrasse 26
CH – 8625 Gossau ZH

m.boller@sunrise.ch

Inhaltsverzeichnis

1 Abstract

Die Software *Smtp Mailer* wurde für die comparis.ch AG, dem führenden Internet-Vergleichsdienst in der Schweiz, entwickelt. Über den *Smtp Mailer* wird sämtlicher E-Mail Verkehr der compairs.ch AG verwaltet und verschickt. Funktional muss die Applikation das zeit- und prioritätsgesteuerte Versenden von Nachrichten erlauben, welche Signaturen, Attachments und Multipart-Bodys enthalten können. Das Versandvolumen kann in Spitzenzeiten (z.B. beim Newsletterversand) bis zu 300‘000 Nachrichten pro Stunde betragen. Diese Versandleistung muss das *Smtp Mailer* System erreichen können.

Mittels eines konfigurierbaren Bereinigungsprozesses werden von versendeten Nachrichten nach Ablauf der Archivierungszeit speicherintensive Bestandteile wie Nachrichteninhalte und Anhänge in der Datenbank gelöscht.

Fehlerhafte oder temporär nicht zustellbare Nachrichten erzeugen einen Non Delivery Report (NDR), welcher vom *Smtp Mailer* System automatisch ausgewertet wird. Aus den so ermittelten Daten lassen sich Rückschlüsse auf die Qualität der E-Mail Adressen ziehen. Das Ziel ist es, E-Mail Adressen von schlechter Qualität beim E-Mail Versand nicht mehr berücksichtigen zu müssen.

Sämtliche standortunabhängigen Konfigurationseinstellungen können über eine moderne und intuitiv zu bedienende Silverlight Webapplikation vorgenommen werden. Die Einstellungen werden zur Laufzeit übernommen und erfordern kein Neustart der Anwendungen. Standortabhängige Konfigurationseinstellungen werden in XML Konfigurationsdateien verwaltet. Je nach Situation werden die Einstellungen in den Konfigurationsdateien während dem Betrieb der Applikationen angewendet.

Sämtliche Funktionalitäten des *Smtp Mailers* sind über eine API verfügbar, welche in Form von zwei Programmbibliotheken bereitgestellt werden. Die Versand API erlaubt das Versenden und Verfolgen von Nachrichten und bildet die Schnittstelle für sämtliche Applikationen der comparis.ch AG, welche in irgendeiner Form Nachrichten versenden müssen. Die zweite API stellt Funktionalitäten bereit, welche für Auswertungen und die Wartung des *Smtp Mailer* Systems notwendig sind.

2 Management Summary

2.1 Ausgangslage

Auf der Suche nach einem Thema für die Bachelorarbeit sind wir dank Danny Meier auf das *Smtp Mailer* Projekt bei der Firma comparis.ch AG aufmerksam geworden. Danny Meier arbeitete im Sommer 2008 einige Wochen bei comparis.ch AG. Wir waren auf der Suche nach einem konkreten Arbeitsvorschlag für eine Software, welche später auch tatsächlich eingesetzt und genutzt werden soll. Weil das Projekt *Smtp Mailer* unseren Vorstellungen bezüglich eingesetzten Technologien, Umfang und Themengebiet entsprach, entschieden wir uns für den Themenvorschlag der comparis.ch AG.

2.2 Ziele

Das Ziel der Arbeit ist es, die bestehende Mail Lösung der comparis.ch AG, der so genannte NLNMailer, abzulösen. Mit dem *Smtp Mailer* soll eine Lösung implementiert werden, welche alle fehlenden Funktionalitäten des NLNMailers anbietet und zusätzlich über ein Webinterface vollständig konfigurierbar und wartbar ist. Über das zu implementierende System wird sämtlicher E-Mail Verkehr der compairs.ch AG verwaltet und verschickt. Dabei muss ein priorisierbarer, zeitgesteuerter Versand von Plaintext- und Multipart E-Mails möglich sein. Es müssen Attachments und der Versand von signierten Nachrichten unterstützt werden. Dabei ist auf eine effiziente Auslegung der Software zu achten, ist der geschäftlich bedingte E-Mail Verkehr (z.B. Newsletterversand an alle Kunden) der comparis.ch AG doch sehr massiv. Zusätzlich soll mit dem *Smtp Mailer* System ein automatisches Adressqualitätsmanagement eingeführt werden. Automatisch generierte Fehlermeldungen, welche beim Versand einer unzustellbaren E-Mail Nachricht zurückerhalten werden, sollen ausgewertet werden. Aufgrund der so ermittelten Daten wird für eine E-Mail Adresse festgehalten, wie zuverlässig sie funktioniert. Mit der automatischen Behandlung dieser Fehlermeldungen soll erreicht werden, dass einmal erfasste E-Mail Adressen welche mittlerweile nicht mehr funktionieren, automatisch herausgefiltert werden können resp. dass an solche E-Mail Adressen keine unnötigen E-Mails mehr geschickt werden müssen. Desweiteren soll ein Aufräumprozess implementiert werden, welcher regelbasiert die Anhänge und Nachrichteninhalte von versendeten E-Mails nach einer gewissen Archivierungsdauer aus der Datenbank löscht. So kann das Datenvolumen der Datenbank jeweils verringert werden. Für die Überwachung und Konfiguration des *Smtp Mailers* muss ein Webinterface implementiert werden. Die Benutzeroberfläche muss dabei den Arbeitsweisen und Eigenschaften der zukünftigen Benutzer entsprechen. Das Webinterface erlaubt es, diverse Einstellungen am *Smtp Mailer* System zur Laufzeit vorzunehmen und Real-Time Auswertungen über den E-Mail Verkehr, welcher über das System läuft, anzeigen zu lassen. Eine weitere Schnittstelle, welche das *Smtp Mailer* System anbieten muss, ist eine Programmierschnittstelle. Diese Programmierschnittstelle muss von internen comparis.ch AG Applikationen genutzt werden können, um E-Mail Nachrichten verschicken und den Versandstatus von E-Mail Nachrichten über das *Smtp Mailer* System abfragen zu können. Zur Aufgabe gehört die Konzeption, das Design, die Umsetzung sowie der Funktions- und Performancetest der Software.

2.3 Vorgehen

Das Projekt wurde nach dem Vorgehensmodell Rational Unified Process (kurz RUP) durchgeführt. Dabei handelt es sich um eine iterative Softwareentwicklungsvorgehensweise. Der RUP Prozess sieht die vier Phasen Inception, Elaboration, Construction und Transition im Projektablauf vor. Jede der Phasen besitzt eine oder mehrere Iterationen.

In der Inception Phase verschafften wir uns einen Überblick über das zu lösende Problem und wie sich die Software in die Geschäftswelt bei comparis.ch AG eingliedert. Es wurden also die Systemgrenzen ermittelt. Schliesslich wurden erste Kernrisiken evaluiert und dokumentiert.

In der Elaboration Phase wurde ein Projektplan erstellt. Im Projektplan wurde die Projektlaufzeit in die RUP Phasen und Iterationen eingeteilt und dokumentiert, was in den einzelnen Iterationen erreicht werden soll. Zudem wurde das Risikomanagement verfeinert und festgelegt, welche Risiken bis wann minimiert sein müssen. Eine Meilensteinplanung definierte Ziele und die Termine für deren Erreichung. Des Weiteren begannen wir in der Elaboration Phase mit der Anforderungsspezifikation. Die Anforderungsspezifikation spezifiziert detailliert die funktionalen und nicht funktionalen Anforderungen. Zusätzlich wurden die Use Cases und Aktoren des zu implementierenden Systems definiert. Use Cases beschreiben, welche Personen wie mit dem System interagieren und welche Ziele sie mit dem System erreichen wollen. Mit einer Domainanalyse erarbeiteten wir uns ein tieferes Verständnis für die verschiedenen Konzepte in den Geschäftsprozessen der comparis.ch AG. Erste Kernrisiken wurden bereits in der Elaboration Phase abgedeckt. So wurden Prototypen implementiert, welche die Einhaltung der Performanceanforderungen beweisen. Es wurde ein Architektur Prototyp entwickelt, der uns beispielhaft aufzeigte, wie schlussendlich das zu implementierende System softwaremässig aufgebaut sein muss.

Die Construction Phase wurde in drei Iterationen aufgeteilt. So wurde das Versandmodul, das Fehlerbehandlungsmodul von nicht zustellbaren Nachrichten sowie die Webapplikation in dieser Reihenfolge in den drei Iterationen implementiert. Jede Iteration bestand daraus, die dokumentierten Anforderungen zu studieren, daraus ein Design abzuleiten, die Software gemäss dem Design zu implementieren und schlussendlich zu testen.

Die Transition Phase war dazu da, die System- und Integrationstests über das ganze *Smtp Mailer* System abzuschliessen und dem Code mittels Reviews den letzen Schliff zu geben. Schlussendlich wurde die Dokumentation abgeschlossen, gedruckt und gebunden.

2.4 Beteiligte Personen

Bei den Meilensteinsitzungen waren sowohl der Betreuer (Prof. Dr. Markus Stolze) als auch der Industriepartner (comapris.ch, vertreten durch Benedikt Unold) anwesend. Ein Meilenstein ist ein Zeitpunkt, an welchem ein Teil der Software oder ein Dokument fertiggestellt war. So konnte der Industriepartner sich vom Fortschritt der Arbeit überzeugen lassen und bei Entscheidungen Einfluss nehmen. Wöchentlich fanden mit dem Betreuer Projektfortschrittsitzungen statt. Dabei wurden offene Fragen oder Probleme diskutiert und der Stand der Arbeit besprochen.

2.5 Ergebnisse

Aus der Bachelorarbeit resultiert ein performancemässig als auch funktional getestetes System, welches das bestehende System bei comparis.ch AG nach einer Test- und Einführungsphase mit realen Daten und Datenvolumen ablösen soll. Sämtliche Anforderungen, welche gemäss der Anforderungsspezifikation festgehalten wurden, konnten umgesetzt werden. Mit dem *Smtp Mailer* erhält die Firma comparis.ch AG ein System, das sämtliche fehlende Funktionalitäten der bestehenden Lösung abdeckt und vollständig über ein Webinterface konfigurierbar ist (siehe Abbildung 1 - Detailanzeige einer Nachricht über die Webapplikation).

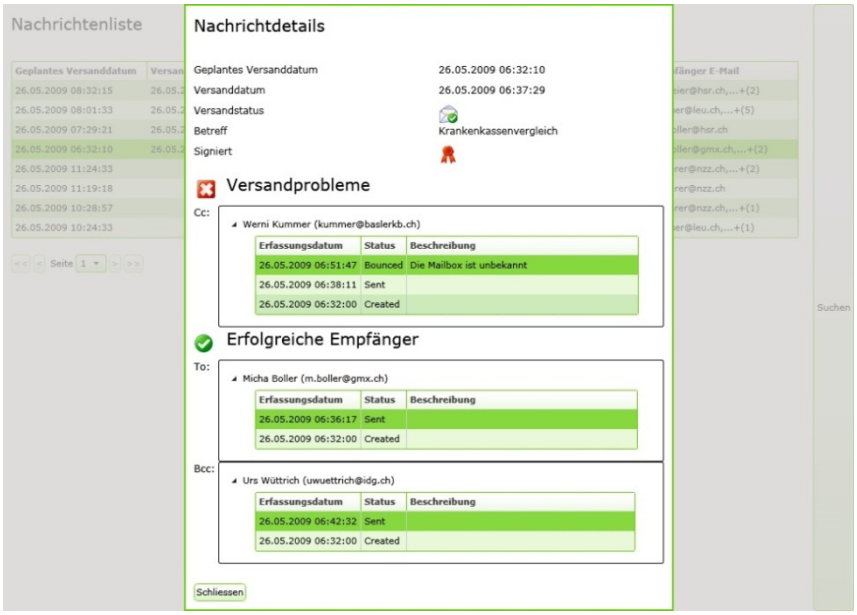


Abbildung 1 - Detailanzeige einer Nachricht über die Webapplikation

Für das Versenden von Nachrichten als auch für die Auswertung der Fehlermeldungen konnte eine Drittherstellerkomponente in Form einer Programmbibliothek verwendet werden. Die Evaluierung von solchen Komponenten war ebenfalls Teil der Bachelorarbeit.

2.6 Abweichungen

In die Webapplikation hätte bezüglich optischer Aufmachung noch beliebig Zeit investiert werden können. Jedoch resultiert in Anbetracht der investierten Zeit eine ansehnliche und sehr intuitiv zu bedienende Webapplikation, welche auf die Bedürfnissen und Arbeitsweisen der Benutzer zugeschnitten ist. Zusätzlich kann mit tieferem fachlichem Wissen und Verständnis bezüglich Performance das *Smtp Mailer* System noch optimiert werden.

2.7 Kosten

Sämtliche Meilensteintermine konnten eingehalten werden. Der Totalaufwand für die Entwicklung des *Smtp Mailers* beläuft sich auf 1106.25 Stunden. In der Abbildung 2 - Zeitauswertung RUP Iterationen ist der Arbeitsaufwand verteilt auf die RUP Iterationen ersichtlich.

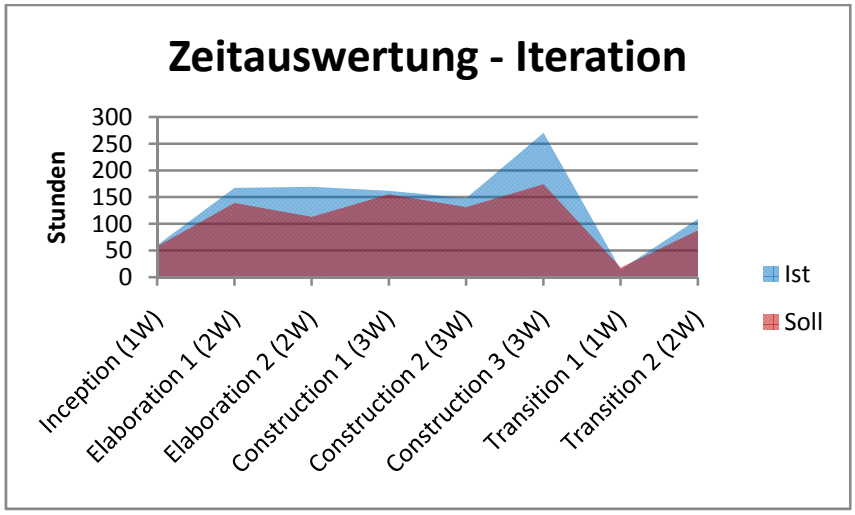


Abbildung 2 - Zeitauswertung RUP Iterationen

2.8 Lernpunkte

Die Bachelorarbeit ermöglichte uns, weitere Erfahrungen mit der iterativen Softwareentwicklungsmethode RUP zu sammeln. Zudem konnten wir unser Wissen und unsere Fähigkeiten in der Teamarbeit vertiefen. Die Erfahrungen welche wir mit der Firma comparis.ch AG sammeln konnten, sehen wir als sehr wertvoll. So spielte die Firma comparis.ch AG die Rolle des Kunden. Wir mussten lernen, über Dinge mit dem Kunden zu verhandeln, unsere Meinung einzubringen, Präsentationen und Demonstrationen kompetent und überzeugend zu halten und Termine einzuhalten. Wir haben uns beim Arbeiten unter Druck oder in Stresssituationen verbessern können.

Auch fachlich haben wir dazugelernt. So konnten wir unser Wissen in der .NET Entwicklungsumgebung vertiefen. Wir lernten die für uns neue Technologie Silverlight kennen. Zudem konnten wir unser Datenbankwissen am MS SQL Server weiter ausbauen. Einige Spezialitäten wie der Distributed Transaction Coordinator (DTC) für das Transaktionsmanagement oder das Tüfteln an Workarounds bei fehlenden Funktionalitäten von Silverlight blieben uns auch nicht vorenthalten.

2.9 Ausblick

Die *Smtp Mailer* Software muss als nächster Schritt bei der comparis.ch AG in einer realen Testumgebung mit realen Daten und realem Datenvolumen getestet werden. Dabei sind sicherlich noch gewisse Optimierungen in der Datenbankebene der Software möglich. Diese Schritte werden vom Team der comparis.ch AG durchgeführt. Hier hat uns etwas die Erfahrung in effizienter Datenbankprogrammierung und Datenbankoptimierung gefehlt. Jedoch hält die Software gemäss unseren Tests funktional und performancemässig die Anforderungen vollständig ein.

3 Aufgabenstellung

Abteilung Informatik Frühjahrssemester 2009
Bachelorarbeit für Herrn Danny Meier und Herrn Micha Boller
„Comparis Smtp Mailer“

1. Auftraggeber und Betreuer

Diese Bachelorarbeit findet in Zusammenarbeit mit der Firma Comparis statt.

Die Firma Comparis, vertreten durch Herrn B. Unold (benedikt.unold@comparis.ch) ist Auftraggeber ist Auftraggeber dieser Arbeit.

Prof. M. Stolze betreut diese Arbeit von der Seite der HSR.

2. Ausgangslage und Aufgabenstellung

In der beiliegenden Aufgabenstellung des Auftraggebers [1] findet sich eine detaillierte Beschreibung der Ausgangslage sowie die konkrete Aufgabenstellung, die hiermit unverändert übernommen wird. Als Zusatzinformation liegen der Aufgabenstellung weitere Beilagen bei, siehe [2].

3. Zur Durchführung

Mit den HSR-Betreuern finden in der Regel wöchentliche Besprechungen statt. Zusätzliche Besprechungen sind nach Bedarf durch die Studierenden zu veranlassen.

Alle Besprechungen sind von den Studenten mit einer Traktandenliste vorzubereiten und die Ergebnisse in einem Protokoll zu dokumentieren, das dem Betreuer und dem Auftraggeber (bei Bedarf) per EMail zugestellt wird.

Die Studenten und Betreuer unterzeichen individuelle Vereinbarungen mit Comparis welche die Rechte der Parteien an den erarbeiteten Resultaten regeln.

Für die Durchführung der Arbeit ist ein Projektplan zu erstellen. Dabei ist auf einen kontinuierlichen und sichtbaren Arbeitsfortschritt zu achten. An Meilensteinen gemäss Projektplan sind einzelne Arbeitsresultate in vorläufigen Versionen abzugeben. Über die abgegebenen Arbeitsresultate erhalten die Studierenden ein vorläufiges Feedback. Eine definitive Beurteilung erfolgt auf Grund der am Abgabetermin abgelieferten Dokumentation.

4. Dokumentation

Über diese Arbeit ist eine Dokumentation gemäss den Richtlinien der Abteilung Informatik zu verfassen. Die zu erstellenden Dokumente sind im Projektplan festzuhalten. Alle Dokumente sind nachzuführen, d.h. sie sollten den Stand der Arbeit bei der Abgabe in konsistenter Form dokumentieren. Die Dokumentation ist vollständig auf CD/DVD in 5 Exemplaren abzugeben. Die Arbeit ist in einem *einzelnen* PDF Dokument auf der CD/DVD abzulegen. Auf Wunsch ist für den Auftraggeber eine gedruckte Version zu erstellen.

5. Termine

Siehe auch Terminplan auf http://www.hsr.ch/Diplom-und-Studienarbeiten.1825.0.html?&L=3

Montag, den 16. Februar 2009 Ausgabe der Aufgabenstellung durch die Betreuer	Beginn der Bachelorarbeit,
3. Juni 2009	Abgabe Kurzbeschreibung an das Abteilungssekretariat mit folgendem Formular: http://www.hsr.ch/Vorlagen.1327.0.html
12. Juni 2009, 12:00	**Abgabe** des Berichtes an die Betreuer bis 12.00 Uhr. Fertigstellung des A0-Posters bis 12.00 Uhr. Abgabe der Posters im Abteilungssekretariat 6.113.
15.06 - 28.08.2009	Mündliche Prüfung zur Bachelorarbeit
28.08.2009	Präsentation der Bachelorarbeiten, 13 bis 18 Uhr

Beurteilung

Eine erfolgreiche Bachelorarbeit zählt 12 ECTS-Punkte pro Studierenden. Für 1 ECTS Punkt ist eine Arbeitsleistung von ca. 25 bis 30 Stunden budgetiert. Siehe auch http://unterricht.hsr.ch/staticWeb/allModules/10939_M_BAI.html für die Modulbeschreibung der Bachelorarbeiten.

Für die Beurteilung sind die HSR-Betreuer verantwortlich unter Einbezug des Feedbacks des Auftraggebers.

Gesichtspunkt	Gewicht
1. Organisation, Durchführung	1/6
2. Berichte (Abstract, Mgmt Summary, technischer u. persönliche Berichte) sowie Gliederung, Darstellung, Sprache der gesamten Dokumentation	1/6
3. Inhalt (*)	3/6
4. Mündliche Prüfung zur Bachelorarbeit	1/6

*) Die Unterteilung und Gewichtung von 3. Inhalt wird im Laufe dieser Arbeit mit den Studierenden festgelegt.

Weiterhin sind die Informationen und Richtlinien in [3] zu beachten.
Im Übrigen gelten die Bestimmungen der Abteilung Informatik zur Durchführung von Bachelorarbeiten.

3 Beilagen

[1] „Projektarbeit Smtp Mailer.pdf" Projektbeschrieb "SMTP Mailer" der Comparis AG. Version 11.12.2008

[2] Smtp Mailer System Design.pdf: Architektur des Compris Mail Systems Design. Version 11.12.2008

[3] Tipps für die Strukturierung und Planung von Studien-, Diplom- und Bachelorarbeiten, Stefan Keller, IFS, 4 März 2008.

25. Feb 2009

4 Erklärung zur Erarbeitung

Ich erkläre hiermit,

- dass ich die vorliegende Arbeit selber und ohne fremde Hilfe durchgeführt habe, ausser derjenigen, welche explizit in der Aufgabenstellung erwähnt ist oder mit dem Betreuer schriftlich vereinbart wurde,
- dass ich sämtliche verwendeten Quellen erwähnt und gemäss gängigen wissenschaftlichen Zitierregeln korrekt angegeben habe.

Ort, Datum: _____

Name: _____

Unterschrift: _____

Ort, Datum: _____

Name: _____

Unterschrift: _____

5 Einführung

5.1 Allgemeine Beschreibung

5.1.1 System Übersicht

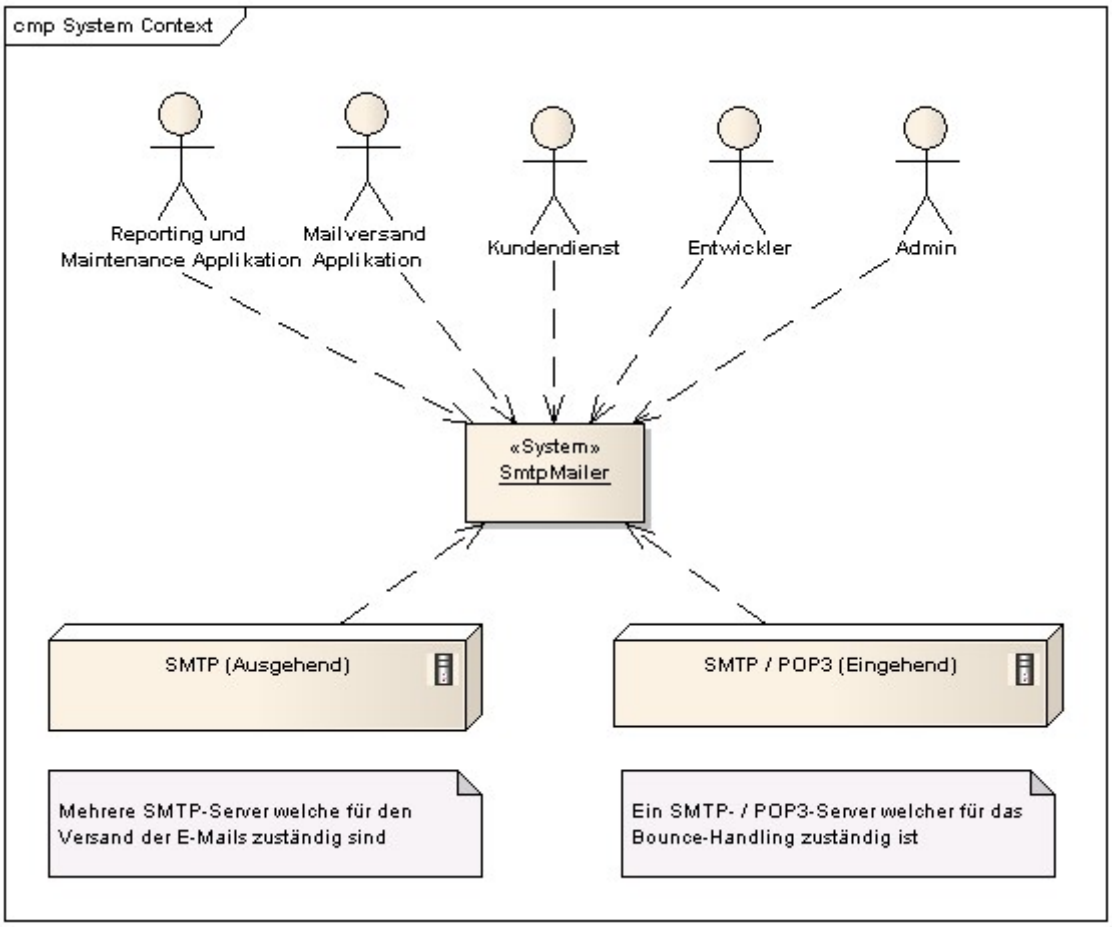


Abbildung 3 - System Kontext

Das System *Smtp Mailer* interagiert mit den menschlichen Aktoren Kundendienst, Entwickler und Admin durch eine Webapplikation als auch mit Reporting und Maintenance sowie mit Mailversand Applikationen über zwei getrennte Entwickler APIs (siehe Abbildung 3 - System Kontext).

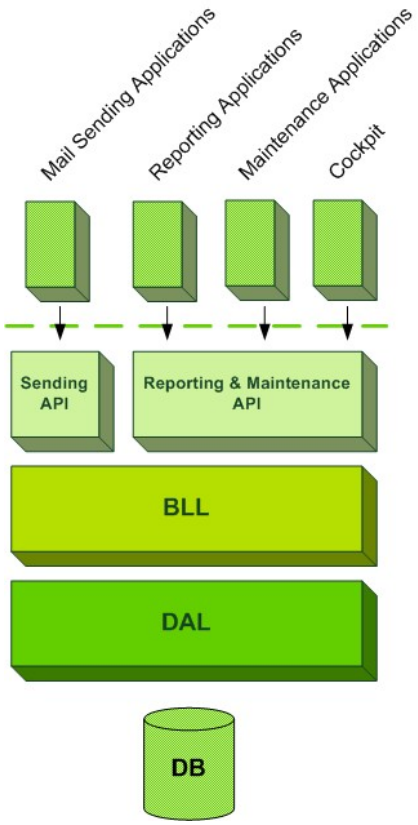


Abbildung 4 - Architekturübersicht

Die eine API dient den Mailversand Applikationen, um Nachrichten zu versenden und den Versandstatus abzufragen (Sending API). Die andere API erlaubt es, die Reporting- und Maintenance-Dienste des *Smtp Mailers* zu nutzen (Reporting & Maintenance API). Dazu gehört das Abfragen von Nachrichten, Auswertungen und des Qualitätsindikators sowie das Durchführen einer Datenbankbereinigung.

Für den Versand der Nachrichten sind eine oder mehrere SMTP Server zuständig, welche die zu versendenden E-Mails durch das *Smtp Mailer* System übermittelt bekommen und schlussendlich über das SMTP Protokoll versenden.

Fehlermeldungen (NDRs), welche vom Mailserver generiert werden wenn eine Nachricht nicht zustellbar ist, werden über einen weiteren SMTP Server entgegen genommen. Das *Smtp Mailer* System holt dort die NDRs per POP3 oder IMAP ab und wertet diese aus.

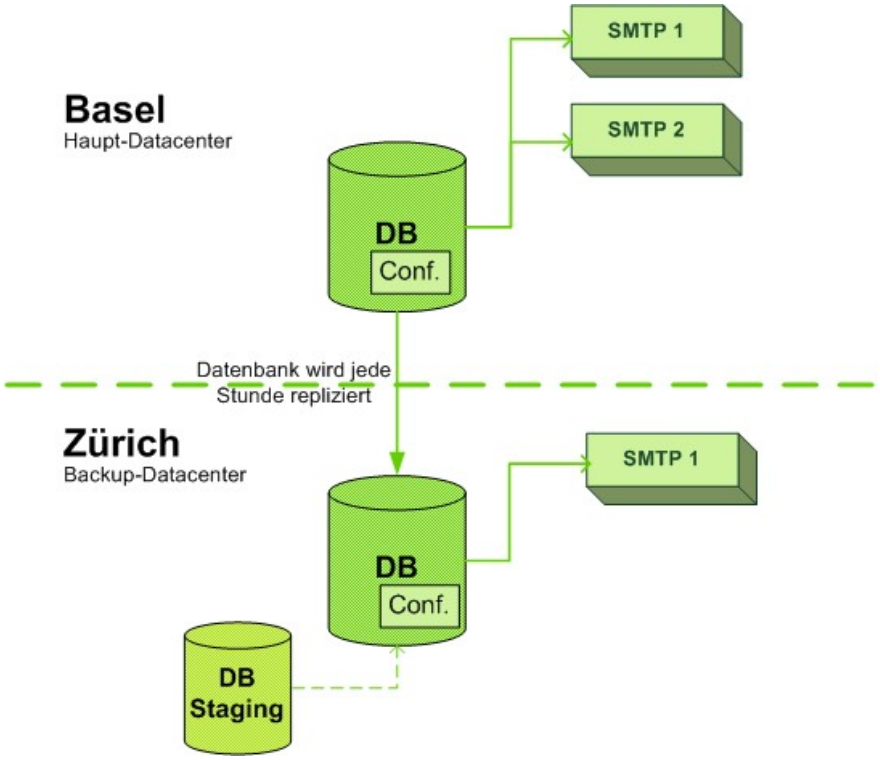


Abbildung 5 - Standortübersicht

Die Firma comparis.ch AG betreibt zwei Datacenter. Beide verfügen über einen identischen Aufbau, nur dass im Haupt-Datacenter Basel pro Komponente mehr Hardware für die Lastverteilung und die Redundanz vorhanden ist. Um bei einem Ausfall in Basel den Betrieb der Dienstleistungen weiter garantieren zu können, werden die Datenbanken in Basel stündlich auf die Datenbanken nach Zürich repliziert. Aus diesem Grund ist es wichtig, dass alle Konfigurationen, welche standortunabhängig sind, in der Datenbank gespeichert werden. Konfigurationen welche jedoch standortspezifische Parameter enthalten, werden in Konfigurationsdateien ausgelagert, damit beim Replizieren der Datenbank von Basel nach Zürich kein undefiniertes Verhalten auftritt.

5.1.2 Produkt Perspektive

Die vorhandene Mail Lösung NLNMailer der comparis.ch AG soll durch die in diesem Projekt zu entwickelnde E-Mail Lösung *Smtp Mailer* abgelöst werden. Der bestehenden Lösung NLNMailer fehlt es an Funktionalitäten, welche mit diesem Projekt abgedeckt werden sollen. Dazu zählt eine Framework Integration für die Konfiguration und das Logging, welche es erlaubt, dass comparis.ch Applikationen direkt über eine API den *Smtp Mailer* konfigurieren und auf Logging-Daten zugreifen können. Zudem sind keine E-Mails im MIME-Format möglich und es können keine Attachments versendet werden. Ein Bounce-Handling soll ein Qualitätsindikator für jede E-Mail Adresse einführen, damit z.B. falsch eingetragene E-Mail Adressen erkannt werden können und in Zukunft an diese E-Mail Adresse keine E-Mails gesendet werden müssen. Die Lösung soll vollständig über ein Webinterface und über Konfigurationsfiles konfigurierbar sein. Es soll nach Möglichkeit die Unterstützung für die verschlüsselte E-Mail Übertragung beachtet werden.

5.1.3 Benutzer Charakteristik

Der *Smtp Mailer* bietet verschiedene Schnittstellen für die Benutzung an. Bestehende comparis.ch Applikationen können den *Smtp Mailer* über zwei verschiedene APIs verwenden, für menschliche Benutzer wird eine Webapplikation für die Konfiguration des *Smtp Mailers* und das Reporting angeboten. Zudem können gewisse Einstellungen über Konfigurationsfiles vorgenommen werden

(solche Einstellungen, welche standortabhängig sind und nicht zur Laufzeit konfigurierbar sein müssen).

5.1.3.1 API

Vom *Smtp Mailer* werden zwei APIs angeboten:

- API für Mailversand Applikationen, um Nachrichten zu versenden und den Versandstatus abzufragen
- API für Reporting und Maintenance Applikationen, um Nachrichten, Reports und den Qualitätsindikators abzufragen sowie die Datenbankbereinigung durchzuführen. Zudem können CleanupRules und NdrPatterns verwaltet werden und die Black/Whitelist konfiguriert werden

Diese APIs sollen von bestehenden comparis.ch Applikationen genutzt werden können.

5.1.3.2 Webapplikation

Jede Person, welche minimale Web Kenntnisse besitzt, soll die Webapplikation benutzen können. Die Webapplikation ist eine Website, über welche der *Smtp Mailer* konfiguriert werden kann und Reports generiert und dargestellt werden können.

5.1.3.3 Konfigurationsfiles

Die Konfigurationsfiles sollen einfach und verständlich aufgebaut sein. Eine Konfiguration mittels eines Konfigurationsfiles soll von Personen mit Systemkenntnissen vorgenommen werden können.

5.1.4 Einschränkungen

Die Grenzen der zu entwickelnden Lösung sind durch präzise Anforderungen von comparis.ch AG im Dokument (Anforderungen SMTP Mailer für comparis.ch, 2009) definiert.

5.1.5 Annahmen

Es wird angenommen, dass die vom Kunden formulierten Anforderungen komplett sind. Weitere Anforderungswünsche können aber ohne weiteres berücksichtigt werden. Jedoch müsste die Erweiterung dieser Anforderungsspezifikation um weitere zu erfüllende Punkte unter Berücksichtigung des Zeitplanes diskutiert werden.

5.1.6 Abhängigkeiten

Das Projekt *Smtp Mailer* ist unabhängig von anderen Projekten oder Resultaten.

5.1.7 Use Cases Überblick

- UC 1 Nachricht versenden
- UC 2 Versandstatus abfragen
- UC 3 Nachricht abfragen
- UC 4 Filterliste CRUD
- UC 5 NDR Pattern CRUD
- UC 6 Qualitätsindikator abfragen
- UC 7 Report abfragen
- UC 8 Datenbank bereinigen
- UC 9 CleanupRule CRUD
- UC 10 NDR Nachricht verarbeiten

6 Resultate

6.1 Zielerreichung

Sämtliche gemäss Anforderungsspezifikation geforderten funktionalen und nicht funktionalen Anforderungen konnten erreicht werden. Die verschiedenen Bestandteile des *Smtp Mailer* Systems wurden durch die comparis.ch AG, vertreten durch Benedikt Unold, an Meilensteinsitzungen abgenommen.

6.2 Ausblick für Weiterentwicklungen

Der *Smtp Mailer* muss in einer nächsten Phase mittels realen Testdaten und realem Datenvolumen getestet werden. Eventuell können an gewissen Schlüsselstellen im System, wie z.B. im Data Access Layer (Datenbank Zugriffsschicht), noch Optimierungen vorgenommen werden.

Nach Abschluss dieser Testphase kann die Überführung des Systems in die produktive Umgebung erfolgen.

6.3 Persönliche Berichte

6.3.1 Schlussbericht Danny Meier

Die selbst ausgesuchte Bachelorarbeit ist eines der Dinge auf die ich sehr gerne und positiv gestimmt zurückblicken werde. Diese Bachelorarbeit ermöglichte es ein Produkt zu erstellen, welches nach unserem Abschluss produktiv eingesetzt wird und mir viele Möglichkeiten bot, die Kenntnisse in .NET zu vertiefen. Zum einen lernte ich die neue Technologie Silverlight 2 kennen, welche gleich den Einsatz von WCF erforderte, die ich schon lange mal in einem realen Projekt ausprobieren wollte. Weitere neue Punkte die ich dazulernen konnte sind der Umgang mit dem DTC (Distributed Transaction Coordinator) welcher es erlaubt Transaktionen schichten- und threadübergreifend zu halten. Bisher waren mir nur die normalen Datenbanktransaktionen bekannt, welche aber spätestens mit dem Schliessen der Datenbankverbindung abgeschlossen sind. Und das starke Ausrichten der Software nach Performance-Kriterien. Es war nervenaufreibend und spassig zugleich zu sehen, dass minimale Einstellungen in der MSSQL Datenbank oder im T-SQL Query zu viel effizienteren Laufzeiten führten.

Den Risiken habe ich früher nicht viel Beachtung geschenkt. Ich habe diese wahrgenommen, jedoch nicht versucht mit zeitintensiven Massnahmen diese zu minimieren resp. ganz zu eliminieren. Da der RUP Vorgehensprozess die Risiken jedoch als integraler Bestandteil sieht, haben wir uns auch intensiv um die Risiken gekümmert. Eines der Hauptrisiken war das Erreichen der Performanz. Dieses Risiko zu minimieren kostete viel Zeit, lohnte sich aber doppelt, als wir mit der Implementierung begonnen haben. Viele Quellcodebestandteile konnten übernommen werden und wir waren sicher, dass diese sehr effizient funktionieren. Ohne die Risikoabdeckung hätte es bestimmt starke Verzögerungen bei der Implementierung gegeben und uns hätte die Zeit in den relevanten Schlussiterationen gefehlt.

Die Zusammenarbeit mit Benedikt Unold von der comparis.ch AG, unserem Industriepartner, hat sehr gut funktioniert und Spass gemacht. Benedikt Unold war stets bestens informiert und hat sich gut auf unsere Sitzungen vorbereitet. Unklarheiten bezüglich Aufgabe und Umsetzung konnten immer gleich beseitigt werden und er gab uns nützliche Tipps, wie die Performanz weiter gesteigert werden kann. Die Teamarbeit mit Micha Boller war grossartig. Er arbeitete sehr zielstrebig, zuverlässig und ausdauernd und konnte mich zu jeder Zeit motivieren, wenn mal etwas nicht ganz nach Plan lief. Da wir uns über einige Jahre schon kannten, wusste ich genau, welche Qualitäten er ins Team mitbringt und konnten uns entsprechend auch gut abstimmen.

Unsere Zeitplanung ist sehr gut aufgegangen, jedoch wussten wir von Anfang an, dass der theoretische Zeitaufwand von 360 Stunden pro Personen bei uns nicht reichen würde. Da mir die Aufgabenstellung sowie die Zusammenarbeit mit Micha Boller und dem Industriepartner grossen Spass bereitet hat, war ich auch bereit mehr zu leisten, auch wenn dies zum Schluss nicht honoriert wird.

6.3.2 Schlussbericht Micha Boller

Eine sehr intensive aber lehrreiche Zeit geht mit der Abgabe dieser Bachelorarbeit zu Ende. Sowohl fachlich als auch in der Disziplin Projektmanagement konnte ich einiges dazulernen und mehr Routine dazugewinnen. Ich werde mit einem guten Gefühl diese Bachelorarbeit in Erinnerung halten, weil ich weiss, dass Danny Meier und ich eine sehr gute und den Kunden zufriedenstellende Arbeit geleistet haben.

Da wir das Thema zur Bachelorarbeit selber gesucht und bestimmt haben, fehlte es mir nie an Motivation oder daran, mich im Thema und den Technologien weiter zu vertiefen und optimale Lösungen zu erarbeiten. So denke ich an die Schlagwörter und eingesetzten Technologien wie Silverlight, ADO.NET, Distributed Transaction Coordinator oder Table Hints mit welchen wir uns während dieser Arbeit beschäftigen durften. Eine weitere Neuheit für mich war, während dem Programmieren auf die Performance achten zu müssen. Alles bisherige was wir während dem Studium entwickelt hatten musste lediglich funktionale Anforderungen einhalten. Hauptmotivationsfaktor dieser Bachelorarbeit war es allerdings, dass die während dieser Bachelorarbeit erstellte Software *Smtp Mailer* tatsächlich bei comparis.ch AG gebraucht und eingesetzt wird.

Einen sehr spannenden Teil dieser Arbeit sah ich in der Zusammenarbeit mit dem Kunden. Gerade hier konnte ich sicher sehr viel für das spätere Berufsleben profitieren und mitnehmen. Wir erlebten einige Situationen, in welchen wir unsere Lösungen kompetent und überzeugend vor dem Kunden präsentieren mussten oder unsere Vorschläge und Meinungen in Diskussionen einbringen mussten.

Das Arbeiten im Projektteam zusammen mit Danny Meier klappte hervorragend. Es herrschte eine sehr angenehme, arbeitswillige aber doch lockere Stimmung während dem Arbeiten an der Bachelorarbeit. Entscheidungen oder das Erarbeiten von Lösungsvorschlägen erfolgten immer in Teamarbeit. Die Arbeitsaufteilung funktionierte ebenfalls sehr gut, wir konnten häufig parallel am Projekt arbeiten ohne voneinander abhängig zu sein. Das war sehr effizient und so kamen wir sehr gut und schnell voran. Danny Meier schaffte es immer wieder, uns bei Problemen weiterzubringen oder Vorschläge zu Lösungsansätzen zu liefern. Ich wusste schon zu Beginn der Bachelorarbeit, dass er hier sehr wertvoll sein kann.

Ich möchte mich beim Industriepartner, vertreten durch Benedikt Unold, bedanken für die gute Zusammenarbeit und die Unterstützung während der Arbeit wo es nur ging. Wir konnten einiges von Benedik Unold bezüglich Architektur, Datenbankdesign und Datenbanktricks lernen.

Ebenfalls bedanken möchte ich mich bei Prof. Dr. Markus Stolze für die kompetente Betreuung der Arbeit. Auch Prof. Dr. Markus Stolze unterstützte uns hervorragend bei Fragen und Problemen.

6.4 Methodenreflexion

Der nachfolgende Abschnitt dokumentiert während dieser Bachelorarbeit eingesetzte Methoden. Dabei wird beschrieben, was gut und was weniger gut funktionierte.

6.4.1 Rational Unified Process

Mit der Software Engineering Methode Rational Unified Process, kurz RUP, waren wir schon vor der Bachelorarbeit sehr vertraut. Dies kommt daher, dass in den Fächern Software Engineering 1 und Software Engineering 2 an der HSR der ganze Prozess behandelt wird. Im Software Engineering 2 Projekt hat man die Möglichkeit, ein beliebiges Projekt nach RUP durchzuspielen. Eine weitere Vertiefungsmöglichkeit war die Semesterarbeit, welche ebenfalls nach RUP abgehandelt wurde. Somit war die Anwendung von RUP bereits schon eine Routineangelegenheit. Die zu erstellenden Dokumente und Abläufe waren jederzeit bekannt. Dies vereinfachte und erleichterte die Projektabwicklung sehr.

6.4.2 Test First Prinzip

Wo möglich wurde nach dem Test First Prinzip gemäss der Test Driven Development Vorgehensweise gearbeitet. Das Test First Prinzip bezieht sich auf die Unit-Tests. Zuerst wird der Unit Test zu der Methode geschrieben, welche implementiert werden soll. Der Unit-Test beweist das korrekte Funktionieren der Methode. Erst im zweiten Schritt wird der eigentliche Produktionscode geschrieben. So kann sofort überprüft werden, ob der Code wie gedacht funktioniert. Zudem hat man zwangsläufig eine 100% Code Coverage d.h. sämtlicher Business Code ist mit Unit-Tests getestet.

Vorteile mit der Test Driven Development Methode liegen auf der Hand. Gemäss dem Artikel (Crispin, 2006) kann mit den durch das Test First Prinzip entwickelten Unit-Tests nach einem Refactoring das korrekte Funktionieren der Software einfach verifiziert werden. Ebenfalls sagt der Artikel aus, dass Entwickler, welche Test Driven Development praktizieren, weniger am debuggen sind, weil sich weniger Fehler im Code befinden.

(Janzen, 2005) besagt sogar, dass Code der mit Test Driven Development entwickelt wurde, 18-50% mehr externe, d.h. unvorhergesehene Test Cases besteht.

Jedoch ist nicht alles mit Unit-Tests testbar. Z.B. sind multithreaded Abläufe wie wir sie beim Mail Modul oder dem Bounce Modul mit den Workerthreads hatten nicht testbar. Somit kann auch nicht jede Entwicklung nach dem Test First Prinzip erfolgen.

6.4.3 Einsatz von einem Wiki

Der Einsatz vom Wiki bietet den grossen Vorteil, dass die Information sehr einfach und schnell auf aktuellstem Stand gehalten werden kann und auch von überall her einsehbar ist. Zudem gibt es weniger Probleme beim gleichzeitigen Bearbeiten des Wikis durch mehrere Projektmitarbeiter als wenn man z.B. ein Word Dokument gleichzeitig bearbeitet, welches per SVN in eine Versionsverwaltung eingecheckt ist.

Im Wiki verwalteten wir unseren Fragenkatalog, das Glossar, Todo Listen für die Projektmitarbeiter, Pendenzenlisten sowie weitere projektbezogene Informationen.

Der Einsatz eines Wikis empfiehlt sich sehr.

6.4.4 Codereviews

Das Durchführen von Code Reviews verhalf uns dazu, dass beide Projektmitarbeiter ein gutes Verständnis vom gesamten Code besitzen. So erhielt man auch Einblicke in Codeteile, welche man nicht selber entwickelt hat.

6.5 Silverlight Reflexion

Dieser Abschnitt beschreibt Probleme und deren Lösungen, welche uns bei der Entwicklung mit Silverlight (Version 2) beschäftigten.

6.5.1 AutoCompleteBox (aus Toolkit)

Die Auto Complete Box stellte aus unserer Sicht ein sehr nützliches Feature dar. Das Ziel war es z.B. dass dem Benutzer beim Suchen nach einer Nachricht anhand der Empfänger E-Mail Adresse bereits erfasste E-Mail Adressen in der Auto Complete Box angezeigt werden. Das Problem war allerdings, dass das Laden von viel Auto Complete Daten sehr lange dauern kann.

Problem	Das Abfragen nach darzustellenden Auto Complete Daten führt zu einem Timeout auf Seiten des WCF Services
Lösung	Auch wenn z.B. erst nach drei eingegebenen Zeichen die Daten für die Auto Complete Box geladen wurden, erhielten wir immer wieder Timeouts beim WCF Service. Wir haben uns nun darauf geeinigt, die Auto Complete Funktion nicht zu nutzen. Sie wäre bei uns z.B. bei E-Mail Adressen zum Einsatz gekommen. Jedoch ist hier mit einem sehr grossen Datenvolumen zu rechnen.

6.5.2 Itemssource von Datagrid

Ein typisches Anwendungsszenario von einem Datagrid in unseren Webapplikationen war es, dass Elemente im Datagrid editiert, gelöscht oder neue Elemente hinzugefügt werden können.

Problem	Beim Editieren oder Löschen eines Eintrages im Datagrid erscheint die entsprechende Zeile als weisse Zeile
Lösung	Das Problem war, dass jedesmal beim Editieren oder Löschen eines Eintrages die Elemente neu von der Datenbank gelesen wurden und diese gelesene Collection erneut als Itemssource vom Datagrid gesetzt wurde. Die Lösung bestand darin, dass die Itemssource Collection auf dem Datagrid nur einmal beim Instanziieren des Datagrids gesetzt wird. Danach wird beim erneuten Lesen der Einträge aus der Datenbank die als Itemssource gesetzte Collection geleert und die gelesenen Einträge wieder in die Collection abgefüllt

6.5.3 Objektdarstellung im Datagrid mit IValueConverter Interface steuern

Wenn in einem Datagrid Objekte, welche per Data Binding als Quelle des Datagrids angegeben wurden, dargestellt werden möchten, so wird einfach die Objektreferenz (ToString() Methode des Objektes) im Datagrid ausgegeben.

Problem	Von Objekten, welche im Datagrid dargestellt werden sollen, wird die Objektreferenz ausgegeben
Lösung	Damit dem Datagrid gesagt werden kann, wie ein Objekt dargestellt werden soll, muss eine Klasse erstellt werden, welche das Interface System.Windows.Data.IValueConverter implementiert. In der zu implementierenden Convert Methode erhält man das Objekt, welches per Data Binding gebunden ist. Als Returnwert gibt man den Wert zurück, welcher dargestellt werden soll. Dabei muss beim Data Binding im XAML Code der Converter für die Spalte angegeben werden (siehe Code Statement unten)

```
<data:DataGridTextColumn Header="Empfänger Vor-/Nachname" Width="180" Binding="{Binding
Recipients, Converter={StaticResource RecipientNameConverter}}"/>
```

6.5.4 Paging im Datagrid

Bei unseren Datagrids ist es durchaus möglich, dass ein Datagrid bis zu 1000 Datensätze enthält. Würde man kein Paging einsetzen, so wäre das Datagrid sehr lange und eine extrem kleine Scrollbar

würde im Browser resultieren. Ein Paging stellt hier die richtige Lösung dar, bei welchem zur nächsten Resultatseite geschaltet werden kann.

Problem	Das Datagrid besitzt kein automatisches Paging
Lösung	Das Paging musste manuell implementiert werden. Dabei hielten wir uns an den folgenden Ansatz: (Dean, 2009)

6.5.5 Sortierung im Datagrid

Das Datagrid UserControl von Silverlight Version 2 erlaubt es nicht, benachrichtig zu werden, wenn der Benutzer die Sortierung der Spalten des Datagrids ändert. Weil wir ein clientseitiges Paging der Einträge im Datagrid implementiert haben, dürfen nicht nur die in der aktuellen Page dargestellten Einträge sortiert werden, sondern es müsste die ganze Collection entsprechend sortiert werden, welche dem Datagrid als Itemssource dient.

Problem	Es gibt beim Datagrid keinen Event, um benachrichtigt zu werden, wenn der Benutzer die Sortierung der Spalten ändert
Lösung	Wir haben für dieses Problem keine Lösung gefunden. Eine Sortierung wirkt sich also nur auf die Einträge in der aktuellen Page aus.

6.5.6 Input Validierung

Eine fertig zu verwendende Lösung für die Input-Validierung gibt es bei Silverlight (Version 2) nicht (z.B. für die Validierung von Webformular Daten).

Problem	Es gibt keinen Input-Validierungsmechanismus für Webformulare
Lösung	Wir implementierten eine eigene Lösung, indem wir die Daten aus dem Webformular manuell validieren und gegebenenfalls eine Fehlermeldung ausgeben.

6.5.7 Problem mit fehlendem Cross-Domain Policy File

Wenn beide Projektmitglieder am gleichen Silverlight Projekt im Visual Studio arbeiteten und zwischendurch ein Commit auf dem SVN Server machten, so kam es häufig vor, dass nach dem Update das andere Projektmitglied eine CommunicationException hatte. Die Fehlermeldung besagte, dass kein Cross-Domain Policy File vorhanden ist. Dies, obwohl die Silverlight Applikation lokal laufengelassen wird.

Problem	Plötzlich (z.B. nach einem SVN Update) tritt eine CommunicationException zur Laufzeit auf, welche besagt, dass kein Cross-Domain Policy File vorhanden ist
Lösung	Unsere Lösung zum Problem war es, die Service Reference zum WCF Service zu löschen und neu zu generieren.

6.5.8 Probleme bei Aktualisierung von Service Reference

Um die Service Reference zu aktualisieren, klickt man mit der rechten Maustaste im Visual Studio auf die Service Reference und sagt ‚Update Service Reference.

Problem	Nach dem Aktualisieren der Service Reference befinden sich im ServiceReference.ClientConfig File mehrere Bindings und Endpoints. Beim Starten der Webapplikation weiss die Runtime nicht, welches Binding und welcher Endpoint zu verwenden ist.
Lösung	Nach jeder Aktualisierung der Service Reference muss überprüft werden, ob das alte Binding und der alte Endpoint korrekt aus dem File ServiceReference.ClientConfig gelöscht wurden resp. dass sich nur eine Definition darin befindet.

6.5.9 Styles und Templates

Für die Erstellung der Styles und Templates wurde Microsoft Expression Blend 2 verwendet.

Problem	**Beim Erstellen von eigenen Styles ist es immer wieder vorgekommen, dass gewisse XAML-Elemente, die automatisch erstellt wurden, nicht gültig sind. Dies wurde in Blend 2 sofort ersichtlich, indem anstelle der Preview der Seite nur eine rote Fehlermeldungsbox erschien.**
Lösung	Man musste sehr umständlich Eigenschaft um Eigenschaft verändern und beobachten, ob es sich um eine erlaubte Eigenschaft handelt. Einige Male konnten wir den Style nicht verändern und mussten den Style der Komponente auf dem Standard-Style (Blau) belassen.

Problem	**Um einer einzelnen Komponente eine andere Farbe (Style) zu verpassen, ist ein enorm hoher Aufwand zu tätigen**
Lösung	Jede Komponente setzt sich aus vielen Unterkomponenten zusammen. Leider wurde kein schnellerer Weg gefunden, als durch jedes Template zu gehen und diesem einen Style zu zuweisen.

6.5.10 Exception Handling über WCF

Für die Kommunikation zwischen Silverlight (Version 2) und dem *Smtp Mailer* wurde ein Silverlight-enabled WCF Service mit einem BasicHttpBinding verwendet.

Problem	**WCF in Kombination mit Silverlight (Version 2) erlaubt es nicht, das Exception Handling wie gewohnt über FaultException abzuhandeln.**
Lösung	Stattdessen haben wir eine generische ResultData Klasse implementiert, welche ein Objekt von generischem Typ besitzt (Returntyp der WCF Methode) und ein String, welcher eine Error Message enthalten kann (siehe Codestatement unten). Ist der Error Message String nicht leer, so weiss die Silverlight Applikation, dass eine Exception auf Seiten vom *Smtp Mailer* aufgetreten ist.

```
[DataContract]
public class ResultData<T>
{
    /// <summary>
    /// The return type of the WCF method
    /// </summary>
    [DataMember]
    public T Result { get; set; }

    /// <summary>
    /// Contains a textual error message, if
    /// an exception is occured. If error message
    /// is empty, no exception occured
    /// </summary>
    [DataMember]
    public string ErrorMessage { get; set; }
}
```

7 Anforderungsspezifikation

7.1 Funktionale Anforderungen

Einen Grossteil der funktionalen Anforderungen und nicht funktionalen Anforderungen sind bereits in der Aufgabenstellung aufgeführt. Ergänzend wurden mit einem zusätzlichen Dokument (Anforderungen SMTP Mailer für comparis.ch, 2009) beim Kickoff Meeting bei der comparis.ch AG weitere Anforderungen aufgelistet.

7.1.1 Versenden von E-Mail Nachrichten

- E-Mail Nachricht im Plaintext- und MIME-Format möglich
- E-Mail Nachricht kann an mehrere Empfänger gleichzeitig versendet werden, klassiert über die E-Mail Felder TO, CC und BCC
- Unterstützung von Attachements und Signaturen
- Zeitgesteuerter Versand von einer E-Mail Nachricht gemäss einer Angabe ‚nicht früher als'. Die E-Mail Nachricht wird also nicht früher als die angegebene Zeit versendet
- Versandpriorisierung aufgrund der Zuweisung eines Applikationscodes und Mailtyps zu einem SMTP Server
- Unterstützen einer Blacklist. Die Blacklist ist immer aktiv. An alle E-Mail Adressen resp. Domains auf der Blacklist werden keine E-Mail Nachrichten versendet, auch wenn dies über den *Smtp Mailer* versucht wird. E-Mail Nachrichten werden als erledigt markiert, wenn sie aufgrund eines Blacklist-Eintrages nicht versendet werden (das heisst solche Nachrichten können bereinigt werden)
- Unterstützen einer Whitelist. Die Whitelist kann aktiviert / deaktiviert werden. Es können bei aktiver Whitelist nur an E-Mail Adressen oder Domains E-Mail Nachrichten gesendet werden, welche auf der Whitelist stehen

7.1.2 API

Grundsätzlich werden zwei verschiedene APIs angeboten:

API für Mailversand Applikationen:

- Das Versenden einer E-Mail Nachricht muss über eine API möglich sein
- Eine Abfrage des aktuellen Status des Versands einer E-Mail Nachricht muss über eine API möglich sein

API für Reporting und Maintenance Applikationen:

- Der Qualitätsindikator einer E-Mail Adresse muss über eine API abgefragt werden können
- Sämtliche Daten für die verschiedenen angebotenen Reports müssen über eine API ausgelesen werden können
- Das Durchführen einer Datenbankbereinigung muss ermöglicht werden
- Das Verwalten der NdrPatterns, CleanupRules, Filterlisten muss möglich sein

7.1.3 Ablage von E-Mail Nachrichten

- Ablage der E-Mail Nachrichten in einer Datenbank, wobei Header und Content einer E-Mail Nachricht unterschieden werden
- Pro E-Mail Nachricht wird ein Status pro Empfänger abgelegt (nicht versendet, versendet, blacklisted, failed)
- Pro E-Mail Nachricht können Metadaten in die Datenbank abgelegt werden (Applikationscode, Mailtyp und Referenz)
- Pro E-Mail Adresse wird ein Qualitätsindikator abgelegt (good, poor, failed)

- Bei einem Fehler beim Versenden einer E-Mail Nachricht wird die Fehlermeldung abgelegt

7.1.4 Konfiguration des *Smtp Mailers* über Webinterface

- Konfiguration der Blacklist (hinzufügen, löschen und bearbeiten von Einträgen)
- Konfiguration der Whitelist (hinzufügen, löschen und bearbeiten von Einträgen)
- Konfiguration von Bounce-Handling (hinzufügen, löschen und bearbeiten von NdrPatterns anhand von nicht erkannten NdrMails)
- Konfiguration der Datenbankbereinigung (Bereinigung pro Applikationscode und Mailtyp konfigurierbar)

7.1.5 Konfiguration des *Smtp Mailers* über Konfigurationsfiles

- Definieren der Anzahl Prozesse, welche High Priority resp. Low Priority E-Mail Nachrichten abarbeiten und Zuordnung von Prozess zu SMTP Server
- Zuweisen von Prioritäten (High und Low) zu Applikationscode und Mailtyp Tupel. Dabei müssen Wildcards erlaubt sein (*).
- Konfiguration der Übergabemethode (SMTP Protokoll oder Message Queue) zwischen *Smtp Mailer* und dem SMTP Server
- Aktivierung und Deaktivierung der Whitelist

7.1.6 Errorhandling

- Automatisches Bounce-Handling über einen eigenen SMTP Service unabhängig von der FROM oder REPLY-TO Adresse
- Unterscheidung von verschiedenen Bounce-Typen

7.1.7 Debugging- und Reporting-Tool (Webapplikation)

- Abfragen von Fehlermeldungen (Bounces), welche vom Bounce-Handling nicht erkannt wurden
- Anzeigen von E-Mail Nachrichten (als Liste und Zusammenfassung) gefiltert nach Datum / Uhrzeit, E-Mail Adresse(n), Betreff, Text, Status, Applikationscode und Mailtyp
- Detailanzeige von einzelnen E-Mail Nachrichten
- Anzeigen von Qualitätsindikator der E-Mail Adressen, gefiltert nach E-Mail Adresse und Qualitätsindikator. Die Darstellung soll als Übersicht, als Liste oder als Detailanzeige möglich sein
- Anzeigen eines Reports, welcher Auskunft darüber gibt, wie viele E-Mails über einen Zeitraum verschickt wurden und in welchem Status diese sind. Das Resultat soll auf gewisse Applikationscodes, Mailtypen sowie auf den Status eingeschränkt werden können
- Anzeigen eines Reports, welcher Auskunft darüber gibt, wie viele E-Mail Nachrichten in den nächsten n Stunden verschickt werden sollen. Das Resultat soll auf gewisse Applikationscodes und Mailtypen eingeschränkt werden können
- Als Nice-To-Have kann das nochmalige Versenden von Nachrichten, welche mit einer Bounce Nachricht zurückgekommen sind, über die Webapplikation angeschaut werden

7.1.8 Maintenance

- Regelmässige, automatisierte Bereinigung der Datenbank durch Löschen von Content und Attachements von versendeten oder blacklisted Nachrichten. Zudem können Nachrichten bereinigt werden, welche mit einem Hardbounce zurückgekommen sind. Dabei muss die Archivierungszeit (StorageDuration) der Nachricht beachtet werden.

7.2 Nicht funktionale Anforderungen

7.2.1 Allgemein

7.2.1.1 Lizenzanforderungen

Kostenpflichtige Komponenten von Drittherstellern dürfen in der *Smtp Mailer* Software verwendet werden. Eine beabsichtigte Verwendung einer Komponente muss aber mit comparis.ch AG abgesprochen werden. Die Anschaffung würde in einem solchen Fall über comparis.ch AG laufen.

7.2.2 Schnittstellen

7.2.2.1 Hardware Schnittstelle

Die entwickelte Software muss auf einem Dual-Core XEON Server mit 2 GB RAM lauffähig sein. Es handelt sich dabei um eine x64 Architektur (64-Bit).

7.2.2.2 Software Schnittstelle

Die entwickelte Software muss auf einer Microsoft Windows Server 2003 Standard Umgebung lauffähig sein. Die Implementierung der *Smtp Mailer* Software soll in .NET Framework 3.5, C#, ASP.NET und Silverlight geschehen. Auf der Windows Server 2003 Umgebung wird mit dem .NET Framework x64 gearbeitet.

7.2.2.3 Datenbank Schnittstelle

Die Speicherung der Daten geschieht in einer Microsoft SQL Server 2005 Datenbank.

7.2.3 Funktionalität

7.2.3.1 Interoperabilität

Die *Smtp* Mailer Komponente muss mit einer dazuzukaufenden Mail Komponente und einer NDR Komponente zusammenwirken können (beide Komponenten sind in .NET geschrieben).

7.2.3.2 Sicherheit

Damit die Applikationen, welche lediglich über den *Smtp Mailer* Nachrichten versenden und den Versandstatus abfragen wollen, nicht auch Zugriff auf Reporting- und Maintenance-Dienste des *Smtp Mailers* haben, müssen zwei getrennte APIs angeboten werden.

7.2.4 Zuverlässigkeit

7.2.4.1 Fehlertoleranz

Bei Fehlern werden über die API Exceptions an den Benutzer des *Smtp Mailers* geworfen. Dort wo menschliche Aktoren den *Smtp Mailer* nutzen, müssen textuelle Beschreibungen des Fehlers an den Benutzer zurückgegeben werden.

7.2.4.2 Wiederherstellbarkeit

Bei einem Fehler im System (z.B. Absturz) gehen keine Nachrichten verloren. Nicht versendete Nachrichten müssen wieder in die Queue aufgenommen werden und verarbeitet werden.

7.2.5 Benutzbarkeit

7.2.5.1 Verständlichkeit

Die Benutzeroberfläche der Webapplikation muss intuitiv zu bedienen sein. Die Webapplikation muss für einen Benutzer mit Weberfahrung einfach zu handhaben sein.

7.2.5.2 Erlernbarkeit
Die Webapplikation muss nach kurzer Einarbeitungszeit (1/2 Tag) verstanden werden.

7.2.5.3 Bedienbarkeit
Die Oberfläche der Webapplikation muss vollständig mit der Maus bedienbar sein.

7.2.6 Effizienz

7.2.6.1 Zeitverhalten
Die Versandleistung des *Smtp Mailers* muss 2‘000‘000 E-Mail Nachrichten pro Tag und 300‘000 E-Mail Nachrichten pro Stunde betragen. Diese Leistungen müssen auf der gegeneben Hardware und Netzwerkinfrastruktur von comparis.ch AG erreicht werden.

Es kann davon ausgegangen werden, dass 10 % der versendeten Nachrichten beim Versenden fehlschlagen und somit mit einem NDR zurückkommen. Die Verarbeitungsleistung des *Smtp Mailers* muss somit 200‘000 NDR Nachrichten pro Tag und 30‘000 NDR Nachrichten pro Stunde erreichen.

7.2.7 Änderbarkeit

7.2.7.1 Naming Guidelines
Der Code, welcher für die Software *Smtp Mailer* geschrieben wird, muss sich an die Naming Guidelines (Microsoft Guidelines for Names, 2009) gemäss Microsoft halten.

Für Assembly Namen müssen folgende comparis.ch Naming Guidelines verwendet werden: Alle Assembly Namen müssen nach dem Schema *Comparis.Framework.SmtpMailer.** aufgebaut sein.

Der Datenbankname muss folgendermassen lauten: SMTP_MAILER_DB

7.2.7.2 Analysierbarkeit
Das System muss Fehler an einer zentralen Stelle festhalten (Logging). So bietet das System Zugriff auf sämtliche festgehaltene Fehler und Störungen.

7.2.7.3 Modifizierbarkeit
Die gemäss den funktionalen Anforderungen konfigurierbaren Teile des Mail Moduls und NDR Moduls müssen ohne Änderungen im Code konfiguriert und angepasst werden können. Einstellungen, welche zwingend zur Laufzeit übernommen werden müssen, müssen über das Webinterface vorgenommen werden können.

7.2.7.4 Testbarkeit
Über Unit-Tests müssen fehlerhafte Codeteile erkannt werden können und bei Änderungen im Code die Lauffähigkeit des Codes verifiziert werden können.

Über die Use Cases muss die Business Logik 100 % Statement Coverage aufweisen (Unit Tests). Das System wird nach dem Test-First Prinzip entwickelt.

7.2.8 Übertragbarkeit

7.2.8.1 Anpassbarkeit
Umgebungsänderungen müssen über Konfigurationsfiles (z.B. App.config) vorgenommen werden können. Dabei kann die Anforderung, dass solche Änderungen ohne Neustart der Applikation übernommen werden können, als Nice-To-Have angeschaut werden.

7.2.8.2 Koexistenz

Es müssen mehrere Instanzen des *Smtp Mailers* gleichzeitig betrieben werden können. Als Möglichkeit kann das Zugreifen von zwei oder mehr *Smtp Mailer* Instanzen auf dem gleichen Server auf dieselbe Datenbank oder auf verschiedene Datenbanken angeschaut werden.

7.3 Use Cases

7.3.1 Übersicht

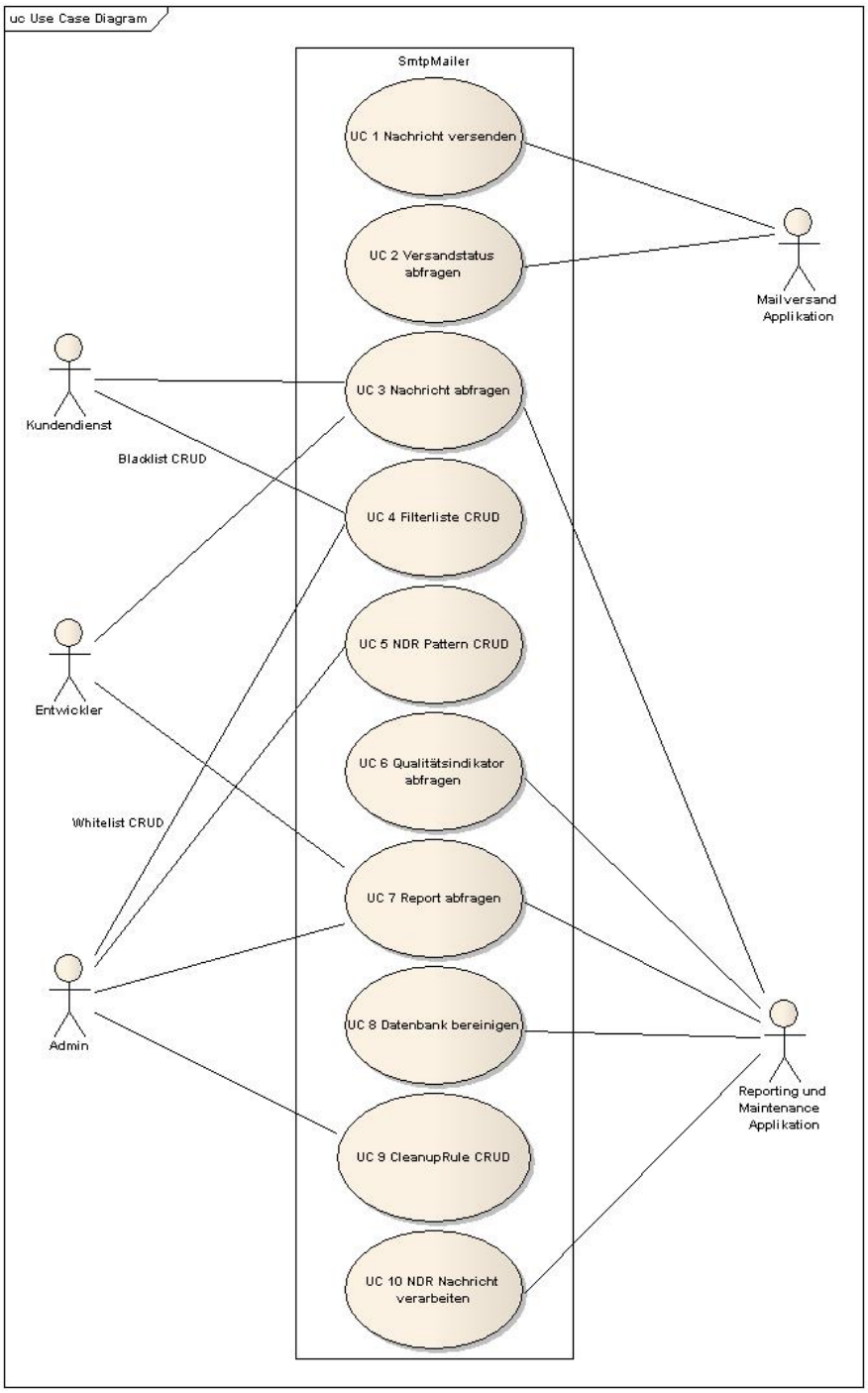


Abbildung 6 - Use Case Diagramm

7.3.2 Aktoren

Aktor	Beschreibung
Admin	Der Admin ruft gelegentlich Reports (Auswertungen) auf, konfiguriert die SMTP Server Zuordnung, die NDR Patterns und pflegt die Whitelist. Die CleanupRules werden ebenfalls vom Admin konfiguriert.
Entwickler	Der Entwickler schaut sich versendete E-Mails an oder ruft Reports (Auswertungen) auf
Kundendienst	Der Büroangestellte überprüft versendete oder zu versendende E-Mails und konfiguriert die Blacklist
Mailversand Applikation	Applikationen nutzen die Dienste des *Smtp Mailers*, um Nachrichten zu versenden und den Versandstatus abzufragen (z.B. Krankenkassen Applikation oder Autoversicherung Applikation)
Reporting und Maintenance Applikation	Applikationen nutzen die Dienste des *Smtp Mailers*, um Nachrichten und Reports abzufragen, den Qualitätsindikator abzufragen oder die Datenbank zu bereinigen.

Tabelle 1 - Aktoren

7.3.3 UC 1 Nachricht versenden

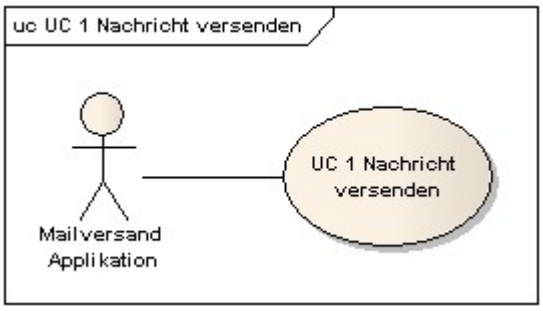


Abbildung 7 - UC1 Nachricht versenden

Name	Nachricht versenden	
Scope	*Smtp Mailer* – Mail Modul	
Level	User Goal	
Primary Actor	Mailversand Applikation	
Stakeholder and Interests	Der Benutzer möchte eine Nachricht versenden im Plaintext- oder MIME-Format inklusive Anhänge und digitaler Signatur	
Preconditions	Das Zertifikat für die digitale Signatur ist vorhanden	
Success Guarantee	Die Nachricht wurde zum versenden geplant und der Benutzer erhielt vom System eine Kennung (guid) um die Nachricht zu verfolgen	
Main Success Scenario	***User Intention***	***System Response***
	1. Der Benutzer möchte eine Nachricht versenden	
	2. Der Benutzer übergibt dem System die Empfänger E-Mail Adressen, Betreff, Inhalt, Anhänge und wählt ob die Nachricht signiert werden muss. Der Benutzer gibt an, zu welcher Zeit der Versand stattfinden soll	
		3. Das System nimmt die Nachricht entgegen
	4. Der Benutzer bekommt eine E-Mail Kennung zurück	
		5. Das System fügt die Nachricht einer Warteschlange hinzu

Extensions	*a. Zu jeder Zeit wenn der Benutzer abbricht: 1. Das System führt keine Aktionen durch *b. Zu jeder Zeit wenn das System fehlschlägt: 1. Das System wirft eine Exception 2. Das System loggt den Fehler 2a. Der Benutzer möchte eine bereits erstellte E-Mail Nachricht erneut versenden: 1. Der Benutzer gibt die E-Mail Kennung der Nachricht an, welche erneut versendet werden soll 2. Das System sucht nach der Nachricht mit der vom Benutzer angegebenen E-Mail Kennung 3. Das System fügt die Nachricht einer Warteschlange hinzu 2b. Die übergebenen Werte für die E-Mail Nachricht sind ungültig: 1. Das System wirft eine Exception 2. Das System bricht die Verarbeitung ab 3a. Der Benutzer hat eine signierte Nachricht gewünscht: 1. Das System erzeugt eine Signatur und bindet Sie in die Nachricht ein
Frequency of Occurence	2‘000‘000x pro Tag
Spezielle nicht funktionale Anforderungen	• Nachrichten dürfen bei einem Fehler im System nicht verloren gehen

Tabelle 2 - UC 1 Nachricht versenden

7.3.4 UC 2 Versandstatus abfragen

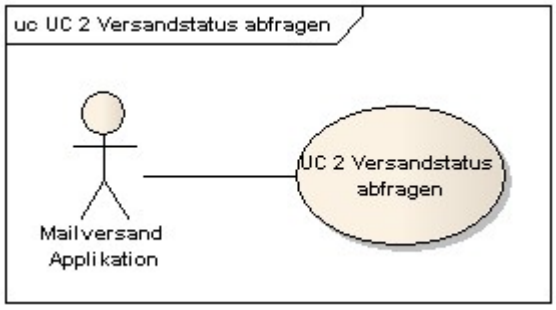


Abbildung 8 - UC 2 Versandstatus abfragen

Name	Versandstatus abfragen	
Scope	*Smtp Mailer* – Mail Modul	
Level	User Goal	
Primary Actor	Mailversand Applikation	
Stakeholder and Interests	Der Benutzer möchte den Versandstatus einer Nachricht abfragen	
Preconditions	Die abzufragende Nachricht besteht im System	
Success Guarantee	Der Versandstatus für die abgefragte Nachricht wurde dem Benutzer vom System geliefert	
Main Success Scenario	***User Intention***	***System Response***
	1. Der Benutzer möchte für eine beliebige Nachricht den Versandstatus abfragen	

	2. Der Benutzer gibt die E-Mail Kennung ein	3. Das System zeigt den Versandstatus für die vom Benutzer abgefragte Nachricht an
Extensions	*a. Zu jeder Zeit wenn der Benutzer abbricht: 1. Das System führt keine Aktionen durch *b. Zu jeder Zeit wenn das System fehlschlägt: 1. Das System wirft eine Exception 2. Das System loggt den Fehler 2a. Die E-Mail Kennung ist ungültig: 1. Das System wirft eine Exception	
Frequency of Occurence	50x pro Woche	
Spezielle nicht funktionale Anforderungen	-	

Tabelle 3 - UC 2 Versandstatus abfragen

7.3.5 UC 3 Nachricht abfragen

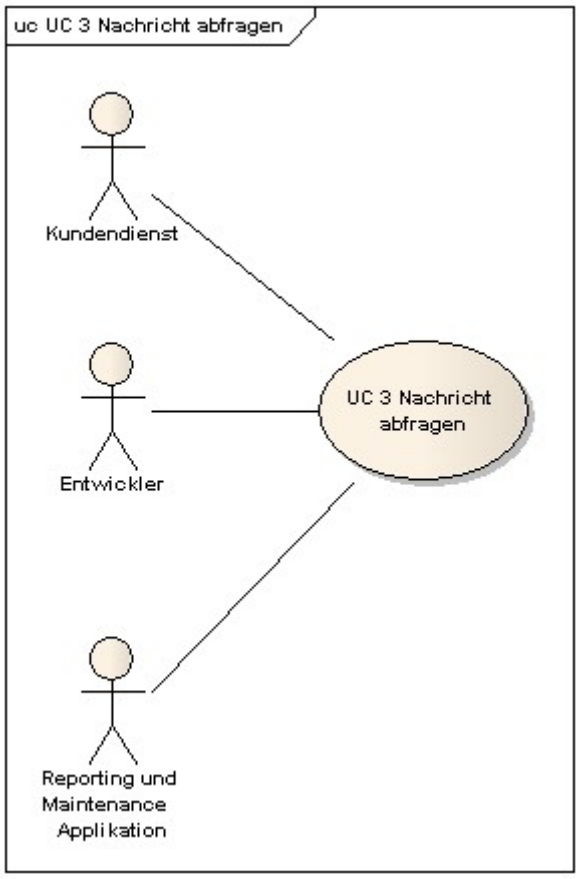


Abbildung 9 - UC 3 Nachricht abfragen

Name	Nachricht abfragen	
Scope	*Smtp Mailer* – Mail Modul und Web Modul	
Level	User Goal	
Primary Actor	Kundendienst, Entwickler und Reporting und Maintenance Applikation	
Stakeholder and Interests	Der Benutzer möchte eine Nachricht einsehen inklusive dem Versandstatus und (Fehler-) Meldungen	
Preconditions	Die abzufragende Nachricht besteht im System	
Success Guarantee	Die Details für die abgefragte Nachricht wurde dem Benutzer vom System geliefert	
Main Success Scenario	***User Intention***	***System Response***
	1. Der Benutzer möchte die Details einer Nachricht ansehen 2. Der Benutzer gibt die E-Mail Kennung ein	3. Das System zeigt dem Benutzer die Details (Datum / Uhrzeit, E-Mail Adresse(n), Subject, Text, Status, Applikationscode und Mailtyp) der Nachricht an

Extensions	*a. Zu jeder Zeit wenn der Benutzer abbricht: 1. Das System führt keine Aktionen durch *b. Zu jeder Zeit wenn das System fehlschlägt: 1. Das System informiert den Benutzer in einer sinnvollen Form (Textuelle Beschreibung des Fehlers bei menschlichen Benutzern, Exception bei API Benutzer) 2. Das System loggt den Fehler 2a. Die E-Mail Kennung ist ungültig: 1. Das System informiert den Benutzer in einer sinnvollen Form (Textuelle Beschreibung des Fehlers bei menschlichen Benutzern, Exception bei API Benutzer)
Frequency of Occurence	20x pro Woche
Spezielle nicht funktionale Anforderungen	-

Tabelle 4 - UC 3 Nachricht abfragen

7.3.6 UC 4 Filterliste CRUD

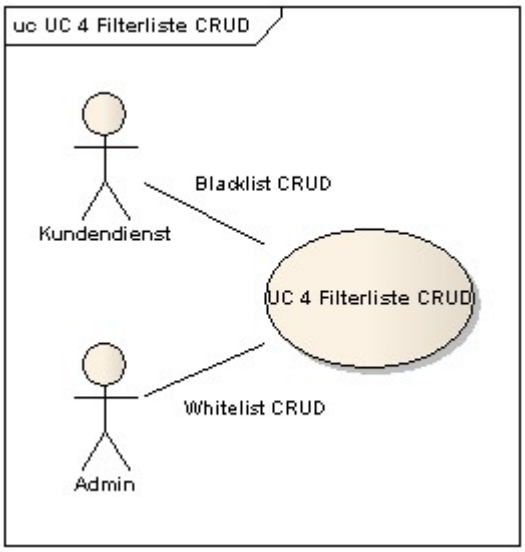


Abbildung 10 - UC 4 Filterliste CRUD

Name	Filterliste CRUD	
Scope	*Smtp Mailer* – Mail Modul und Web Modul	
Level	User Goal	
Primary Actor	Kundendienst und Admin	
Stakeholder and Interests	Der Benutzer möchte eine E-Mail Adresse oder eine ganze Domain in der Filterliste erstellen, abfragen, aktualisieren oder löschen.	
Preconditions	-	
Success Guarantee	Die eingetragenen Änderungen sind gespeichert und werden per sofort angewendet.	
Main Success Scenario	***User Intention***	***System Response***
	1. Der Benutzer möchte einen Filterlisteeintrag erstellen, abfragen, aktualisieren oder löschen	
		2. Das System zeigt Objekte an, auf welchen CRUD Operationen ausgeführt werden können.
		3. Das System zeigt mögliche CRUD Operationen an
	4. Der Benutzer führt eine CRUD Operation aus.	
		5. Das System gibt dem Benutzer ein Feedback zur ausgeführten CRUD Operation

Extensions	*a. Zu jeder Zeit wenn der Benutzer abbricht: 1. Das System führt keine Aktionen durch 2. Das System verwirft die Änderungen und behält die bestehende Konfiguration bei *b. Zu jeder Zeit wenn das System fehlschlägt: 1. Das System informiert den Benutzer in Form einer textuellen Beschreibung 2. Das System loggt den Fehler 4a. Der vom Benutzer hinzugefügte oder aktualisierte Filterlisteeintrag ist ungültig: 1. Das System informiert den Benutzer in Form einer textuellen Beschreibung
Frequency of Occurence	10x pro Woche
Spezielle nicht funktionale Anforderungen	-

Tabelle 5 - UC 4 Filterliste CRUD

7.3.7 UC 5 NDR Pattern CRUD

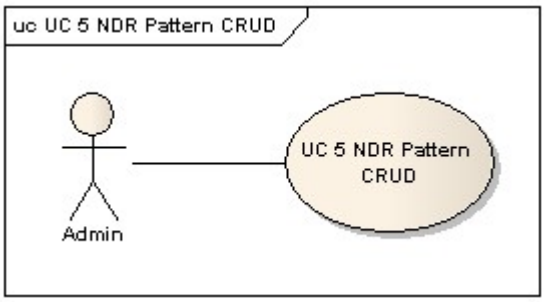


Abbildung 11 - UC 5 NDR Pattern CRUD

Name	NDR Pattern CRUD	
Scope	*Smtp Mailer* – NDR Modul und Web Modul	
Level	User Goal	
Primary Actor	Admin	
Stakeholder and Interests	Der Benutzer möchte ein NDR Pattern für die Erkennung der NDR Nachrichten erstellen, abfragen, aktualisieren oder löschen.	
Preconditions	-	
Success Guarantee	Die Konfiguration wurde gespeichert und wird per sofort verwendet	
Main Success Scenario	***User Intention***	***System Response***
	1. Der Benutzer möchte ein NDR Pattern erstellen, abfragen, aktualisieren oder löschen	
		2. Das System zeigt Objekte an, auf welchen CRUD Operationen ausgeführt werden können.
		3. Das System zeigt mögliche CRUD Operationen an
	4. Der Benutzer führt eine CRUD Operation aus	
		5. Das System gibt dem Benutzer ein Feedback zur ausgeführten CRUD Operation
Extensions	*a. Zu jeder Zeit wenn der Benutzer abbricht: 1. Das System führt keine Aktionen durch *b. Zu jeder Zeit wenn das System fehlschlägt: 1. Das System informiert den Benutzer in Form einer textuellen Beschreibung 2. Das System loggt den Fehler 4a. Das vom Benutzer hinzugefügte oder aktualisierte NDR Pattern ist ungültig: 1. Das System informiert den Benutzer in Form einer textuellen Beschreibung	
Frequency of Occurence	2x pro Monat	
Spezielle nicht funktionale Anforderungen	-	

Tabelle 6 - UC 5 NDR Pattern CRUD

7.3.8 UC 6 Qualitätsindikator abfragen

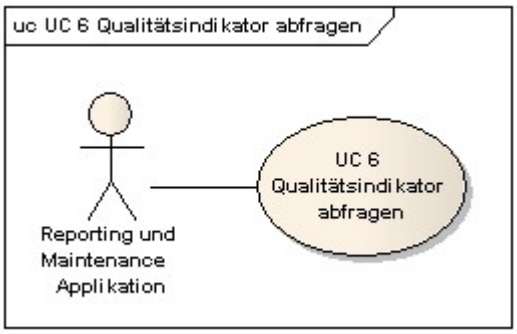


Abbildung 12 - UC 6 Qualitätsindikator abfragen

Name	Qualitätsindikator abfragen	
Scope	*Smtp Mailer* – Mail Modul	
Level	User Goal	
Primary Actor	Reporting und Maintenance Applikation	
Stakeholder and Interests	Der Benutzer möchte den Qualitätsindikator einer E-Mail Adresse abfragen	
Preconditions	Die abzufragende E-Mail Adresse besteht im System	
Success Guarantee	Der Qualitätsindikator für die abgefragte E-Mail Adresse wird dem Benutzer vom System geliefert	
Main Success Scenario	***User Intention***	***System Response***
	1. Der Benutzer möchte für eine beliebige E-Mail Adresse den Qualitätsindikator abfragen 2. Der Benutzer gibt die E-Mail Adresse an	3. Das System zeigt den Qualitätsindikator für die vom Benutzer angegebene E-Mail Adresse an 4. Das System zeigt zur Begründung des Qualitätsindikators eine History an.
Extensions	*a. Zu jeder Zeit wenn der Benutzer abbricht: 1. Das System führt keine Aktionen durch *b. Zu jeder Zeit wenn das System fehlschlägt: 1. Das System wirft eine Exception 2. Das System loggt den Fehler 2a. Die E-Mail Adresse ist ungültig: 1. Das System wirft eine Exception	
Frequency of Occurence	10x pro Woche	
Spezielle nicht funktionale Anforderungen	-	

Tabelle 7 - UC 6 Qualitätsindikator abfragen

7.3.9 UC 7 Report abfragen

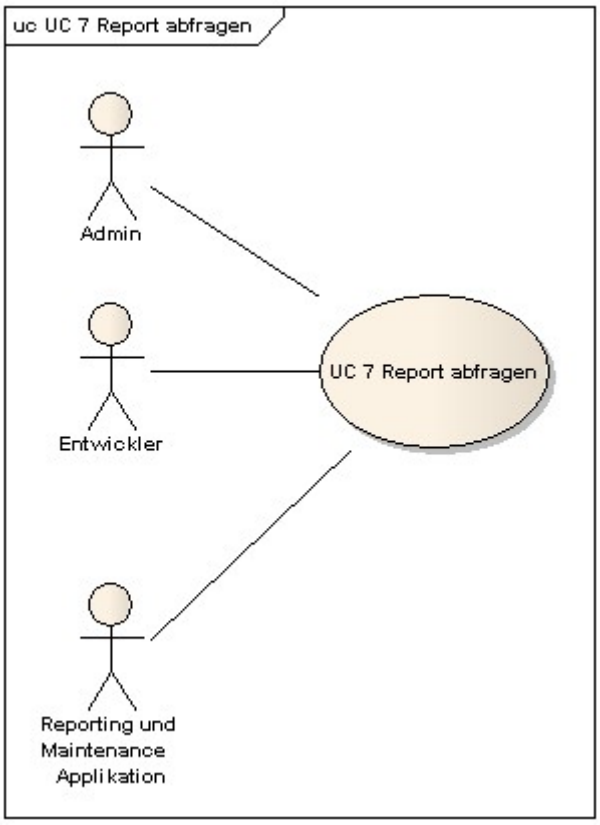


Abbildung 13 - UC 7 Report abfragen

Name	Report abfragen	
Scope	*Smtp Mailer* – Mail Modul und Web Modul	
Level	User Goal	
Primary Actor	Admin, Entwickler und Reporting und Maintenance Applikation	
Stakeholder and Interests	Der Benutzer möchte einen Report generieren und anzeigen lassen.	
Preconditions	-	
Success Guarantee	Dem Benutzer wird der gewünschte Report vom System geliefert.	
Main Success Scenario	***User Intention***	***System Response***
	1. Der Benutzer möchte einen Report anzeigen lassen	
		2. Das System zeigt die verfügbaren Reports an
	3. Der Benutzer wählt einen Report aus und gibt dem System die Suchkriterien an	
		4. Das System stellt den Report dar

Extensions	*a. Zu jeder Zeit wenn der Benutzer abbricht: 1. Das System führt keine Aktionen durch *b. Zu jeder Zeit wenn das System fehlschlägt: 1. Das System informiert den Benutzer in einer sinnvollen Form (Textuelle Beschreibung des Fehlers bei menschlichen Benutzern, Exception bei API Benutzer) 2. Das System loggt den Fehler 3a. Die vom Benutzer angegebenen Suchkriterien sind ungültig: 1. Das System informiert den Benutzer 4a. Es sind keine den Suchkriterien entsprechenden Resultate vorhanden: 1. Das System informiert den Benutzer in einer sinnvollen Form (Textuelle Beschreibung des Fehlers bei menschlichen Benutzern, Exception bei API Benutzer)
Frequency of Occurence	100x pro Woche
Spezielle nicht funktionale Anforderungen	-

Tabelle 8 - UC 7 Report abfragen

7.3.10 UC 8 Datenbank bereinigen

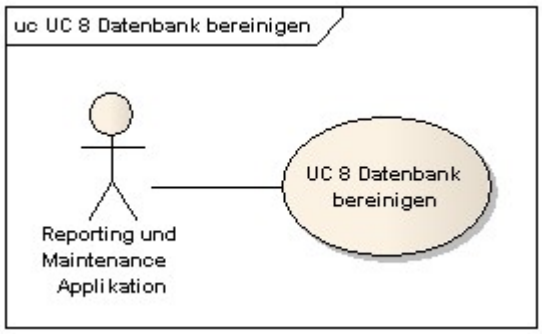


Abbildung 14 - UC 8 Datenbank bereinigen

Name	Datenbank bereinigen	
Scope	*Smtp Mailer* – Mail Modul	
Level	User Goal	
Primary Actor	Reporting und Maintenance Applikation	
Stakeholder and Interests	Der Benutzer möchte eine Datenbankbereinigung durchführen	
Preconditions	Das System verfügt über Nachrichten, die gemäss CleanupRule gelöscht werden müssen (Content und Attachements)	
Success Guarantee	Der Content und die Attachements aller Nachrichten wurde gelöscht, welche gemäss CleanupRule gelöscht werden müssen	
Main Success Scenario	***User Intention***	***System Response***
	1. Der Benutzer möchte eine Datenbankbereinigung durchführen	2. Das System löscht den Content und die Attachements aller Nachrichten, welche gemäss CleanupRule gelöscht werden müssen
Extensions	*a. Zu jeder Zeit wenn der Benutzer abbricht: 1. Das System führt keine Aktionen durch *b. Zu jeder Zeit wenn das System fehlschlägt: 1. Das System wirft eine Exception 2. Das System loggt den Fehler	
Frequency of Occurence	1x pro Tag	
Spezielle nicht funktionale Anforderungen	-	

Tabelle 9 - UC 8 Datenbank bereinigen

7.3.11 UC 9 CleanupRule CRUD

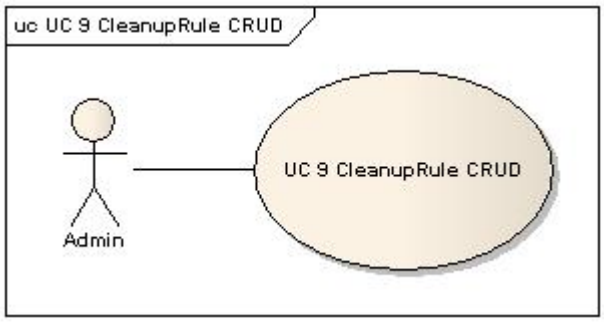


Abbildung 15 - UC 9 CleanupRule CRUD

Name	CleanupRule CRUD	
Scope	*Smtp Mailer* – Mail Modul und Web Modul	
Level	User Goal	
Primary Actor	Admin	
Stakeholder and Interests	Der Benutzer möchte eine CleanupRule (nach welcher Zeit dass der Content und die Attachements eines Applikationscodes / Mailtyps gelöscht werden) erstellen, abfragen, aktualisieren oder löschen.	
Preconditions	-	
Success Guarantee	Die Konfiguration wurde gespeichert und wird per sofort verwendet	
Main Success Scenario	***User Intention***	***System Response***
	1. Der Benutzer möchte ein CleanupRule erstellen, abfragen, aktualisieren oder löschen	
		2. Das System zeigt Objekte an, auf welchen CRUD Operationen ausgeführt werden können.
		3. Das System zeigt mögliche CRUD Operationen an
	4. Der Benutzer führt eine CRUD Operation aus	
		5. Das System gibt dem Benutzer ein Feedback zur ausgeführten CRUD Operation

Extensions	*a. Zu jeder Zeit wenn der Benutzer abbricht: 1. Das System führt keine Aktionen durch *b. Zu jeder Zeit wenn das System fehlschlägt: 1. Das System informiert den Benutzer in Form einer textuellen Beschreibung 2. Das System loggt den Fehler 4a. Die vom Benutzer hinzugefügte oder aktualisierte CleanupRule ist ungültig: 1. Das System informiert den Benutzer in Form einer textuellen Beschreibung
Frequency of Occurence	2x pro Monat
Spezielle nicht funktionale Anforderungen	-

Tabelle 10 - UC 9 CleanupRule CRUD

7.3.12 UC 10 NDR Nachricht verarbeiten

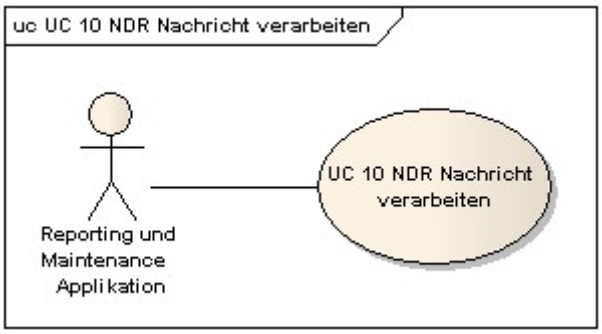


Abbildung 16 - UC 10 NDR Nachricht verarbeiten

Name	NDR Nachricht verarbeiten	
Scope	*Smtp Mailer* – NDR Modul	
Level	User Goal	
Primary Actor	Reporting und Maintenance Applikation	
Stakeholder and Interests	Der Benutzer möchte eine NDR Nachricht verarbeiten, damit der Grund für den Fehler einer Nachrichtenversendung erfasst werden kann.	
Preconditions	-	
Success Guarantee	Alle verfügbaren NDR Nachrichten sind verarbeitet und der Grund für den Fehler einer Nachrichtenversendung im System erfasst	
Main Success Scenario	***User Intention***	***System Response***
	1. Der Benutzer möchte die NDR Nachrichten verarbeiten	2. Das System verarbeitet die NDR Nachrichten und speichert den Grund für den Fehler einer Nachrichtenversendung ab
Extensions	*a. Zu jeder Zeit wenn der Benutzer abbricht: 1. Das System führt keine Aktionen durch *b. Zu jeder Zeit wenn das System fehlschlägt: 1. Das System wirft eine Exception 2. Das System loggt den Fehler	
Frequency of Occurence	30‘000x pro Stunde	
Spezielle nicht funktionale Anforderungen	-	

Tabelle 11 - UC10 NDR Nachricht verarbeiten

7.4 Webapplikation

Bei der Entwicklung der Webapplikation wird ein User Centered Design (UCD) angewendet. Das heisst, dass der Endbenutzer der Webapplikation in den Design Prozess mit einbezogen wird.

Anhand eines Contextual Inquirys wird versucht, den Endbenutzer und seine Arbeitsabläufe zu verstehen und seine Bedürfnisse sowie Arbeitsweisen und Tätigkeiten zu erkennen. So kann ein fundiertes Verständnis für die späteren Benutzer erarbeitet werden. Mit diesem Wissen wird die Benutzeroberfläche den Ansprüchen angepasst.

Der Artikel (Ji-Ye Mao, 2002) zeigt auf, dass das Contextual Inquiry resp. die Feldstudie beim Kunden als sehr wichtig angesehen wird und eine der am häufigsten eingesetzten UCD Methoden ist.

Während den Benutzerbeobachtungen werden Daten vom Endbenutzer gesammelt um daraus eine Persona abzuleiten. Eine Persona ist eine fiktive aber konkret beschriebene Person, welche einen möglichen Endbenutzer verkörpert. Es werden die Ziele und Verhaltensweisen eines möglichen Endbenutzers dokumentiert.

Szenarien beschreiben, wie eine Persona mit dem zu implementierenden, geplanten System zusammenarbeitet und interagiert, um ihre Ziele zu erreichen.

Mittels eines Paper Prototypes, welcher an einem realen Endbenutzer getestet wird, kann festgestellt werden, ob der Endbenutzer die Benutzeroberfläche versteht, ob die Beschriftungen von UI Elementen und UI Übergängen klar sind sowie ob er die Arbeitsabläufe durch die Masken optimal findet. Man erhält also ein Feedback vom Endbenutzer, bevor der Code geschrieben ist. So können Probleme schon vor der Entwicklung der Benutzeroberfläche gelöst werden.

7.4.1 Contextual Inquiry

Am 06.05.2009 wurde bei der Firma comparis.ch AG ein Contextual Inquiry sowie ein Paper Prototype Test mit einem Kundendienst Mitarbeiter (Sven Häfliger) durchgeführt.

Es wird vor Ort beim Kunden ein möglicher Benutzer beobachtet und bei Unklarheiten befragt.

Mögliche Themen zur Klärung sind folgende (nicht abschliessend)

- Wie häufig treten die mit dem Paper Prototype getesteten Szenarien auf?
  Das Szenario Nachricht abfragen findet sehr häufig statt (mehrmals täglich), das Szenario Blacklist konfigurieren tritt eher selten auf (bei 98% dieser Anfragen wird der Kunde aus dem Newslettersystem ausgetragen und nicht auf die Blacklist gesetzt).
- Welche Ausbildung besitzen Kundendienstmitarbeiter?
  Typischerweise eine kaufmännische Ausbildung. Sehr gute Kenntnisse mit den Office-Produkten von Microsoft (vor allem Word und Outlook) werden vorausgesetzt.
- Mit welchen Tools arbeitet ein Kundendienstmitarbeiter? Welche Formulare/Werkzeuge werden im Kundendienst benutzt?
  Outlook wird sehr häufig eingesetzt.
- Wie sucht ein Kundendienstmitarbeiter im Outlook nach einer E-Mail Nachricht?
  Meistens durch Sortieren der entsprechenden Spalten.
- In was für einer Umgebung arbeitet ein Kundendienstmitarbeiter?
  In einer ruhigen Umgebung. Jedoch muss sich ein Kundendienstmitarbeiter auch zum Teil um den Empfang von Gästen kümmern.
- Welche Hilfsmittel besitzt ein Kundendienstmitarbeiter?
  Telefon, Computer, Headset
- Wie gut kennt sich ein Kundendienstmitarbeiter mit dem Webbrowser und dessen Bedienung aus?

Sehr gute Kenntnisse. Auch privat wird das Web häufig genutzt.

- Sind in gewissen Situationen Symbole/Icons gerne gesehen oder wird Text bevorzugt?
  Symbole werden sofern der Sinn erkannt wird bevorzugt.
- Was für Rollen gibt es im Kundendienst? Macht jeder Kundendienst Mitarbeiter alle Tätigkeiten oder gibt es eine Aufteilung?
  Es findet eine Rotation statt bei der Kundenberatung und beim Empfang von Gästen.
- Welche Ausnahmesituationen gibt es bei den getesteten Szenarien?
  Sowohl beim Szenario Nachricht abfragen als auch Blacklist konfigurieren: Der anrufende Benutzer weiss weder seine E-Mail Adresse, um welche es geht, noch seine Benutzer-Id. Zudem hat er seinen Vornamen und Nachnamen in seinem Benutzerprofil nicht erfasst. Somit ist der Benutzer für den Kundendienstmitarbeiter nicht auffindbar. Der Kundendienstmitarbeiter muss über ein weiteres Tool die Kontaktdaten herausfinden.
- Gibt es irgendwelche Wünsche vom Kundendienst Mitarbeiter?
  Alles muss möglichst klar sein. Es dürfen keine Abkürzungen oder IT-spezifische Fachbegriffe in der Webapplikation zu finden sein.
- Zitate und Aussagen vom Kundendienst Mitarbeiter:
  Zitat 1: „Die E-Mail Adresse des Kunden ist das wichtigste. Nach der E-Mail Adresse werden sämtliche Prozesse in der Webapplikation ausgerichtet“
  Zitat 2: „Abkürzungen haben für mich meist keine Bedeutung“
  Zitat 3: „Im Outlook suche ich nach einer E-Mail Nachricht, in dem ich die entsprechenden Spalten sortiert darstellen lasse und dann zu der gesuchten Nachricht scrolle oder indem ich die eingebaute Suche nutze.“

7.4.2 Persona Daniel Müller

7.4.2.1 Beschreibung

Daniel Müller ist 30 Jahre alt, verheiratet und hat eine dreijährige Tochter. Daniel Müller ist im Kundendienst bei comparis.ch AG tätig. Zuvor arbeitete Daniel Müller drei Jahre im Backoffice Bereich. Vor zwei Jahren hat Daniel Müller in den Kundendienst gewechselt, weil er sich mehr Kundenkontakt wünschte. Daniel Müller absolvierte eine kaufmännische Lehre. Er kennt sich gut mit den Office Produkten von Microsoft aus. Auch im Umgang mit Webbrowsern hat Daniel Müller gute Kenntnisse. So surft er privat regelmässig im Netz und muss geschäftlich gewisse Dinge im Intranet oder Internet erledigen. Es ist ihm sehr wichtig, den Kunden ganzheitlich und nach einem systematischen Prozess beraten zu können. Bei einer Beratung laufen die meisten Prozesse über die E-Mail Adresse. Die E-Mail Adresse ist das zentrale Identifikationsmerkmal eines Kunden für den Kundendienst.

„Die E-Mail Adresse des Kunden ist das wichtigste. Nach der E-Mail Adresse werden sämtliche Prozesse in der Webapplikation ausgerichtet“

Für komplexere Arbeitsabläufe und Prozesse muss sich Daniel Müller dauernd Notizen machen, sonst kommt er schon bald nicht mehr klar. Wenn ihm ein Arbeitsmittel als zu kompliziert erscheint, sucht er sich eine Alternative oder tätigt die Arbeiten wenn möglich manuell. Auch mit Abkürzungen in Begriffen auf der Webapplikation kann Daniel Müller wenig anfangen.

„Abkürzungen haben für mich meist keine Bedeutung“

Daniel Müller erscheint es als wichtig, dass die Suchmasken der Webapplikation ähnlich funktionieren wie die Suche im Outlook. Denn Outlook ist das wichtigste Tool im Kundendienst. Zum Suchen von E-Mail Nachrichten sortiert er meist die Spalten. So kann er schnell zur gesuchten E-Mail Nachricht navigieren. Ebenfalls nutzt er häufig die eingebaute Suche von Outlook.

„Im Outlook suche ich nach einer E-Mail Nachricht, in dem ich die entsprechenden Spalten sortiert darstellen lasse und dann zu der gesuchten Nachricht scrolle.“

In der Freizeit treibt Daniel Müller viel Sport, so kommt er im Sommer gerne mit dem Fahrrad zur Arbeit.

7.4.2.2 Behavioral Variables

Merkmal	Pole	
Weberfahrung	Gering	Hoch
Risikobereitschaft	Gering	Hoch
Wartetoleranz	Ungeduldig	Geduldig
Suchstil	Link-orientiert	Suchhilfen-orientiert
Handlungsziel	Nicht vorhanden	Genau definiert
Technisches Wissen	Laie	Experte
Wissen über dargestellte Inhalte	Laie	Experte
Nutzungsfrequenz	Selten	Häufig

Tabelle 12 - Behavioral Variables von Persona Daniel Müller

Weberfahrung

Risikobereitschaft

Wartetoleranz

Suchstil

Handlungsziel

Technisches Wissen

Wissen über dargestellte Inhalte

Nutzungsfrequenz

7.4.3 Szenarien

7.4.3.1 Szenario Nachricht abfragen

Es ist 10 Uhr am 28.04.2009. Ein verzweifelter Kunde meldet sich beim Kundendienst der comparis.ch AG. Daniel Müller kümmert sich um den Kunden.

Der Kunde schildert Daniel Müller, dass er das Passwort für den Benutzerbereich der comparis.ch Webplattform vergessen habe. Er habe soeben das Passwort neu angefordert. Jedoch erhalte er das E-Mail mit dem neu generierten Passwort nicht.

Daniel Müller fragt nach dem Namen und der E-Mail Adresse des Kunden. Der Kunde antwortet, dass er Florian Sprunger heisse und seine E-Mail Adresse fsprunger@migros.ch laute. Als erstes fragt Daniel Müller alle E-Mail Nachrichten am Tag des 28.04.2009 ab, welche an Florian Sprunger gesendet wurden. Das System findet keine Nachrichten. Anscheinend hat der Kunde seinen Vornamen und Nachnamen nicht in seinem Profil erfasst. Daniel Müller fragt nun alle E-Mail Nachrichten mittels der Webapplikation ab, welche an die genannte E-Mail Adresse am Tag des 28.04.2009 gesendet wurden. Nun liefert das System einen Treffer. Daniel Müller sieht, dass die zuletzt versendete Nachricht nicht versendet werden konnte, weil der Benutzer ein volles Postfach hat. Daniel Müller gibt dem Kunden die Anweisung, dass er zuerst das Postfach leeren und danach das Passwort nochmal anfordern soll.

7.4.3.2 Szenario Blacklist konfigurieren

Es ist 17 Uhr. Beim Kundendienst meldet sich ein verärgerter Kunde. Daniel Müller erfährt vom Kunden, dass er immer noch Newsletter von comparis.ch erhalte, obwohl er diese Newsletter letzte Woche alle abgemeldet habe.

Daniel Müller fragt nach der E-Mail Adresse des Kunden. Der Kunde weiss zuerst nicht, wie seine E-Mail Adresse lautet, auf welche er die Newsletter erhält. Also fragt Daniel Müller nach dem Namen des Kunden. Dieser antwortet mit dem Namen Florian Sprunger. Daniel Müller findet die E-Mail Adresse von Florian Sprunger (fsprunger@migros.ch) über ein anderes System von comparis.ch heraus. Diese E-Mail Adresse wird daraufhin von Daniel Müller über die Webapplikation in die Blacklist eingetragen. So wird der Benutzer keine E-Mails mehr von comparis.ch erhalten.

Daniel Müller entschuldigt sich beim Kunden für die Unannehmlichkeiten und versichert ihm, dass nun keine Newsletter mehr an ihn verschickt werden.

8 Analyse

8.1 Domain Modell

8.1.1 Strukturdiagramm

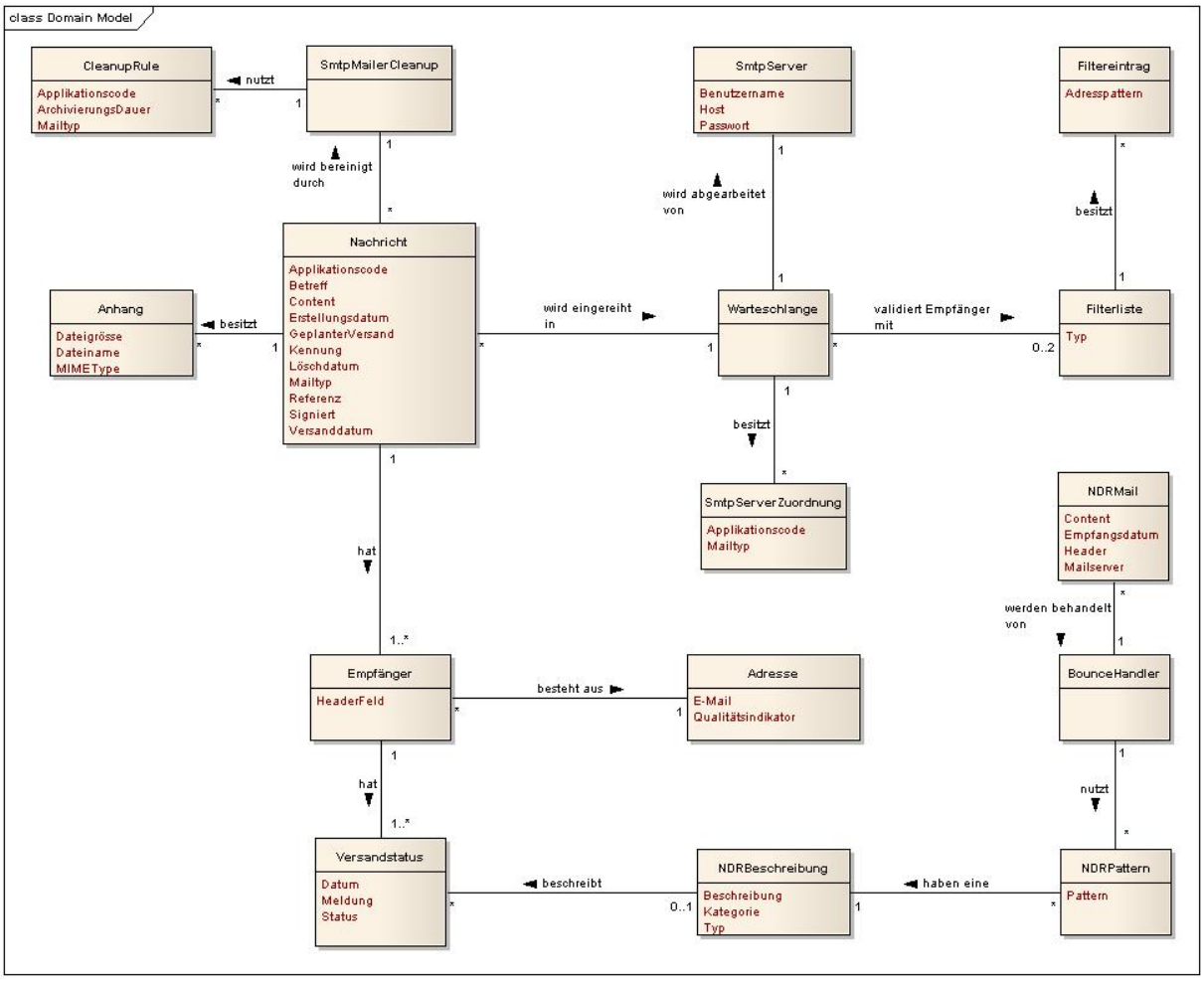


Abbildung 17 - Domain Modell

8.1.2 Konzeptbeschreibung

Nachricht		
Beschreibung	Stellt die zu versendende E-Mail Nachricht dar	
Attribute	Applikationscode	Kurzzeichen der Applikation, welche die Nachricht versendet
	Betreff	Betreffzeile der Nachricht
	Content	Inhalt der Nachricht
	Erstellungsdatum	Datum und Uhrzeit der Generierung der Nachricht
	GeplanterVersand	Datum und Uhrzeit, wann die Nachricht frühestens versendet werden darf (nicht früher als)
	Kennung	Eindeutiger Schlüssel um auf die Nachricht zu verweisen

	Löschdatum	Datum und Uhrzeit der Löschung des Content und der Anhänge
	Mailtyp	Kennzeichnet die Art der Nachricht (z.B. Newsletter)
	Referenz	Interne Referenz von comparis.ch
	Signiert	Beschreibt, ob die Nachricht mit einer Signatur versehen ist
	Versanddatum	Datum und Uhrzeit wann die Nachricht versendet wurde
Beziehungen	Empfänger	Eine Nachricht wird an mindestens einen Empfänger versendet
	Anhang	Eine Nachricht kann Anhänge besitzen
	SmtpMailerCleanup	Eine Nachricht wird durch eine Datenbankbereinigung bereinigt
	Warteschlange	Eine Nachricht wird in eine Warteschlange eingereiht. So wird sie zum Versand freigegeben

Tabelle 13 - Konzept Nachricht

Empfänger		
Beschreibung	Stellt einen Nachrichtenempfänger dar	
Attribute	HeaderFeld	Definiert, ob der Empfänger auf der TO, CC oder BBC Liste steht
Beziehungen	Nachricht	An einen oder mehrere Empfänger wird eine Nachricht gesendet
	Adresse	Ein Empfänger besitzt eine Adresse
	Versandstatus	Ein Empfänger besitzt für eine Nachricht einen oder mehrere Versandstatus (History)

Tabelle 14 - Konzept Empfänger

Adresse		
Beschreibung	Stellt eine E-Mail Adresse dar	
Attribute	E-Mail	E-Mail Adresse des Empfängers
	Qualitätsindikator	Klassierung der E-Mail Adresse in good, poor, failed
Beziehungen	Empfänger	Eine Adresse beschreibt den Empfänger

Tabelle 15 - Konzept Adresse

Versandstatus		
Beschreibung	Beschreibt den Status der Nachrichtenversendung	
Attribute	Datum	Datum und Uhrzeit des Versandstatuseintrages
	Meldung	Textuelle Beschreibung des Versandstatus (Rückmeldung, Fehlermeldung)
	Status	Klassierung des Status (nicht versendet, versendet, blacklisted, failed)
Beziehungen	Empfänger	Ein Versandstatus beschreibt den Status der Nachrichtenversendung für einen Empfänger für eine Nachricht
	NDRBeschreibung	Ein Versandstatus kann einer NDR Beschreibung zugeordnet sein (im Fehlerfall)

Tabelle 16 - Konzept Versandstatus

NDRBeschreibung		
Beschreibung	Beschreibt einen Non Delivery Report, also eine Fehlermeldung beim Versenden einer Nachricht	
Attribute	Beschreibung	Beschreibung der Fehlermeldung
	Kategorie	Kategorie der Fehlermeldung
	Typ	Fehlertyp
Beziehungen	Versandstatus	Eine NDRBeschreibung beschreibt den aufgetretenen Fehler beim Versenden einer Nachricht
	NDRPattern	Eine NDRBeschreibung beschreibt den Fehler, welcher durch ein oder mehrere NDRPattern gefunden wurde

Tabelle 17 - Konzept NDRBeschreibung

SmtpServer		
Beschreibung	Beschreibt einen SMTP Server (Der SMTP Server versendet die E-Mail Nachrichten)	
Attribute	Benutzername	Zur Authentifizierung am SMTP Servers
	Host	Host Adresse des SMTP Servers
	Passwort	Zur Authentifizierung am SMTP Servers
Beziehungen	Warteschlange	Ein SMTP Server besitzt eine Warteschlange, welche er abarbeitet

Tabelle 18 - Konzept SmtpServer

Anhang		
Beschreibung	Beschreibt ein Nachrichtenanhang (z.B. File)	
Attribute	Dateigrösse	Grösse des Anhangs
	Dateiname	Dateiname des Anhangs
	MIMEType	MIME Type des Anhangs
Beziehungen	Nachricht	Ein Anhang wird an eine Nachricht angefügt

Tabelle 19 - Konzept Anhang

Filterliste		
Beschreibung	Beschreibt eine Liste von Adresspatterns, welche entweder für die Whitelist oder die Blacklist gelten	
Attribute	Typ	Definiert, ob es sich bei der Filterliste um die Whitelist oder die Blacklist handelt
Beziehungen	Filtereintrag	Eine Filterliste besitzt mehrere Filtereinträge
	Warteschlange	Eine Filterliste bietet einer Warteschlange Validierungsfunktionen für Empfängeradressen an

Tabelle 20 - Konzept Filterliste

Filtereintrag		
Beschreibung	Beschreibt eine E-Mail Adresse oder eine Domain, an welche keine E-Mail Nachrichten versendet werden dürfen (im Falle der Blacklist). Im Falle der Whitelist dürfen nur an solche E-Mail Adressen oder Domains Nachrichten versendet werden.	
Attribute	Adresspattern	Definiert eine E-Mail Adresse oder eine Domain.
Beziehungen	Filterliste	Ein Filtereintrag befindet sich in einer Filterliste

Tabelle 21 - Konzept Filtereintrag

NDRPattern		
Beschreibung	Beschreibt ein Erkennungspattern für die Behandlung und Kategorisierung von Bounce-Messages	
Attribute	Pattern	Definiert ein Muster für die Erkennung und Kategorisierung von Bounce-Messages
Beziehungen	NDRBeschreibung	Ein NDRPattern besitzt eine NDRBeschreibung
	BounceHandler	Ein NDRPattern dient einem BounceHandler für die Erkennung und Kategorisierung von Bounce-Messages

Tabelle 22 - Konzept NDRPattern

BounceHandler		
Beschreibung	Wertet die NDRMails (Bounce-Messages) aus	
Attribute	-	
Beziehungen	NDRPattern	Benutzt die NDRPatterns für die Erkennung und Kategorisierung von Bounce-Messages
	NDRMail	Der BounceHandler wertet die NDRMails aus

Tabelle 23 - Konzept BounceHandler

NDRMail		
Beschreibung	Beschreibt eine Bounce-Message, welche als Fehlermeldung auf einen fehlgeschlagenen Versendungsversuch empfangen wird	
Attribute	Content	Inhalt der Nachricht
	Empfangsdatum	Datum und Uhrzeit wann die Nachricht empfangen wurde
	Header	Header der Nachricht
	Mailserver	Mailserver, welcher die NDRMail verschickt hat (Absender der NDRMail)
Beziehungen	BounceHandler	Ein NDRMail wird von einem BounceHandler behandelt

Tabelle 24 - Konzept NDRMail

Warteschlange		
Beschreibung	Beschreibt eine Liste, in welche die Nachrichten eingereiht werden, damit sie vom SmtpServer versendet werden können	
Attribute	-	
Beziehungen	SmtpServer	Eine Warteschlange wird durch einen SmtpServer abgearbeitet
	Filterliste	Eine Warteschlange validiert die Empfänger mittels Filterlisten
	SmtpServerZuordnung	Eine SmtpServerZuordnung teilt mit, welche Nachricht in welche Warteschlange eingereiht wird
	Nachricht	Eine Warteschlange nimmt Nachrichten auf, damit sie vom SmtpServer versendet werden können

Tabelle 25 - Konzept Warteschlange

SmtpServerZuordnung		
Beschreibung	Eine SmtpServerZuordnung definiert, welche Nachricht in welche Warteschlange eingereiht wird	
Attribute	Applikationscode	Kurzzeichen der Applikation, welche die Nachricht versendet
	Mailtyp	Kennzeichnet die Art der Nachricht (z.B. Newsletter)
Beziehungen	Warteschlange	Eine SmtpServerZuordnung definiert, in welche Warteschlange Nachrichten mit gewissen Applikationscodes / Mailtypen kommen

Tabelle 26 - Konzept SmtpServerZuordnung

SmtpMailerCleanup		
Beschreibung	Führt eine Datenbankbereinigung durch, indem der Content und die Anhänge von Nachrichten gelöscht werden	
Attribute	-	
Beziehungen	Nachricht	Die Datenbankbereinigung bereinigt die Nachrichten
	CleanupRule	Die Datenbankbereinigung bereinigt die Nachrichten aufgrund von CleanupRules

Tabelle 27 - Konzept SmtpMailerCleanup

CleanupRule		
Beschreibung	Eine CleanupRule definiert, welche Nachrichten nach welcher Zeit gelöscht werden müssen (Content, Anhänge)	
Attribute	Applikationscode	Kurzzeichen der Applikation, welche die Nachricht versendet
	ArchivierungsDauer	Definiert die Dauer in Tagen, wie lange für eine Nachricht der Content und Anhänge gespeichert werden müssen
	Mailtyp	Kennzeichnet die Art der Nachricht (z.B. Newsletter)
Beziehungen	SmtpMailerCleanup	Die CleanupRule definiert, nach welcher Dauer in Tagen die Datenbankbereinigung Nachrichten eines gewissen Applikationscodes / Mailtyps bereinigen muss

Tabelle 28 - Konzept CleanupRule

8.2 System Sequenzdiagramme

Die nachfolgenden Sequenzdiagramme zeigen die Use Case Abläufe auf einer konzeptuellen Ebene.

8.2.1 SSD UC 1 Nachricht versenden

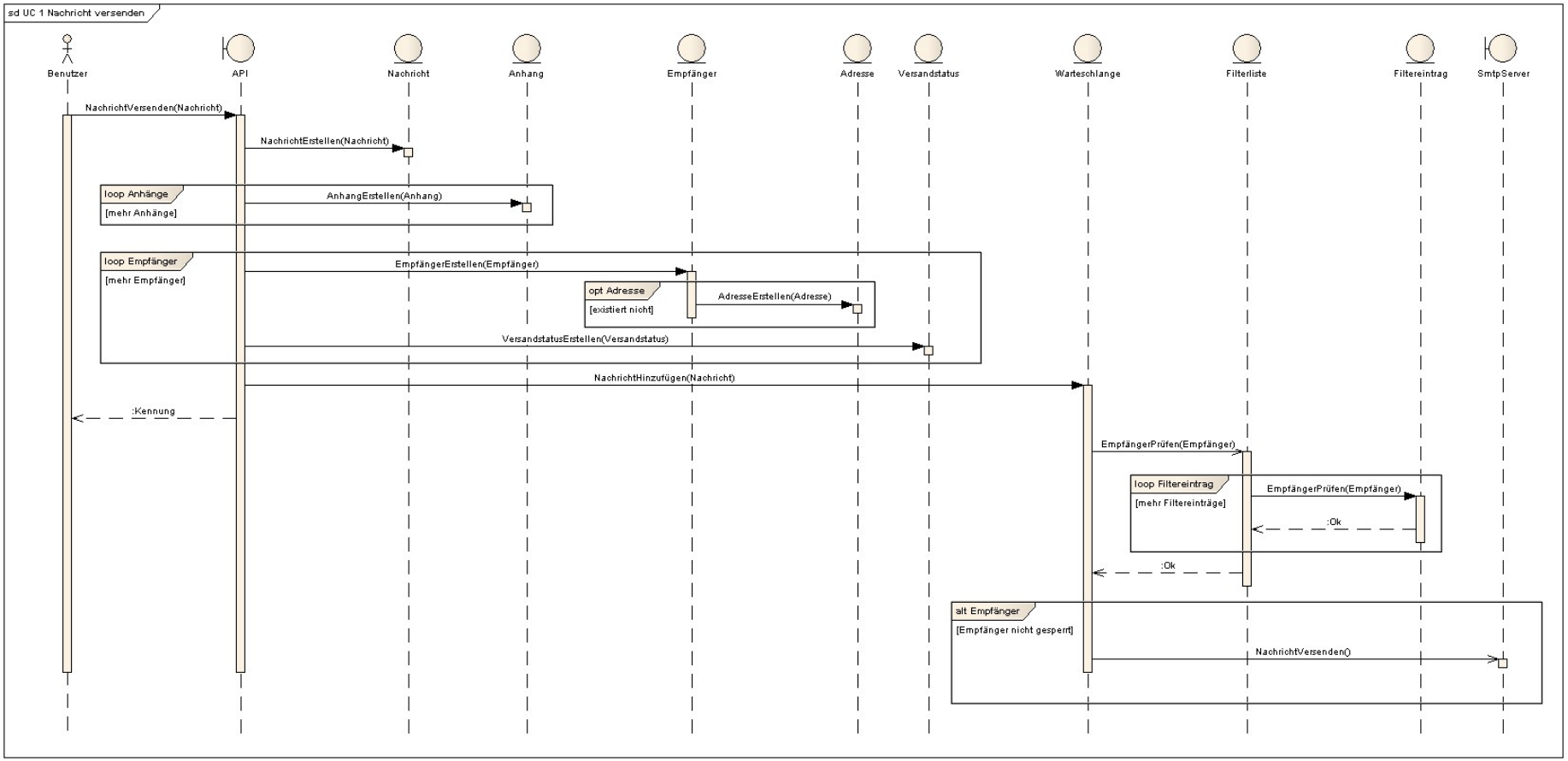


Abbildung 18 - Sequenzdiagramm detailliert - UC 1 Nachricht versenden

8.2.2 SSD UC 2 Versandstatus abfragen

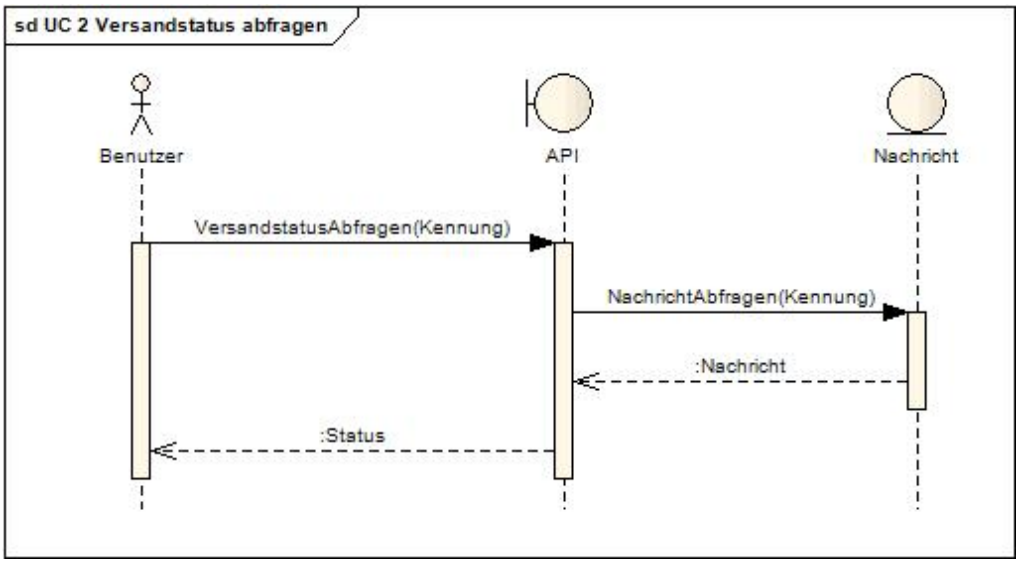


Abbildung 19 - Sequenzdiagramm detailliert - UC 2 Versandstatus abfragen

8.2.3 SSD UC 3 Nachricht abfragen

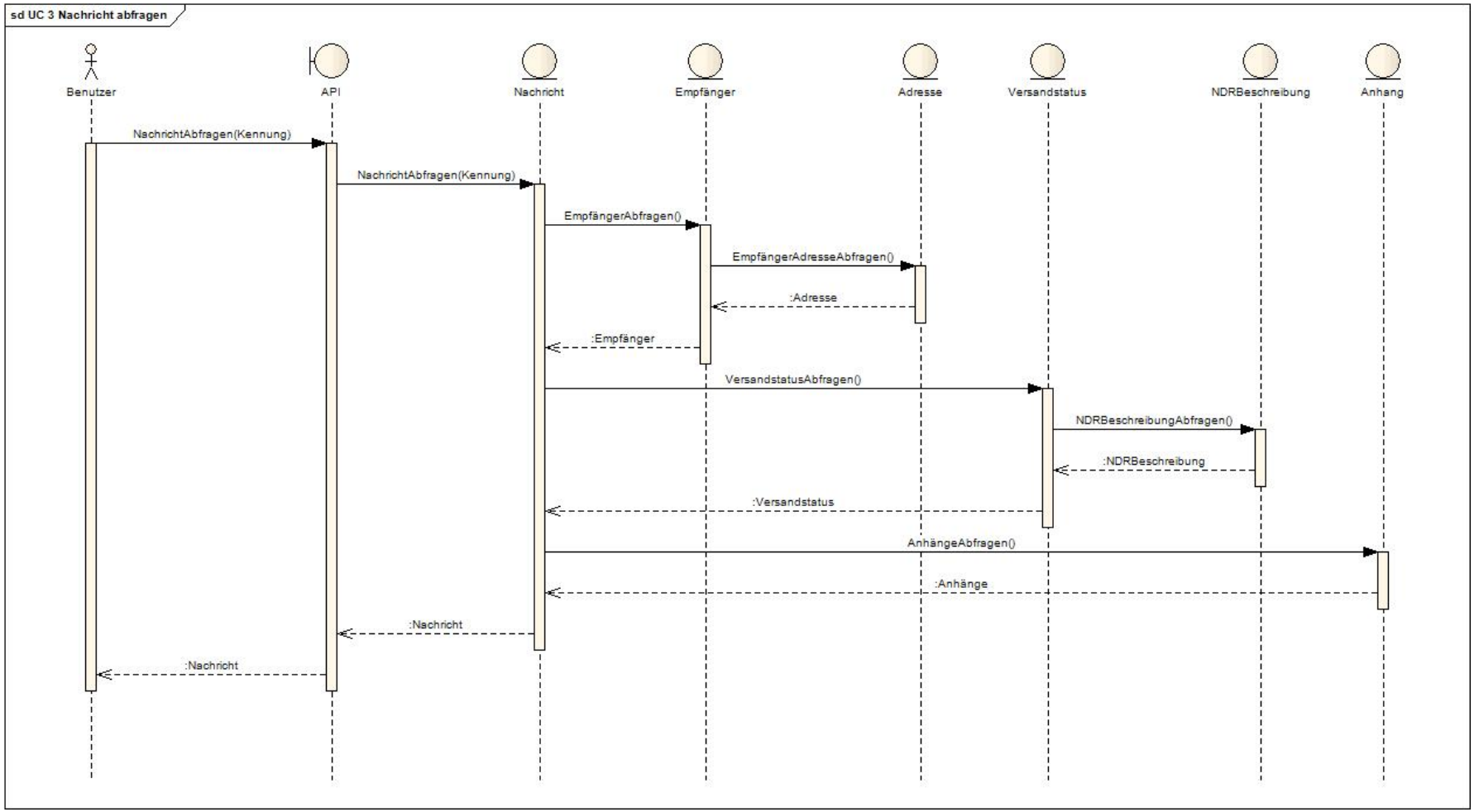


Abbildung 20 - Sequenzdiagramm detailliert - UC 3 Nachricht abfragen

8.2.4 SSD UC 4 Filterliste CRUD

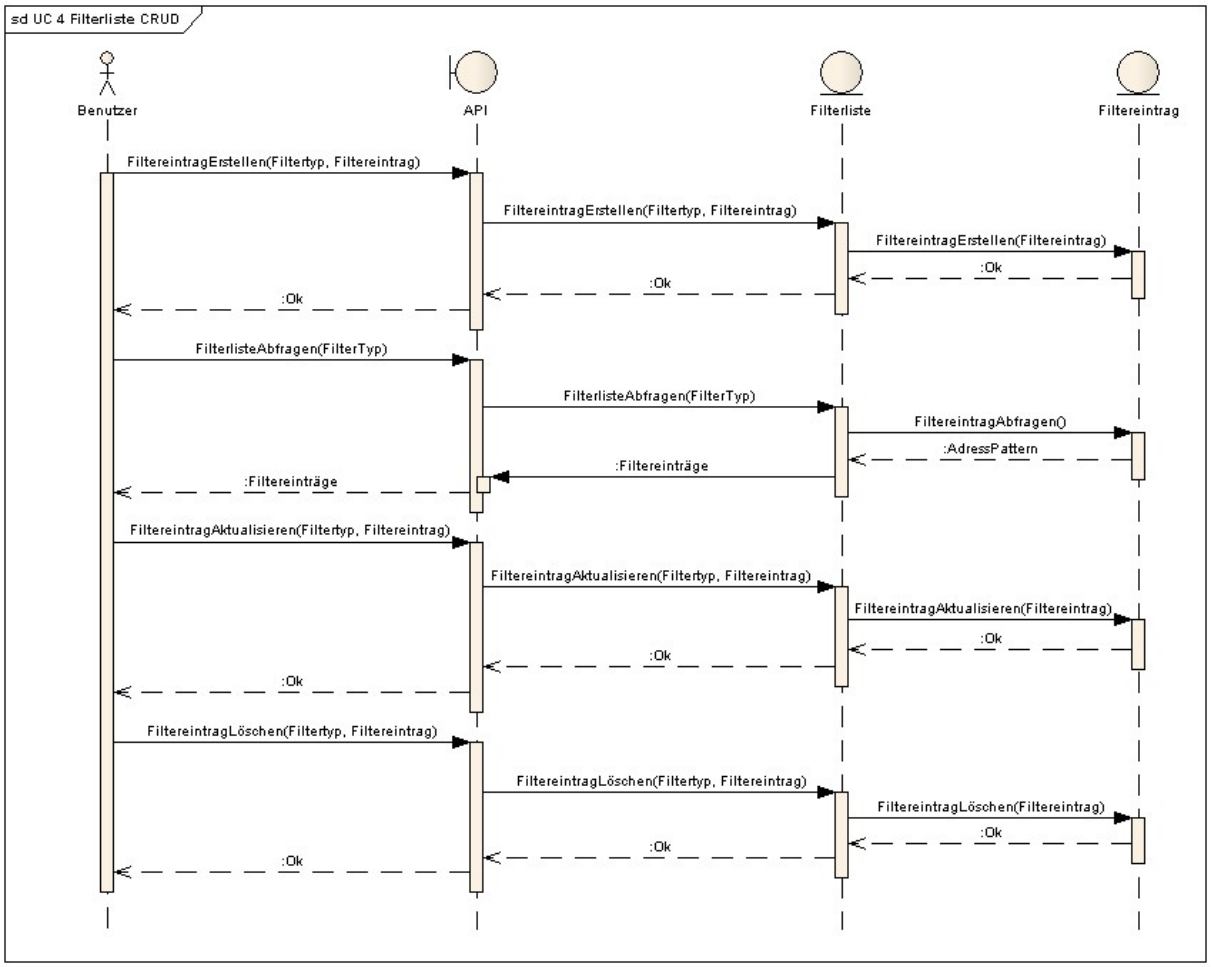


Abbildung 21 - Sequenzdiagramm detailliert - UC 4 Filterliste CRUD

8.2.5 SSD UC 5 NDR Pattern CRUD

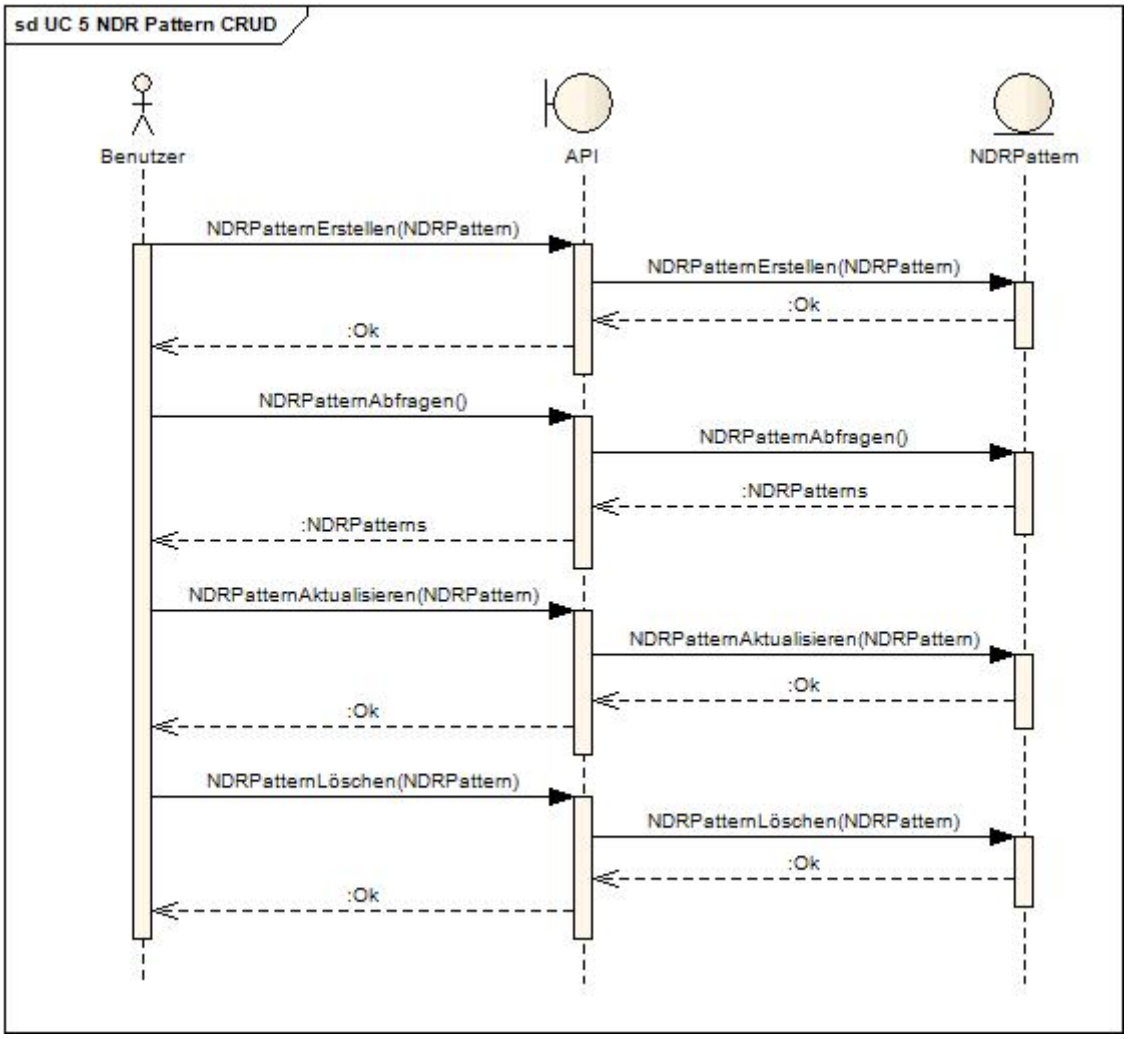


Abbildung 22 - Sequenzdiagramm detailliert - UC 5 NDR Pattern CRUD

8.2.6 SSD UC 6 Qualitätsindikator abfragen

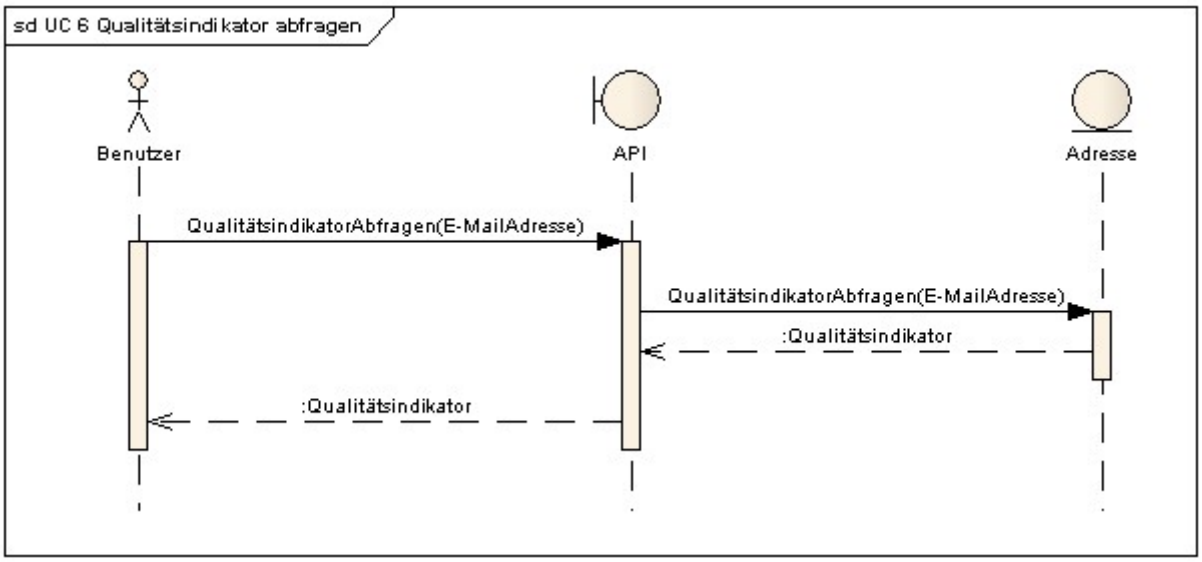


Abbildung 23 - Sequenzdiagramm detailliert - UC 6 Qualitätsindikator abfragen

8.2.7 SSD UC 7 Report abfragen

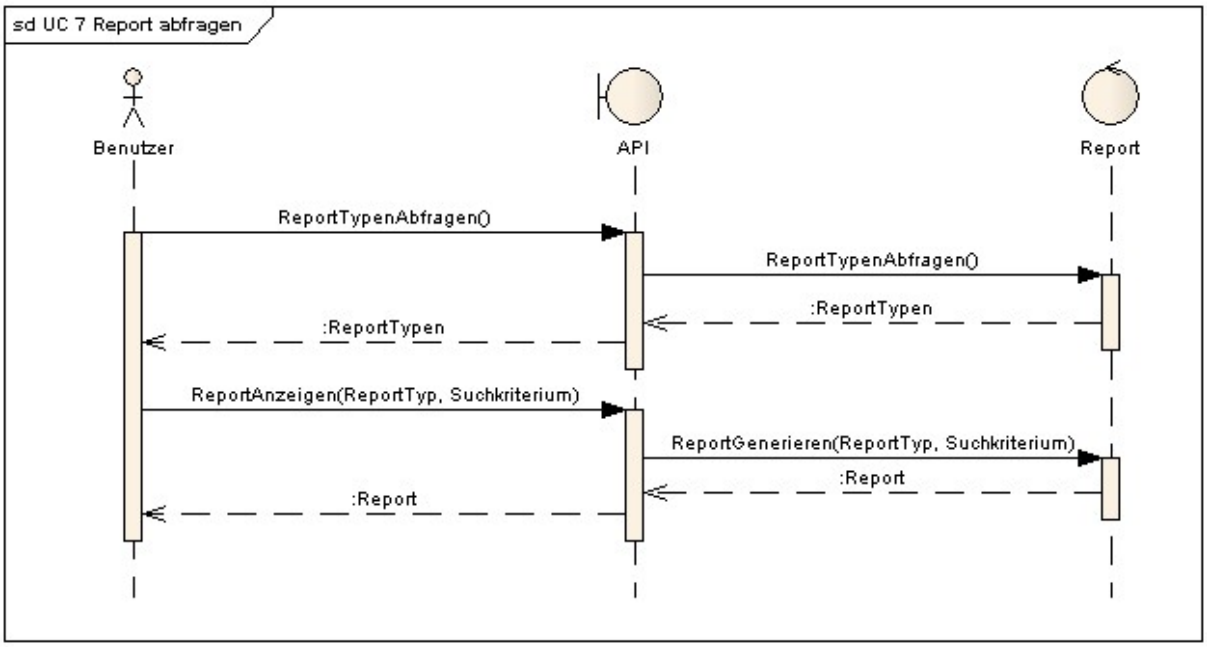


Abbildung 24 - Sequenzdiagramm detailliert - UC 7 Report abfragen

8.2.8 SSD UC 8 Datenbank bereinigen

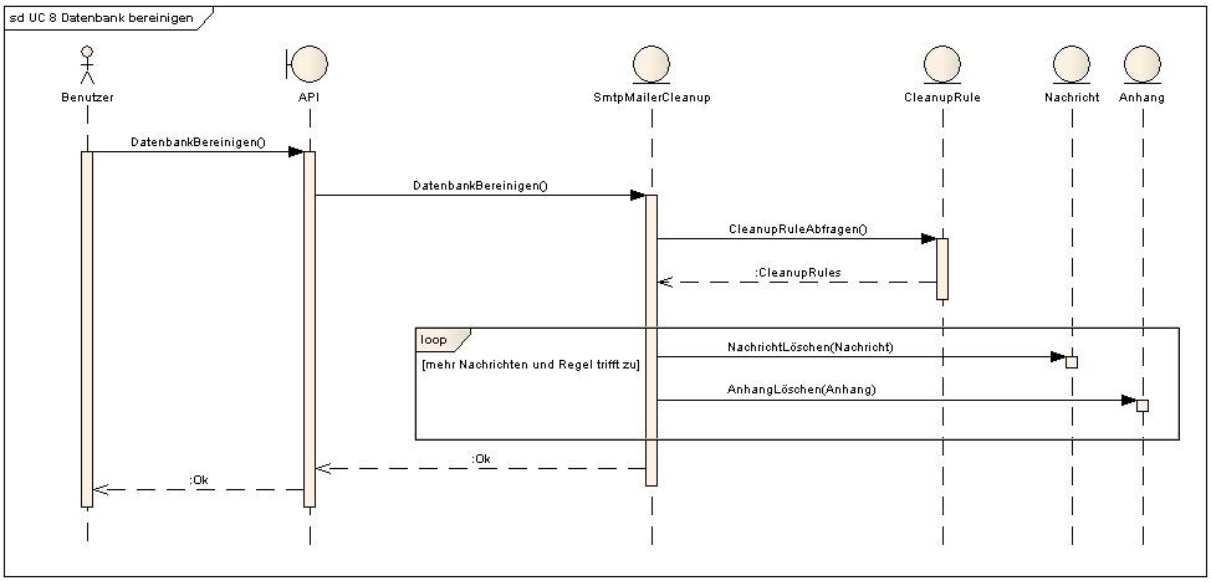


Abbildung 25 - Sequenzdiagramm detailliert - UC 8 Datenbank bereinigen

8.2.9 SSD UC 9 CleanupRule CRUD

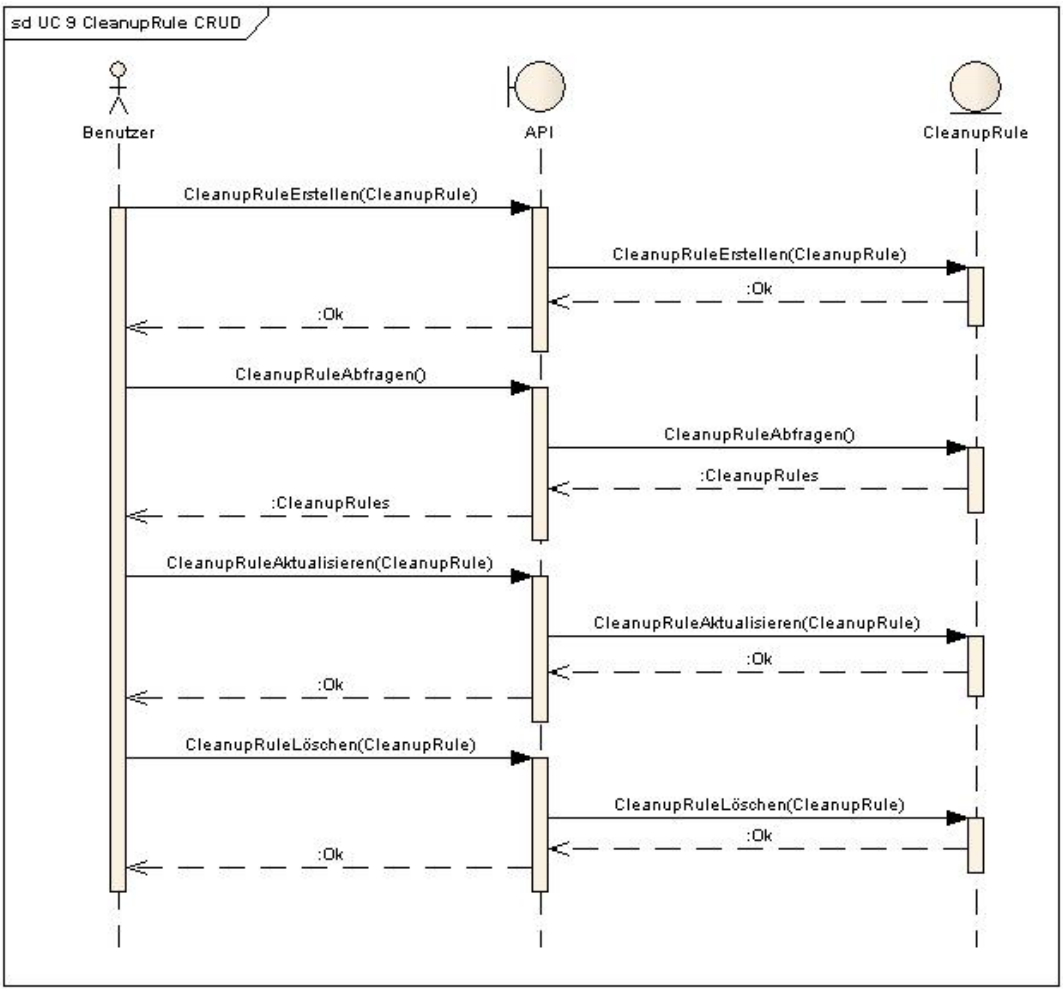


Abbildung 26 - Sequenzdiagramm detailliert - UC 9 CleanupRule CRUD

8.2.10 SSD UC 10 NDR Nachricht verarbeiten

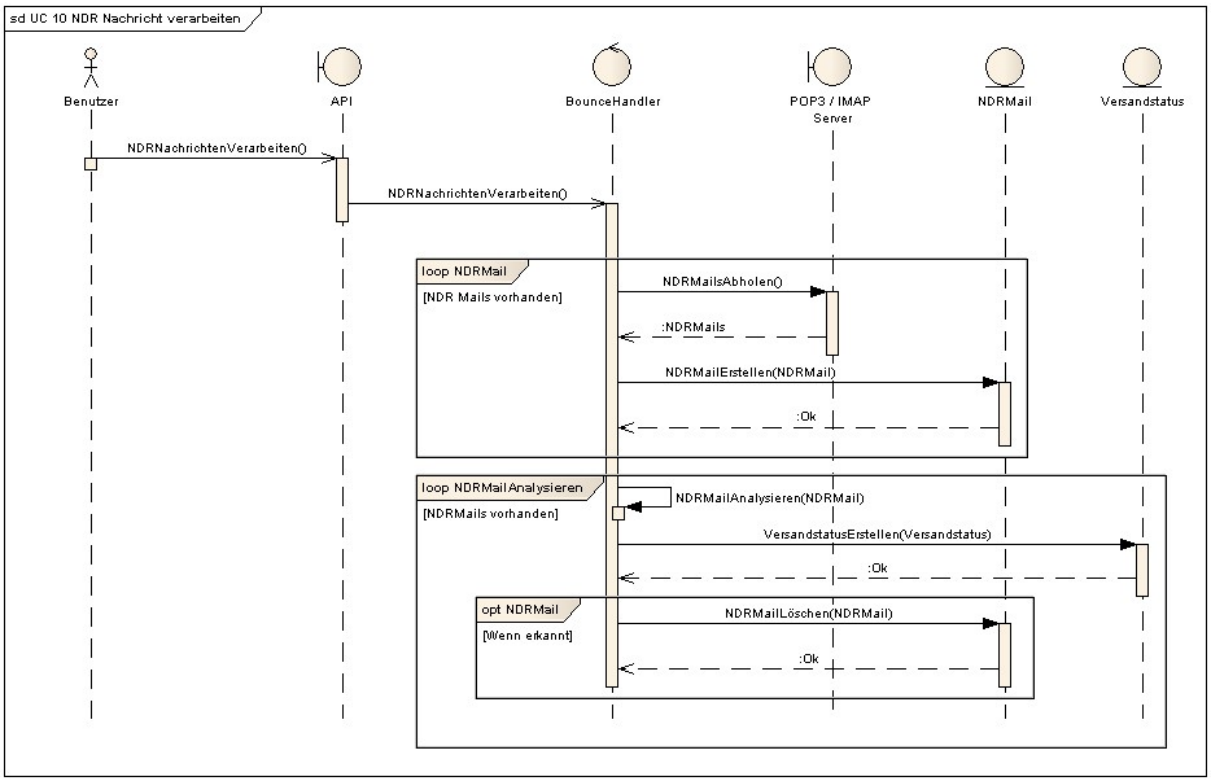


Abbildung 27 - Sequenzdiagramm detailliert - UC 10 NDR Nachricht verarbeiten

8.3 Contracts

8.3.1 Contracts UC 1 Nachricht versenden

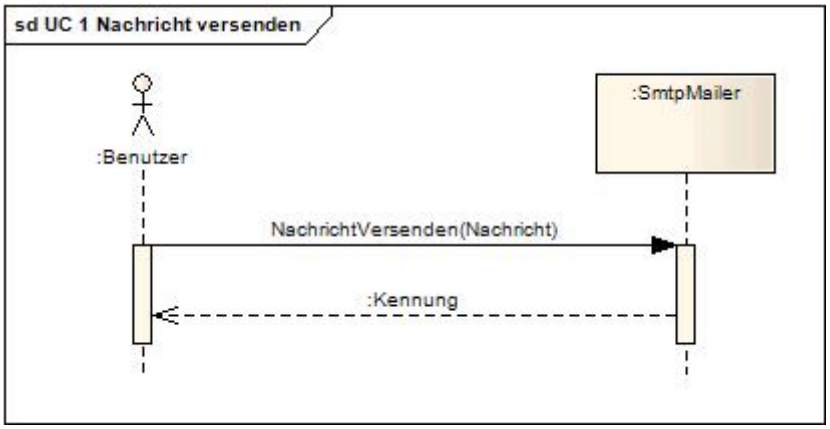


Abbildung 28 - Sequenzdiagramm - UC 1 Nachricht versenden

Operation	NachrichtVersenden(Nachricht)
Cross References	Use Case: Nachricht versenden
Preconditions	• Es ist eine SmtpServer-Instanz *smtpServer* vorhanden • Es ist eine Warteschlange-Instanz *queue* vorhanden
Postconditions	• Die Nachricht-Instanz *msg* wurde im System abgelegt • Die Nachricht-Instanz *msg* wurde mit *queue* assoziiert • Eine oder mehrere Anhang-Instanzen wurden erstellt und mit *msg* assoziiert

	• Eine oder mehrere Empfänger-Instanzen *recipient* wurden erstellt und mit *msg* assoziiert • Wenn die Adresse für *recipient* nicht existierte wurde eine Adresse-Instanz erstellt und mit *recipient* assoziiert • Für jeden Empfänger wurde eine Versandstatus-Instanz *status* erstellt und mit *recipient* assoziiert • Jeder erstellte Anhang wurde mit der Nachrichten-Instanz *msg* assoziiert

8.3.2 Contracts UC 2 Versandstatus abfragen

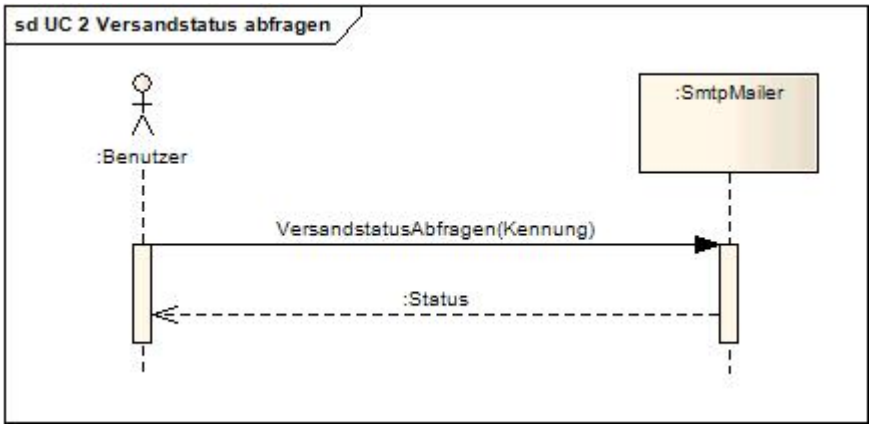


Abbildung 29 - Sequenzdiagramm - UC 2 Versandstatus abfragen

Operation	VersandstatusAbfragen(Kennung)
Cross References	Use Case: Versandstatus abfragen
Preconditions	• Es ist eine Nachricht-Instanz *msg* vorhanden
Postconditions	-

8.3.3 Contracts UC 3 Nachricht abfragen

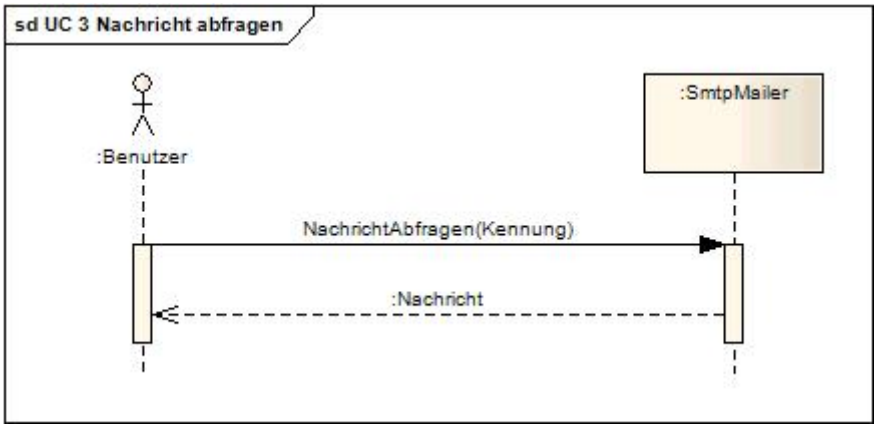


Abbildung 30 - Sequenzdiagramm - UC 3 Nachricht anzeigen

Operation	NachrichtAbfragen(Kennung)
Cross References	Use Case: Nachricht abfragen
Preconditions	• Es ist eine Nachricht-Instanz *msg* vorhanden
Postconditions	-

8.3.4 Contracts UC 4 Filterliste CRUD

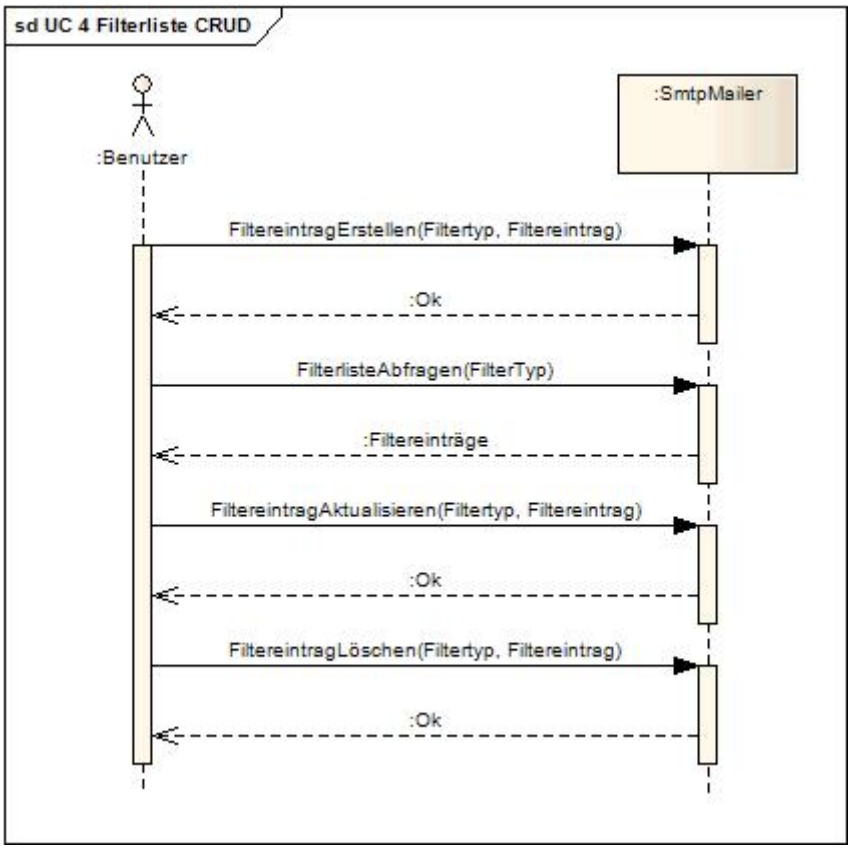


Abbildung 31 - Sequenzdiagramm - UC 4 Filterliste konfigurieren

Operation	FiltereintragErstellen(Filtertyp, Filtereintrag)
Cross References	Use Case: Filterliste CRUD
Preconditions	• Es ist eine Filterliste-Instanz *filterList* vom Typ Filtertyp vorhanden
Postconditions	• Die Filtereintrag-Instanz *filterEntry* wurde im System abgelegt • Die Filtereintrag-Instanz *filterEntry* wurde mit *filterList* assoziiert

Operation	FilterlisteAbfragen(FilterTyp)
Cross References	Use Case: Filterliste CRUD
Preconditions	• Es ist eine Filterliste-Instanz *filterList* vom Typ Filtertyp vorhanden
Postconditions	-

Operation	FiltereintragAktualisieren(Filtertyp, Filtereintrag)
Cross References	Use Case: Filterliste CRUD
Preconditions	• Es ist eine Filterliste-Instanz *filterList* vom Typ Filtertyp vorhanden • Die übergebene Filtereintrag-Instanz *filterEntry* ist im System vorhanden
Postconditions	• Die übergebene Filtereintrag-Instanz *filterEntry* wurde mit der bestehenden Instanz ersetzt

Operation	FiltereintragLöschen(Filtertyp, Filtereintrag)
Cross References	Use Case: Filterliste CRUD
Preconditions	• Es ist eine Filterliste-Instanz *filterList* vom Typ Filtertyp vorhanden • Die übergebene Filtereintrag-Instanz *filterEntry* ist im System vorhanden
Postconditions	• Die übergebene Filtereintrag-Instanz *filterEntry* wurde im System gelöscht

8.3.5 Contracts UC 5 NDR Pattern CRUD

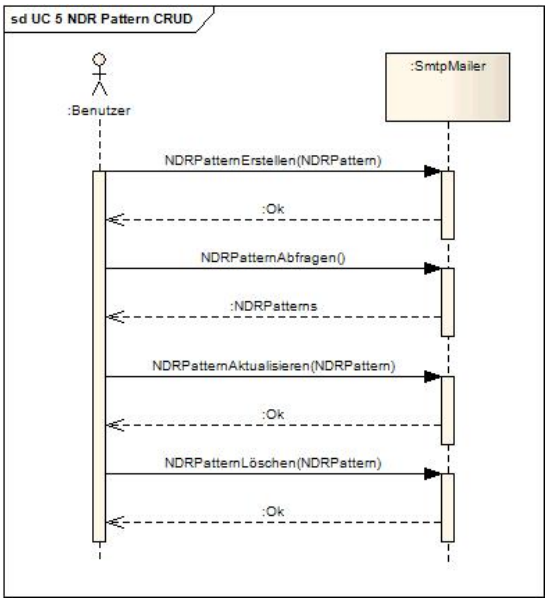


Abbildung 32 - Sequenzdiagramm - UC 5 NDR Pattern CRUD

Operation	NDRPatternErstellen(NDRPattern)
Cross References	Use Case: NDR Pattern CRUD
Preconditions	• Es ist eine BounceHandler-Instanz *bounce* vorhanden
Postconditions	• Die NDRPattern-Instanz *pattern* wurde im System abgelegt • Die Instanz *pattern* wurde mit *bounce* assoziiert

Operation	NDRPatternAbfragen()
Cross References	Use Case: NDR Pattern CRUD
Preconditions	• Es ist eine NDRPattern-Instanz *pattern* vorhanden
Postconditions	-

Operation	NDRPatternAktualisieren(NDRPattern)
Cross References	Use Case: NDR Pattern CRUD
Preconditions	• Die übergebene NDRPattern-Instanz *pattern* ist im System vorhanden
Postconditions	• Die übergebene NDRPattern-Instanz *pattern* wurde mit der bestehenden Instanz ersetzt

Operation	NDRPatternLöschen(NDRPattern)
Cross References	Use Case: NDR Pattern CRUD

Preconditions	• Die übergebene NDRPattern-Instanz *pattern* ist im System vorhanden
Postconditions	• Die übergebene NDRPattern-Instanz *pattern* wurde im System gelöscht

8.3.6 Contracts UC 6 Qualitätsindikator abfragen

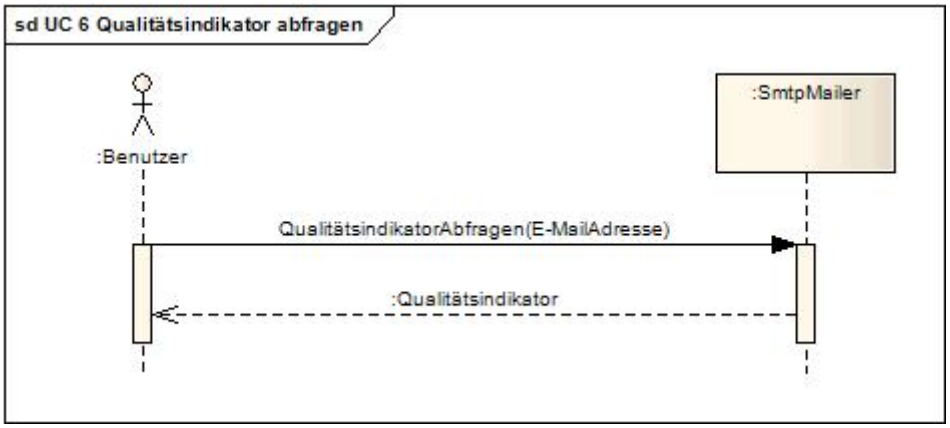


Abbildung 33 - Sequenzdiagramm - UC 6 Qualitätsindikator abfragen

Operation	QualitätsindikatorAbfragen(E-MailAdresse)
Cross References	Use Case: Qualitätsindikator abfragen
Preconditions	• Es ist eine Adresse-Instanz *adr* vorhanden
Postconditions	-

8.4 Contracts UC 7 Report abfragen

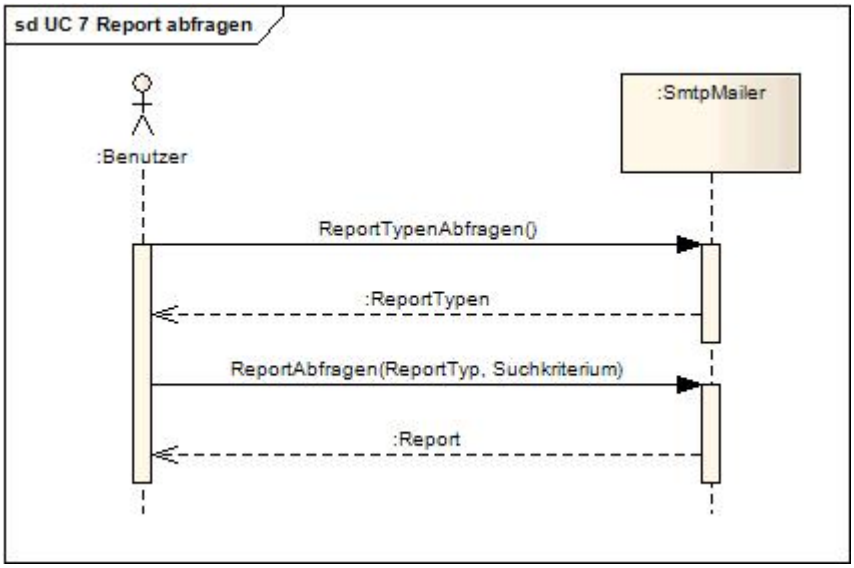


Abbildung 34 - Sequenzdiagramm - UC 7 Report abfragen

Operation	ReportTypenAbfragen()
Cross References	Use Case: Report abfragen
Preconditions	• Es ist eine Report-Instanz *report* vorhanden
Postconditions	-

Operation	ReportAnzeigen(ReportTyp, Suchkriterium)
Cross References	Use Case: Report abfragen

Preconditions	• Es ist eine Report-Instanz *report* vorhanden
Postconditions	-

8.4.1 Contracts UC 8 Datenbank bereinigen

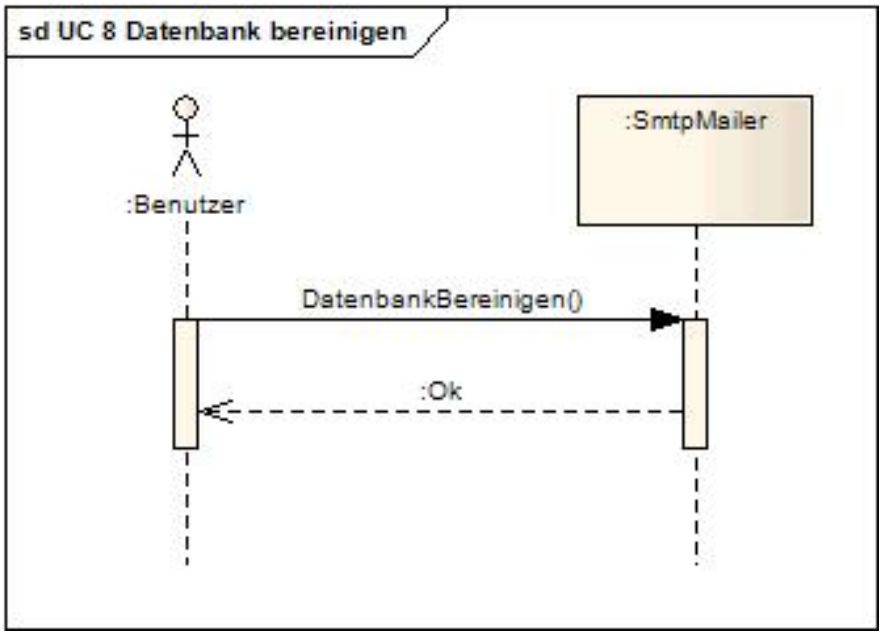


Abbildung 35 - Sequenzdiagramm - UC 8 Datenbank bereinigen

Operation	DatenbankBereinigen()
Cross References	Use Case: Datenbank bereinigen
Preconditions	• Es ist eine SmtpMailerCleanup-Instanz *smtpMailerCleanup* vorhanden • Es ist eine CleanupRule-Instanz *cleanupRule* vorhanden
Postconditions	• In der Nachricht-Instanz *msg*, welche auf die CleanupRule-Instanz *cleanupRule* passt (*msg.Applikationscode, msg.Mailtyp)*, wurde der *msg.Content* gelöscht • Alle mit *msg* assoziierten Anhang-Instanzen *anhang* wurden gelöscht

8.4.2 Contracts UC 9 CleanupRule CRUD

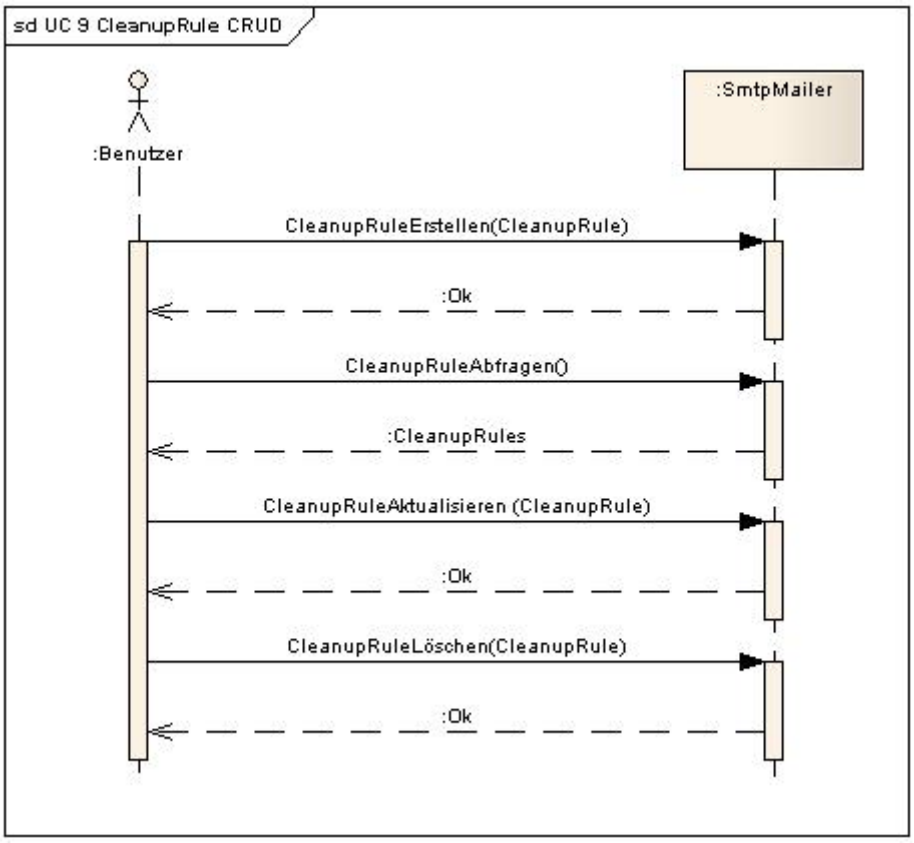


Abbildung 36 - Sequenzdiagramm - UC 9 CleanupRule CRUD

Operation	CleanupRuleErstellen(CleanupRule)
Cross References	Use Case: CleanupRule CRUD
Preconditions	• Es ist eine SmtpMailerCleanup-Instanz *smtpMailerCleanup* vorhanden
Postconditions	• Die übergebene CleanupRule-Instanz *cleanupRule* wurde im System abgelegt • Die CleanupRule-Instanz *cleanupRule* wurde mit *smtpMailerCleanup* assoziiert

Operation	CleanupRuleAbfragen()
Cross References	Use Case: CleanupRule CRUD
Preconditions	• Es ist eine SmtpMailerCleanup-Instanz *smtpMailerCleanup* vorhanden
Postconditions	-

Operation	CleanupRuleAktualisieren(CleanupRule)
Cross References	Use Case: CleanupRule CRUD
Preconditions	• Es ist eine SmtpMailerCleanup-Instanz *smtpMailerCleanup* vorhanden
Postconditions	• Die übergebene CleanupRule-Instanz *cleanupRule* wurde mit der bestehenden Instanz ersetzt

Operation	CleanupRuleLöschen(CleanupRule)
Cross References	Use Case: CleanupRule CRUD
Preconditions	• Es ist eine SmtpMailerCleanup-Instanz *smtpMailerCleanup* vorhanden
Postconditions	• Die übergebene CleanupRule-Instanz *cleanupRule* wurde im System gelöscht

8.4.3 Contracts UC 10 NDR Nachricht verarbeiten

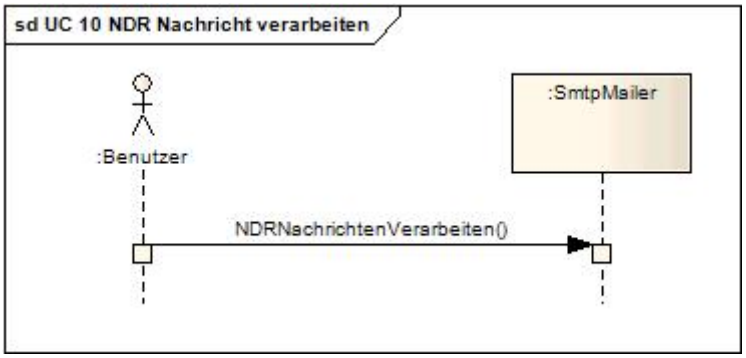


Abbildung 37 - Sequenzdiagramm - UC 10 NDR Nachricht verarbeiten

Operation	NDRNachrichtVerarbeiten()
Cross References	Use Case: Datenbankbereinigung CRUD
Preconditions	• Die zu verarbeitende NDRMail-Instanz *ndrMail* ist bereits in der Datenbank gespeichert oder der POP3/IMAP Server steht betriebsbereit zur Verfügung um die Bounce-Nachrichten abzurufen
Postconditions	• Der Versandstatus *status* der Nachricht, zu welcher die NDRMail-Instanz *ndrMail* empfangen wurde, wurde aktualisiert und mit der entsprechenden NDRBeschreibung *desc* assoziiert • Die NDRMail-Instanz *ndrMail* wurde gelöscht, wenn die Bounce-Nachricht erkannt und klassiert werden konnte

9 Evaluation

9.1 Komponenten Evaluation

9.1.1 Mailer Komponente

Mailer Komponenten				
	Produktname	***EasyMail.NET***	***MailBee***	***aspNetMail***
	Hersteller	*QuikSoft*	*AfterLogic*	*Advanced Intellect LLC*
Funktionale Anforderungen		***0=nicht erfüllt 1=erfüllt 1*=erfüllt mit Zusatz-Plug-In***		
	Plaintext	1	1	1
	MIME Format	1	1	1
	Mehrere Empfänger To, Cc, Bcc	1	1	1
	Priorität	1	1	1
	Empfangs- und Lesebestätigung	1	1	1
	Angabe von Reply-To	1	1	1
	Attachments	1	1	1
	Validierung von E-Mail Adresse	1	1	1
	Signaturen	1* Benötigt S/MIME Plug-In	1	0
	ReturnPath	1	1	1
	SMTP Authentifizierung	1 Auth-Login, CRAM-MD5, NTLM User Defined	1 LOGIN, PLAIN, CRAM-MD5, DIGEST-MD5, NTLM, Kerberos, POP before SMTP, Windows User Authentication, User Defined	1 Nicht direkt ersichtlich

	POP3 / IMAP Unterstützung	1* Benötigt Full Edition	1	1* Benötigt aspNetPOP3 / aspNetIMAP
	Verschlüsselung	*1** *Benötigt S/MIME Plug-In*	*1*	*0*
	SSL	*1** *Benötigt SSL Plug-In*	*1*	*1*
Kriterien		***1=schlecht 2=mittel 3=gut***		
	Support	3 Forum, KB, Tickets	3 Forum, KB, Tickets	2 Forum, E-Mail
	Dokumentation und Beispiele	3 Sehr viele Beispiele, Forum, API Dokumentation, FAQ	3 Tutorials, Forum, API Dokumentation, Beispiele, FAQ	1 FAQ, Forum, Beispielprojekte Eher minimalistisch und unübersichtlich
	Weitere vorhandene Funktionen	3	3	1
	Synchrone / Asynchrone Methodenaufrufe	Synchron / Asynchron	Synchron / Asynchron	Synchron / Asynchron
	Preis	$249.00 / $799.00 SMTP Edition $499.00 / $1999.00 Full Edition $ 374.25 / 1499.25 Security Plug-In	$199.00 / $349.00 MailBee.NET Object $ 69.00 / $129.00 MailBee.NET SMTP $ 69.00 / $129.00 MailBee.NET POP3/IMAP $ 60.00 / $129.00 MailBee.NET Security	$250.00 aspNetMail $350.00 aspNetPOP3 $350.00 aspNetIMAP
	Bemerkungen	SMTP Version unterstützt nur das Versenden. Parsing, Validation, POP3/IMAP, … wird nur durch die Full Edition unterstützt	In MailBee.Net Object ist alles enthalten. Alternative nur die Komponenten zu kaufen die benötigt werden.	Benötigt zum Lesen und Parsen von Nachrichten aspNetMIME. Enthalten bei POP3/IMAP sonst separat mit aspNetMime ($250.00)

Tabelle 29 - Auswertung Mailer Komponenten

9.1.1.1 Entscheidung

Die Entscheidung der Mailer Komponente fiel auf das Produkt EasyMail.NET Full Edition von QuikSoft.

Der Entscheid ist dadurch zu begründen, dass für dieses Produkt bei der Firma comparis.ch AG bereits Lizenzen vorhanden sind und die gesamten funktionalen wie auch nicht funktionalen Anforderungen vollständig vom Produkt EasyMail.NET erfüllt werden. Preise für Versionsupgrades werden aus diesem Grund ausser Acht gelassen.

9.1.2 Bounce Mail Komponente

9.1.2.1 Übersicht

Bei nicht erfolgreich zugestellten E-Mail Nachrichten wird der Absender mittels einer Bounce Nachricht informiert, wieso es bei der Zustellung Probleme gegeben hat. Die zu evaluierende Komponente muss das Auswerten und Zuordnen des ausschlaggebenden Grundes für die Nichtzustellung vereinfachen.

Diese Auswertung wird benötigt, um einen Qualitätsindikator für E-Mail Adressen einzuführen und so die Adressqualität zu erhöhen.

9.1.2.2 Anforderungen

- Bounce Mails müssen nach Kategorie und Typ ausgewertet werden
- Die Komponente verfügt über einen bereits vordefinierten Erkennungskatalog
- Der Katalog muss mit eigenen Signaturen erweitert werden können
- Es können eigene Kategorien und Typen definiert und zugeordnet werden
- Das Auslesen der ursprünglichen E-Mail Nachricht muss möglich sein
- Die zurückerhaltene Bounce Mail kann über eine der folgenden Varianten an die Bounce Mail Komponente übergeben werden:
    - Stream
    - Dateipfad
    - E-Mail Objekt (Voraussetzung dafür ist, dass es sich um denselben Hersteller wie bei der Mailer Komponente handelt)
    - Direkte Abfrage eines Postfachs mit den Bounce Mails via POP3 / IMAP
- Synchrone und asynchrone Methodenaufrufe
- Die Bounce Mail Komponente muss eine Verarbeitung von 30‘000 E-Mails pro Stunde bewältigen können

9.1.2.3 Ausgangslage

Um die Erkennungsqualität der verschiedenen Bounce Mail Komponenten zu prüfen, wurden 22 verschiedene Bounce Mails ausgesucht, die möglichst verschiedene, zum Teil spezielle Fehler aufweisen und von unterschiedlichen MTAs stammen. Die 22 Bounce Mails wurden uns von Benedikt Unold, comparis.ch AG zur Verfügung gestellt und sind originale Bounce Messages vom comparis.ch SMTP Server.

Die Performance der Bounce Mail Komponente wurde durch das Abarbeiten von 10’000 E-Mail Nachrichten ermittelt. So konnte von der benötigten Abarbeitungszeit auf die mittlere Verarbeitungszeit pro Bounce Mail geschlossen werden. In einer ersten Testphase werden die E-Mails als lokal gespeicherte *.eml* Dateien eingelesen, in einer zweiten Testphase mittels POP3 / IMAP vom Server abgeholt, in der Datenbank zwischengespeichert und verarbeitet.

9.1.2.4 Produkte

QuickSoft EasyMail BounceBuster

Die Produktinformationen wurden von der Webseite des Herstellers QuikSoft (QuikSoft BounceBuster, 2009) entnommen.

Typen und Kategorien

Name	Beschreibung
Undeliverable	The message was undeliverable.
Blocked	The message was blocked.
Informational	The bounce is informational only. The means the message may have been delivered.
Custom	The bounce was detected using a custom signature.
Unkown	The bounce type is unknown.

Tabelle 30 - BounceType

Jeder BounceType enthält mehrere BounceCategory Typen.

Name	Beschreibung
Hard	The server was unable to deliver the message due to a permanent error. A permanent failure is one which is not likely to be resolved by resending the message in the current form. Some change to the message or the destination must be made for successful delivery.
Soft	The server was unable to deliver the message but it is a temporary error. This type of failure is one in which the message as sent is valid, but some temporary event prevents the successful sending of the message. Sending in the future may be successful
Spam	Your message has been identified as spam and was blocked.
Virus	Your message contained a virus and was blocked.
Modified	Your message was modified prior to delivery. (i.e. illegal attachment removed)
BlockedOther	Your message was blocked for an unknown or general reason. (i.e. A word in the message triggered content filtering to fail the message.)
Challenge	The message is a challenge response message which requires further action for the original message to be delivered. For example, "Click this link to verify the message is coming from a human."
Transient	The message has been delayed because there is a problem. This does not mean the message will fail it's only a warning.
AutoReply	The message is an auto-reply. For example out of office, or message received confirmation.
Subscribe	The message is a subscribe request. The Address will be set to the from address.
Unsubscribe	The message is an unsubscribe request. The Address will be set to the from address.
AddressChange	The message is an address change notification.
Delivered	A confirmation stating the message was delivered.
Custom	The message was detected by a custom signature.
Unknown	The type could not be determined.

Tabelle 31 - BounceCategory

Eigene Typen und Kategorien können hinzugefügt werden. Dazu kann aber lediglich ein benutzerdefinierter String bei der AddSignature() Methode der BounceBuster Klasse übergeben werden.

Tests

Mit folgendem C# Quellcode wurde die Erkennungsqualität getestet:

```
public static void BounceBusterNdrQualityTest()
{
    try
    {
        BounceBuster bb = new BounceBuster();

        bb.OnBounceProcessed += new BounceProcessed(OnBounceBusterProcessed);
        bb.Process(@"C:\Temp\quality", "*.eml");
    }
    catch (Exception exp)
    {
        Console.WriteLine("Msg: " + exp.Message);
    }
}

static void OnBounceBusterProcessed(object sender, BounceProcessedEventArgs e)
{
    if (e.Status != ResultStatus.Identified)
    {
        Console.WriteLine(e.Filename);
        Console.WriteLine("Status: " + e.Status);
    }
    else
    {
        Console.WriteLine(e.Filename);
        Console.WriteLine("Kategorie: " + e.Results[0].BounceCategory);
        Console.WriteLine("Typ: " + e.Results[0].BounceType);
        Console.WriteLine("Kommentar: " + e.Results[0].UserData);
    }
}
```

Mit folgendem C# Quellcode wurde die Performance getestet:

```
public static void BounceBusterNdrPerformanceTest()
{
    Stopwatch watch;

    Console.WriteLine("BounceBuster\n------------");
    watch = Stopwatch.StartNew();

    try
    {
        BounceBuster bb = new BounceBuster();

        bb.OnBounceProcessed += new BounceProcessed(OnBounceBusterProcessed);
        bb.Process(@"C:\Temp\performance", "*.eml");
    }
    catch (Exception exp)
    {
        Console.WriteLine("Msg: " + exp.Message);
    }

    watch.Stop();
    Console.WriteLine("Benötigte Zeit für 10'000 Analysen: " +
         watch.Elapsed.TotalMinutes + " Minuten");
    Console.WriteLine("Benötigte Zeit pro Analyse: " +
         (watch.ElapsedMilliseconds / 10000) + "ms");

}
```

```
static void OnBounceBusterProcessed(object sender, BounceProcessedEventArgs e)
{
    if (e.Status != ResultStatus.Identified)
    {
        // Do Nothing
    }
    else
    {
        // Do Nothing
    }
}
```

Nachfolgend das erhaltene Resultat der durchgeführten Tests:

```
Nicht erkannt:          11
Fehlerhaft erkannt:     2
Richtig erkannt:        6
Nicht getestet:         1 (korrupte .eml Datei)

Verarbeitungszeit 10‘000 E-Mails:             02:34:37
Mittlere Verarbeitungszeit pro E-Mail:        927ms
```

AfterLogic MailBee.NET Objects

Aufgrund der sehr eingeschränkten Funktionalität wurde entschlossen, die Komponente AfterLogic MailBee.NET Objects nicht genauer zu evaluieren und zu testen.

Der Auszug der unterstützten Funktionalitäten ist unter *9.1.2.5 Auswertung der* Bounce Mail Komponenten auf Seite 86 ersichtlich.

Advanced Intellect ListNanny

Die Produktinformationen wurden von der Webseite des Herstellers Advanced Intellect (Advanced Intellect ListNanny, 2009) entnommen.

Typen und Kategorien

Name	Beschreibung
HardBounce	NDRPatternCollection containing NDRType.HardBounce types
Transient	NDRPatternCollection containing NDRType.Transient types
Unsubscribe	NDRPatternCollection containing NDRType.Unsubscribe types
Subscribe	NDRPatternCollection containing NDRType.Subscribe types
AutoResponder	NDRPatternCollection containing NDRType.AutoResponder types
AddressChange	NDRPatternCollection containing NDRType.AddressChange types
DnsError	NDRPatternCollection containing NDRType.DnsError types
SpamNotification	NDRPatternCollection containing NDRType.SpamNotification types
OpenRelayTest	NDRPatternCollection containing NDRType.OpenRelayTest types
SoftBounce	NDRPatternCollection containing NDRType.SoftBounce types
VirusNotification	NDRPatternCollection containing NDRType.VirusNotification types
Priority	NDRPatternCollection containing very specific, high probability NDRType matches.
LowestPriority	NDRPatternCollection containing NDRTypes that have less specific patterns, but still match various NDR types.
Custom	NDRPatternCollection containing new patterns defined by the end developer.
ChallengeVerification	NDRPatternCollection containing NDRType.ChallengeNotification types

Tabelle 32 - NDRCategory

Name	Beschreibung
AddressChange	The recipient has requested an address change.
AutoResponder	Automatic email responder (ex: 'Out of Office' or 'On Vacation')
DnsError	A temporary DNS error.

HardBounce	The server was unable to deliver your message (ex: unknown user, mailbox not found)
OpenRelayTest	The NDR is actually a test email message to see if the mail server is an open relay.
SoftBounce	Unable to temporarily deliver message (i.e. mailbox full, account disabled, exceeds quota, out of disk space)
SpamNotification	The message was delivered, but was either blocked by the user, or classified as spam, bulk mail, or had rejected content.
Subscribe	Subscribe request from someone wanting to get added to the mailing list.
Transient	The server couldn't temporarily deliver your message (ex: Message is delayed due to network troubles)
Unkonwn	Unable to classify the NDR
Unsubscribe	Unsubscribe or Remove request
VirusNotification	The bounce is actually a virus notification warning about a virus/code infected message.
ChallengeVerification	The bounce is a challange asking for verification you actually sent the email. Typcial challenges are made by Spam Arrest, or MailFrontier Matador
Custom1	To be used by the developer for their own designation, when creating additional patterns
Custom2	To be used by the developer for their own designation, when creating additional patterns

Tabelle 33 - NDRType

Eigene Typen und Kategorien sind bei ListNanny nicht möglich.

Tests

Mit folgendem C# Quellcode wurde die Erkennungsqualität getestet:

```
public static void ListNannyNdrQualityTest()
{
    try
    {
        string[] files = Directory.GetFiles(@"C:\Temp\quality", "*.eml");

        NDR.ImportDefinitionFile(@"C:\Temp\ndr1.41.def.xml");
        NDR ndr;

        foreach (string file in files)
        {
            ndr = NDR.ParseFile(file);

            if (ndr.Type == NDRType.Unknown)
            {
                Console.WriteLine(ndr.Filename);
                Console.WriteLine("Wurde nicht erkannt!");
            }
            else
            {
                Console.WriteLine(ndr.Filename);
                Console.WriteLine("Hilfe: " + ndr.HelpMessage);
                Console.WriteLine("Typ: " + ndr.Type);
                Console.WriteLine("Kommentar: " + ndr.PatternComments);
            }
        }
    }
    catch (Exception exp)
    {
        Console.WriteLine("Msg: " + exp.Message);
    }
}
```

Mit folgendem C# Quellcode wurde die Performance getestet:

```
public static void ListNannyNdrPerformanceTest()
{
    Stopwatch watch;

    Console.WriteLine("\n\nListNanny\n---------\n");
    watch = Stopwatch.StartNew();

    try
    {
        string[] files = Directory.GetFiles(@"C:\Temp\performance", "*.eml");

        NDR.ImportDefinitionFile(@"C:\Temp\ndr1.41.def.xml");
        NDR ndr;

        foreach (string file in files)
        {
            ndr = NDR.ParseFile(file);

            if (ndr.Type == NDRType.Unknown)
            {
                // Do Nothing
            }
            else
            {
                // Do Nothing
            }
        }
    }
    catch (Exception exp)
    {
        Console.WriteLine("Msg: " + exp.Message);
    }
    Console.WriteLine("Benötigte Zeit für 10'000 Analysen: " +
         watch.Elapsed.TotalMinutes + "Minuten");
    Console.WriteLine("Benötigte Zeit pro Analyse: " +
         (watch.ElapsedMilliseconds / 10000) + "ms");

}
```

Nachfolgend das erhaltene Resultat der durchgeführten Tests:

```
Nicht erkannt:          2
Fehlerhaft erkannt:     3
Richtig erkannt:        16
Nicht getestet:         1 (korrupte .eml Datei)

Verarbeitungszeit 10‘000 E-Mails:             00:06:43
Mittlere Verarbeitungszeit pro E-Mail:        40ms
```

Safabyte NetXtremeBounceFilter

Die Produktinformationen wurden von der Webseite des Herstellers Safabyte (Safabyte NetXtremeBounceFilter, 2009) entnommen.

Zu den unten aufgelisteten Typen und Kategorien führt der Hersteller keine genauen Beschreibungen auf.

Typen und Kategorien

Name	Beschreibung
AutoReply	
Blocked	
Generic	
Hard	
Soft	
Temporary	
Unknow	

Tabelle 34 - BounceType

Name	Beschreibung
AntiSpam	
Autoreply	
Challenging	
Defer	
Delayed	
DnsLoop	
DnsUnknow	
Full	
Generic	
Inactive	
Internal Error	
LatinOnly	
MailboxUnkown	
NotResponding	
Oversize	
RejectedCommand	
RejectedContent	
RejectedUser	
Subscribe	
Unknow	
Unsubscribe	

Tabelle 35 - BounceCategory

Eigene Typen und Kategorien können hinzugefügt werden. Dies ist über die Methoden AddBounceCategory(int, String) und AddBounceType(int, String) der BounceFilter Klasse möglich.

Tests

Mit folgendem C# Quellcode wurde die Erkennungsqualität getestet:

```
public static void NetXtremeBounceFilterNdrQualityTest()
{
    try
    {
        BounceFilter bf = new BounceFilter();
        BounceFilterResultCollection results =
                bf.ProcessMessages(@"C:\Temp\quality");

        foreach (BounceFilterResult result in results)
        {
            if (result.Identified)
            {
                Console.WriteLine(result.FileName);
                Console.WriteLine("Kategorie: " + result.BounceCategory);
                Console.WriteLine("Typ: " + result.BounceType);
            }
            else
            {
                Console.WriteLine(result.FileName);
                Console.WriteLine("Wurde nicht erkannt!");
                Console.WriteLine("Exception: " + result.Error);
            }
        }
    }
    catch (Exception exp)
    {
        Console.WriteLine("Msg: " + exp.Message);
    }
}
```

Mit folgendem C# Quellcode wurde die Performance getestet:

```
public static void NetXtremeBounceFilterNdrPerformanceTest()
{
    Stopwatch watch;

    Console.WriteLine("\n\nBounceFilter\n------------\n");
    watch = Stopwatch.StartNew();

    try
    {
        BounceFilter bf = new BounceFilter();
        BounceFilterResultCollection results =
                bf.ProcessMessages(@"C:\Temp\performance");

        foreach (BounceFilterResult result in results)
        {
            if (result.Identified)
            {
                // Do Nothing
            }
            else
            {
                // Do Nothing
            }
        }
    }
    catch (Exception exp)
    {
        Console.WriteLine("Msg: " + exp.Message);
    }

    Console.WriteLine("Benötigte Zeit für 10'000 Analysen: " +
         watch.Elapsed.TotalMinutes + "Minuten");
    Console.WriteLine("Benötigte Zeit pro Analyse: " +
         (watch.ElapsedMilliseconds / 10000) + "ms");

}
```

Nachfolgend das erhaltene Resultat der durchgeführten Tests:

```
Nicht erkannt:          16
Fehlerhaft erkannt:     1
Richtig erkannt:        4
Nicht getestet:         1 (korrupte .eml Datei)

Verarbeitungszeit 10‘000 E-Mails:             00:03:02
Mittlere Verarbeitungszeit pro E-Mail:        18ms
```

9.1.2.5 Auswertung der Bounce Mail Komponenten bezüglich Anforderungen

Bounce Mail Komponenten					
		EasyMail BounceBuster	***MailBee***	***ListNanny***	***NetXtremeBounceFilter***
	Hersteller	*QuikSoft*	*AfterLogic*	*Advanced Intellect LLC*	*Safabyte*
Funktionale Anforderungen		***0=nicht erfüllt 1=erfüllt 1*=erfüllt mit Zusatz-Plug-In***			
	Behandeln und kategorisieren	1	0	1	1
	Bestehender NDR Katalog	1	1	1	1
	Aktualisierung von bestehendem NDR Katalog	Hersteller aktualisiert den NDR Katalog. Download von neuestem NDR Definition File über Homepage möglich	Hersteller aktualisiert den NDR Katalog. Deployment von NDR Katalog unklar	Hersteller aktualisiert den NDR Katalog. Download von neuestem NDR Definition File über Homepage möglich	Aktualisierung des NDR Katalog erfolgt über die jeweilige aktuellste Assembly
	Erweiterung von bestehendem NDR Katalog	Über Regular Expressions mittels der Methode AddSignature() in BounceBuster Klasse (Patterns werden aber nur im Body gesucht)	Dem Hersteller müssen fehlende Patterns gemeldet werden. Der Katalog wird vom Hersteller erweitert	Über Regular Expressions und Strings mittels der Methoden AddNDRStringPattern() und AddNDRRegexPattern() in NDR Klasse Der Suchort der eigenen Patterns kann auch selber definiert werden (Subject, Body, HeaderValues, BodyIntroText, DecodedHeaders)	Über Regular Expressions mittels der Methode AddBounceSignature() in BounceFilter Klasse. Der Suchort der eigenen Patterns kann auch selber definiert werden

	Wie und wo werden die Patterns verwaltet	Proprietäres Dateiformat	XML Datenbank	XML Datei mit binärem Inhalt. Der Hersteller bringt laufend neue NDR Pattern Files heraus	Verwaltung von NDR Katalog unklar.
	Hinzufügen von Typen und Kategorien von NDR Mails	Indem ein User defined Objekt (String) beim Erstellen der Signatur mittels AddSignature() an der Klasse BounceBuster mitgegeben wird	0	0	Indem die Methoden AddBounceType() und AddBounceCategory() an der Klasse BounceFilter aufgerufen wird
	Auslesen der ursprünglichen Header und Content Daten	1	1	1	1
	Varianten zur Übergabe von auszuwertenden Bounce-Messages	Übergabe von Filepfad, Stream, Verzeichnisname und Filemaske an Methode Process() in BounceBuster Klasse	Übergabe von MailMessage Objekt an die Methode Process() in DeliveryStatusParser Klasse MailMessage Objekt kann über Filepfad, Stream oder byte[] erstellt werden	Übergabe von String, Filepfad oder Stream an die Methode Parse(), ParseFile() resp. ParseStream() in NDR Klasse	Übergabe von Filepfad oder MailMessage Objekt an Process() Methode in BounceFilter Klasse Übergabe von MailMessage[], String[] von Filepfaden oder Verzeichnisname an ProcessMessages() in BounceFilter Klasse zum Verarbeiten von mehreren Bounce-Messages Übergabe von Pop3Client oder ImapClient an ProcessMessages() in BounceFilter für direktes Zugreifen auf Postfach.
	Synchrone / Asynchrone Methodenaufrufe	Asynchron	Synchron	Synchron	Synchron / Asynchron

Kriterien		***1=schlecht 2=mittel 3=gut***			
	Dokumentation und Beispiele	3 Sehr viele Beispiele, Forum, API Dokumentation, FAQ	3 Wenige und nicht ausführliche Beispiele, Forum, API Dokumentation	3 Beispiele, Forum, API Dokumentation	3 Beispiele, Forum, API Dokumentation
	Support	2 Forum, KB, Tickets	3 Forum, KB, Tickets	2 Forum, E-Mail	1 Forum, E-Mail
	Preis	$399.00 für eine Developer Licence	$199.00 für eine Developer Licence	$350.00 für eine Developer Licence	$269.00 für eine Developer Licence

Tabelle 36 - Auswertung Anforderungen Bounce Mail Komponenten

9.1.2.6 Auswertung der Performance und Qualität

Bounce Mail Komponente					
	Produktname		***EasyMail BounceBuster***	***ListNanny***	***NetXtremeBounceFilter***
	Hersteller		*QuikSoft*	*Advanced Intellect LLC*	*Safabyte*
Test der Erkennungsgenauigkeit sowie Kategorisierung von realen Bounce-Messages					
Dateiname	**Fehlernachricht im Text**	**Typ**	***0 =Nicht erkannt 1=Fehlerhaft erkannt 2=Richtig erkannt***		
1.eml	Out of Office Message	AutoReply	0	2	2
2.eml	Out of Office Message	AutoReply	0	0	0
3.eml	Out of Office Message	AutoReply	0	2	2
4.eml	Delivery is delayed	Temporary	0	0	2
5.eml	Delivery failed	Hard	2	2	2
6.eml	Can't open mailbox. Temporary Error	Soft	1	2	2
7.eml	Wasn't able to deliver your message to the following addresses. User don't want to receive mails from your address. Permanent Error	Hard	2	2	2
8.eml	Wasn't able to deliver your message to the following addresses. User don't want to receive mails from your address. Permanent Error	Hard	2	2	2
9.eml	Mailbox disk quota exceeded	Soft	1	2	2
10.eml	Delivery not authorized	Soft	-	-	-
11.eml	E-Mail-System des Empfängers unbekannt oder ungültig. Invalid Recipient	Hard	-	-	-
12.eml	mailbox is full and can't accept messages now	Soft	0	2	1
13.eml	The recipient's e-mail address was not found	Hard	0	2	1
14.eml	Mailbox full	Soft	0	2	1
15.eml	The recipient's e-mail address was not found	Hard	0	2	1

16.eml	Mailbox unavailable. (A communication failure occurred during the delivery)	Hard	0	2	1
17.eml	Delivery failed 48 attempts	Hard	2	2	2
18.eml	Unknown host	Hard	0	2	1
19.eml	undeliverable to e-mail address	Hard	2	1	0
20.eml	undeliverable to e-mail address	Hard	2	1	0
21.eml	Unkown User	Hard	2	2	2
22.eml	Your message wasn't delivered because of security policies	Hard	0	2	2
Test der Performance mit 10'000 Bounce-Messages					
	Benötigte Dauer		02:34:37 h	00:06:43 h	00:03:02 h
	Durchschnittliche Dauer pro Bounce Message		927 ms	40 ms	18 ms

Tabelle 37 - Auswertung Performance und Qualität Bounce Mail Komponenten

BounceBuster

```
------------

C:\Temp\quality\1.eml
Status: Unidentified
-----------------------------------
C:\Temp\quality\2.eml
Status: Unidentified
-----------------------------------
C:\Temp\quality\3.eml
Status: Unidentified
-----------------------------------
C:\Temp\quality\4.eml
Status: Unidentified
-----------------------------------
C:\Temp\quality\5.eml
Kategorie: Hard
Typ: Undeliverable
-----------------------------------
C:\Temp\quality\6.eml
Kategorie: Hard
Typ: Undeliverable
-----------------------------------
C:\Temp\quality\7.eml
Kategorie: Hard
Typ: Undeliverable
-----------------------------------
C:\Temp\quality\8.eml
Kategorie: Hard
Typ: Undeliverable
-----------------------------------
C:\Temp\quality\9.eml
Kategorie: Hard
Typ: Undeliverable
-----------------------------------
C:\Temp\quality\10.eml
Kategorie: Hard
Typ: Undeliverable
```

```
-----------------------------------
C:\Temp\quality\12.eml
Status: Unidentified
-----------------------------------
C:\Temp\quality\13.eml
Status: Unidentified
-----------------------------------
C:\Temp\quality\14.eml
Status: Unidentified
-----------------------------------
C:\Temp\quality\15.eml
Status: Unidentified
-----------------------------------
C:\Temp\quality\16.eml
Status: Unidentified
-----------------------------------
C:\Temp\quality\17.eml
Kategorie: Hard
Typ: Undeliverable
-----------------------------------
C:\Temp\quality\18.eml
Status: Unidentified
-----------------------------------
C:\Temp\quality\19.eml
Kategorie: Hard
Typ: Undeliverable
-----------------------------------
C:\Temp\quality\20.eml
Kategorie: Hard
Typ: Undeliverable
-----------------------------------
C:\Temp\quality\21.eml
Kategorie: Hard
Typ: Undeliverable
-----------------------------------
C:\Temp\quality\22.eml
Status: Unidentified
---------------------------------
```

ListNanny

```
---------

C:\Temp\quality\1.eml
Hilfe: autoreply
Typ: AutoResponder
-----------------------------------
C:\Temp\quality\2.eml
Wurde nicht erkannt!
-----------------------------------
C:\Temp\quality\3.eml
Hilfe: this is an automated response
Typ: AutoResponder
-----------------------------------
C:\Temp\quality\4.eml
Wurde nicht erkannt!
-----------------------------------
C:\Temp\quality\5.eml
Hilfe: delivery to the following
recipients failed
Typ: HardBounce
-----------------------------------
C:\Temp\quality\6.eml
Hilfe: unable to deliver
Typ: SoftBounce
-----------------------------------
C:\Temp\quality\7.eml
Hilfe: this is a permanent error
Typ: HardBounce
-----------------------------------
C:\Temp\quality\8.eml
Hilfe: this is a permanent error
Typ: HardBounce
```

```
-----------------------------------
C:\Temp\quality\9.eml
Hilfe: smtp; 552 RCPT
TO:<therese.gilomen@bluemail.ch>
Mailbox disk quota exceeded
Typ: SoftBounce
-----------------------------------
C:\Temp\quality\10.eml
Hilfe: smtp;
Typ: SoftBounce
-----------------------------------
C:\Temp\quality\12.eml
Hilfe: mailbox is full
Typ: SoftBounce
-----------------------------------
C:\Temp\quality\13.eml
Hilfe: address was not found
Typ: HardBounce
-----------------------------------
C:\Temp\quality\14.eml
Hilfe: mailbox is full
Typ: SoftBounce
-----------------------------------
C:\Temp\quality\15.eml
Hilfe: address was not found
Typ: HardBounce
-----------------------------------
C:\Temp\quality\16.eml
Hilfe: mailbox unavailable
Typ: HardBounce
-----------------------------------
```

```
C:\Temp\quality\17.eml
Hilfe: delivery failed 48 attempts
Typ: HardBounce
-----------------------------------
C:\Temp\quality\18.eml
Hilfe: unknown host
Typ: HardBounce
-----------------------------------
C:\Temp\quality\19.eml
Hilfe: undeliverable to hanka@old.ch
Typ: SoftBounce
-----------------------------------
```

```
C:\Temp\quality\20.eml
Hilfe: undeliverable to
steffi@yahoo.com
Typ: SoftBounce
-----------------------------------
C:\Temp\quality\21.eml
Hilfe: unknown user
Typ: HardBounce
-----------------------------------
C:\Temp\quality\22.eml
Hilfe: 5.7.1
Typ: Transient
```

```
BounceFilter
------------

C:\Temp\quality\1.eml
Kategorie: Auto Reply
Typ: Generic
-----------------------------------
Wurde nicht erkannt:
C:\Temp\quality\2.eml
Exception:
-----------------------------------
C:\Temp\quality\3.eml
Kategorie: Auto Reply
Typ: Generic
-----------------------------------
C:\Temp\quality\4.eml
Kategorie: Delayed
Typ: Soft
-----------------------------------
C:\Temp\quality\5.eml
Kategorie: Mailbox Unknow
Typ: Hard
-----------------------------------
C:\Temp\quality\6.eml
Kategorie: Defer
Typ: Soft
-----------------------------------
C:\Temp\quality\7.eml
Kategorie: Anti Spam
Typ: Blocked
-----------------------------------
C:\Temp\quality\8.eml
Kategorie: Anti Spam
Typ: Blocked
-----------------------------------
C:\Temp\quality\9.eml
Kategorie: Full
Typ: Soft
-----------------------------------
C:\Temp\quality\10.eml
Kategorie: Mailbox Unknow
Typ: Hard
-----------------------------------

C:\Temp\quality\12.eml
Kategorie: Delayed
Typ: Soft
-----------------------------------
C:\Temp\quality\13.eml
Kategorie: Delayed
Typ: Soft
-----------------------------------
C:\Temp\quality\14.eml
Kategorie: Delayed
Typ: Soft
-----------------------------------
C:\Temp\quality\15.eml
Kategorie: Delayed
Typ: Soft
```

```
-----------------------------------
C:\Temp\quality\16.eml
Kategorie: Delayed
Typ: Soft
-----------------------------------
C:\Temp\quality\17.eml
Kategorie: Mailbox Unknow
Typ: Hard
-----------------------------------
C:\Temp\quality\18.eml
Kategorie: Delayed
Typ: Soft
-----------------------------------
C:\Temp\quality\19.eml
Kategorie: Delayed
Typ: Soft
-----------------------------------
C:\Temp\quality\20.eml
Kategorie: Delayed
Typ: Soft
-----------------------------------
C:\Temp\quality\21.eml
Kategorie: Mailbox Unknow
Typ: Hard
-----------------------------------
C:\Temp\quality\22.eml
Kategorie: Delayed
Typ: Soft
```

9.1.2.7 Entscheidung

Die Entscheidung der Bounce Mail Komponente fiel auf das Produkt NetXtremeBounceFilter von Safabyte.

Aufgrund der sehr unzuverlässigen Ergebnisse bei der Erkennung von Bounce Typen und Kategorien konnte die Komponente BounceBuster von QuikSoft ausgeschlossen werden. ListNanny von Advanced Intellect wie auch die Komponente NetXtremeBounceFilter von Safabyte liegen bei der korrekten Erkennung der Bounce Nachrichten in etwa gleich auf. Beide zeigen ihre Schwächen bezüglich Erkennungsqualität aber klar.

Das ausschlaggebende Argument für die Wahl des Produktes von Safabyte war die Anforderung, die Bounce Nachrichten genauer erkennen und kategorisieren zu können (das heisst, NetXtremeBounceFilter erlaubt eine genaue Unterscheidung von Hard-Bounces und Soft-Bounces in z.B. Mailbox full etc.). Nur die NetXtremeBounceFilter Komponente erlaubt diese Art der Auswertung ohne eigene Logik implementieren zu müssen. Des Weiteren spricht die schnelle Verarbeitungszeit sowie die etwas geringeren Kosten für Safabyte's NetXtremeBounceFilter.

9.2 Prototypen

9.2.1 Prototyp Risiko 04

Dieser Prototyp hat zum Ziel, das Performance Risiko seitens der verwendeten Mail Komponente zu beseitigen. Die Komponente muss in der Lage sein, mindestens 300‘000 E-Mail Nachrichten pro Stunde zu erzeugen und zu versenden, wobei der Versandprozess für den *Smtp Mailer* beim Übergeben der E-Mail Nachricht an den MTA endet. Der effektive Durchsatz an E-Mails, begonnen beim Generieren der E-Mail Nachricht bis zur tatsächlichen Auslieferung beim Empfänger, kann nicht getestet oder nachvollzogen werden, da viele Fremdparameter mitspielen die nicht im Einflussbereich des Betreibers liegen (SMTP Server Performance, Netzwerkauslastung etc.).

9.2.1.1 Szenario

1. Der Benutzer wählt wie die E-Mail Nachricht versendet werden soll. Dabei stehen folgende Möglichkeiten zur Verfügung:
   - SMTP Message Queue
   - Direkt
   - Multi-Threaded mittels Message Queue
2. Der Benutzer wählt, wie viele E-Mail Nachrichten generiert und versendet werden sollen
3. Der Mail Generator erstellt nach dem Zufallsprinzip die Anzahl E-Mail Nachrichten. Dabei haben die generierten E-Mail Nachrichten eine oder mehrere dieser Eigenschaften:
   - Signatur zum verifizieren der Echtheit
   - Anhang
   - Klartext oder HTML Content
4. Der Mail Generator versendet die generierten E-Mail Nachrichten mittels der vom Benutzer ausgewählten Methode
5. Die Konsolenanwendung gibt für alle Teilschritte die benötigte Ausführungszeit aus

9.2.1.2 Namespaces

Namespace	Beschreibung
Comparis.Framework.SmtpMailer.Prototype.R04	Enthält den Mail Generator, welcher E-Mail Nachrichten generieren und versenden kann sowie eine ausführbare Konsolenanwendung

Tabelle 38 - Namespaces R04 Prototyp

9.2.1.3 Bestandteile

Komponente	Beschreibung
Mail Generator	Generiert E-Mail Nachrichten und versendet die E-Mail Nachrichten
Konsolenanwendung	Nimmt vom Benutzer Generierungs- und Versandeinstellungen entgegen und gibt die Ausführungszeiten aus

Tabelle 39 - Bestandteile R04 Prototyp

9.2.1.4 Probleme

Beim direkten versenden der E-Mail Nachrichten über das SMTP Protokoll konnte in keiner Konfiguration die geforderte Performance von 300‘000 E-Mail Nachrichten pro Stunde erreicht werden. Diese Tests wurden unter verschiedenen Hardwarekonfigurationen sowohl single- wie auch multithreaded durchgeführt. Hier zeichnete sich eindeutig der zu grosse Overhead des SMTP Protokolls als Flaschenhals ab. Auch mittels Bulk-Versand und Pipelining konnte keine Verbesserung erzielt werden.

Die Lösung des Problems liegt darin, die E-Mail Nachrichten direkt in ein Pickup-Verzeichnis beim MTA abzulegen, welches anschliessend vom MTA abgearbeitet wird. Diese Art des Versands wird als Mail Queueing bezeichnet.

9.2.1.5 Performancemessung

Versandart	Beschreibung
SMTP	Die Mail-Komponente verwendet für den Versand einen konfigurierbaren SMTP Server welcher als zentraler Relay dient für den Versand
Queue (Mail Queueing)	Die Mail-Komponente schreibt die zu versendende E-Mail in ein spezielles Datei-Verzeichnis welche dann vom SMTP Server periodisch ausgelesen und für den Versand vorgemerkt wird
Direkt	Die Mail-Komponente führt für sämtliche E-Mail Empfänger eine DNS-Abfrage (MX-Record) durch und versendet die E-Mail Nachricht direkt an den für die Domain zuständigen SMTP Server

Tabelle 40 - Beschreibung der Versandarten

Single-Threaded				
Anzahl	*SMTP*	*Queue*	*Direkt*	
1‘000	~32‘500 M / h	~247‘500 M / h	~96‘000 M / h	
10‘000	~33‘000 M / h	~318‘500 M / h	~69‘000 M / h	
Multi-Threaded über SMTP				
Anzahl	*2 Threads*	*4 Thread*	*8 Threads*	*16 Threads*
1‘000	~35‘000 M / h	~45‘000 M / h	~80‘000 M / h	~65‘000 M / h
10‘000	~38‘500 M / h	~43‘500 M / h	~81‘500 M / h	~58‘000 M / h
Multi-Threaded Queue				
Anzahl	*2 Threads*	*4 Threads*	*8 Threads*	*16 Threads*
1‘000	~541‘000 M / h	~561‘000 M / h	~628‘000 M / h	~399‘000 M / h
10‘000	~495‘000 M / h	~397‘000 M / h	~358‘000 M / h	~467‘000 M / h

Tabelle 41 - Performancemessung R04 Prototyp

9.2.1.6 Erkenntnisse

Durch das gezielte Testen verschiedener Threading Strategien und Auslieferungsarten konnte das Optimum gefunden werden. Dabei wurden Peaks von bis zu 700‘000 E-Mail Nachrichten pro Stunde auf unseren Schulrechnern (Intel Core 2 Duo E6750, 2.66GHz, 4GB RAM, HDD mit 7200rpm) erreicht.

9.2.2 Prototyp Risiko 05

Der Prototyp Risiko 05 hat zum Ziel, zu beweisen, dass die geforderte Performance von 300‘000 Mails pro Stunde zu verarbeiten, erreicht werden kann.

Bis eine E-Mail Nachricht komplett verarbeitet ist, müssen verschiedene Schritte getätigt werden:

- Lesen der nächsten zu versendenden E-Mail Nachricht aus der Datenbank
- Setzen von DispatchDate bei der gelesenen E-Mail Nachricht
- E-Mail Nachricht wird an alle Empfänger versendet
- Bei jedem Empfänger wird das Feld SuccessfulSent auf true gesetzt
- Pro Empfänger wird ein DispatchingState in der Datenbank abgelegt, welcher anzeigt, dass die E-Mail Nachricht an den Empfänger versendet wurde
- Die Message wird als versendet markiert

Damit diese Vorgänge abgebildet werden konnten, musste für den Prototyp bereits schon relativ viel Business- und Datenbank Logik implementiert werden.

9.2.2.1 Szenario

1. Ein Prozess generiert permanent Message DTO Objekte und übergibt diese dem Data Access Layer, damit die Message DTOs in die Datenbank geschrieben werden
    a. Der BLL generiert Message DTOs und assoziiert die Message DTOs korrekt mit den dazugehörigen Recipient DTOs, MessageBody DTOs, Attachment DTOs
    b. Der BLL übergibt die Message DTOs an den DAL, damit die Objekte abgespeichert werden

   c. Der DAL generiert zum Message DTO eine Guid, setzt das CreateDate und speichert die Message DTO inklusive assoziierte DTOs in der Datenbank
   d. Der DAL erstellt pro Recipient ein DispatchingState DTO, welcher anzeigt, dass die Message in der Datenbank abgelegt wurde (DispachtingState Created)
   e. Der DAL gibt die Guid an den BLL zurück
2. Ein bis mehrere Threads, welche sich um unterschiedliche Mail Prioritäten (High, Low) kümmern, fragen die Queue im DAL permanent nach der nächsten zu versendenden Message mit der entsprechenden Priorität.
   a. Der BLL fragt den DAL unter Angabe einer Priorität nach der nächsten zu versendenden Nachricht
   b. Der DAL liest die nächste zu versendende Message anhand der Kriterien ScheduledDispatch und Priorität des Versandprozesses
   c. Der DAL liest die Message sowie assoziierte Recipient, MessageBody und Attachment Datensätze. Der DAL generiert ein Message DTO und assoziiert die dazugehörigen Recipient DTOs, MessageBody DTOs und Attachment DTOs.
   d. Der DAL gibt das Message DTO zurück an den BLL
3. Aus der aus der Queue abgefragten Message DTO wird ein Mail Objekt der verwendeten Mail Komponente erzeugt und versendet
   a. Der BLL erstellt aus der Message DTO ein Mail Objekt der Mail Komponente
   b. Der BLL versendet das Mail Objekt mit der Mail Komponente
   c. Der BLL setzt beim Recipient DTO das SuccessfulSent Feld auf true
   d. Der BLL übergibt das geänderte Recipient DTO an den DAL
   e. Der DAL speichert die Änderungen am Recipient DTO in Datenbank
   f. Der BLL erstellt pro Recipient der zu versendenden Message ein DispatchingState DTO (DispatchingState Sent) und übergibt dieses dem DAL zur Speicherung
   g. Der DAL speichert das DispatchingState DTO
   h. Der BLL markiert die Message als versendet und übergibt die Message an den DAL
   i. Der DAL speichert die geänderte Message

9.2.2.2 Namespaces

Namespace	Beschreibung
Comparis.Framework.SmtpMailer.Prototype.R05.Core	Enthält die Business Logik. Diese beschränkt sich bei diesem Prototyp auf das Delegieren des Aufrufs in den DAL. Zudem befindet sich im Core eine SendingController Klasse, welche Kontrollerthreads und einen Threadpool steuert und so die E-Mail Nachrichten vom DAL abruft und versendet.
Comparis.Framework.SmtpMailer.Prototype.R05.Core.Service	Enthält die Main Methode, um den SendingController zu starten.
Comparis.Framework.SmtpMailer.Prototype.R05.Data	Enthält den Data Access Layer, welcher die Anbindung an die Datenbank mittels ADO.NET realisiert.
Comparis.Framework.SmtpMailer.Prototype.R05.DTO	Enthält die Business Objekte sowie die Enumerationen.
Comparis.Framework.SmtpMailer.Prototype.R05.Mailgenerator	Enthält die Mailgenerierungslogik.
Comparis.Framework.SmtpMailer.Prototype.R05.Service	Beinhaltet die Sending API.

Tabelle 42 - Namespaces R05 Prototyp

9.2.2.3 Bestandteile

Komponente	Beschreibung
MessageHandler (Core)	Enthält die Business Logik und delegiert die Aufrufe an den DAL
SendingController (Core)	Unterhält pro Priorität (High, Low) ein Kontrollerthread und ein Threadpool mit Workerthreads. Die Threadpoolgrösse wurde auf dem Standardwert belassen (250 Threads pro verfügbarem Kern). Über die Kontrollerthreads werden die zu versendenden Nachrichten aus dem DAL abgeholt und zur Versendung den Workerthreads aus dem Threadpool übergeben
MessageDALC (Data)	Enthält die Datenbank Logik. Der MessageDALC (Data Access Layer Component) erlaubt es, die nächste zu versendende Nachricht einer gewissen Priorität aus der Datenbank zu lesen. Des Weiteren werden sonstige CRUD Operationen angeboten (DispatchingState speichern, Recipient updaten, Message updaten)
PriorityQueue (Data)	Stellt eine generische, synchronisierte PriorityQueue dar
Attachment (DTO)	Attachment Data Transfer Object
DispatchState (DTO)	DispatchState Data Transfer Object
Message (DTO)	Message Data Transfer Object. Implementiert das IComparable Interface. Als Vergleichskriterium wird ScheduledDispatch und bei Gleichheit die MessageId verwendet.
MessageBody (DTO)	MessageBody Data Transfer Object.
Recipient (DTO)	Recipient Data Transfer Object.
MailGenerator (MailGenerator)	Enthält Logik, um verschiedenste E-Mail Nachrichten zu generieren (PlainText E-Mails, Multipart E-Mails, E-Mails mit verzögertem ScheduledDispatch)
Sending (Service)	Repräsentiert die Sending API. Sie erlaubt es, E-Mail Nachrichten zu versenden (Übergabe eines Message DTO)

Tabelle 43 - Bestandteile Prototyp Risiko 05

9.2.2.4 Prototypvarianten

Variante 1

Eine erste Variante des Prototyps arbeitet die in der Datenbank abgebildete Queue an Messages sequentiell ab. Das heisst, es wird eine Message gelesen, den Status (dispatched) von 0 auf 1 gesetzt (d.h. die Message wird demnächst versendet) und diese Message dem Thread übergeben, welcher die Message versendet.

Aufgrund von Performanceproblemen (siehe Abschnitt Probleme auf Seite 98) mit der Architektur von Variante 1 wurde eine Variante 2 entwickelt.

Nachtrag:

Eine Lösung des Performanceproblems hängt damit zusammenhängen, dass die in der Datenbank gesetzten Indizes vom Query Optimizer nicht genutzt werden. Mit dem Hint ‚WITH INDEX indexname' kann im SQL Statement die Benutzung des Index befohlen werden. Zusätzlich erhielten wir von Christian Tarnutzer, comparis.ch den Tipp, die Indizes Dispatched und Mailtype als Key Index zusammenzunehmen, Scheduled Dispatch ebenfalls als Index ohne Key einzuführen sowie die Fill Faktoren (20%) entsprechend zu setzen.

Variante 2

Bei jedem Lesevorgang in der Queue (d.h. in der Tabelle Message in der Datenbank) werden alle möglichen zu versendenden E-Mail Nachrichten gelesen, welche auf die Kriterien passen. Diese E-Mail Nachrichten werden in eine Collection (PriorityQueue) abgelegt. Der Business Layer liest

schliesslich die E-Mail Nachrichten aus der Collection im Speicher und verarbeitet die E-Mail Nachrichten.

Auch mit dieser Variante blieb ein Problem bestehen. Es müssten Absicherungsmechanismen eingebaut werden, um ein Abstürzen einer *Smtp Mailer* Instanz zu erkennen und entsprechend reagieren zu können (siehe Abschnitt Probleme auf Seite 98). Um das Problem zu lösen wären grössere Absicherungsmechanismen notwendig. Deshalb wurde eine Variante 3 entwickelt.

Variante 3

Die Variante 3 implementiert die Queue der zu versendenden Nachrichten direkt in der MS SQL Datenbank als Tabelle. Pro Priorität (High, Low) wird eine Queue resp. Tabelle geführt. Alle Versandprozesse greifen auf die Tabellen zu um die nächste zu versendende Nachricht zu erhalten.

9.2.2.5 Probleme

Mit der Variante 1 konnte die Performanceanforderung von 300‘000 E-Mail Nachrichten pro Stunde (resp. rund 85 E-Mail Nachrichten pro Sekunde) nicht erreicht werden. Die Verarbeitungsgeschwindigkeit nahm mit der Anzahl bereits verarbeiteter E-Mail Nachrichten in der Queue ab. Nun war sehr viel Nachforschungsarbeit notwendig um das Problem zu lokalisieren. Es wurden verschiedenste Performancemessungen durchgeführt. Das Problem war nicht sehr offensichtlich. Schlussendlich ist es aber damit zu begründen, dass die Datenbank ein Full Scan macht und immer länger nach einer nicht versendeten Nachricht suchen musste, je mehr E-Mail Nachrichten bereits verarbeitet waren.

Nachtrag:
Mit den Tipps von Christian Tarnutzer, comparis.ch AG, konnten die Performanceprobleme von Variante 1 behoben werden. Weil die Variante 1 am wenigsten Komplexität benötigt, um die Message Queue abzuarbeiten, haben wir uns zusammen mit Benedikt Unold dafür entschieden, mit dieser Variante weiterzufahren.

Durch die Architekturumstellung auf die Variante 2 konnten die Performanceanforderungen erreicht werden. Zudem bleibt die Verarbeitungsgeschwindigkeit nun konstant. Das Problem bei dieser Variante 2 stellt aber die verteilte Queue dar. Denn es können mehrere Prozesse mit einer Queue gestartet werden, welche auf dieselbe Datenbank zugreifen und Messages in die Queue laden um die Nachrichten zu versenden. Wenn nun ein solcher Prozess abstürzt, so bleiben diese Messages unversendet, weil die Messages in der Datenbank markiert werden sobald sie in eine Queue geladen werden. Markierte Messages werden nicht mehr gelesen, weil sie ja demnächst versendet werden sollten.

9.2.2.6 Erkenntnisse

Dank der Implementierung dieses Prototyps konnte bewiesen werden, dass die Performanceanforderungen eingehalten werden können. Die Messungen wurden auf einem Schulrechner durchgeführt (Intel Core 2 Duo E6750, 2.66 GHz, 4GB RAM, HDD mit 7200rpm). Weil bei comparis.ch die Software schlussendlich auf wesentlich performanteren Systemen zum Einsatz kommt, werden dort erst recht die Anforderungen eingehalten.

Zudem wurde eine geeignete Architektur gefunden, welche nun die Implementierung des Mail Moduls vereinfacht, weil viel Business Logik und Datenbank Logik Code aus dem Prototyp übernommen werden kann. Es konnten durch die Entwicklung des Prototyps bereits einige Probleme aus dem Weg geschafft werden.

9.2.2.7 Performancemessungen

Hinweis zu den nachfolgenden Messprotokollen: Es wurden teilweise nicht alle Messungen durchgeführt sowie nicht überall der genaue Performancewert berechnet. Manchmal brauchte es nur

wenige Messungen um die Tendenz der getesteten Konfiguration zu erkennen. Die Einheit M / h bedeutet Anzahl verarbeiteter Messages pro Stunde.

Lesen der Nachricht und Statusupdate mit Variante 1 und 2

In einem ersten Test wurde das reine Lesen der zu versendenden Nachrichten aus der Datenbank und das Updaten des Status der Nachricht auf Dispatched = 1 getestet.

Der Test wurde auf einem Laptop durchgeführt (Intel Core 2 Duo T7200, 2.0 GHz, 2 GB RAM, HDD 5400rpm).

Konfiguration:

Es wurden vor jedem Testschritt 5000 E-Mail Nachrichten generiert und in die Datenbank geschrieben. Danach wurde die Zeit gemessen, um die Nachrichten alle abzuarbeiten. Zu beachten ist, dass die Abarbeitung von einem Prozess erfolgte und somit es sich um eine Single-Threaded Anwendung handelt.

Single-Threaded (Variante 1)				
Testschritt	*Anzahl Datensätze in Datenbank*	*Anzahl bereits verarbeiteter Datensätze in Datenbank*	*Dauer der Verarbeitung*	*Performance*
1	5‘000	0	07.12 sec	~2‘500‘000 M / h
2	10‘000	5‘000	09.61 sec	~1‘870‘000 M / h
3	15‘000	10‘000	11.06 sec	~1‘620‘000 M / h
4	20‘000	15‘000	14.51 sec	~1‘240‘000 M / h

Tabelle 44 - Performancemessung Risiko 05 Prototyp Variante 1

Es ist an der Performance deutlich zu erkennen, dass die Verarbeitung der jeweils 5000 Nachrichten immer länger dauert, je mehr bereits verarbeitete Nachrichten sich in der Datenbank befinden.

Der gleiche Test wurde mit der umgestellten Architektur durchgeführt (Variante 2). Dabei werden alle Nachrichten von der Datenbank in eine Collection gelesen, welche verarbeitet werden können. Aus dieser Collection werden schliesslich die Messages gelesen und verarbeitet.

Single-Threaded (Variante 2)				
Testschritt	*Anzahl Datensätze in Datenbank*	*Anzahl bereits verarbeiteter Datensätze in Datenbank*	*Dauer der Verarbeitung*	*Performance*
1	5‘000	0	05.13 sec	3‘500‘000 M / h
2	10‘000	5‘000	05.08 sec	3‘500‘000 M / h
3	15‘000	10‘000	04.99 sec	3‘600‘000 M / h
4	20‘000	15‘000	04.74 sec	3‘800‘000 M / h
5	25‘000	20‘000	05.37 sec	3‘400‘000 M / h
6	30‘000	25‘000	04.96 sec	3‘600‘000 M / h
7	35‘000	30‘000	05.69 sec	3‘200‘000 M / h
8	40‘000	35‘000	05.48 sec	3‘400‘000 M / h
9	45‘000	40‘000	05.43 sec	3‘400‘000 M / h
10	50‘000	45‘000	05.05 sec	3‘500‘000 M / h
11	55‘000	50‘000	05.84 sec	3‘100‘000 M / h
12	60‘000	55‘000	05.39 sec	3‘400‘000 M / h
13	65‘000	60‘000	05.78 sec	3‘100‘000 M / h
14	70‘000	65‘000	05.38 sec	3‘400‘000 M / h
15	75‘000	70‘000	04.87 sec	3‘700‘000 M / h
16	80‘000	75‘000	04.63 sec	3‘900‘000 M / h
17	85‘000	80‘000	05.54 sec	3‘200‘000 M / h

18	90‘000	85‘000	04.80 sec	3‘800‘000 M / h
19	95‘000	90‘000	05.07 sec	3‘500‘000 M / h
20	100‘000	95‘000	04.92 sec	3‘600‘000 M / h
21	50‘000	0	50.06 sec	3‘600‘000 M / h
22	100‘000	0	111.13 sec	3‘250‘000 M / h

Tabelle 45 - Performancemessung Risiko 05 Prototyp Variante 2

Komplettes Verarbeiten von Nachrichten mit Variante 1

Der Test wurde auf mehreren Schulrechnern durchgeführt (Intel Core 2 Duo E6750, 2.66 GHz, 4GB RAM, HDD mit 7200rpm).

Auf einem Schulrechner läuft der Service, welcher die Kontrollerthreads und den Threadpool startet und verwaltet. Auf einem zweiten Schulrechner läuft der MS SQL Server. Auf einen dritten Schulrechner werden die E-Mail Nachrichten versendet (in ein Directory abgelegt).

Konfiguration:

Pro Priorität (High, Low) wird ein Kontrollerthread gestartet. Der Threadpool enthält 250 Threads pro verfügbarem CPU Kern (Standard).

Die generierten Testmails in der Datenbank sind zur Hälfte High Priority Mails und zur Hälfte Low Priority Mails.

Messung mit einzelnen Indizes auf Dispatched und Mailtype in der Datenbank:

Multi-Threaded (Variante 1)				
Testschritt	*Anzahl Datensätze in Datenbank*	*Anzahl bereits verarbeiteter Datensätze in Datenbank*	*Dauer der Verarbeitung*	*Performance*
1	5‘000	0	41.00 sec	~M / h
2	10‘000	5‘000	45.20 sec	~M / h
3	15‘000	10‘000	48.81 sec	~M / h
4	20‘000	15‘000	50.58 sec	~M / h
5	25‘000	20‘000	60:00 sec	~M / h
6	30‘000	25‘000	75.50 sec	~M / h
7	50‘000	0	1119.78 sec	~M / h

Tabelle 46 - Performancemessung Risiko 05 Prototyp Variante 1

Messung mit einzelnen Indizes auf Dispatched, Mailtype und ScheduledDispatch in der Datenbank:

Multi-Threaded (Variante 1)				
Testschritt	*Anzahl Datensätze in Datenbank*	*Anzahl bereits verarbeiteter Datensätze in Datenbank*	*Dauer der Verarbeitung*	*Performance*
1	5‘000	0	112.57 sec	~M / h
2	10‘000	5‘000	229.84 sec	~M / h
3	15‘000	10‘000	sec	~M / h
4	25‘000	20‘000	sec	~M / h
5	50‘000	0	sec	~M / h

Tabelle 47 - Performancemessung Risiko 05 Prototyp Variante 1

Messung mit Index nur auf Dispatched in der Datenbank:

Multi-Threaded (Variante 1)				
Testschritt	*Anzahl Datensätze in Datenbank*	*Anzahl bereits verarbeiteter Datensätze in Datenbank*	*Dauer der Verarbeitung*	*Performance*
1	5‘000	0	47.68 sec	~M / h
2	10‘000	5‘000	46.68 sec	~M / h
3	15‘000	10‘000	44.11 sec	~M / h
4	20‘000	15‘000	49.38 sec	~M / h
5	25‘000	20‘000	41.77 sec	~M / h
6	30‘000	25‘000	41.05 sec	~M / h
7	50‘000	0	1090.09 sec	~M / h

Tabelle 48 - Performancemessung Risiko 05 Prototyp Variante 1

Messung mit kombinierten Indizes sowie Angabe von Fill Faktoren (gemäss Christian Tarnutzer, comparis.ch AG):

Multi-Threaded (Variante 1)				
Testschritt	*Anzahl Datensätze in Datenbank*	*Anzahl bereits verarbeiteter Datensätze in Datenbank*	*Dauer der Verarbeitung*	*Performance*
1	5‘000	0	27.27 sec	~660‘000 M / h
2	10‘000	5‘000	20.66 sec	~870‘000 M / h
3	15‘000	10‘000	29.26 sec	~620‘000 M / h
4	20‘000	15‘000	37.89 sec	~480‘000 M / h
5	25‘000	20‘000	29.14 sec	~620‘000 M / h
6	30‘000	25‘000	28.65 sec	~620‘000 M / h
7	35‘000	30‘000	23.24 sec	~780‘000 M / h
8	40‘000	35‘000	25.51 sec	~700‘000 M / h
9	45‘000	40‘000	27.10 sec	~660‘000 M / h
10	50‘000	45‘000	36.48 sec	~480‘000 M / h
11	55‘000	50‘000	21.05 sec	~850‘000 M / h
12	60‘000	55‘000	25.60 sec	~700‘000 M / h
13	65‘000	60‘000	23.48 sec	~780‘000 M / h
14	70‘000	65‘000	23.40 sec	~780‘000 M / h
15	75‘000	70‘000	24.64 sec	~750‘000 M / h
16	80‘000	75‘000	39.28 sec	~460‘000 M / h
18	90‘000	85‘000	19.08 sec	~940‘000 M / h
19	95‘000	90‘000	21.37 sec	~850‘000 M / h
20	100‘000	95‘000	27.07 sec	~650‘000 M / h
21	50‘000	0	471.18 sec	~382‘000 M / h
22	100‘000	50‘000	446.12 sec	~400‘000 M / h
23	100‘000	0	1056.45 sec	~340‘000 M / h
24	200‘000	100‘000	1127.93 sec	~320‘000 M / h

Tabelle 49 - Performancemessung Risiko 05 Prototyp Variante 1

Messung wie oben aber mit DTC (TransactionScope) wobei als Isolation Level ReadCommitted genutzt wird:

Multi-Threaded (Variante 1)				
Testschritt	*Anzahl Datensätze in Datenbank*	*Anzahl bereits verarbeiteter Datensätze in Datenbank*	*Dauer der Verarbeitung*	*Performance*
1	5‘000	0	42.12 sec	~430‘000 M / h

2	10‘000	5‘000	39.94 sec	~450‘000 M / h
3	15‘000	10‘000	43.65 sec	~430‘000 M / h
4	20‘000	15‘000	42.14 sec	~430‘000 M / h
5	25‘000	20‘000	35.42 sec	~510‘000 M / h
6	100‘000	95‘000	sec	~M / h
7	50‘000	0	549.54 sec	~330‘000 M / h
8	100‘000	50‘000	sec	~M / h
9	100‘000	0	sec	~M / h
10	200‘000	100‘000	sec	~M / h

Tabelle 50 - Performancemessung Risiko 05 Prototyp Variante 1

Messung wie oben mit DTC (TransactionScope) wobei als Isolation Level ReadUncommitted genutzt wird:

Multi-Threaded (Variante 1)				
Testschritt	*Anzahl Datensätze in Datenbank*	*Anzahl bereits verarbeiteter Datensätze in Datenbank*	*Dauer der Verarbeitung*	*Performance*
1	5‘000	0	40.54 sec	~450‘000 M / h
2	10‘000	5‘000	40.80 sec	~450‘000 M / h
3	15‘000	10‘000	44.18 sec	~400‘000 M / h
4	20‘000	15‘000	46.91 sec	~350‘000 M / h
5	25‘000	20‘000	51.00 sec	~M / h
6	30‘000	25‘000	sec	~M / h
7	35‘000	30‘000	sec	~M / h
8	40‘000	35‘000	sec	~M / h
9	45‘000	40‘000	sec	~M / h
10	50‘000	45‘000	sec	~M / h
11	50‘000	0	554.42 sec	~325‘000 M / h
12	100‘000	50‘000	sec	~M / h
13	100‘000	0	sec	~M / h
14	200‘000	100‘000	sec	~M / h

Tabelle 51 - Performancemessung Risiko 05 Prototyp Variante 1

Messung wie oben mit DTC (TransactionScope) wobei als Isolation Level RepeatableRead genutzt wird:

Multi-Threaded (Variante 1)				
Testschritt	*Anzahl Datensätze in Datenbank*	*Anzahl bereits verarbeiteter Datensätze in Datenbank*	*Dauer der Verarbeitung*	*Performance*
1	5‘000	0	43.40 sec	~ 420‘000 M / h
2	10‘000	5‘000	38.68 sec	~ 470‘000 M / h
3	15‘000	10‘000	35.87 sec	~ 520‘000 M / h
4	20‘000	15‘000	35.31 sec	~ 520‘000 M / h
5	25‘000	20‘000	34.01 sec	~ 530‘000 M / h
6	50‘000	0	556.37 sec	~320‘000 M / h
7	100‘000	50‘000	sec	~M / h
8	100‘000	0	sec	~M / h
9	200‘000	100‘000	sec	~M / h

Tabelle 52 - Performancemessung Risiko 05 Prototyp Variante 1

Komplettes Verarbeiten von Nachrichten mit Variante 2

Der Test wurde auf mehreren Schulrechnern durchgeführt (Intel Core 2 Duo E6750, 2.66 GHz, 4GB RAM, HDD mit 7200rpm).

Auf einem Schulrechner läuft der Service, welcher die Kontrollerthreads und den Threadpool startet und verwaltet. Auf einem zweiten Schulrechner läuft der MS SQL Server. Auf einen dritten Schulrechner werden die E-Mail Nachrichten versendet (in ein Directory abgelegt).

Konfiguration:

Pro Priorität (High, Low) wird ein Kontrollerthread gestartet. Der Threadpool enthält 250 Threads pro verfügbaren CPU Kern (Standard).

Die generierten Testmails in der Datenbank sind zur Hälfte High Priority Mails und zur Hälfte Low Priority Mails.

Testschritt	*Anzahl Datensätze in Datenbank*	*Anzahl bereits verarbeiteter Datensätze in Datenbank*	*Dauer der Verarbeitung*	*Performance*
1	5‘000	0	32.07 sec	~550‘000 M / h
2	10‘000	5‘000	31.52 sec	~570‘000 M / h
3	15‘000	10‘000	32.34 sec	~550‘000 M / h
4	20‘000	15‘000	38.47 sec	~470‘000 M / h
5	25‘000	20‘000	33.76 sec	~530‘000 M / h
6	30‘000	25‘000	30.95 sec	~570‘000 M / h
7	35‘000	30‘000	28.30 sec	~630‘000 M / h
8	40‘000	35‘000	27.94 sec	~630‘000 M / h
9	45‘000	40‘000	31.37 sec	~570‘000 M / h
10	50‘000	45‘000	28.55 sec	~630‘000 M / h
11	50‘000	0	420.38 sec	~430‘000 M / h
12	100‘000	0	906.36 sec	~400‘000 M / h

Tabelle 53 - Performancemessung Risiko 05 Prototyp Variante 2

Komplettes Verarbeiten von Nachrichten mit Variante 3

Der Test wurde auf mehreren Schulrechnern durchgeführt (Intel Core 2 Duo E6750, 2.66 GHz, 4GB RAM, HDD mit 7200rpm).

Auf einem Schulrechner läuft der Service, welcher die Kontrollerthreads und den Threadpool startet und verwaltet. Auf einem zweiten Schulrechner läuft der MS SQL Server. Auf einen dritten Schulrechner werden die E-Mail Nachrichten versendet (in ein Directory abgelegt).

Konfiguration:

Pro Priorität (High, Low) wird ein Kontrollerthread gestartet. Der Threadpool enthält 250 Threads pro verfügbaren CPU Kern (Standard).

Die generierten Testmails in der Datenbank sind zur Hälfte High Priority Mails und zur Hälfte Low Priority Mails.

Testschritt	*Anzahl Datensätze in Datenbank*	*Anzahl bereits verarbeiteter Datensätze in Datenbank*	*Dauer der Verarbeitung*	*Performance*
1	5‘000	0	29.32 sec	~610‘000 M / h
2	10‘000	5‘000	30.62 sec	~570‘000 M / h
3	15‘000	10‘000	31.10 sec	~580‘000 M / h
4	20‘000	15‘000	34.53 sec	~520‘000 M / h
5	25‘000	20‘000	32.73 sec	~550‘000 M / h

6	30‘000	25‘000	31.28 sec	~580‘000 M / h
7	35‘000	30‘000	29.92 sec	~610‘000 M / h
8	40‘000	35‘000	32.57 sec	~550‘000 M / h
9	45‘000	40‘000	35.35 sec	~510‘000 M / h
10	50‘000	45‘000	34.64 sec	~520‘000 M / h
11	55‘000	50‘000	36.32 sec	~500‘000 M / h
12	50‘000	0	421.63 sec	~170‘000 M / h
12	100‘000	0	926.87 sec	~390‘000 M / h

Tabelle 54 - Tabelle 15 - Performancemessung Risiko 05 Prototyp Variante 3

9.3 Mail Modul

9.3.1 Bestimmung der Mail Priorität

Die Mail Priorität wird jeweils beim Erfassen der zu versendenden Nachricht in der Datenbank vermerkt. Somit muss beim Lesen der nächsten zu versendenden Nachricht nicht mehr zuerst die Mail Priorität der Nachrichten bestimmt werden.

Die Bestimmung der Mail Priorität leitet sich aus den in der Konfiguration definierten Priority Rules ab. Diese Priority Rules werden von oben nach unten durchlaufen. Trifft die zu versendende Nachricht auf eine Priority Rule zu, so wird diese angewendet resp. gibt diese die Mail Priorität an.

9.3.2 Beispiel

Es soll für die folgende Beispielkonfiguration beschrieben werden, welche Regel für nachfolgende Nachrichten angewendet wird resp. welch Mail Priorität sich aus der Beispielkonfiguration ergibt:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <smtpmailer>
   <priorityrules>
    <rule applicationcode="APPX" mailtype="PWFORGOTTEN" priority="High"/>
    <rule applicationcode="*" mailtype="NEWSLETTERHIGH" priority="High"/>
    <rule applicationcode="*" mailtype="*" priority="Low"/>
   </priorityrules>
  </smtpmailer>
</configuration>
```

Code Snippet 1 - Beispielkonfiguration

Applikationscode	Mailtyp	Priorität	Erklärung
APPX	PWFORGOTTEN	High	Die erste Regel trifft zu
APPY	PWFORGOTTEN	Low	Die letzte Regel (catch-all Regel trifft zu)
APPZ	NEWSLETTERHIGH	High	Die zweite Regel trifft zu
APPX	KK	Low	Die letzte Regel (catch-all Regel trifft zu)

Tabelle 55 - Beispielnachrichten und ihre Mail Priorität

9.4 Ndr Modul

9.4.1 Qualitätsindikator Algorithmus

In diesem Abschnitt soll festgelegt werden, wie der Qualitätsindikator für eine E-Mail Adresse bestimmt wird.

Name	Beschreibung
Good	Die E-Mail Adresse ist von guter Qualität.
Poor	Die E-Mail Adresse ist von mittelmässiger Qualität. Es treten ab und zu Fehler auf (das heisst, es kommen Bounce Nachrichten von E-Mail Nachrichten zurück, welche an die E-Mail Adresse gesendet wurden)
Failed	Die E-Mail Adresse ist von unbrauchbarer Qualität. Es treten permanent Hard-Bounces auf (z.B. Empfänger unbekannt)

Tabelle 56 - Qualitätsstufen einer E-Mail Adresse

9.4.2 Definition des Algorithmus

In welche Qualitätsstufe eine E-Mail Adresse fällt soll mittels eines Zahlenwerts bestimmt werden. Dazu werden Schwellenwerte definiert. Anhand des Schwellenwerts kann bestimmt werden, welche Qualitätsstufe der E-Mail Adresse zugeordnet wird. Es wird also festgelegt, ab welchem Zahlenwert die E-Mail Adresse von Good in die Qualitätsstufe Poor fällt und bei welchem Zahlenwert von Poor in die Qualitätsstufe Failed. Zu Beginn, das heisst, wenn eine E-Mail Adresse in die Datenbank eingetragen wird, besitzt sie die Qualitätsstufe Good.

Für die verschiedenen Bounce Typen (Hard, Soft, Blocked etc.), die gemäss der verwendeten Bounce Komponente NetXtremeBounceFilter von Safabyte auftreten können, werden Gewichtungen definiert.

Zur Bestimmung des Zahlenwerts wurden verschiedene Alternativen erarbeitet. Diese sind in den nachfolgenden Abschnitten beschrieben (Variante 1 – 3 zur Bestimmung des Zahlenwertes).

Diese Gesamtsumme bestimmt dann gemäss der Schwellenwerte für Poor und Failed, in welche Qualitätsstufe die E-Mail Adresse fällt.

9.4.2.1 Konfigurationswerte

Es ergeben sich aus obiger Beschreibung folgende konfigurierbaren Werte:

Konfigurationsvariable	Beschreibung
Alter	Definiert die Anzahl Tage, welche seit der letzten Berechnung vergangen sein müssen, damit der Qualitätsindikator für eine E-Mail Adresse neu berechnet wird
Strategie	Definiert, welche Strategie dass verwendet werden soll um den Qualitätsindikator zu berechnen gemäss dem Strategy Pattern (Wikipedia, 2009)
Schwellenwert	Gibt an, ab wann eine E-Mail Adresse als Poor oder Failed eingestuft werden soll.
Gewichtung für Bounce Typen	Jeder Bounce-Typ hat eine Gewichtung zugeordnet. Zur Auswahl stehen Low, Medium, High oder Ignore.

Tabelle 57 - Konfigurationswerte Qualitätsindikator Algorithmus

9.4.2.2 Zeitpunkt der Berechnung des Qualitätsindikators

Die Applikation, welche den Qualitätsindikator berechnet, wird regelmässig (z.B. wöchentlich) ausgeführt.

9.4.2.3 Variante 1 Bestimmung des Zahlenwertes

Eine erste Variante zur Bestimmung des Zahlenwertes berechnet aus allen zu einer E-Mail Adresse gehörenden DispatchingStates (Versandhistory) einen Wert. Dazu wird die Summe aus den Gewichten für die aufgetretenen Soft- resp. Hardbounces gebildet. Diese Summe wird durch die Anzahl Mails geteilt, welche an die E-Mail Adresse gesendet wurden. Besitzt der zuletzt eingetragene DispatchingState den Status Sent, so wird die Berechnung des Zahlenwertes nicht durchgeführt und der Qualitätsindikator ist good.

Der Vorteil dieser Variante ist es, dass so der Zahlenwert einfach berechnet werden kann. Der Programmieraufwand ist minimal.

Der Nachteil ist, dass bei sehr vielen versendeten E-Mail Nachrichten Hardbounces zu wenig Gewicht erhalten können. Ergaben z.B. die letzten 20 gesendeten Mails alles Hardbounces, wobei zuvor 100 Mails erfolgreich versendet werden können, so erhalten diese 20 Hardbounces wenig Gewicht. Dies obwohl die E-Mail Adresse in letzter Zeit überhaupt nicht mehr zu funktionieren scheint. Genauso als

Nachteil dieser Variante kann angeschaut werden, dass z.B. 20 Hardbounces am Anfang der Analysephase als auch 20 Hardbounces am Ende der Analysephase gleich gewichtet werden.

9.4.2.4 Variante 2 Bestimmung des Zahlenwertes

Eine zweite Variante zur Bestimmung des Zahlenwertes analysiert immer nur diejenigen DispatchingStates bis und mit dem letzten DispatchingState, welcher den Status Sent enthält (d.h. der DispatchingState der letzten erfolgreichen Nachricht). Besitzt der zuletzt eingetragene DispatchingState den Status Sent, so wird die Berechnung des Zahlenwertes nicht durchgeführt und der Qualitätsindikator ist good.

Der Vorteil dieser Variante ist es, dass die zuletzt eingetragenen DispatchingStates mehr Gewicht erhalten (weil nicht immer alle DispatchingStates analysiert werden).

Die Nichtbeachtung der Vergangenheit wirkt sich aber gleichzeitig auch als Nachteil aus. Treten z.B. in den letzten 2 Minuten der Analysephase einige Hardbounces auf, so wird die E-Mail Adresse zu einem Failed klassiert obwohl sonst die ganze Woche die E-Mail Nachrichten an die E-Mail Adresse problemlos gesendet werden konnten.

9.4.2.5 Variante 3 Bestimmung des Zahlenwertes

Diese Variante zur Bestimmung des Qualitätsindikators berücksichtigt sowohl die aktuelle Situation (alle Bounce Nachrichten bis zum letzten erfolgreichen Versenden) wie auch die Vergangenheit (alle Bounce Nachrichten von dem letzten erfolgreichen Versand bis zum Anfang der Aufzeichnung).

Für die Berechnung der aktuellen Situation werden die Bounce Nachrichten gruppiert, welche sich innerhalb eines konfigurierbaren Zeitraumes befinden. Der Schlusswert wird errechnet durch das Summieren aller Gewichtungs-Mittelwerte der Gruppen. Diese Summe führt gemäss der Konfiguration des Schwellwertes zu einem ersten Indikator (Good, Poor oder Failed).

Bei den Bounce Nachrichten aus der Vergangenheit wird wieder das oben beschriebene Schema verwendet um Nachrichten innerhalb eines Zeitraumes zu gruppieren (nur Bounce Nachrichten werden gruppiert, erfolgreiche werden einzeln miteinbezogen). Jedoch wird aus der Summe aller errechneten Werte der Mittelwert genommen, welcher sich entsprechend zwischen 0 und der in der Konfiguration definierten High-Gewichtung befinden wird. Daraus resultiert ein zweiter Indikator (Good, Poor, Failed). Diese Zuordnung ist jedoch fix im Algorithmus hinterlegt als einen von den konfigurierten Gewichtungen abhängigen Wert.

Aus den beiden errechneten Indikatoren ergibt sich das Endresultat nach folgender Tabelle:

Aktuell	*Good*	*Good*	*Good*	*Poor*	*Poor*	*Poor*	*Failed*	*Failed*	*Failed*
Vergangenheit	*Good*	*Poor*	*Failed*	*Good*	*Poor*	*Failed*	*Good*	*Poor*	*Failed*
Ergebnis	Good	Good	Good	Poor	Poor	Poor	Poor	Failed	Failed

9.5 Paper Prototype

Zur Überprüfung der geplanten Benutzeroberfläche der Kundendienst Webapplikation wurde ein Paper Prototype entwickelt. Das Ziel war es, den Kundendienstmitarbeiter beim Arbeiten mit dem Paper Prototype zu beobachten und herauszufinden, ob ihm die Beschriftungen und Übergänge in der Benutzeroberfläche klar sind. Dabei wurden die Szenarien ‚Nachricht abfragen' und ‚Blacklist konfigurieren' verwendet siehe Abschnitt 7.4.3.1 Szenario Nachricht abfragen auf Seite 55 und 7.4.3.2 Szenario Blacklist konfigurieren auf Seite 55.

9.5.1 Fotos vom Paper Prototype

Nachfolgend sind alle Paper Prototypen als Fotografie dokumentiert:

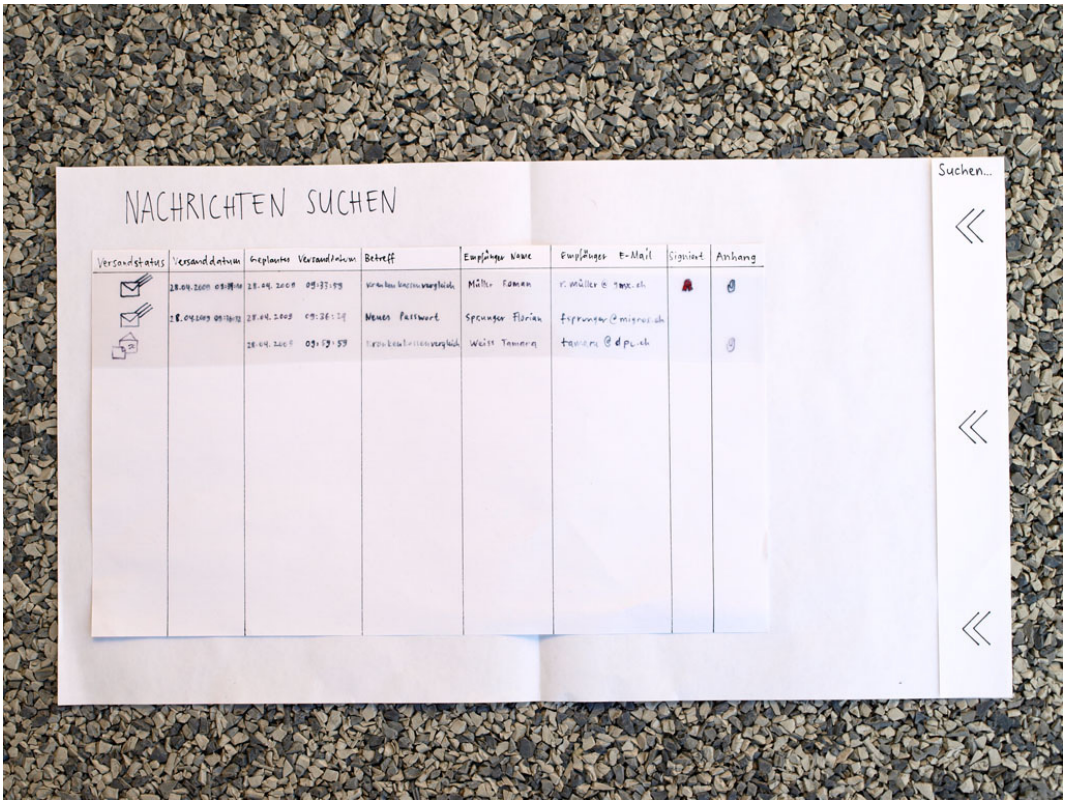


Abbildung 38 - Paper Prototype Nachrichten suchen

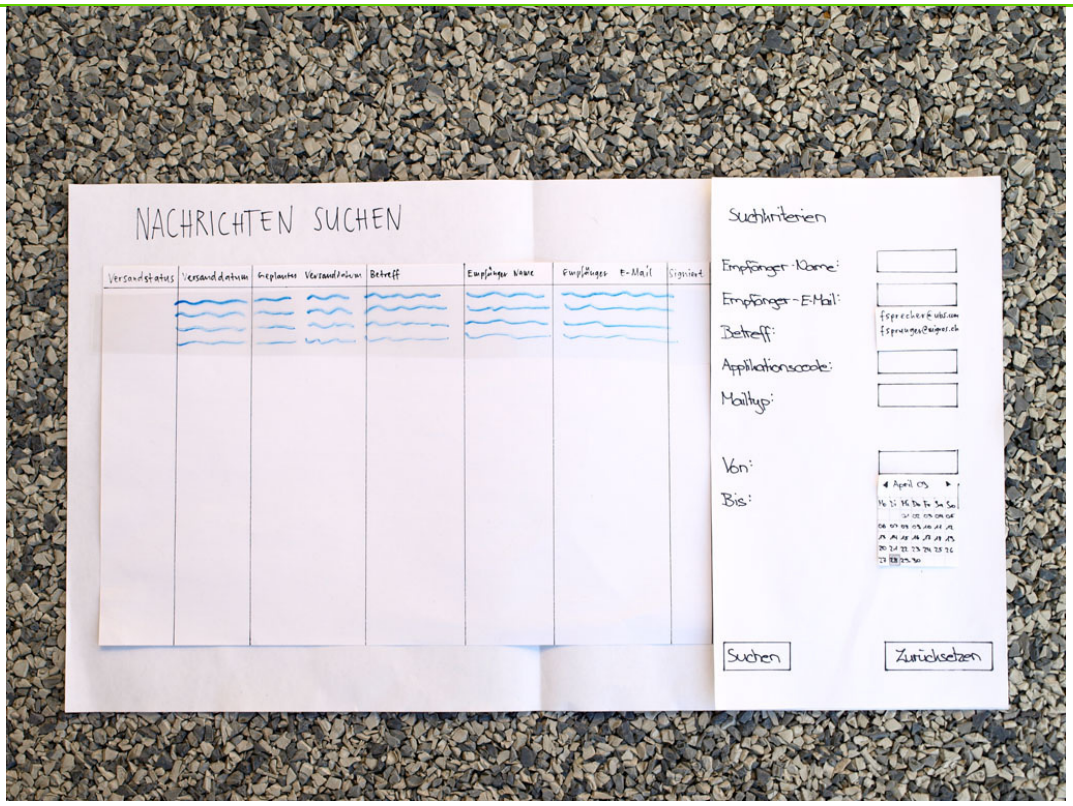


Abbildung 39 - Paper Prototype Nachrichten suchen mit ausgefahrener Search Bar

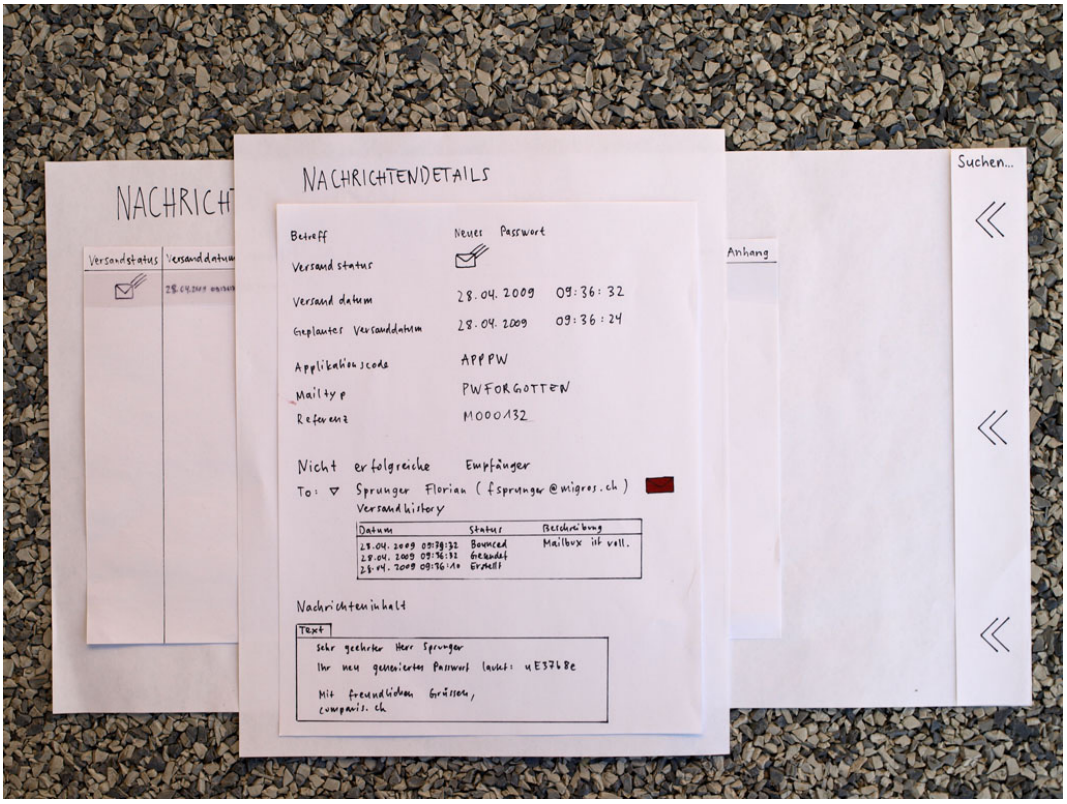


Abbildung 40 - Paper Prototype Nachrichtendetails

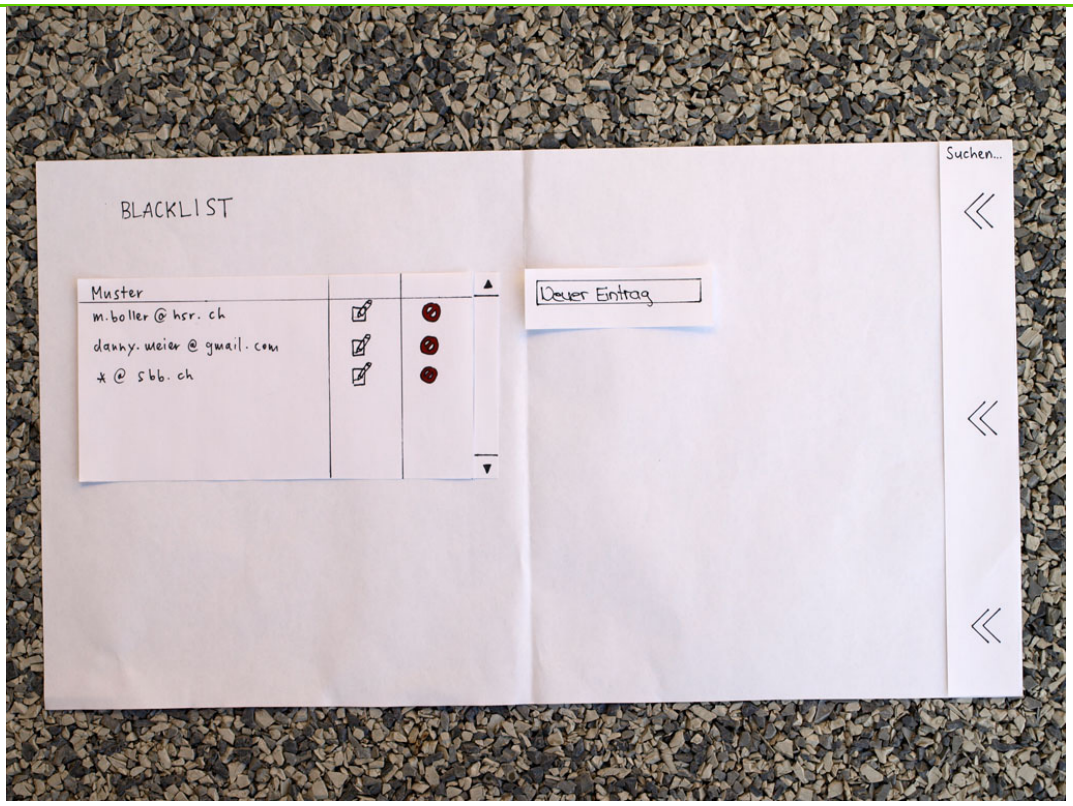


Abbildung 41 - Paper Prototype Blacklist konfigurieren

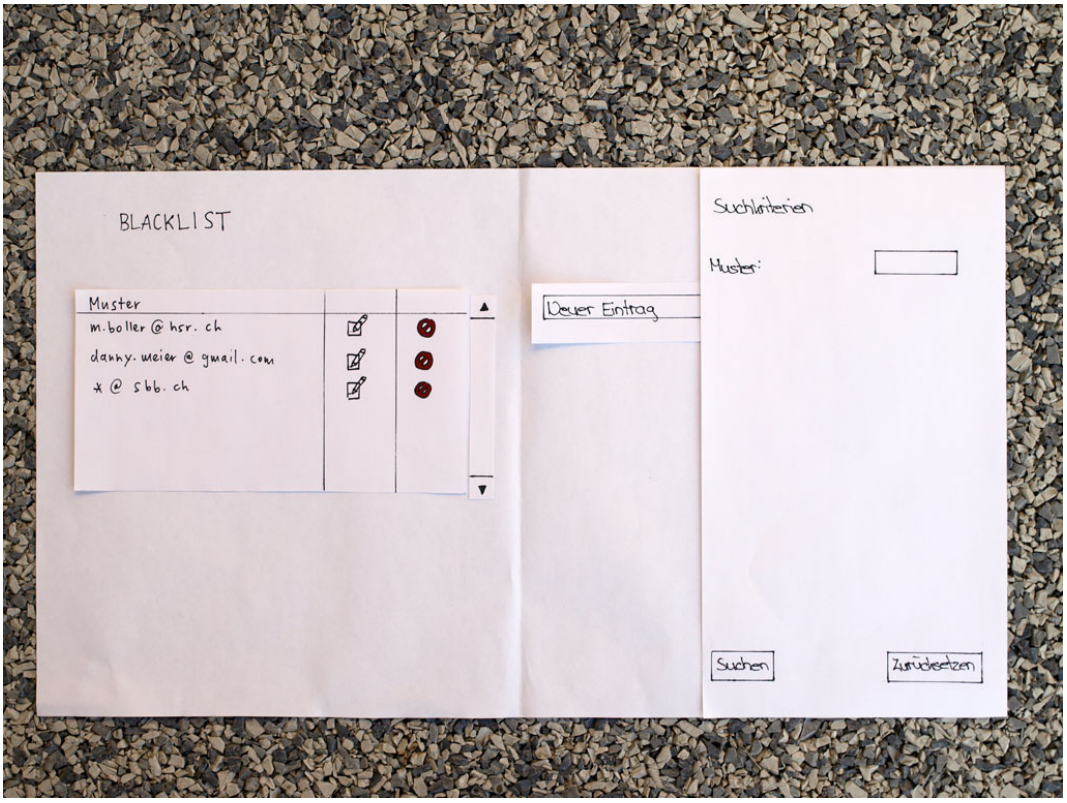


Abbildung 42 - Paper Prototype Blacklist konfigurieren mit ausgefahrener Search Bar

10 Design

10.1 Architektonische Ziele und Einschränkungen

10.1.1 Ziele

- Das Versenden von Nachrichten und Abfragen eines Versandstatus sowie die Reporting – und Maintenance Dienste des *Smtp Mailers* müssen über getrennte APIs angeboten werden
- Die Dienste des *Smtp Mailers* sind von anderen .NET Applikationen nutzbar
- Der *Smtp Mailer* reagiert richtig auf ein Fehlverhalten eines Benutzers
- Die *Smtp Mailer* Software muss stabil laufen
- Bei einem Fehler im *Smtp Mailer* dürfen keine Daten verloren gehen
- Die Business Logik soll vom Persistenzmechanismus unabhängig sein
- Die *Smtp Mailer* Software muss mehrfach parallel laufen gelassen werden können (mehrere Instanzen)
- Sämtliche Geschehnisse (Fehler, Unvorhergesehenes) werden an zentraler Stelle geloggt
- Das Austauschen des Logging Frameworks ist einfach realisierbar
- Das Austauschen der NDR Komponente soll mit vertretbarem Aufwand möglich sein
- Einstellungs- und Konfigurationsänderungen in Konfigurationsfiles (App.config) sollen nach Möglichkeit ohne Neustart vom *Smtp Mailer* übernommen werden können
- Konfigurationsänderungen in der Blacklist und Whitelist, der NDR Patterns und der Datenbankbereinigung (CleanupRules) müssen zur Laufzeit übernommen werden
- Die Datenbankbereinigung muss automatisch ablaufen

10.1.2 Einschränkungen

Die nachfolgenden Einschränkungen wurden bereits in der Anforderungsspezifikation festgehalten, gelten aber als besonders wichtig in Bezug auf die Architektur:

- Der Data Access Layer muss die Daten auf einen MS SQL 2005 abbilden
- Die Kommunikation innerhalb des *Smtp Mailers* muss gewisse Qualitätsanforderungen erfüllen:
  - Speicherung und Versendung von 300‘000 E-Mail Nachrichten pro Stunde
  - Speicherung und Verarbeitung von 30‘000 NDR Nachrichten pro Stunde
- Bei einem Fehler im System müssen menschliche Benutzer mittels einer lesbaren Fehlermeldung informiert werden

10.2 Architektonische Darstellung

10.2.1 Grobübersicht

Die Architektur des *Smtp Mailers* gliedert sich in drei Schichten.

Auf oberster Stufe werden zwei verschiedene APIs angeboten. Diese APIs delegieren die API Aufrufe weiter in den Business Logik Layer. Wie in Abbildung 43 - Schichten Architektur ersichtlich ist, verwenden beide APIs denselben Core (Business Logik Layer).

Der Business Logik Layer (BLL) enthält die Business Logik des *Smtp Mailers* und verwendet Komponenten von Drittherstellern (Third Party Components), um die E-Mail Nachrichten zu versenden oder NDR Nachrichten zu verarbeiten.

Der Business Logik Layer ist möglichst so zu gestalten, dass ein Austausch einer Komponente eines Drittherstellers mit vertretbarem Aufwand möglich ist.

Für die Persistenz ist der Data Access Layer (DAL) zuständig. Der Business Logik Layer nutzt die Dienste des Data Access Layers. Der Data Access Layer kapselt die Datenbank und sämtliche Funktionalität, welche datenbankspezifisch ist. Der Business Logik Layer ist so unabhängig von der darunterliegenden Datenbank und dem Persistenzmechanismus.

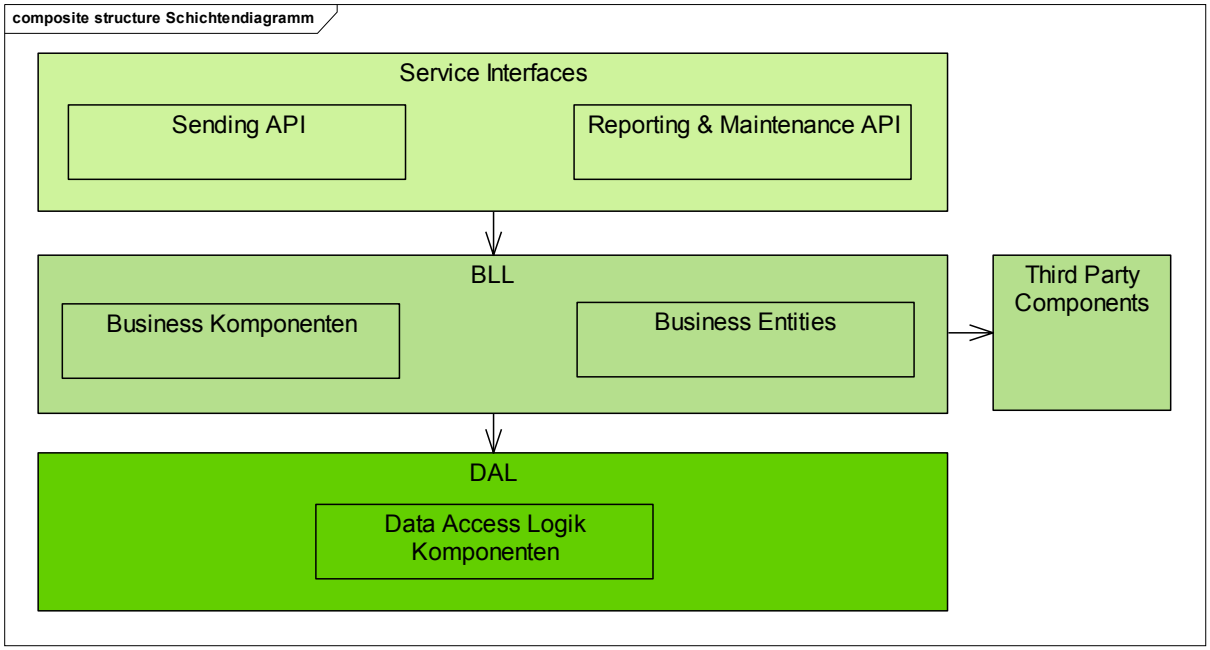


Abbildung 43 - Schichten Architektur

Die durch den *Smtp Mailer* zur Verfügung gestellten APIs werden von anderen .NET Applikationen als auch von den Silverlight Webapplikationen sowie den verschiedenen Engines (Sending Engine, Bounce Engine, Cleanup Engine, AddressQuality Engine) genutzt.

10.2.2 Übersicht über architektonische Elemente

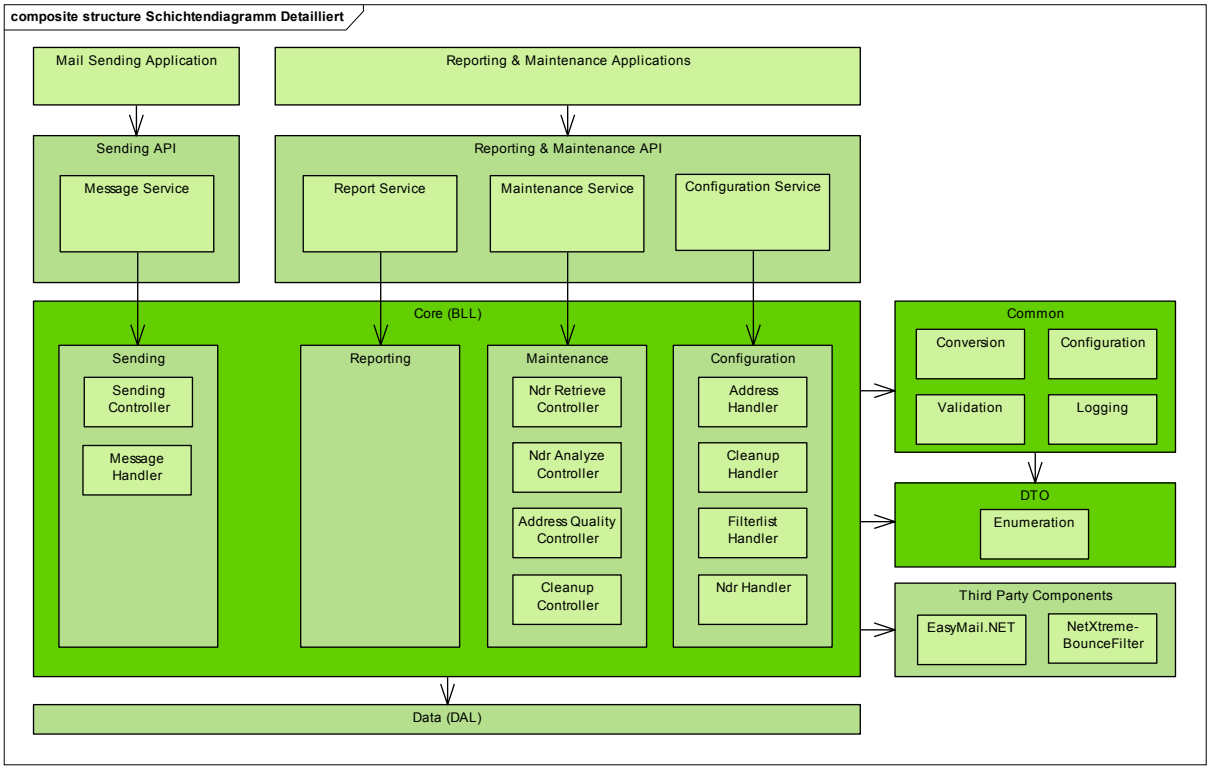


Abbildung 44 - Schichtendiagramm detailliert

Diese detaillierte Abbildung der Architektur (Abbildung 44 - Schichtendiagramm detailliert) zeigt die Kommunikation der verschiedenen Schichten auf. Dabei wurde jede Schicht in einzelne Komponenten aufgeteilt, welche jeweils für eine spezifische Funktionalität im *Smtp Mailer* stehen.

Die beiden APIs werden in zwei Assemblies zur Verfügung gestellt, welche auf einen gemeinsamen Core zurückgreifen. Die APIs delegieren ihre Aufrufe an den Core weiter, welcher wiederum die internen spezifischen Methoden aufruft um die Anfrage zu bearbeiten und die Business Logik abzuhandeln. Die Core Komponenten greifen auf einen gemeinsamen Data Access Layer (DAL) zu, welcher für die Datenhaltung zuständig ist. Sowohl der Core wie auch der DAL befinden sich in einer eigenständigen Assembly.

Die *Smtp Mailer* Software gliedert sich also in folgende Assemblies:

- Sending API
- Reporting & Maintenance API
- Core (BLL)
- Data (DAL)

Dazu kommen Assemblies für die Engines, die Webapplikationen sowie den Common Namespace.

10.2.3 Architektur Prototyp

Das Ziel des Architektur Prototyps ist es, die entworfene Grobarchitektur im Code abzubilden. Es wird ein vertikaler Prototyp entwickelt, das heisst, es werden vom Prototyp sämtliche Architektur Schichten abgedeckt, jedoch nur mit minimaler Funktionalität.

Der Prototyp soll so das Funktionieren der Zusammenarbeit zwischen den verschiedenen Schichten aufzeigen und beweisen.

10.2.3.1 Szenario

1. Eine C# Konsolenapplikation nutzt die Sending API des *Smtp Mailers*, um eine Nachricht zu versenden.
2. Die Sending API delegiert den Aufruf an den Business Logik Layer
3. Die Business Logik delegiert das Speichern der Nachricht an den Data Access Layer
4. Der Data Access Layer legt die Nachricht in der Datenbank ab
5. Der Data Access Layer gibt die E-Mail Kennung an den Business Logik Layer zurück
6. Die Business Logik gibt auf der Konsole aus, dass die Nachricht nun versendet wird
7. Die Business Logik gibt die E-Mail Kennung an die Sending API zurück
8. Die Sending API gibt die E-Mail Kennung an die Testapplikation zurück
9. Die Testapplikation gibt auf der Konsole die E-Mail Kennung aus
10. Eine Silverlight Webapplikation nutzt die Reporting & Maintenance API des *Smtp Mailers*, um eine Nachricht mittels einer E-Mail Kennung abzufragen
11. Die Reporting & Maintenance API delegiert den Aufruf an den Business Logik Layer
12. Die Business Logik delegiert das Abfragen einer Nachricht mittels Kennung an den Data Access Layer
13. Der Data Access Layer fragt die Nachricht mittels Kennung in der Datenbank ab
14. Der Data Access Layer gibt die Nachricht an den Business Logik Layer zurück
15. Die Business Logik gibt die Nachricht an die Reporting & Maintenance API zurück
16. Die Reporting & Maintenance API gibt die Nachricht an die Webapplikation zurück
17. Die Webapplikation stellt die abgefragte Nachricht dar

Es soll auch der Fehlerfall resp. das Werfen einer Exception geprüft und im Architekturprototyp implementiert werden.

10.2.3.2 Namespaces

Namespace	Beschreibung
Comparis.Framework.SmtpMailer.Prototype.Architecture.Core	Enthält die Business Logik und delegiert Aufrufe an den DAL
Comparis.Framework.SmtpMailer.Prototype.Architecture.Data	Repräsentiert den DAL. Bietet das Lesen und Schreiben einer Message an.
Comparis.Framework.SmtpMailer.Prototype.Architecture.DTO	Enthält das Data Transfer Objekt Message und ein ResultData Objekt für den Datenaustausch zwischen WCF Service und Silverlight Applikation.
Comparis.Framework.SmtpMailer.Prototype.Architecture.Service	Repräsentiert die Sending API und die Reporting & Maintenance API
Comparis.Framework.SmtpMailer.Prototype.Architecture.SilverlightApp	Enthält die Silverlight Applikation, welche via WCF Service auf die Reporting & Maintenance API zugreift, um eine Nachricht zu lesen.
Comparis.Framework.SmtpMailer.Prototype.Architecture.SilverlightApp.Web	Enthält einen Silverlight-enabled WCF Service, welcher auf die Reporting & Maintenance API zugreift.
Comparis.Framework.SmtpMailer.Prototype.Architecture.Testapp	Enthält eine Konsolen Applikation, welche über die Sending API eine Nachricht versendet.

Tabelle 58 - Namespaces Architektur Prototyp

10.2.3.3 Architektur Prototyp Bestandteile

Für die Implementierung des Architektur Prototyps gemäss obigem Szenario sind auf den verschiedenen Software Schichten folgende Codeteile zu implementieren:

Service Interface:

- Implementierung einer Sending API mit einer Methode, um eine Nachricht zu versenden. Die Methode delegiert den Aufruf an eine Business Komponente im Business Logik Layer
- Implementierung einer Reporting & Maintenance API mit einer Methode, um eine Nachricht anhand einer E-Mail Kennung abzufragen. Die Methode delegiert den Aufruf an eine Business Komponente im Business Logik Layer
- Implementierung eines Webservices, über welchen die Silverlight Webapplikation auf die APIs (welche sich in einem .NET Assembly befinden) zugreifen kann. Dazu wird ein Silverlight-enabled WCF Service benutzt

Business Logik Layer (BLL):

- Implementierung einer Business Komponente, welche eine Nachricht zum Abspeichern dem DAL übergeben kann, die Nachricht versenden kann und eine Nachricht anhand der E-Mail Kennung der API zurückgeben kann

Data Access Layer (DAL):

- Implementierung einer Nachricht Data Access Layer Komponente, welche die Nachricht in der Datenbank abspeichern kann und eine Nachricht aufgrund einer E-Mail Kennung aus der Datenbank lesen kann

Data Transfer Object (DTO):

- Implementierung eines Message DTO
- Auf Business Logik Layer Ebene und im Service Interface wird mit den Data Transfer Objects gearbeitet. Auch im WCF Service wird mit den DTO gearbeitet. Im DAL werden die DTO in die Datenbank abgebildet.

Datenbank:

- Erstellung einer Nachricht Tabelle, damit eine Message DTO abgelegt werden kann

Testapplikation:

- Implementierung einer C# Testapplikation, welche über die Sending API des *Smtp Mailers* eine Nachricht versendet.

Silverlight Webapplikation:

- Implementierung einer Silverlight Webapplikation, welche über die Reporting & Maintenance API des *Smtp Mailers* eine Nachricht anhand einer E-Mail Kennung abfragt und darstellt

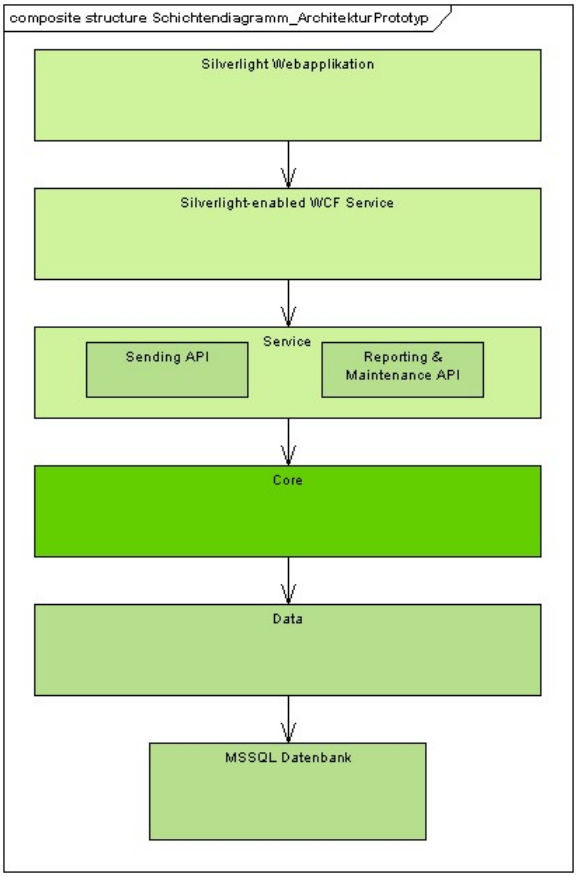


Abbildung 45 - Schichtendiagramm Architektur Prototyp

In Abbildung 45 - Schichtendiagramm Architektur Prototyp ist die Architektur des implementierten Prototyps ersichtlich.

Jeder der Software Layer Data (DAL), Core (BLL) und Service ist über ein Interface von der oberen Schicht abgeschottet. So wird zum Beispiel auf dem Core Layer nur gegen ein Data Layer Interface programmiert. Ein Austausch des Persistenzmechanismus im Data Layers auf ein anderes Framework wäre so einfach machbar.

Die Silverlight Webapplikation greift über einen Silverlight-enabled WCF Service (BasicHttpBinding) auf die vom *Smtp Mailer* zur Verfügung gestellten APIs zu. Die Silverlight Webapplikation hat keinen direkten Zugriff auf ein C# Assembly, weil die Silverlight Webapplikation unter einer eigenen CLR läuft. Deshalb werden die *Smtp Mailer* APIs über einen WCF Service zur Verfügung gestellt, auf welchen die Silverlight Webapplikation zugreifen kann.

10.2.3.4 Probleme bei der Entwicklung des Architektur Prototyp

Beim Silverlight-enabled WCF Service wird das Exception Handling mittels FaultException nicht unterstützt. Also kann das Exception Handling nicht so umgesetzt werden wie man das von einem WCF Service gewohnt ist. Als Workaround kapselt ein ResultData Objekt sowohl das eigentliche Resultatobjekt (Message DTO) als auch ein Exception Objekt (String mit Fehlermeldung). Dieses Resultatobjekt wird von allen WCF Methoden zurückgegeben. Ist das Exception Objekt nicht null oder leer, so ist eine Exception im *Smtp Mailer* System aufgetreten und die Silverlight Applikation kann entsprechend reagieren.

10.2.3.5 Erkenntnisse aus der Entwicklung des Architektur Prototyp

Der Architektur Prototyp hat anhand von zwei Anwendungsfällen (Nachricht versenden, Nachricht abfragen) eine Kommunikation über alle möglichen Software Layers des *Smtp Mailers* gezeigt.

Für die erkannten Probleme konnten bereits jetzt schon mögliche Lösungen und Workarounds ausgearbeitet werden. Dies wird die Implementierung der Webapplikation in der Construction 3. Iteration vereinfachen.

10.3 Logische Architektur

Das *Smtp Mailer* System verwendet als logische Architektur ein geschichtetes System. Einzelne Schichten könnten so einfacher ausgewechselt werden. Verstärkt wird die Kapselung der einzelnen Schichten, indem den Klassen, auf welche von oberen Schichten zugegriffen wird, Interfaces vorgeschaltet werden.

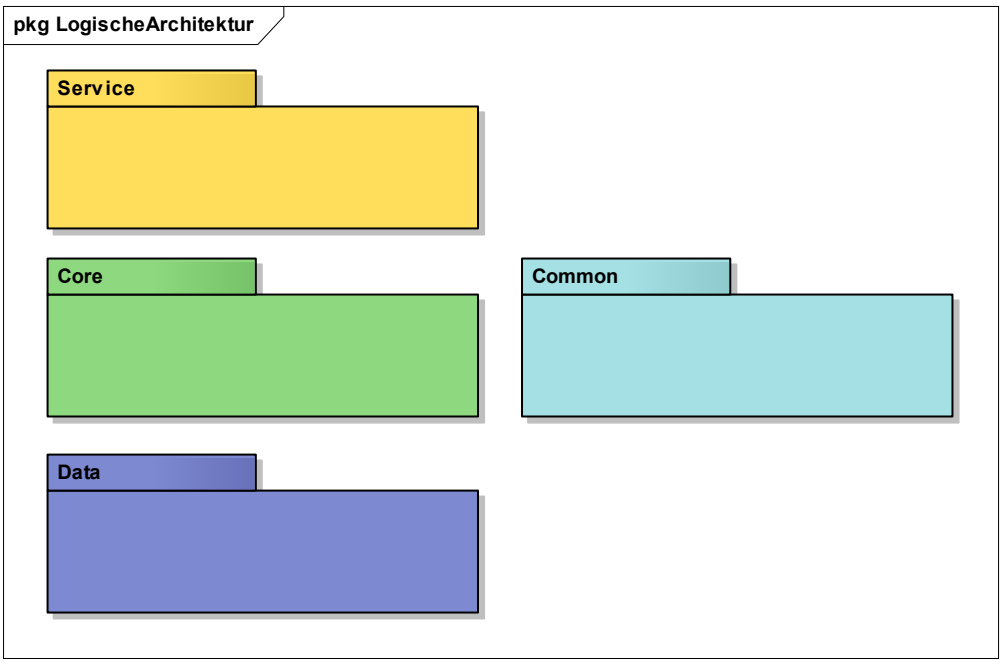


Abbildung 46 - Logische Architektur

10.3.1 Service

Die Service Schicht bildet die API des *Smtp Mailers*. Sowohl andere .NET Applikationen als auch Webservices und die Engines greifen über die API auf die Funktionalität des *Smtp Mailers* zu.

10.4 Core

Im Core wird die Business Logik des *Smtp Mailers* implementiert. Hier werden die Business Prozesse mittels den DTO Objekten (Data Transfer Objects) abgebildet.

10.5 Data

Die Data Schicht bildet die Data Transfer Objects auf die Datenbank ab. Der Data Access Layer kapselt die gesamte Anbindung an die Datenbank. Ein Austausch des Persistenzmechanismus wäre relativ einfach möglich.

10.6 Common

In der Common Schicht befindet sich die Konfiguration des *Smtp Mailers*. Der Layer bietet Zugriff auf die verschiedenen Konfigurationswerte. Zudem wird hier das Logging geregelt sowie Validierungsfunktionen für DTO angeboten.

10.7 Design der Namespaces

Die Architektur des *Smtp Mailers* gliedert sich in verschiedene Namespaces. Dies erhöht die Wiederverwendbarkeit sowie die Wartbarkeit, weil kleine Projekte für sich getestet werden können sowie besser verständlich ist, was in den einzelnen Projekten implementiert ist.

Nachfolgend ist jeder Namespace des *Smtp Mailers* aufgelistet:

Namespace	Beschreibung
Comparis.Framework.SmtpMailer.Common	Beinhaltet verschiedene Hilfsklassen, welche schichtenübergreifend genutzt werden (z.B. Validierungsfunktionen für DTO oder spezielle Konvertierungsmethoden).
Comparis.Framework.SmtpMailer.Common.Configuration	Bildet die Konfiguration des *Smtp Mailers* in Klassen ab, welche mittels Annotations mit den Daten aus dem XML Konfigurationsfiles abgefüllt werden.
Comparis.Framework.SmtpMailer.Common.Logging	Enthält eine Logger Klasse, welche statische Methoden anbietet, um verschiedene Ereignisse zu loggen (Info, Debug, Warn, Error, Fatal). Der Logging Mechanismus ist so gekapselt
Comparis.Framework.SmtpMailer.Common.UnitTests	Enthält Unit-Test Klassen für die Klassen im Common Namespace.
Comparis.Framework.SmtpMailer.Core	Enthält die Business Logik (BLL). Der Core besitzt Handler Klassen, welche die API Aufrufe validieren und diese auf die entsprechenden Klassen weiter delegieren. Des Weiteren enthält der BLL Controller Klassen, welche verschiedene Workflows und Prozesse (z.B. der Engines) abbilden.
Comparis.Framework.SmtpMailer.Core.UnitTests	Enthält Unit-Test Klassen für die Klassen im Core Namespace.
Comparis.Framework.SmtpMailer.Data	In diesem Namespace wird der Data Access Layer (DAL) abgebildet. Dieser stellt sämtliche Funktionalitäten für den Datenbankzugriff bereit. Die verwendete Datenbankschnittstelle ist ADO.NET.
Comparis.Framework.SmtpMailer.Data.UnitTests	Enthält Unit-Test Klassen für die Klassen im Data Namespace.
Comparis.Framework.SmtpMailer.DTO	Definiert die Business Entitäten, welche über alle Schichten der Applikation genutzt werden.
Comparis.Framework.SmtpMailer.DTO.Enumeration	Enthält gemeinsame Enumerationen, welche über alle Schichten der Applikation genutzt werden.
Comparis.Framework.SmtpMailer.Service.ReportingMaintenance	Enthält das Reporting und Maintenance API, welches standortunabhängige Konfigurationen zulässt und verschiedenste Reporting Funktionen anbietet.
Comparis.Framework.SmtpMailer.Service.Sending	Enthält das Entwickler API zum Versenden von E-Mail Nachrichten und zum Prüfen vom Versandstatus der E-Mail Nachrichten.
SmtpMailer.AddressQualityEngine	Diese Engine kümmert sich um die Berechnung des Adressqualitätsindikators. Dieser Indikator beschreibt, wie zuverlässig eine E-Mail Adresse ist. Kommen für eine E-Mail Adresse viele Bounce Nachrichten zurück, so ist der Adressqualitätsindikator schlechter einzustufen, als wenn die Übermittlung der versendeten Nachrichten immer klappt.
SmtpMailer.BounceEngine	Diese Engine steuert die Auswertung von Bounce Nachrichten. Dies beinhaltet das Abholen der Bounce Nachrichten von einem POP3 Konto und das Speichern dieser Nachrichten in der Datenbank. In einem zweiten Schritt werden die in der Datenbank gespeicherten Bounce Nachrichten mittels einer Drittherstellerkomponente analysiert und verarbeitet.
SmtpMailer.CleanupEngine	Die CleanupEngine steuert die Bereinigung der Datenbank. Dazu werden die Attachments und MessageBodies derjenigen Messages in der Datenbank gelöscht, welche auf die

	definierten CleanupRules passen.
SmtpMailer.SendingEngine	Diese Engine kümmert sich um das Versenden der E-Mail Nachrichten.
SmtpMailerConfiguration	Enthält die Silverlight Applikation für die Konfiguration des *Smtp Mailers.* Dazu gehört die Verwaltung der NdrPatterns und der CleanupRules. Zudem kann die Whitelist bearbeitet werden und unerkannte Ndr Mails können angezeigt werden.
SmtpMailerConfiguration.Web	Enthält die ASP.NET Webapplikation, welche die Silverlight Applikation aufruft.
SmtpMailerCustomerService	Enthält die Silverlight Applikation für das Anzeigen von Nachrichten und das Verwalten der Blacklist.
SmtpMailerCustomerService.Web	Enthält die ASP.NET Webapplikation, welche die Silverlight Applikation aufruft.
SmtpMailerReporting	Enthält die Silverlight Applikation für das Anzeigen von Reports sowie um Detailinformationen über das Adressqualitätsmanagement zu erhalten.
SmtpMailerReporting.Web	Enthält die ASP.NET Webapplikation, welche die Silverlight Applikation aufruft.

Tabelle 59 - Design der Namespaces

10.8 Sequenzdiagramme

10.8.1 Mail Modul

10.8.1.1 Nachricht versenden

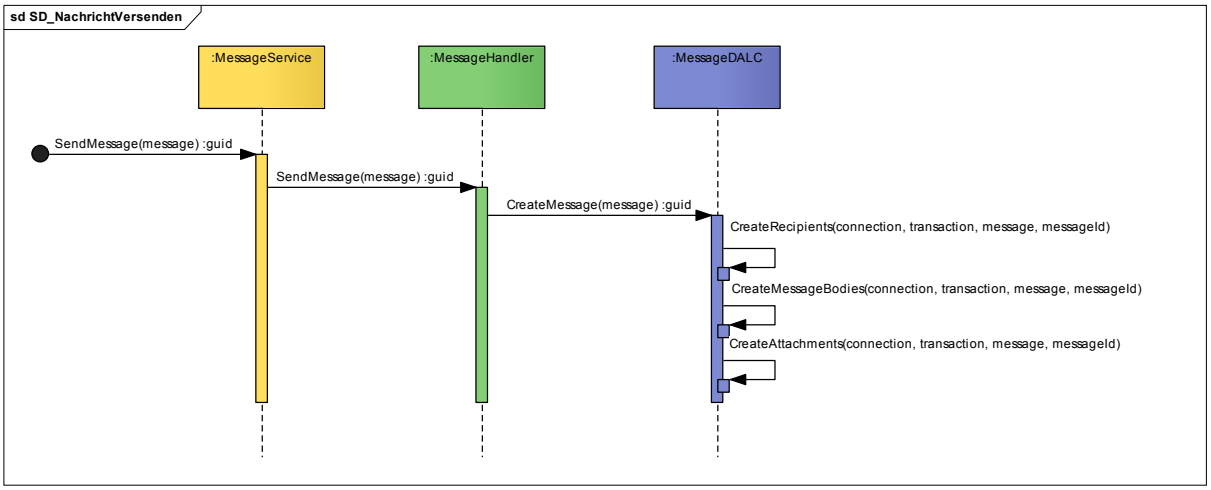


Abbildung 47 - Sequenzdiagramm Nachricht versenden

Das Sequenzdiagramm veranschaulicht die Verwendung des MessageService (ein Teil der öffentlichen API des *Smtp Mailers*). Der MessageService (Software Schicht Service) erlaubt es mittels der Methode SendMessage() eine Nachricht (Message DTO) zu versenden. Dabei wird die Nachricht nicht sofort versendet sondern über den MessageHandler (Software Schicht Core) und schliesslich dem MessageDALC (Software Schicht DAL) in die Message Queue (abgebildet in der Datenbank) eingereiht. Von dort aus wird die Nachricht zum Zeitpunkt des geplanten Versanddatums von einem anderen Prozess (Sending Engine) verschickt. Im MessageDALC wird für die Nachricht eine eindeutige guid generiert. Diese wird dem Benutzer als string zurückgegeben. Damit ist es möglich, den Status der Nachricht mittels der Methode GetMessageStatus(guid) abzufragen.

10.8.1.2 Versandstatus abfragen

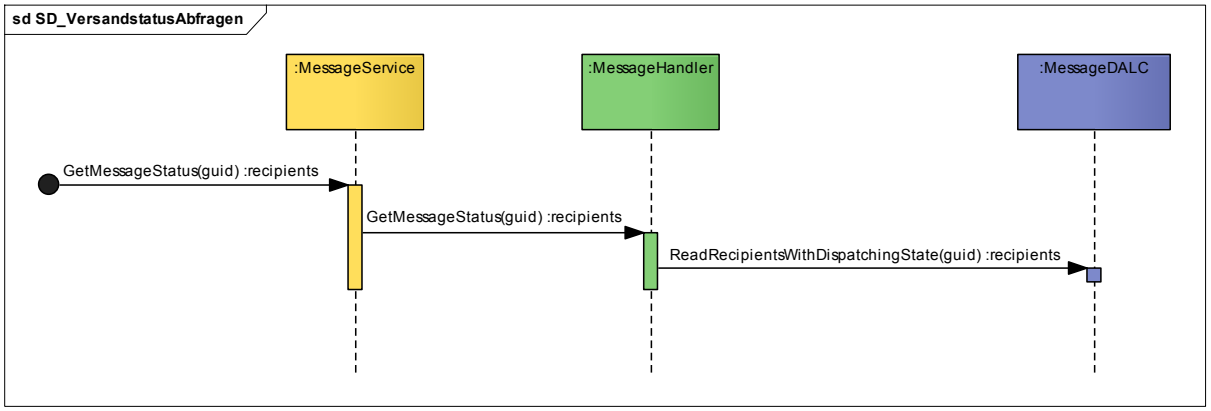


Abbildung 48 - Sequenzdiagramm Versandstatus abfragen

Mittels der Methode GetMessageStatus(guid) der öffentlichen API MessgeService des *Smtp Mailers* ist dem Benutzer möglich, den Status einer Nachricht, identifiziert über die guid, abzufragen. Dabei erhält der Benutzer eine Liste mit Recipient Objekten zurück. Jeder Empfänger der Nachricht ist durch ein Recipient Objekt repräsentiert. Am Recipient Objekt assoziiert ist eine Versandhistory, repräsentiert durch eine Liste mit DispatchingState Objekten. Aus der Versandhistory ist erkennbar, wann eine Nachricht verschickt wurde, ob der Empfänger White- oder Blacklisted war, ob ein Fehler

aufgetreten ist oder ob Bounce Nachrichten zurückerhalten wurden. Zudem besitzt jeder Recipient ein Flag namens SuccessfulSent. Ist es auf true, so wurde die Nachricht erfolgreich verschickt. Ist es auf false, so ist ein Problem aufgetreten. Den Grund für das Problem findet man dann in der Versandhistory.

Der Aufruf in der MessageService Klasse wird in den Core zu der Klasse MessageHandler delegiert. Dieser wiederrum delegiert das Lesen der Recipients und der Versandhistory (DispatchingStates) an die Klasse MessageDALC im Data Access Layer.

10.8.1.3 Sending Engine

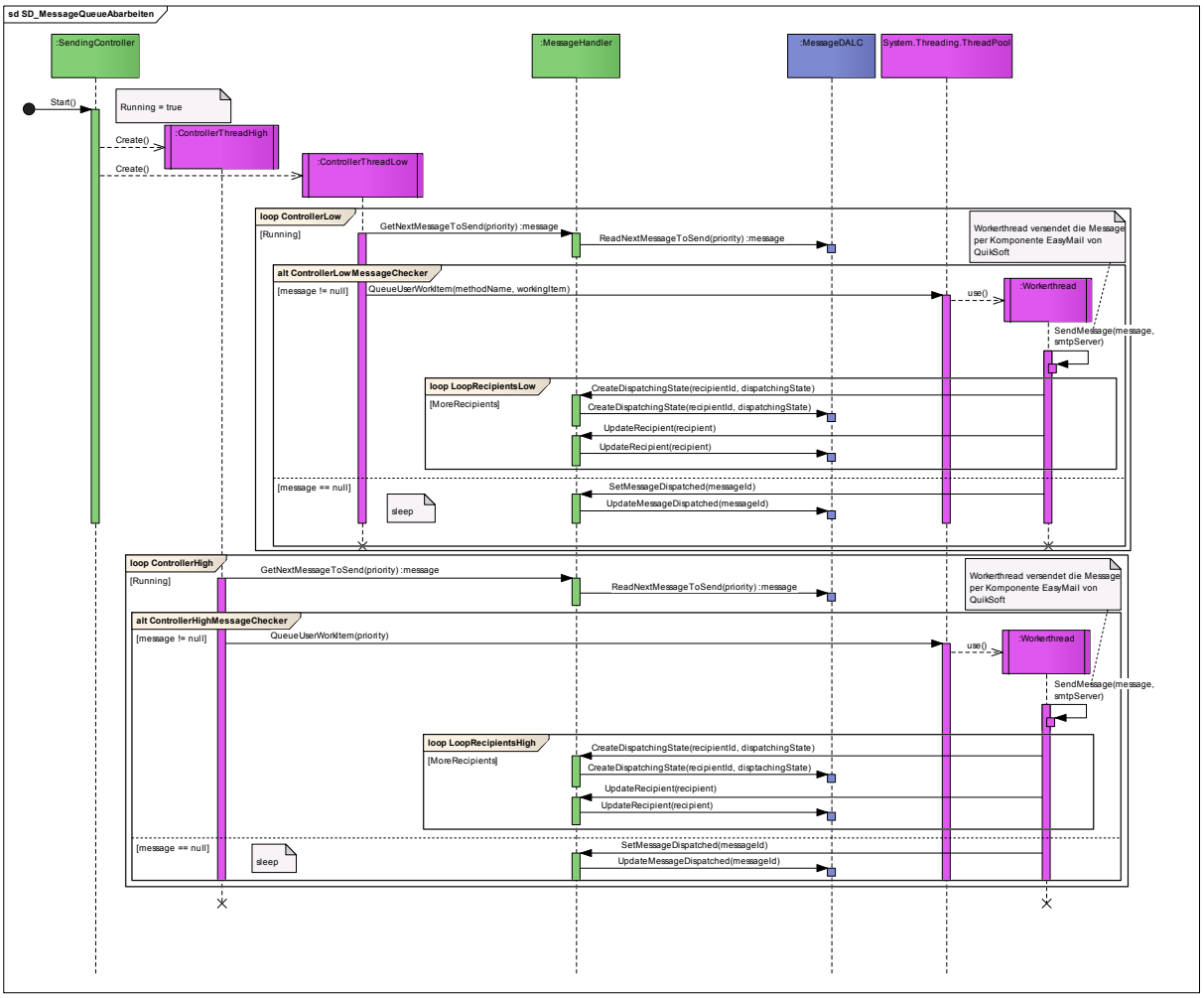


Abbildung 49 - Sequenzdiagramm Message Queue abarbeiten

Die Business Logik der Sending Engine wird von der Klasse SendingController im Core implementiert. Über die Methode Start() wird der Sending Controller gestartet. Dieser wiederrum startet zwei Kontrollerthreads – einer für die Priorität High, einer für die Priorität Low.

Die Kontrollerthreads haben die Aufgabe, die nächste zu versendende Nachricht ihrer Priorität aus der Message Queue abzuholen. Die Message Queue ist in der Datenbank mittels der Tabelle Message abgebildet. Dabei nutzen die Kontrollerthreads die Methode GetNextMessageToSend() unter Angabe der Priorität der Klasse MessageHandler. Der MessageHandler delegiert den Aufruf weiter an die Klasse MessageDALC. Diese liest die nächste zu versendende Nachricht der übergebenen Priorität aus der Datenbank und gibt diese zurück.

Der Kontrollerthread besitzt nun das Message DTO Objekt seiner Priorität, welches als nächstes zu versenden ist. Das Versenden geschieht über einen weiteren Thread, einem so genannten Workerthread aus dem Threadpool. Dazu ruft der Kontrollerthread die Methode QueueUserWorkItem() an der ThreadPool Klasse auf. Dabei wird die Methode angegeben, welche der Workerthread abarbeiten soll und ein WorkingItem. Das WorkingItem kapselt die vom Workerthread zu benutzende Transaktion sowie das Message DTO, welches zu versenden ist. Der Threadpool nutzt einer der vorhandenen Workerthreads und übergibt ihm die Aufgabe des Versendens der Nachricht.

Vom Workerthread wird nun die Nachricht unter Verwendung der EasyMail Komponente verschickt und über den MessageHandler ein DispatchingState DTO erzeugt (DispatchState Sent), der Recipient geupdated (SucessfullSent = true) sowie die Message DTO geupdated (Dispatched = true). Die Nachricht gilt nun als verschickt. Der MessageHandler delegiert dabei die Aufrufe wie bereits oben an den MessgeDALC, welcher die Datenbank manipuliert.

10.8.1.4 Cleanup Controller

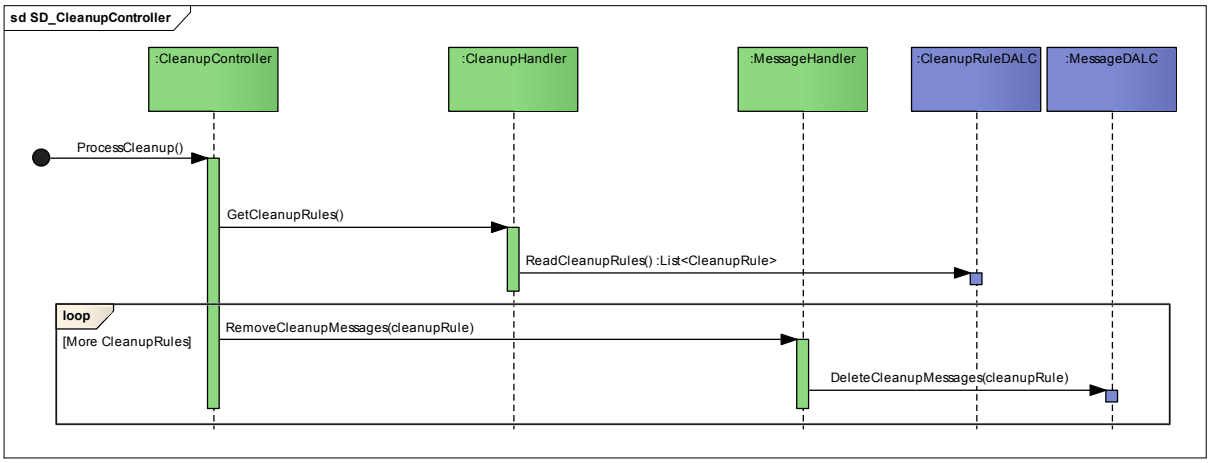


Abbildung 50- Sequenzdiagramm CleanupController

Die CleanupEngine nutzt die Business Logik, welche im CleanupController abgebildet ist. Der Prozess der Nachrichtenbereinigung wird mittels der Methode ProcessCleanup() angestossen. Über den CleanupHandler werden alle CleanupRules (Regeln welche definieren, welche Nachrichten zu bereinigen sind) aus der Datenbank gelesen. Der CleanupHandler delegiert dabei den Aufruf weiter an den CleanupDALC im Data Access Layer.

Der MessageHandler bietet nun die Funktionalität an, anhand einer übergebenen CleanupRule alle zutreffenden Messages in der Datenbank zu bereinigen. Die Bereinigung betrifft dabei die Nachrichteninhalte und Attachments der Nachricht. Die Methode RemoveCleanupMessages() wird für jede gelesene CleanupRule aufgerufen. Der MessageHandler delegiert wiederum den Aufruf in den Data Access Layer zu der Klasse MessageDALC.

10.8.2 Ndr Modul

10.8.2.1 Bounce Engine

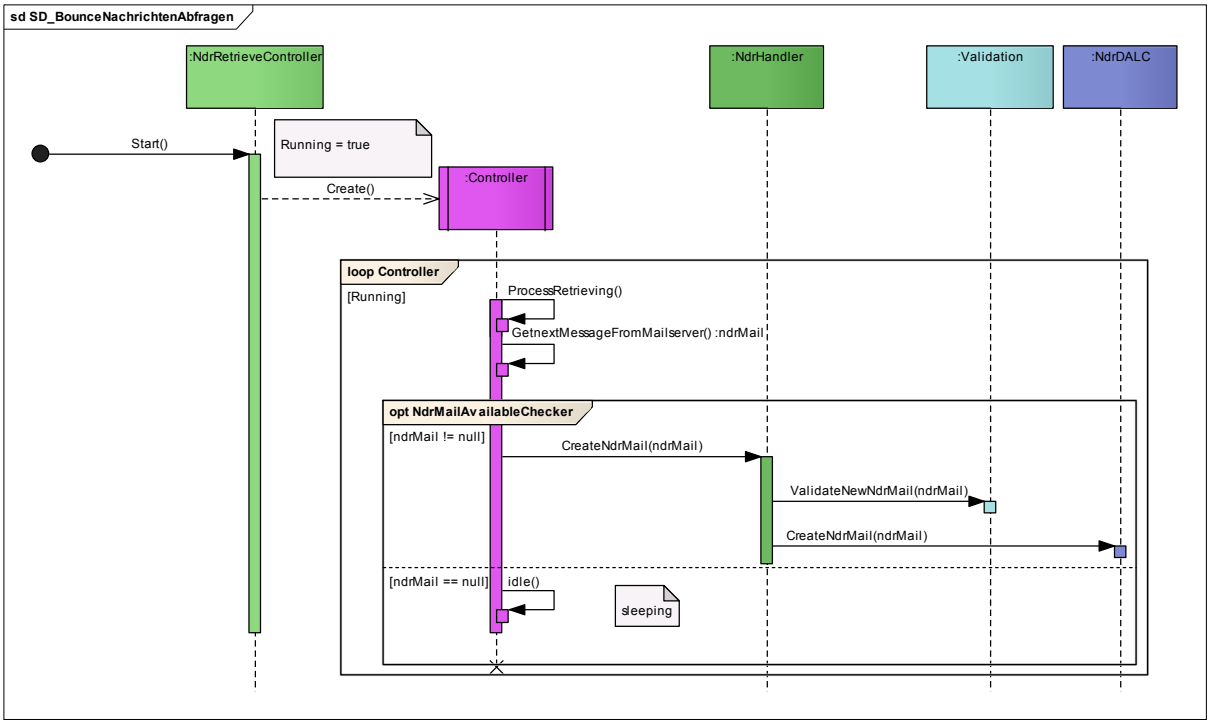


Abbildung 51 - Sequenzdiagramm Bounce Nachrichten abfragen

Ein Teil der Bounce Engine stellt das Abholen der Bounce Nachrichten von einem POP3 Server dar. Diese Funktionalität wird vom NdrRetrieveController im Core implementiert. Über die Methode Start() wird der NdrRetrieveController gestartet. Der NdrRetrieveController beauftragt einen Thread, die nächste Nachricht vom POP3 Server abzuholen und diese in die Datenbank zu speichern. Dabei nutzt der Thread die Funktionalität der EasyMail Komponente, um die Nachrichten von einem POP3 Server zu lesen. Ist keine Bounce Nachricht vorhanden, so legt sich der NdrRetrieveController schlafen. Ist jedoch eine Bounce Nachricht vorhanden, so wird die Bounce Nachricht gelesen und aus der gelesenen Bounce Nachricht ein NdrMail DTO erstellt. Dieses wird in der Datenbank gespeichert. Dazu wird der NdrHandler genutzt, welcher die entsprechende CreateNdrMail() Methode anbietet. Bevor der NdrHandler die Speicherung in der Datenbank an die NdrDALC Klasse im Data Access Layer delegiert, wird das zu speichernde NdrMail validiert. Dies geschieht mittels der Helper Klasse Validation. Diese bietet statische Methoden zur Validierung an. Ist das NdrMail ungültig, so wird eine Exception geworfen.

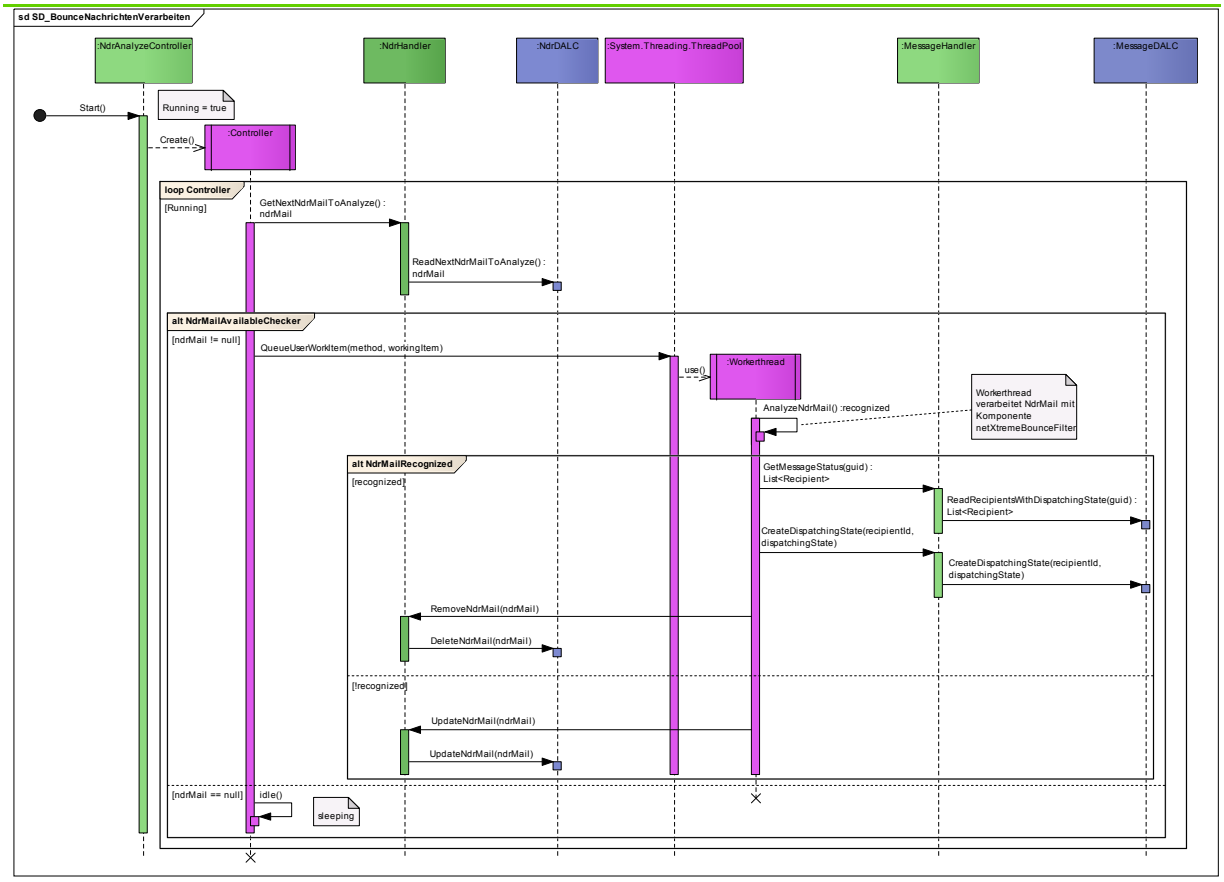


Abbildung 52 - Sequenzdiagramm Bounce Nachrichten verarbeiten

Der zweite Teil der Bounce Engine stellt das Verarbeiten der in der Datenbank gespeicherten NdrMails dar. Die dazu notwendige Business Logik besitzt die Klasse NdrAnalyzeController im Core.

Mittels eines Kontrollerthreads wird die nächste zu verarbeitende NdrMail von der Datenbank geholt. Der NdrHandler bietet dem Kontrollerthread die dazu notwendige Funktionalität. Der NdrHandler delegiert das Lesen aus der Datenbank an den NdrDALC im Data Access Layer.

Ist eine zu verarbeitende NdrMail vorhanden, so wird die Verarbeitung an einen Workerthread im Threadpool delegiert. Das Ablaufmuster ist dabei mit der Verarbeitung von zu versendenden Nachrichten identisch (Siehe Abschnitt 10.8.1.3 Sending Engine auf der Seite 120). Auch hier geschieht die Delegation an einen Workerthread mittels der Methode QueueUserWorkItem(). Der Methode wird als erster Parameter die Methode angegeben, welche der Workerthread abarbeiten soll. Als zweiter Parameter wird ein WorkingItem übergeben, welches die zu nutzende Transaktion als auch das zu verarbeitende NdrMail kapselt.

Der Workerthread nutzt zur Verarbeitung des NdrMails resp. zur Erkennung der ursprünglichen guid und zur Ermittlung des Grundes für die Bounce Nachricht die Komponente NetXtremeBounceFilter von Safabyte.

Wird der Grund für die Bounce Nachricht von der Komponente erkannt, so werden mit der Methode GetMessageStatus(guid) am MessageHandler alle Recipients der ursprünglichen Nachricht anhand der guid ermittelt. Jeder Recipient erhält nun ein DispatchingState DTO mit dem DispatchState Bounced in seine Versandhistory. Dazu nutzt der Workerthread die Methode CreateDispatchingState() am MessageHandler. Beide Methoden werden vom MessageHandler an den MessageDALC im Data Access Layer delegiert. Zuletzt wird das verarbeitete NdrMail aus der Datenbank gelöscht. Die benötigte Funktionalität bietet der NdrHandler an.

Wird der Grund für die Bounce Nachricht jedoch nicht erkannt, so wird der Status der NdrMail auf Unrecognized geändert. Dies geschieht mittels der Methode UpdateNdrMail() am NdrHandler. Das NdrMail bleibt also in der Datenbank. Mittels eines eigens erfassten NdrPattern kann das NdrMail bei einer zukünftigen Verarbeitung erkannt werden.

10.8.2.2 AddressQuality Engine

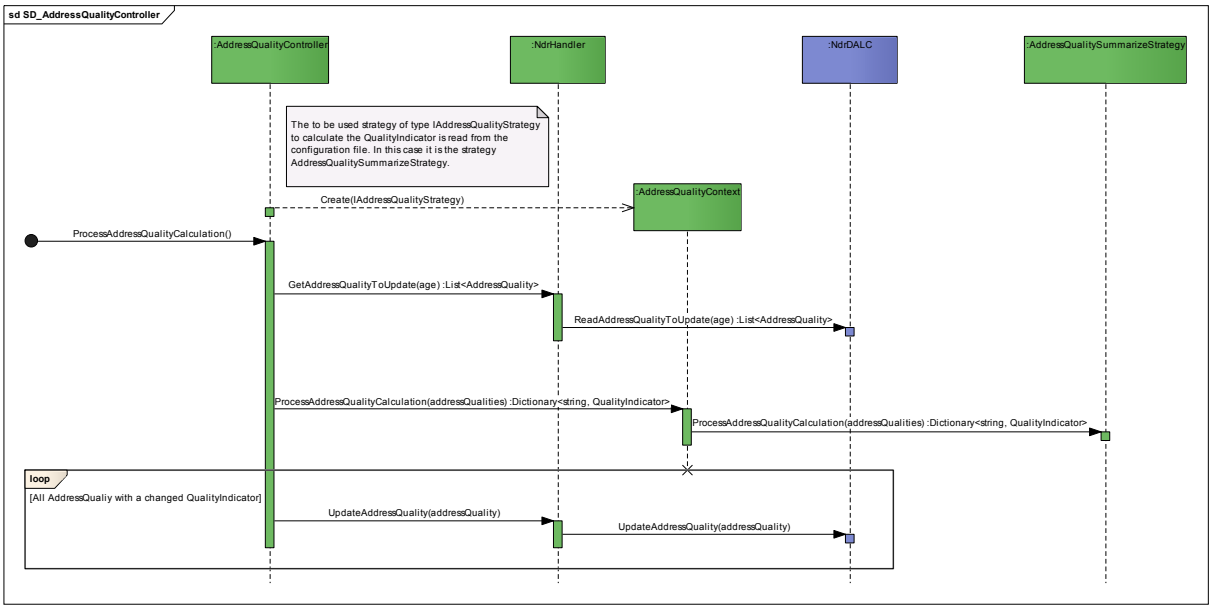


Abbildung 53 - Sequenzdiagramm AddressQualityController

Die AdressQualityEngine nutzt die in der Klasse AddressQualityController abgebildete Business Logik.

Der AddressQualityController erstellt im Konstruktor eine Instanz der AddressQualityContext Klasse. Dabei übergibt der AddressQualityController der Kontextklasse, welche gemäss dem Strategy Pattern (Wikipedia, 2009) arbeitet, ein Objekt vom Typ IAddressQualityStrategy. Die konkret zu verwendende AddressQualityStrategy wird aus der Konfiguration gelesen.

Der NdrHandler bietet mit der Methode GetAddressQualityToUpdate() die Funktionalität, alle zu aktualisierenden AddressQuality DTO Objekte zu erhalten. Dabei wird der Methode ein int übergeben, welcher die minimale Dauer in Tagen (Alter) bestimmt seit der letzten Berechnung des Qualitätsindikators. Alle AddressQuality DTO, deren letzte Berechnung des Qualitätsindikators länger als das übergebene Alter zurückliegt, werden beim Aufruf von GetAddressQualityToUpdate() zur Neuberechnung zurückgegeben. Der NdrHandler delegiert den Aufruf an die Klasse NdrDALC im Data Access Layer.

Bei allen zurückerhaltenen AddressQuality DTO Objekte muss der Qualitätsindikator neu berechnet werden. Dazu wird die Liste mit AddressQuality DTO dem AddressQualityContext übergeben. Dieser delegiert den Aufruf weiter an die konkret zu verwendende AddressQualityStrategy. In diesem Fall ist das die Klasse AddressQualitySummarizeStrategy.

Die Methode ProcessAddressQualityCalculation() gibt ein Dictionary aller neu berechneten Qualitäsindikatoren zurück. Dabei stellt der Key die E-Mail Adresse und der neu berechnete Qualitätsindikator der Value des Dictionary dar. Alle sich geänderten AddressQuality DTO werden nun mittels der UpdateAdressQuality() Methode am NdrHandler in der Datenbank aktualisiert.

10.8.3 Webapplikation

Das nachfolgende Sequenzdiagramm soll zeigen, wie die Kommunikation zwischen der Silverlight Webapplikation und dem *Smtp Mailer* abläuft. Die Kommunikation läuft dabei über einen WCF Service (Windows Communication Foundation), welcher wiederum auf dem Service Layer des *Smtp Mailers* mit den Interfaces IReportService, IMaintenanceService und IConfigurationService aufsetzt (siehe Abschnitt 11.1.3 Comapris.Framework.SmtpMailer.Service.ReportingMaintenance auf der Seite 139).

Repräsentativ als Beispielablauf für alle WCF Aufrufe der Webapplikation soll das Abrufen aller NdrPatterns aus der SmtpMailerConfiguration Webapplikation dienen:

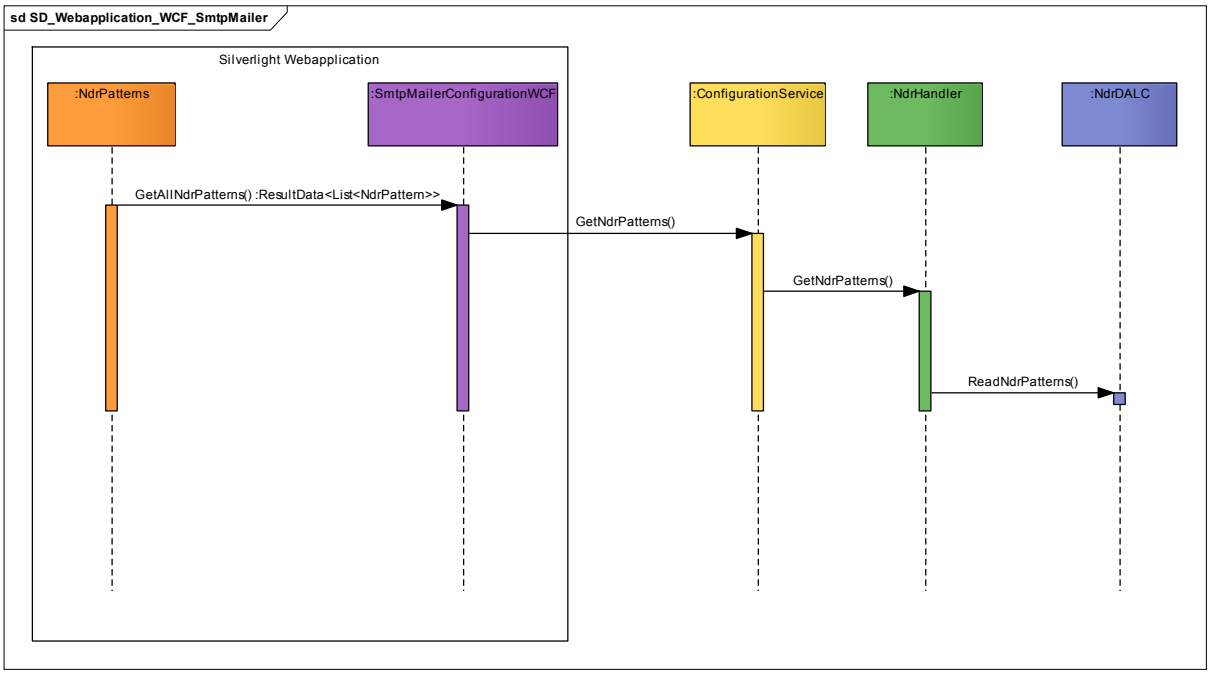


Abbildung 54 - Sequenzdiagramm Webapplikation

Das Code-Behind C# File einer Silverlight Webseite (in diesem Beispiel ist das die Klasse NdrPatterns) fragt mittels der Methode GetAllNdrPatterns() über den WCF Service nach allen NdrPatterns in der Datenbank. Die NdrPatterns sollen in einer Tabelle auf der Silverlight Webseite dargestellt werden.

Der WCF Service nutzt die öffentliche Service Schnittstelle ConfigurationService des *Smtp Mailers*, um alle NdrPatterns zu lesen. Dazu ruft der WCF Service die Methode GetNdrPatterns() auf.

Die Service Schnittstelle delegiert den Aufruf an den NdrHandler im Core weiter, dieser wiederum an den NdrDALC im Data Access Layer.

Der WCF Service übersetzt das von der Service Schnittstelle zurückerhaltene Resultat (Liste von NdrPattern DTO) in ein ResultData Objekt vom generischen Typ List<NdrPattern>. Dies ist notwendig, um ein Exceptionhandling zu implementieren resp. damit die Silverlight Webapplikation über eine aufgetretene Exception im *Smtp Mailer* informiert werden kann. Für mehr Informationen dazu siehe Abschnitt 10.9.7 Exception Handling zwischen WCF Service und Silverlight Applikation auf Seite 127.

10.9 Design Entscheidungen

10.9.1 Kommunikation zwischen Silverlight und *Smtp Mailer* API

Die Kommunikation zwischen der Silverlight Webapplikation und der API vom *Smtp Mailer* geschieht über einen WCF Service. Dabei muss ein vom Visual Studio unterstützter ‚Silverlight-enabled WCF Service' verwendet werden. Dieser nutzt das BasicHttpBinding. D.h. die Kommunikation zwischen der Silverlight Applikation und dem WCF Service läuft über das HTTP Protokoll.

10.9.2 Data Transfer Objects

Auf sämtlichen Schichten des *Smtp Mailers* wird mit den im Namespace Comparis.Framework.SmtpMailer.DTO definierten Data Transfer Objects (gemäss DTO Design Pattern) gearbeitet. Auf unterster Ebene im *Smtp Mailer,* das heisst im DAL, werden die Data Transfer Objects in die Datenbank abgebildet. Beim Herauslesen der Daten aus der Datenbank werden wiederum DTO Objekte erstellt. So werden der nächst höheren, anfragenden Schicht DTO Objekte zurückgegeben.

Eine andere Möglichkeit wäre es gewesen, auf jeder Schicht mit eigenen Business Objekten zu arbeiten. Der Nachteil davon wäre allerdings ein weit grösserer Konvertierungsaufwand. So hätte eine Abbildung von einem Business Objekt auf ein anderes auf jedem Layer geschehen müssen. Mit der verwendeten Lösung wird aber eine solche Abbildung nur im DAL benötigt.

10.9.3 Abarbeiten von nächster zu versendender Nachricht

Sowohl die noch nicht versendeten Nachrichten als auch die bereits versendeten Nachrichten befinden sich in der Tabelle Message in der Datenbank. Die nächste zu versendende Nachricht wird jeweils von dem Business Logik Layer per DAL Aufruf aus der Datenbank geholt. Dabei wird auf die Zeile in der Tabelle Message ein UPDLOCK gelegt. Dieser UPDLOCK gilt solange, bis der entsprechende Datensatz von der Business Logik geupdated wird (d.h. bis der BLL dem DAL sagt, dass die Message verschickt wurde resp. ein Fehler aufgetreten ist) und die in der Business Logik per DTC gestartete Transaktion beendet ist. Durch die Angabe des ROWLOCK Hints wird nur diese eine Zeile gelockt und nicht eine ganze Page oder gar Table. Mittels READPAST werden bei einem SELECT Statement bereits gelockte Datensätze in der Tabelle übersprungen. Dies ist notwendig, weil die Sending Engine aus mehreren Worker Threads auf Business Logik Ebene besteht und somit immer mehrere Threads nach der nächsten zu versendenden Nachricht fragen.

Bei ROWLOCK, UPDLOCK und READPAST handelt es sich um so genannte Table Hints aus dem Transact SQL. Die Table Hints überschreiben das Default Verhalten des Query Optimizers.

10.9.4 Abarbeiten von nächster zu verarbeitender NdrMail

Hier verläuft die Abarbeitung der noch nicht verarbeiteten NdrMails gleich wie beim Abarbeiten der noch nicht versendeten Nachrichten. Es wird mittels den Table Hints ROWLOCK, UPDLOCK und READPAST jeweils eine Zeile für die Dauer der Transaktion, welche in der Business Logik per DTC gestartet wurde, gelockt. Es kümmern sich wiederum mehrere Workerthreads auf Business Logik Ebene um die Verarbeitung der NdrMails.

10.9.5 Kapselung durch Interfaces

Für jede Klasse, welche ihre Dienste einer höheren Schicht anbietet, wurde ein Interface erstellt. Diese Klasse implementiert das Interface. Die nächst höhere Schicht, welche eine Klasse auf tieferer Schicht nutzen will (z.B. eine BLL Klasse nutzt die Dienste der DAL Klasse) programmiert nur gegen das Interface. So erhält man eine geringere Kopplung und die Wartbarkeit wird erhöht, weil grundsätzlich die konkrete Implementierung eines Interfaces ausgetauscht werden kann, ohne dass sich Änderungen auf nächst höherer Schicht ergeben.

10.9.6 Exception Handling im DAL und Core

Exceptions, welche im Data Access Layer bei Datenbankoperationen oder Transaktionsoperationen auftreten, werden im DAL geloggt und dann weitergeworfen.

Der Core macht es grundsätzlich genau gleich wie der DAL: Exceptions, welche im Core auftreten, werden geloggt und weitergeworfen. Auch diejenigen Exceptions, welche bei Aufrufe in den DAL auftreten werden direkt weitergeworfen.

Diese Exception Handling Taktik hat die Auswirkung, dass das Programm mit einem Stacktrace auf der Konsole abstürzt, weil die Exception einfach weitergeworfen wird. Weil schlussendlich niemand die Exception fängt, stürzt das Programm ab.

Diese Taktik ist bewusst so gewählt: Das Programm soll bei einem schwerwiegenden Fehler nicht weiterlaufen und lediglich den Fehler loggen sondern tatsächlich abstürzen. Solche schwerwiegenden Fehler sollten natürlich nur bei Inbetriebnahme des *Smtp Mailers* auftreten. Sobald die Software einmal eingerichtet ist, sind solche Fehler selten (z.B. Datenbank ist nicht verfügbar, POP3 Server ist nicht verfügbar etc.)

10.9.7 Exception Handling zwischen WCF Service und Silverlight Applikation

Mit Silverlight (Version 2) in Kombination mit WCF ist es nicht möglich, wie gewohnt das Exception Handling mittels FaultException vorzunehmen. Dies wird von Silverlight nicht unterstützt. Stattdessen verlassen wir uns auf einen Workaround:

Von jeder WCF Service Methode wird ein spezielles Rückgabeobjekt zurückgegeben (ResultData). Dieses enthält nebst dem eigentlichen Rückgabetyp der WCF Methode ein String, welcher eine Fehlermeldung enthalten kann. Ist dieser String nicht leer, so ist ein Fehler aufgetreten. Der Grund für den Fehler wird dem Aufrufer (die Silverlight Webapplikation) über diesen String mitgeteilt.

10.9.8 Logging

Eine Anforderung an das Logging ist es, dass der konkrete Logging Mechanismus ohne unvertretbaren Aufwand ausgetauscht werden kann. Deshalb werden mittels dem Adapter Pattern (Wikipedia, 2009) die Logaufrufe an einen konkreten Logging Mechanismus delegiert. Der Logging Mechanismus könnte ohne Probleme ausgetauscht werden, indem die Logaufrufe an einen anderen Logging Mechanismus delegiert würden.

10.9.9 DTC (Distributed Transaction Coordinator) beim Abarbeiten von zu versendenden Nachrichten und beim Abarbeiten von zu verarbeitenden NdrMails

Die Message Queue ist in der Datenbank als Tabelle abgebildet (Tabelle Message) und enthält die sowohl versendeten als auch die noch nicht versendeten Nachrichten. Beim Abarbeiten der Message Queue muss jeweils eine Transaktion bereits im Core Layer gestartet werden. Diese Transaktion muss auch Data Access Layer Aufrufe mit einschliessen. Denn wenn z.B. die Nachricht im Core nicht versendet werden kann, müssen gewisse getätigte Änderungen in der Datenbank wieder rückgängig gemacht werden können. Umgekehrt darf z.B. die Nachricht im Core nicht versendet werden, wenn danach der Status der Nachricht in der Datenbank nicht geändert werden kann. Sonst bliebe die Nachricht in der Datenbank als nicht versendet markiert, obwohl sie im Core tatsächlich verschickt wurde. Deshalb wird der Distributed Transaction Coordinator eingesetzt, welcher es ermöglicht, Transaktionen über verschiedene Software Schichten laufen zu lassen.

10.9.10 Zwischenspeicherung von NdrMails

Die von einem POP3 Server abgerufenen Bounce-Mails werden vor der Verarbeitung in die Datenbank zwischengespeichert (Tabelle NdrMail). Als Alternative könnten die Bounce-Mails gleich nach dem Abrufen vom POP3 Server verarbeitet werden.

Die Zwischenspeicherung erlaubt es jedoch, dass zwei getrennte Prozesse das Abrufen und Speichern der Bounce Nachrichten und das Verarbeiten der Bounce Nachrichten vornehmen (NdrRetrieveController, NdrAnalyzeController). Weil die Verarbeitung einer NdrMail fehlschlagen kann (der Grund für das Ndr Mail ist unbekannt) müssen die Nachrichten sowieso in der Datenbank vorhanden sein. Solche Nachrichten müssen zu einem späteren Zeitpunkt nochmals verarbeitet werden können, wenn z.B. eigene Ndr Erkennungssignaturen erfasst wurden.

10.9.11 Strategy Pattern für Qualitätsindikatorberechnung

Die Berechnung des Qualitätsindikators für die E-Mail Adressen aufgrund der Versandhistory wurde nach dem Strategy Pattern (Wikipedia, 2009) implementiert. So kann sichergestellt werden, dass zu einem späteren Zeitpunkt sehr einfach die zu verwendende Strategie durch eine andere ersetzt werden kann. Konkret bedeutet das, dass eine Klasse das Comapris.Framework.SmtpMailer.Core.IAddressQuality Interface implementieren muss. In der Konfiguration App.config kann dann der Klassenname der zu verwendenden Strategie geändert werden. Es sind bei Änderung der Strategie also keine Codeumstellungen notwendig.

10.9.12 Optimistic Concurrency Locking für Webapplikation

Die CustomerService Webapplikation kann von mehreren Kundendienstmitarbeitern gleichzeitig benutzt werden. Bei der Verwaltung der Blacklist kann dann das Problem auftreten, dass zwei Kundendienstmitarbeiter gleichzeitig versuchen, den gleichen Eintrag zu bearbeiten oder der eine Kundendienstmitarbeiter versucht den Eintrag zu löschen während dem ein anderer Kundendienstmitarbeiter den Eintrag versucht zu bearbeiten.

Dazu wurden die Methodenaufrufe über den WCF Service in die Datenbank mittels dem Optimistic Concurrency Locking abgesichert. Jeder Blacklist Eintrag in der Datenbank besitzt ein Versionsfeld mit dem Datentyp timestamp. Anhand des timestamps in der Datenbank kann festgestellt werden, ob in der Zwischenzeit der bearbeitete Eintrag bereits geändert hat und somit diese Änderung einfach überschrieben würde (Dirty Write). Deshalb wird die Version vor dem Schreiben mit der aktuellen Version in der Datenbank verglichen und nur wenn die Versionen identisch sind (d.h. es gab keine Änderungen in der Zwischenzeit) darf die Schreiboperation durchgeführt werden.

10.9.13 Guid

Um das Angriffspotenzial zu erschweren, haben wir uns entschieden auf Guids zur Identifikation von Message DTO zu setzten anstelle des Primärschlüssels, welcher als long Wert abgelegt ist. E-Mail Nachrichten resp. deren Return-Path birgt die Gefahr, dass die E-Mail Adresse erraten werden kann und somit selbsterstellte Bounce Nachrichten zurück gesendet werden können. Sei dies, um das BounceModul zu überlasten (Denial of Service Attacke) oder um mittels manipulierten Antworten die Qualität gewisser E-Mail Adressen zu beinträchtigen.

Daher wird in der Kommunikation nach aussen nur die Guid verwendet, welche einmalig ist und nur schwer zu erraten sein wird. Werden Bounce Nachrichten mit nicht vorhandenen Guids empfangen, werden diese aussortiert und gelöscht.

10.10 Externes Design Webapplikation

Das externe Design wurde mit Microsoft Visio 2007 und den Paper Prototyping Shapes *Sketch GUI Shapes* erstellt. Diese Shapes müssen zusätzlich zu Microsoft Visio installiert werden.

Für alle Screens der Webapplikation wurde jeweils ein Visio Design erstellt. Diese Entwürfe dienten als Grundlage für die Besprechung des Designs mit der Usability-Verantwortlichen Annette Sulzbacher von comparis.ch AG sowie mit Prof. Dr. Markus Stolze.

Dadurch konnte erreicht werden, dass man sich nicht zu früh bereits mit Farben, Schriften und Ausrichtungen auseinandersetzen musste. Es kann so extrem viel Zeit gespart werden. Ebenso muss man sich nicht um konkret zu verwendende GUI Elemente kümmern.

Zudem waren Änderungen im Design sehr einfach und ohne grossen Aufwand möglich. Denn es gab nach der Besprechung des Designs und der Paper Prototype Testresultaten mit Annette Sulzbacher sowie Prof. Dr. Markus Stolze doch noch einiges umzustellen und zu ergänzen.

10.10.1 Entwürfe für die Customer Service Webapplikation

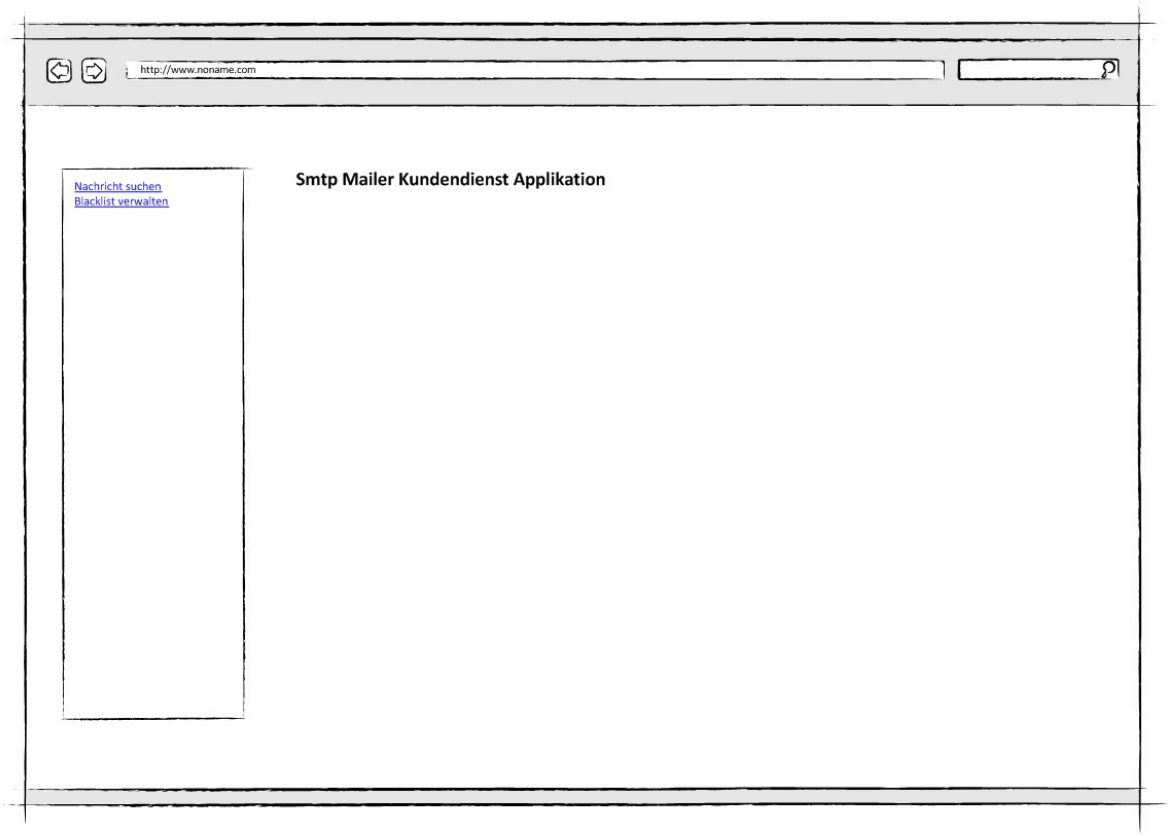


Abbildung 55 - Design Kundendienstapplikation Main

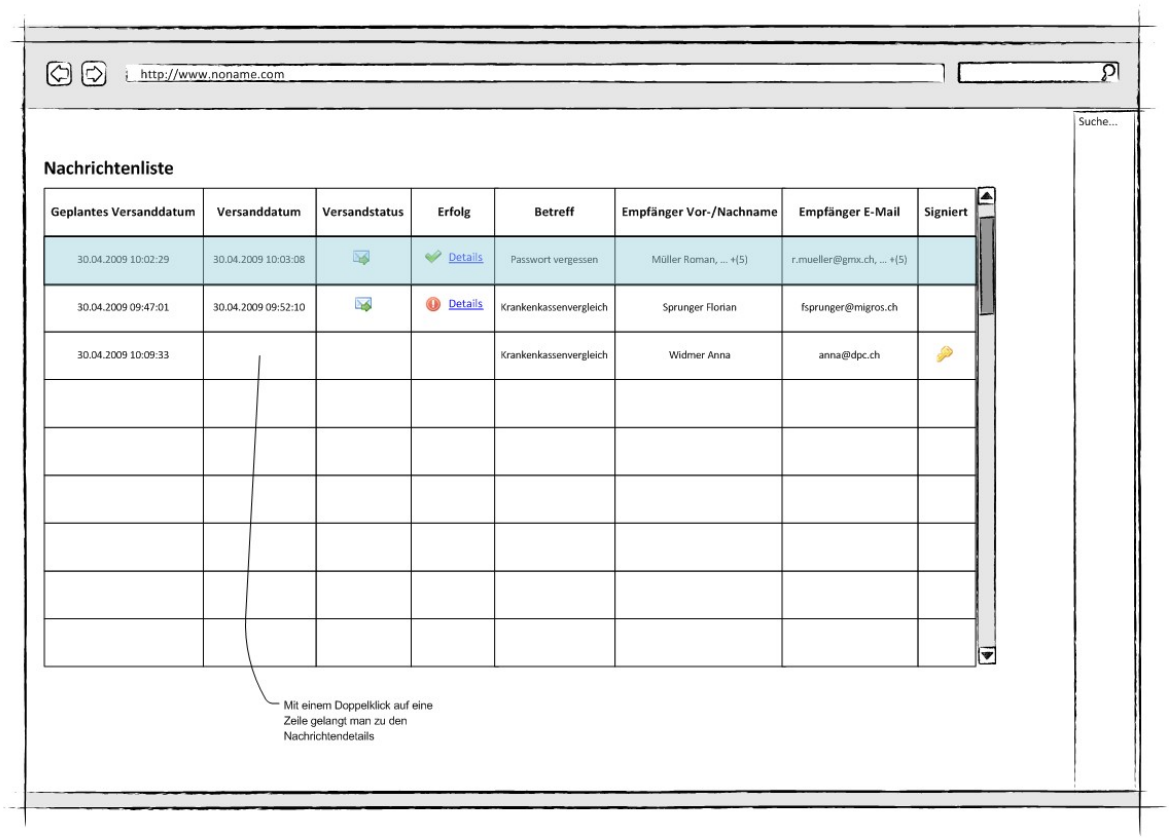


Abbildung 56 - Design Nachrichtenliste

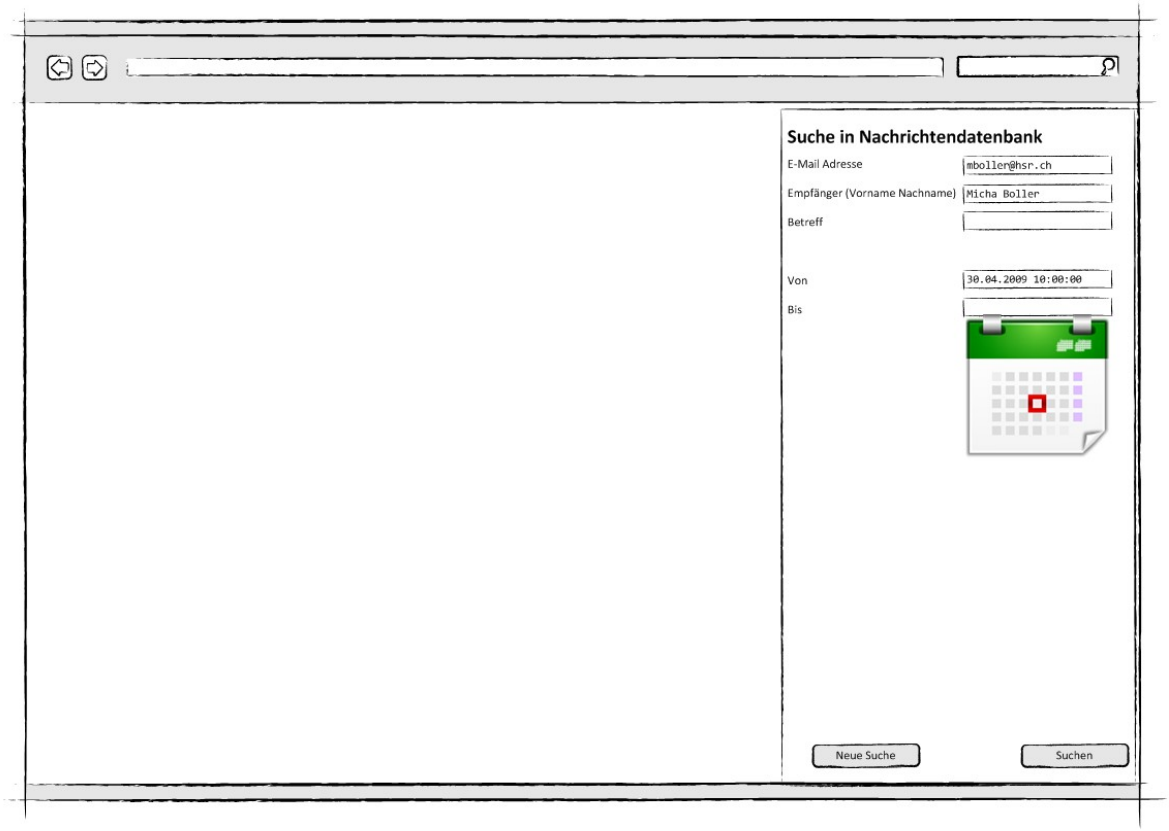


Abbildung 57 - Design Nachricht suchen

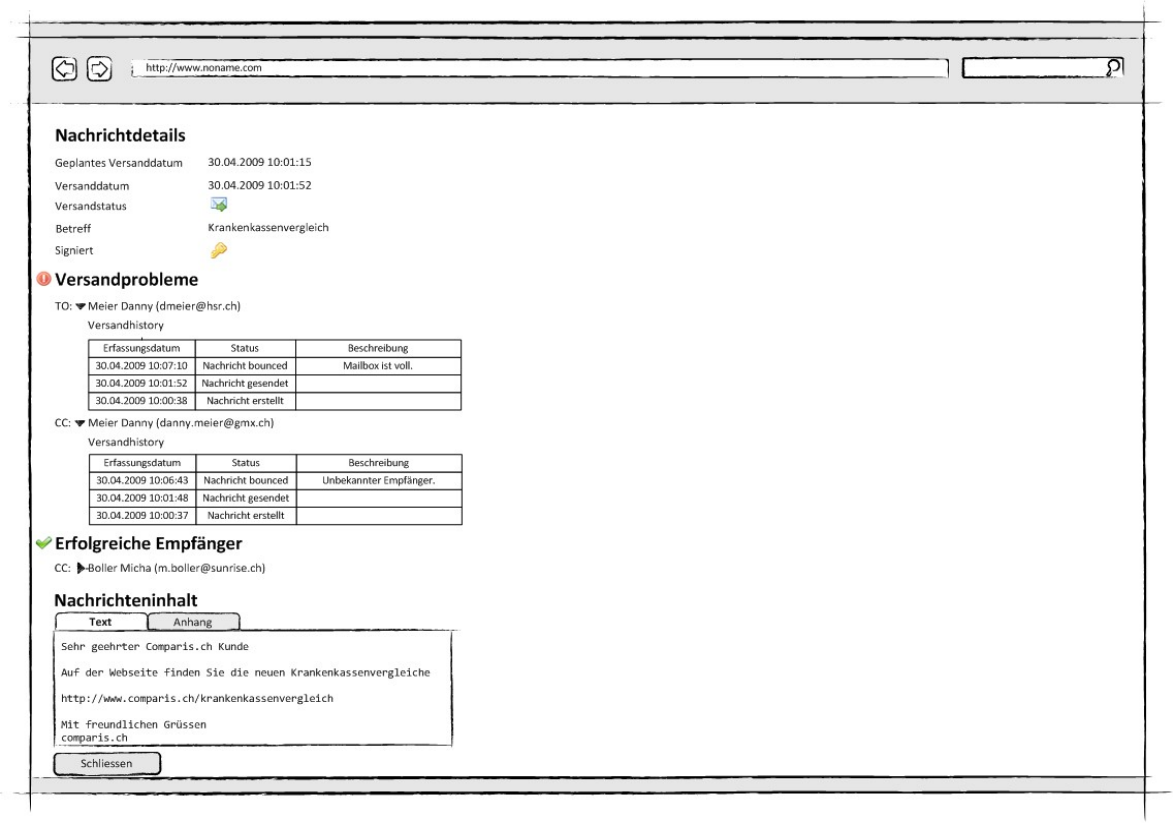


Abbildung 58 - Design Nachrichtdetails

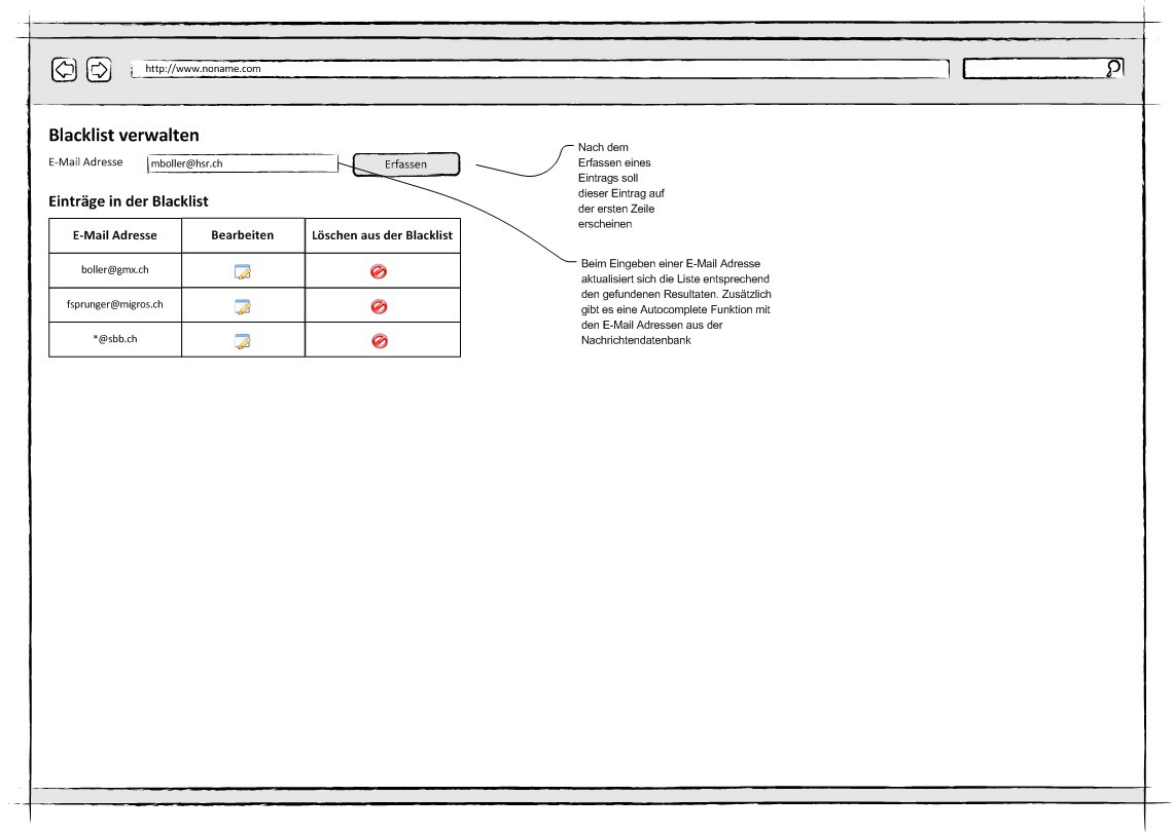


Abbildung 59 - Design Blacklist verwalten

10.10.2 Entwürfe für die Configuration Webapplikation

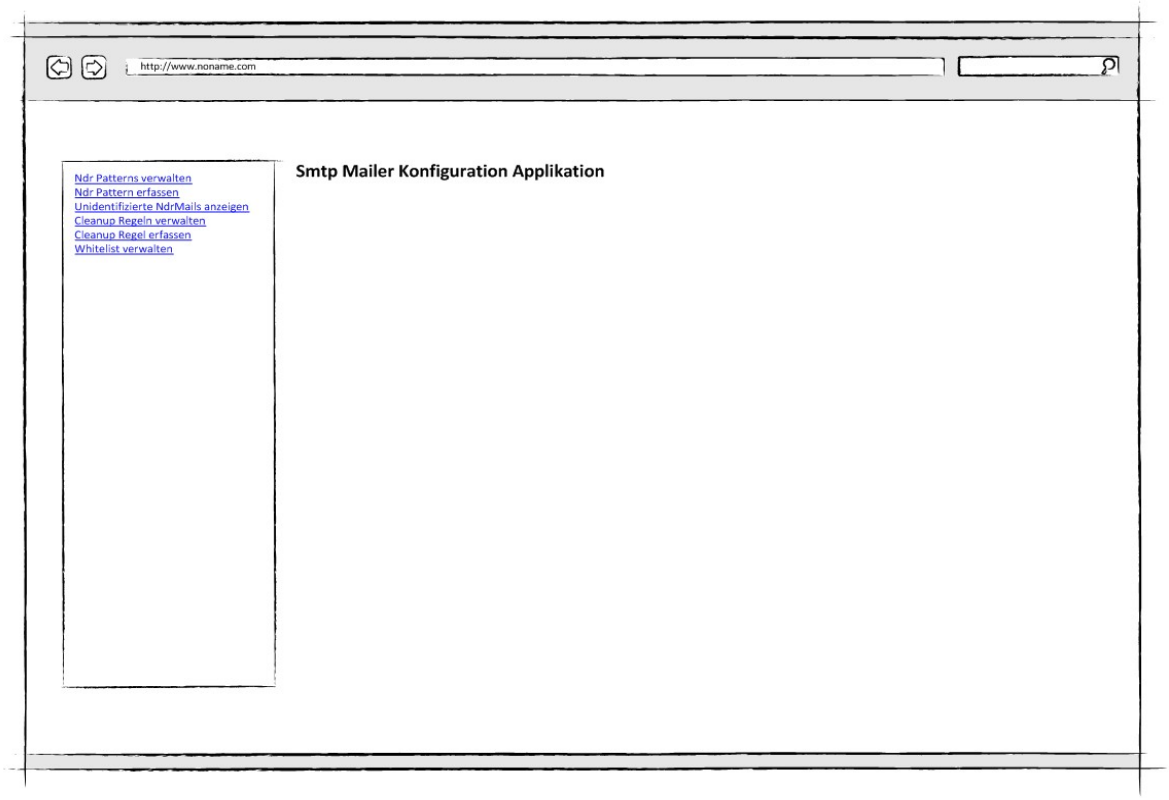


Abbildung 60 - Design Konfigurationsapplikation Main

Abbildung 61 - Design CleanupRules verwalten

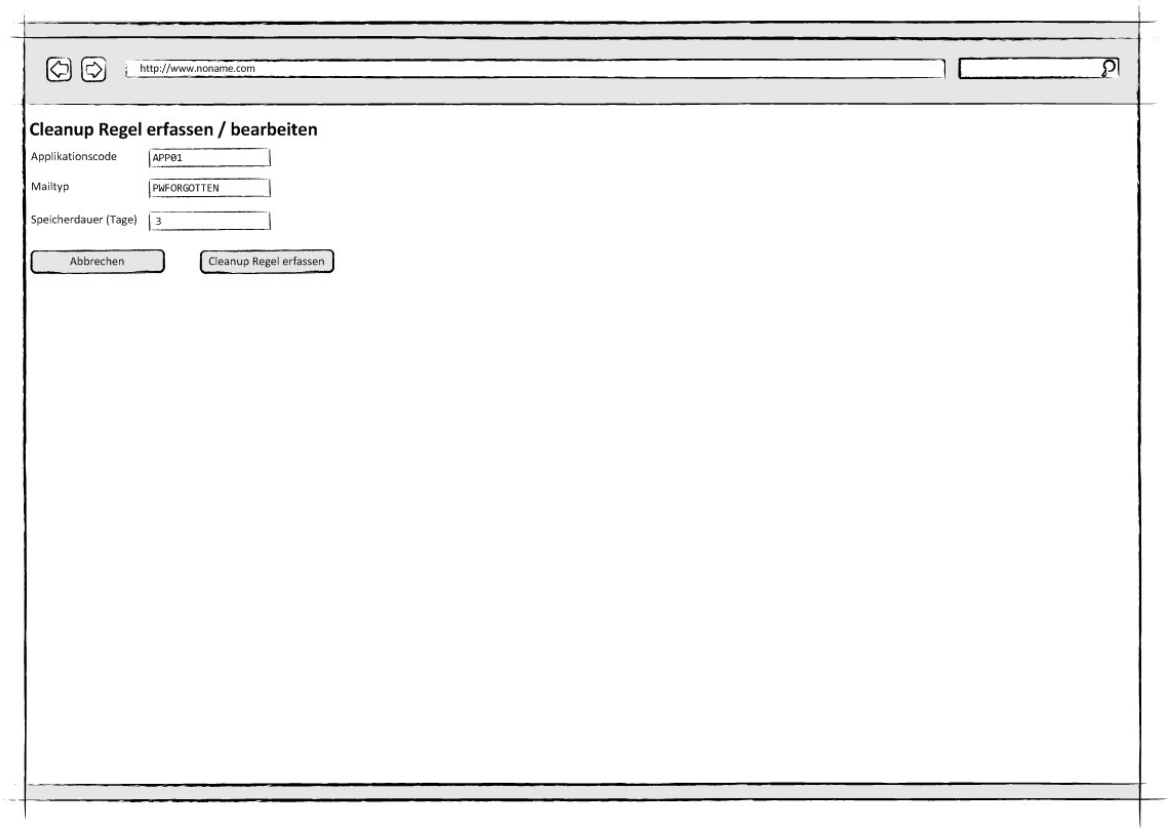


Abbildung 62 - Design CleanupRule new/edit

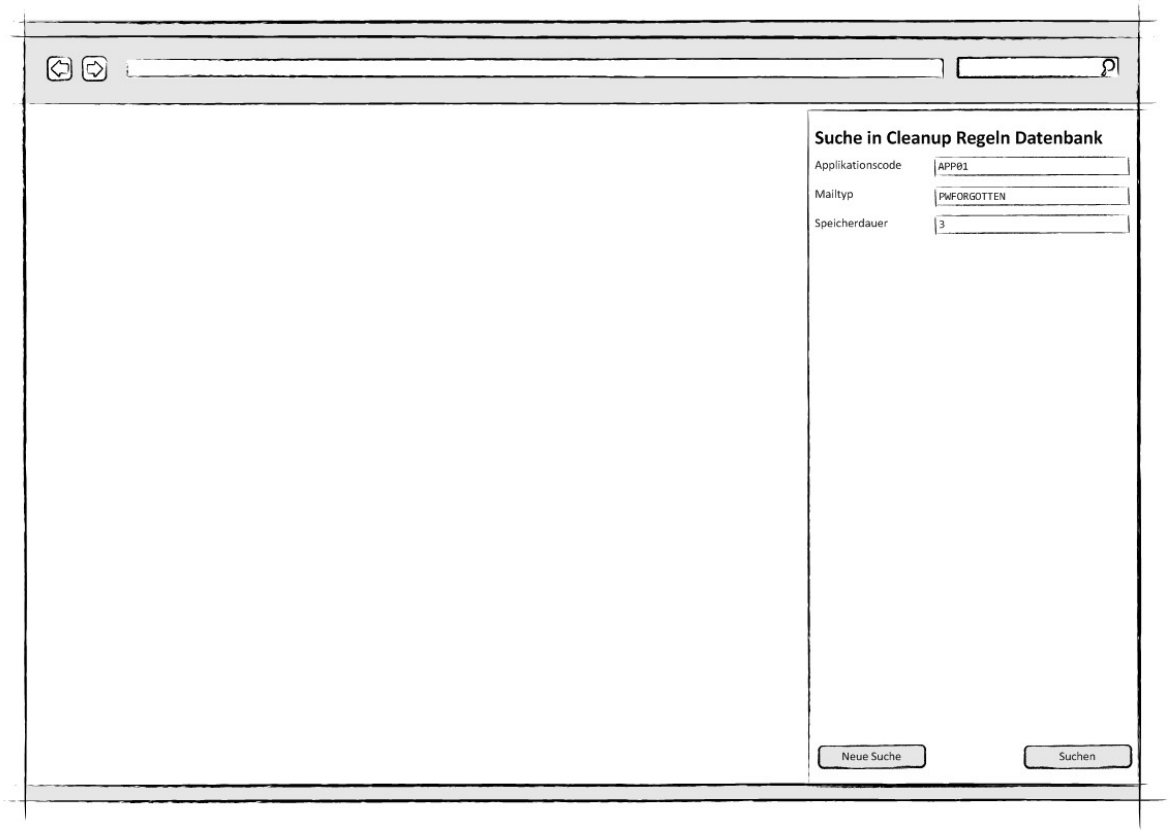


Abbildung 63 - Design CleanupRules suchen

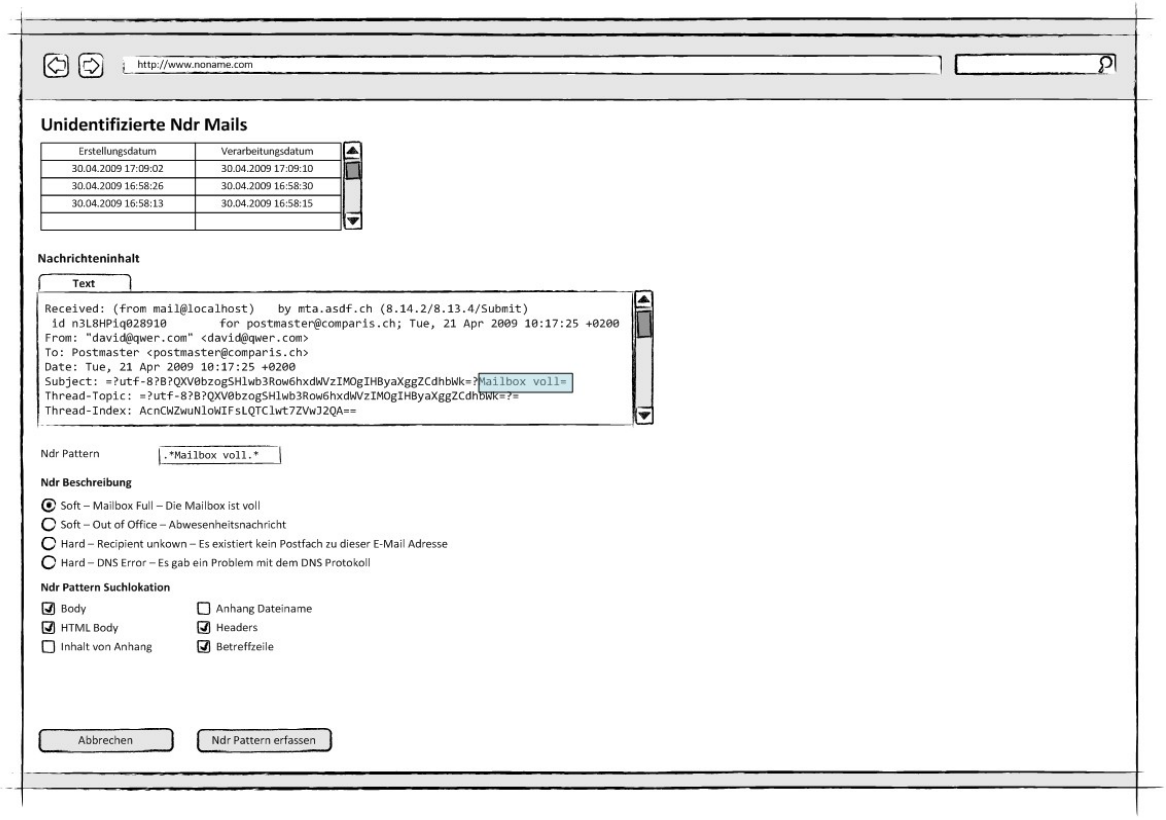


Abbildung 64 - Design unidentifizierte NdrMails

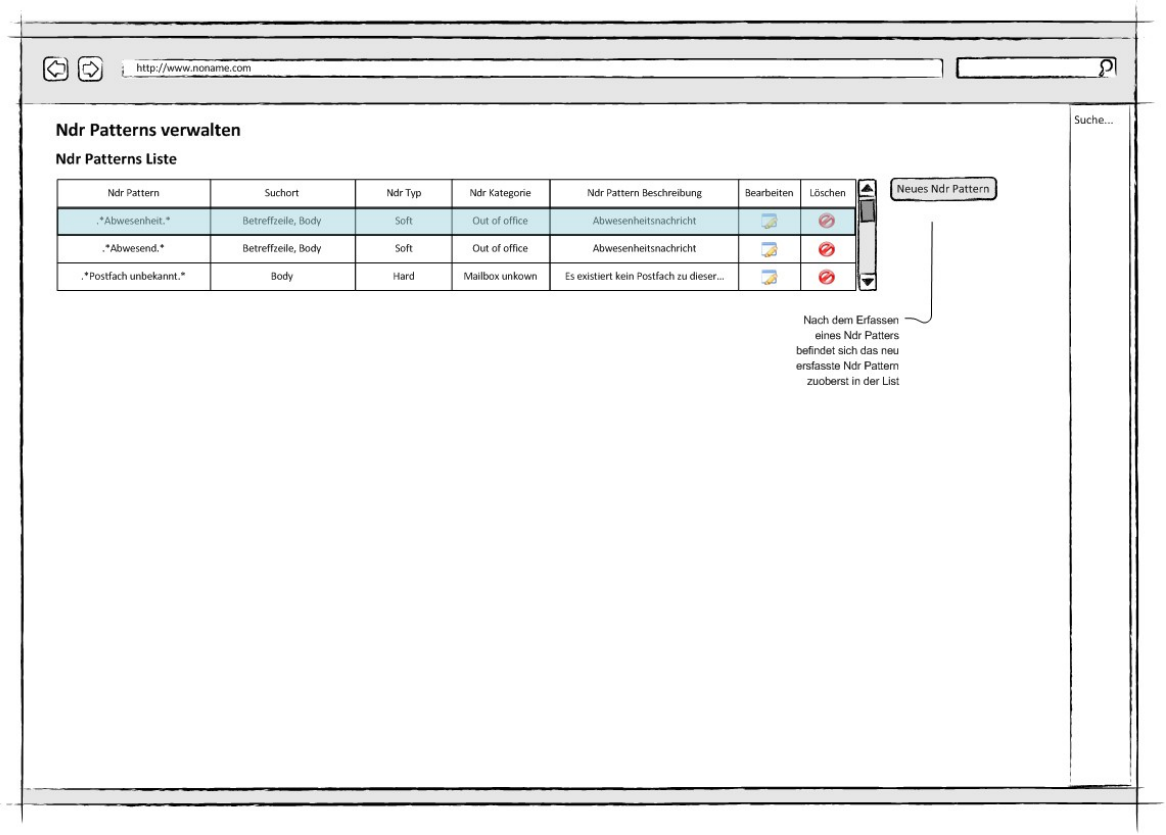


Abbildung 65 - Design NdrPatterns verwalten

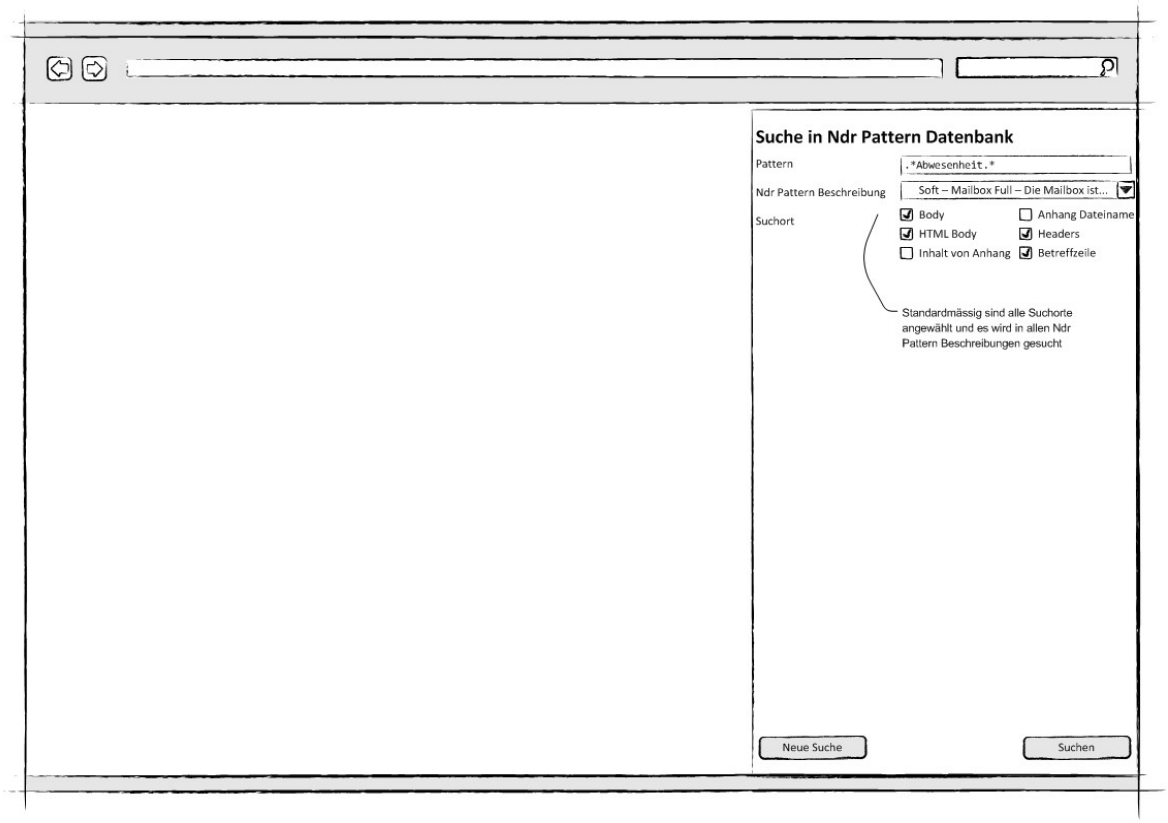


Abbildung 66 - Design NdrPatterns suchen

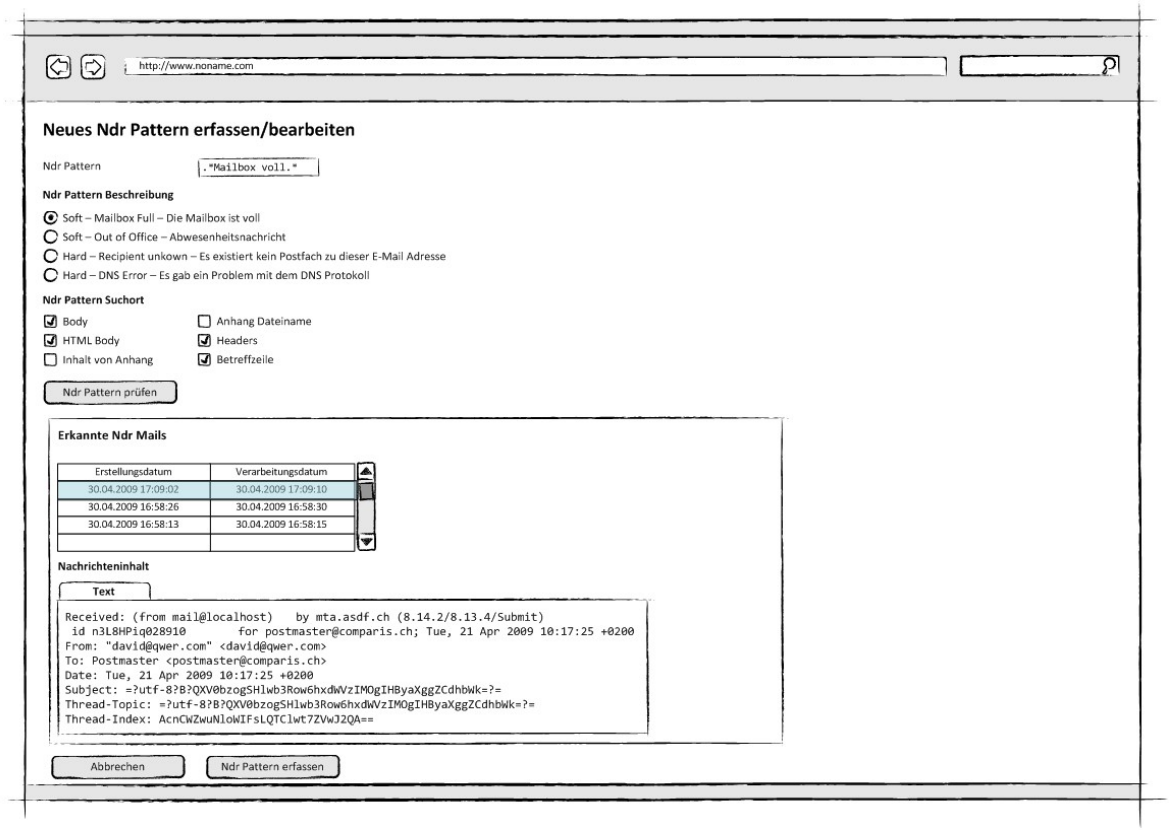


Abbildung 67 - Design NdrPattern new/edit

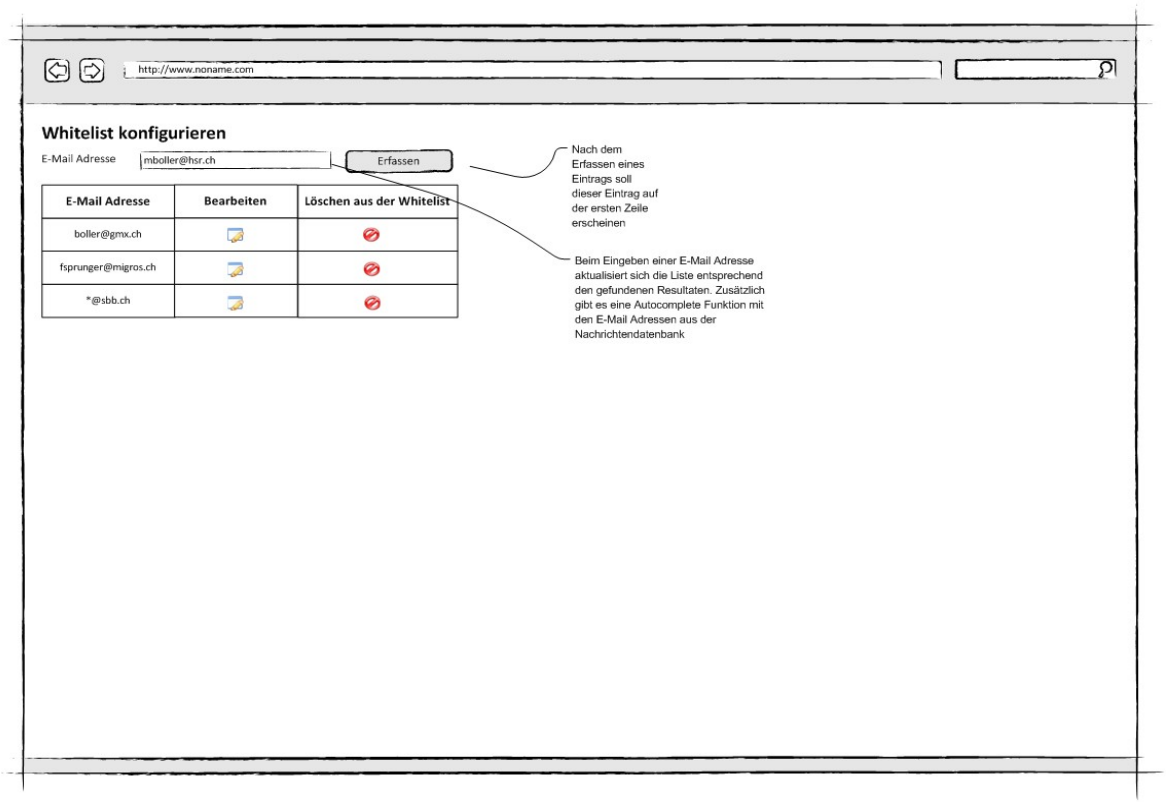


Abbildung 68 - Design Whitelist

10.10.3 Entwürfe für die Reporting Webapplikation

Abbildung 69 - Design Reportingapplikation Main

11 Realisierung

11.1 Klassendiagramm

11.1.1 Gesamtüberblick

Das vollständige Klassendiagramm ist auf der nächsten Seite zu finden.

11.1.2 Comparis.Framework.SmtpMailer.Service.Sending

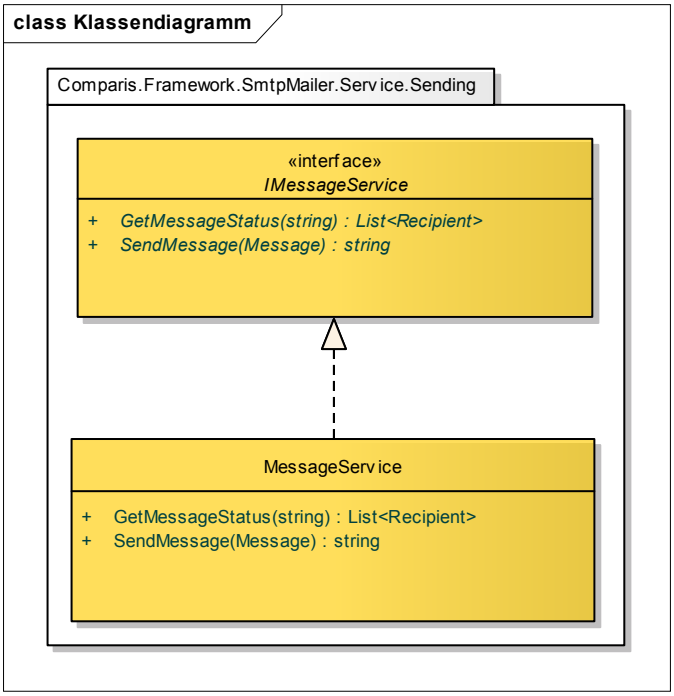


Abbildung 70 - Klassendiagramm Comparis.Framework.SmtpMailer.Service.Sending

11.1.2.1 Beschreibung des Namespace

Der Namespace Comparis.Framework.SmtpMailer.Service.Sending stellt ein Teil der öffentlichen API des *Smtp Mailers* dar, der von Entwicklern von comapris.ch genutzt werden kann.

Diese API erlaubt das Versenden von Nachrichten und das Abfragen vom Versandstatus einer Nachricht. Nachrichten werden über eine guid (String) identifiziert. Die guid wird beim Versenden einer Nachricht von der API zurückgegeben.

11.1.3 Comapris.Framework.SmtpMailer.Service.ReportingMaintenance

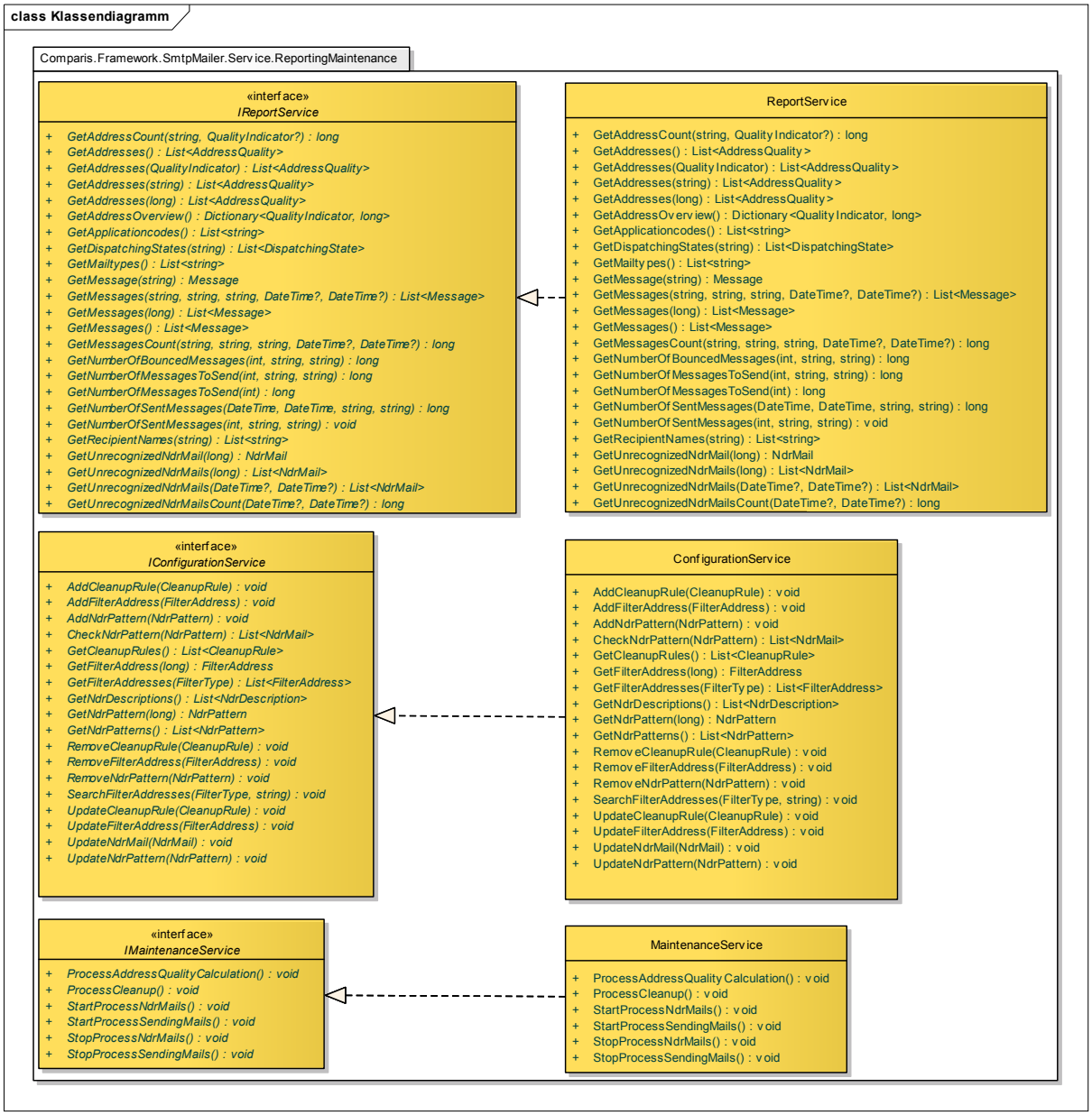


Abbildung 71 - Klassendiagramm Comapris.Framework.SmtpMailer.Service.ReportingMaintenance

11.1.3.1 Beschreibung des Namespace

Der Namespace Comparis.Framework.SmtpMailer.Service.ReportingMaintenance stellt ein Teil der öffentlichen API des *Smtp Mailers* dar, der von Entwickler von comapris.ch genutzt werden kann. Zudem werden diese Service Interfaces auch von den verschiedenen Engines des *Smtp Mailers* (SendingEngine, BounceEngine, AddressQualityEngine und CleanupEngine) und den Webapplikationen genutzt. Das heisst, dass die verschiedenen Engines und die Webapplikationen des *Smtp Mailers* wie andere Applikationen auch die Dienste des *Smtp Mailers* über die Service Interfaces nutzen und nicht direkt auf den Core zugreifen. Dies erlaubt eine bessere Kapselung und Wiederverwendbarkeit.

Die API erlaubt das Konfigurieren des *Smtp Mailers* und das Abfragen von Auswertungen zum E-Mail Verkehr, welcher über den *Smtp Mailer* abgewickelt wird.

11.1.4 Comparis.Framework.SmtpMailer.Core

Der Comparis.Framework.SmtpMailer.Core Namespace ist zur besseren Lesbarkeit in mehreren Bildern dargestellt:

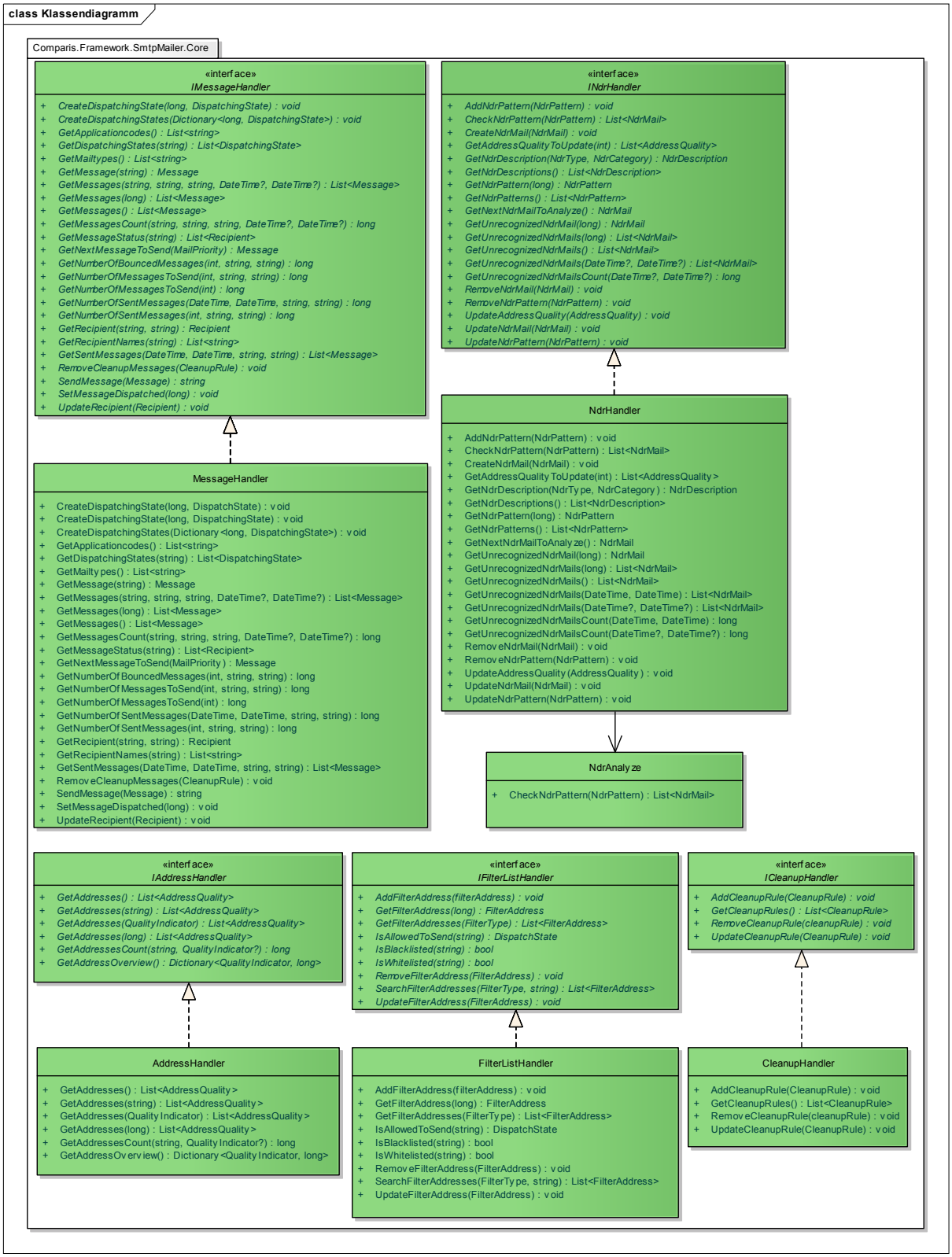


Abbildung 72 - Klassendiagramm Comparis.Framework.SmtpMailer.Core (Handler Klassen)

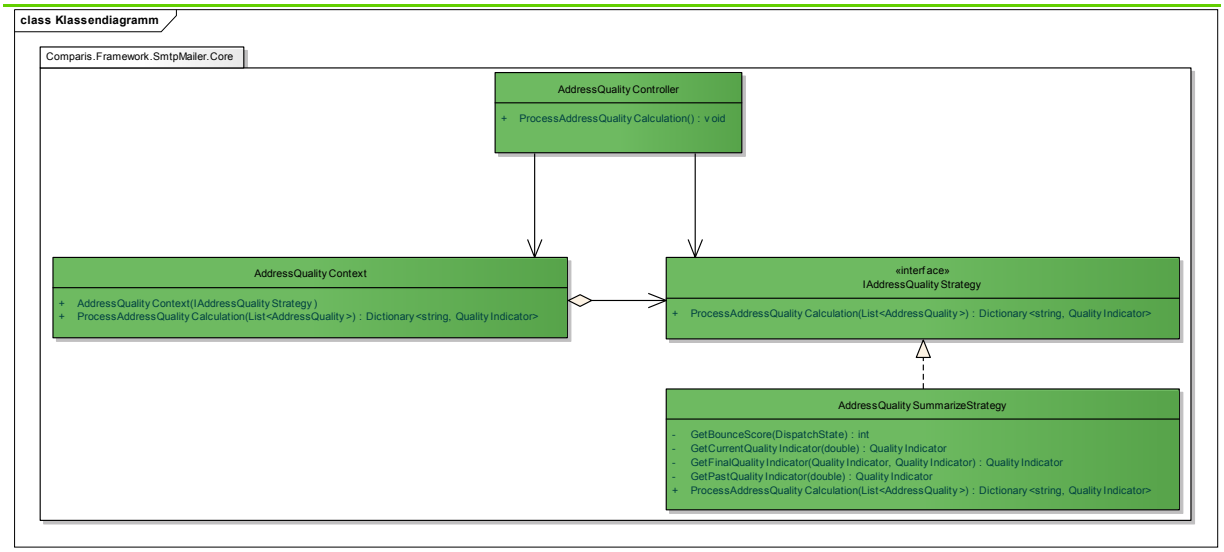


Abbildung 73 - Klassendiagramm Comparis.Framework.SmtpMailer.Core (AddressQuality)

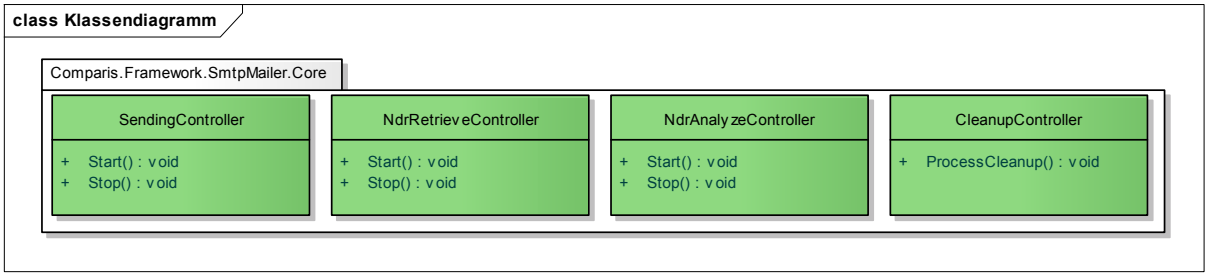


Abbildung 74 - Klassendiagramm Comparis.Framework.SmtpMailer.Core (Controller Klassen)

11.1.4.1 Beschreibung des Namespace

Der Namespace Comparis.Framework.SmtpMailer.Core implementiert die gesamte Business Logik des *Smtp Mailers.* Die Service Interfaces delegieren die ankommenden Aufrufe in den Core des *Smtp Mailers*. Gegenüber den Service Interfaces bietet der Core seine Funktionalität über Interfaces an.

Grundsätzlich behandeln Handler Klassen die vom Service Interface ankommenden Aufrufe. Zudem werden die übergebenen Parameterdaten von den Handlern validiert. Sind die übergebenen Daten ungültig, so wird eine Exception geworfen.

Controller Klassen bilden die Business Logik für die verschiedenen Engines des *Smtp Mailers* ab. So gibt es zu den vier Engines des *Smtp Mailers* (SendingEngine, BounceEngine, AddressQualityEngine und CleanupEngine) verschiedene Controller Klassen im Core (SendingController, NdrAnalyzeController, NdrRetrieveController, AddressQualityController und CleanupController).

11.1.5 Comparis.Framework.SmtpMailer.Data

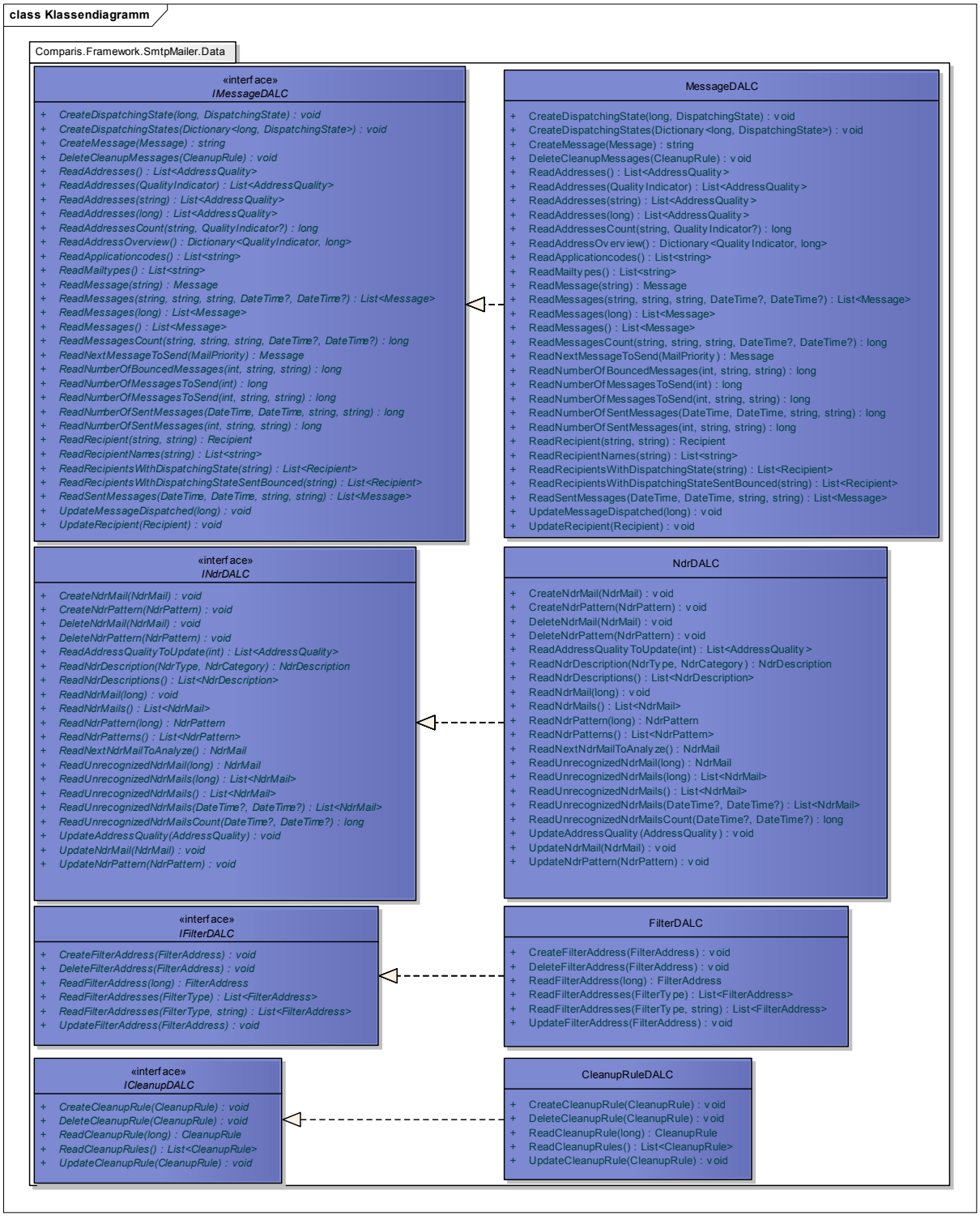


Abbildung 75 - Klassendiagramm Comparis.Framework.SmtpMailer.Data

11.1.5.1 Beschreibung des Namespace

Der Namespace Comparis.Framework.SmtpMailer.Data kümmert sich um die Persistenz der Business Objekte (DTO) des *Smtp Mailers*. Indem für sämtliche Funktionalitäten Interfaces zur Verfügung gestellt werden, kann der konkrete Persistenzmechanismus ohne Änderungen in oberen Schichten ausgetauscht werden. Über die Interfaces wird eine Abschottung resp. Kapselung erreicht.

Eine DALC Klasse (Data Access Layer Component) realisiert grundsätzlich die CRUD Operationen (Create, Read, Update und Delete) eines Data Transfer Objects.

Die DALC Klassen sind gegliedert nach Geschäftsfeldern des *Smtp Mailers*:

- Der MessageDALC kümmert sich um die Persistenz von DTO, welche mit dem Versenden von Nachrichten zu tun haben (Message, Recipient, AddressQuality, DispatchingState). Zudem besitzt der MessageDALC Methoden, um Reportingdaten bezüglich dem Nachrichtenversand aus der Datenbank zu lesen
- Der FilterDALC erlaubt es, CRUD Operationen auf die in der Datenbank abgebildeten Black- und Whitelist durchzuführen
- Der NdrDALC bietet CRUD Operationen für die DTO NdrPattern und NdrMail
- Der CleanupDALC ermöglicht CRUD Operationen auf das DTO CleanupRule

11.1.6 Comparis.Framework.SmtpMailer.Common

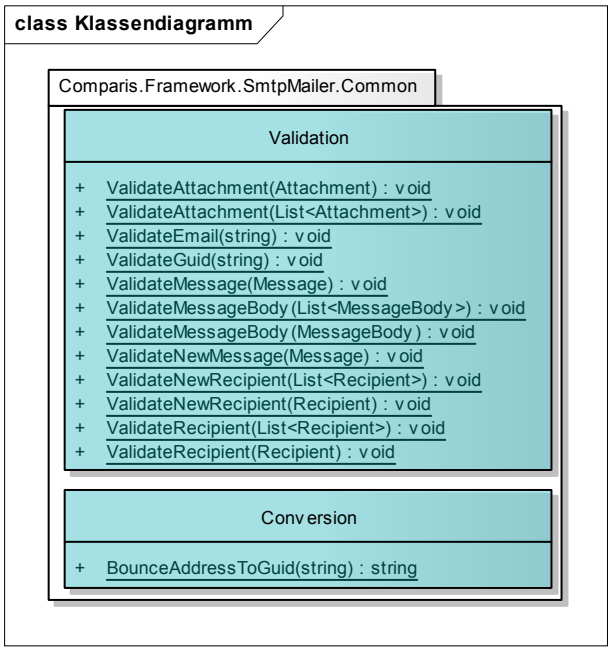


Abbildung 76 - Klassendiagramm Comparis.Framework.SmtpMailer.Common

11.1.6.1 Beschreibung des Namespace

Der Namespace Comparis.Framework.SmtpMailer.Common enthält eine Validierungsklasse, welche von den Handler Klassen im Core genutzt werden kann. So können verschiedene DTO Klassen über die Klasse Validation validiert werden. Es wird geprüft, ob alle Muss-Felder einen gültigen Wert besitzen. Falls ein DTO ungültig ist, wird eine Exception (ArgumentException) geworfen.

Die Klasse Conversion bietet eine Konvertierungsmethode von einer Bounce-Adresse zu einer Guid (Identifikationsmerkmal einer E-Mail Nachricht) an.

Alle Methoden in den Klassen Validation und Conversion des Common Namespaces sind statische Methoden. Dies bietet für die Benutzer der Methoden den Vorteil, dass nicht jedesmal ein Objekt von der Validation resp. Conversion Klasse instanziiert werden muss.

11.1.7 Comparis.Framework.SmtpMailer.DTO

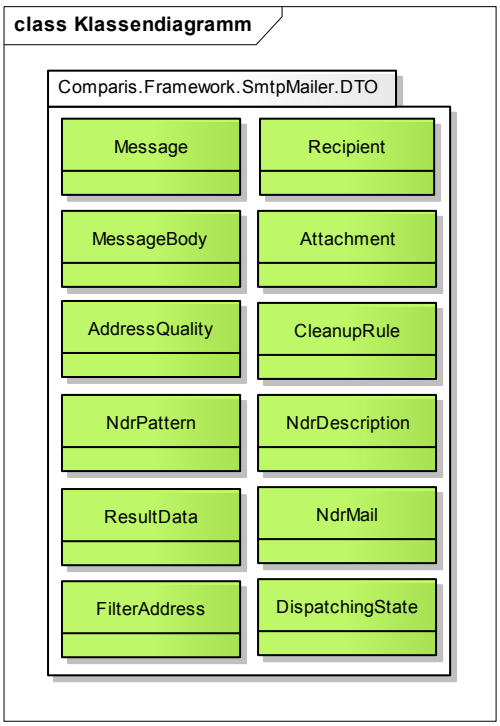


Abbildung 77 - Klassendiagramm Comparis.Framework.SmtpMailer.DTO

11.1.7.1 Beschreibung des Namespace

Der Namespace Comparis.Framework.SmtpMailer.DTO enthält sämtliche Business Objekte des *Smtp Mailers*, welche in Form vom DTO Pattern (Wikipedia, 2009) implementiert wurden.

Auf der Service-Schicht und im Core des *Smtp Mailers* wird mit den DTO Objekten gearbeitet. Der Data Access Layer bildet die DTO Objekte auf eine relationale Datenbank ab. Dadurch dass auf allen Layers mit den DTO Objekten gearbeitet wird, kann ein Konvertierungsaufwand verhindert werden.

11.1.8 Comparis.Framework.SmtpMailer.DTO.Enumerations

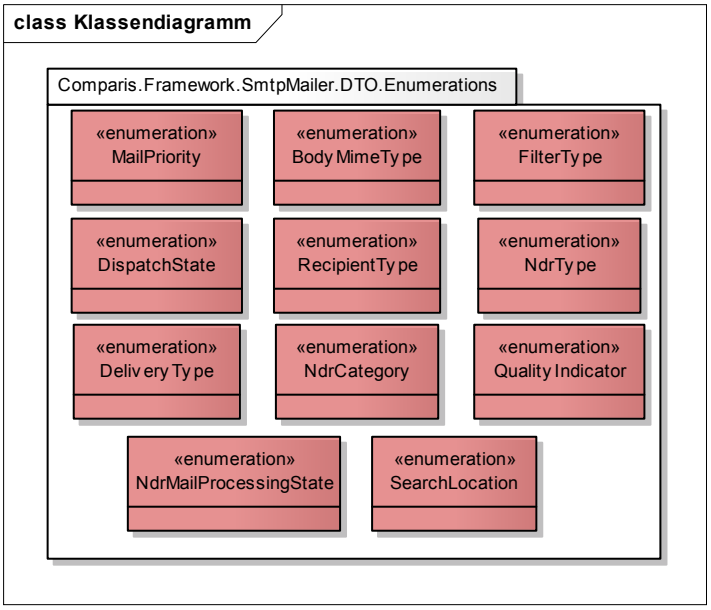


Abbildung 78 - Klassendiagramm Comparis.Framework.SmtpMailer.DTO.Enumerations

11.1.8.1 Beschreibung des Namespace

Der Namespace Comparis.Framework.SmtpMailer.DTO.Enumerations beinhaltet die verschiedenen Enumerations, welche im *Smtp Mailer* benutzt werden.

Die Enumerations sind zusätzlich in der Datenbank abgebildet. Dort wird an den Namen der jeweiligen Enumeration die Endung „Lookup“ angehängt. Die Enumerations werden also doppelt geführt. In der Datenbank stehen die Enumerations jedoch nur zu Dokumentationszwecken.

11.1.9 Comparis.Framework.SmtpMailer.Common.Configuration

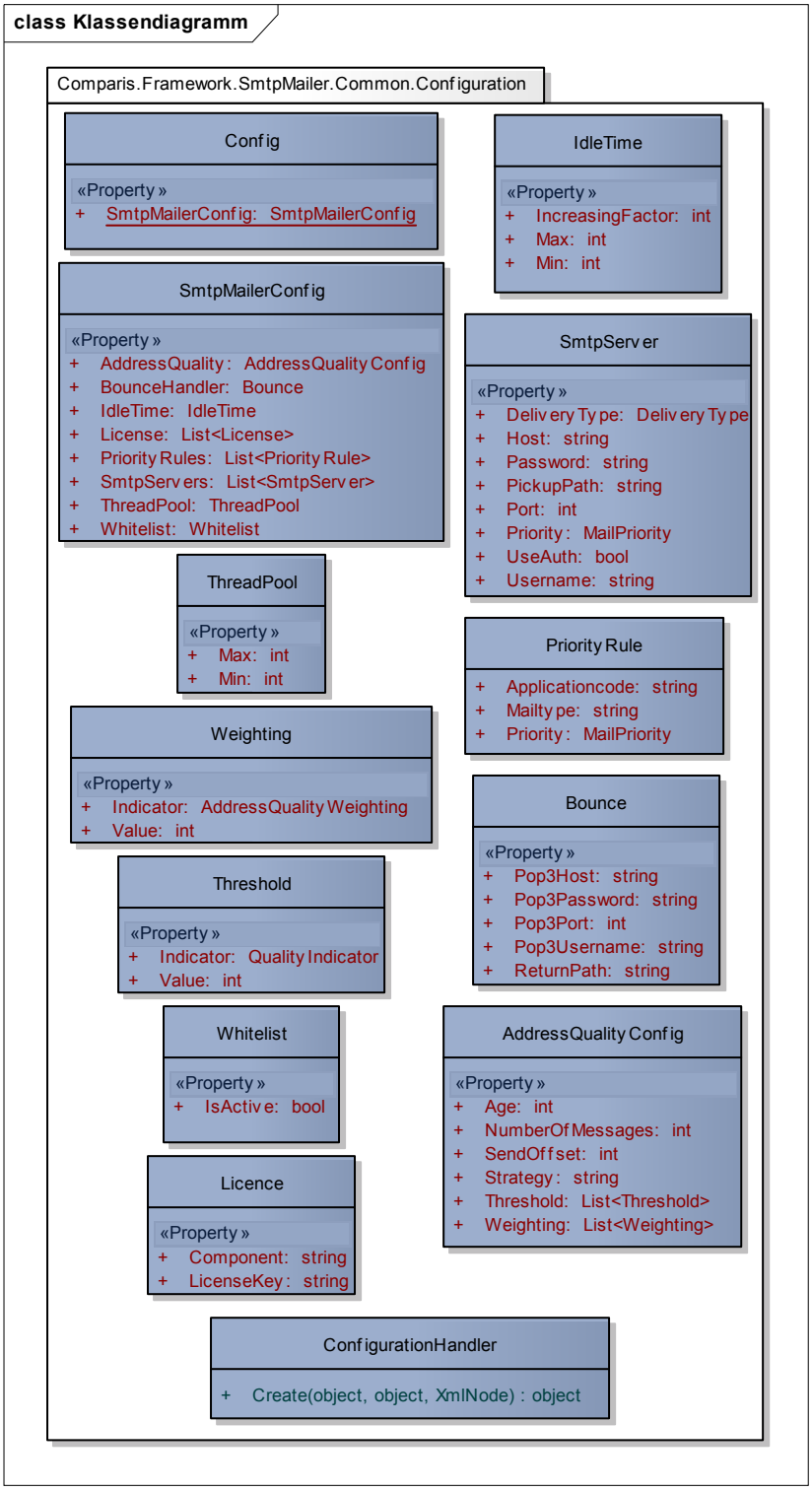


Abbildung 79 - Klassendiagramm Comparis.Framework.SmtpMailer.Common.Configuration

11.1.9.1 Beschreibung des Namespace

Der Namespace Comparis.Framework.SmtpMailer.Common.Configuration besitzt Klassen, welche die vom *Smtp Mailer* verwendete Konfiguration abbildet.

Die Klassen enthalten Attribute, welche es erlauben, die Daten per XML Serialisierung aus der Konfiguration zu lesen und auf die Objekte abzubilden.

11.1.10 Comparis.Framework.SmtpMailer.Common.Logging

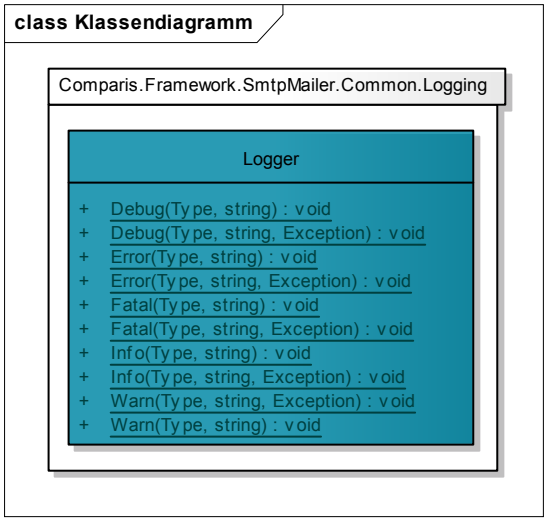


Abbildung 80 - Klassendiagramm Comparis.Framework.SmtpMailer.Common.Logging

11.1.10.1 Beschreibung des Namespace

Der Namespace Comparis.Framework.SmtpMailer.Common.Logging enthält die Logging Funktionalität des *Smtp Mailers.* Über statische Methoden ist es sämtlichen Klassen im *Smtp Mailer* möglich, Logeinträge vorzunehmen. Durch das Anbieten von statischen Methoden ergibt sich für die Benutzer der Logger Klassen den Vorteil, dass nicht jedesmal ein Objekt instanziiert werden muss.

Der konkrete Logging Mechanismus könnte ohne grossen Aufwand ausgetauscht werden, weil die statischen Methoden eine Art Objektadapter gemäss dem Adapter Pattern (Wikipedia, 2009) repräsentieren. Das heisst, in den verschiedenen statischen Methoden wird der Aufruf an ein Objekt des konkreten Loggingmechanismus weiterdelegiert. Momentan geschieht das Logging über die log4net Komponente.

11.1.11 SmtpMailer.AddressQualityEngine

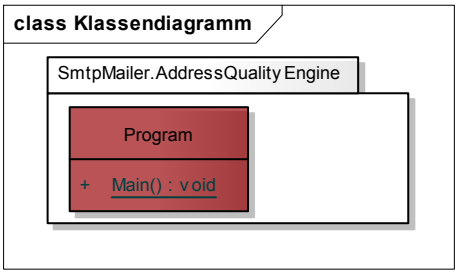


Abbildung 81 - Klassendiagramm SmtpMailer.AddressQualityEngine

11.1.11.1 Beschreibung des Namespace

Der Namespace SmtpMailer.AddressQualityEngine enthält lediglich eine Klasse, welche eine Konsolenapplikation repräsentiert. Mittels der Konsolenapplikation lässt sich die AddressQualityEngine starten. Die AddressQualityEngine kümmert sich um die Berechnung des Qualitätsindikators von E-Mail Adressen.

11.1.12 SmtpMailer.SendingEngine

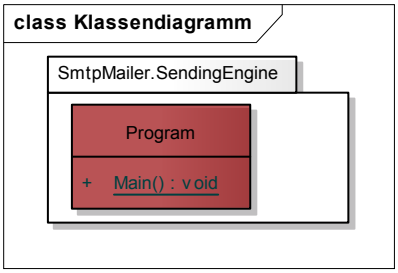


Abbildung 82 - Klassendiagramm SmtpMailer.SendingEngine

11.1.12.1 Beschreibung des Namespace

Der Namespace SmtpMailer.SendingEngine enthält lediglich eine Klasse, welche eine Konsolenapplikation repräsentiert. Mittels der Konsolenapplikation lässt sich die SendingEngine starten. Die SendingEngine verschickt die zu versendenden Nachrichten aus der Message Queue, welche in der Datenbank abgebildet ist.

11.1.13 SmtpMailer.BounceEngine

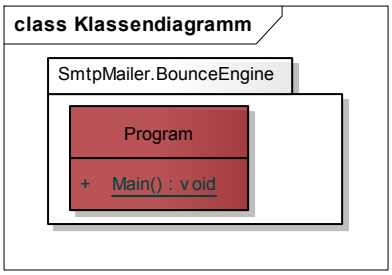


Abbildung 83 - Klassendiagramm SmtpMailer.BounceEngine

11.1.13.1 Beschreibung des Namespace

Der Namespace SmtpMailer.BounceEngine enthält lediglich eine Klasse, welche eine Konsolenapplikation repräsentiert. Mittels der Konsolenapplikation lässt sich die BounceEngine starten. Die BounceEngine kümmert sich um die Verarbeitung der Bounce Nachrichten.

11.1.14 SmtpMailer.CleanupEngine

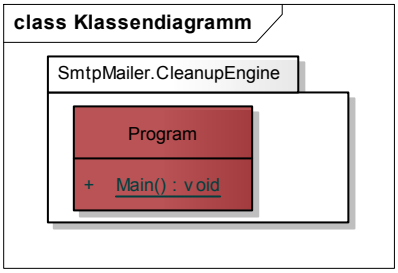


Abbildung 84 - Klassendiagramm SmtpMailer.CleanupEngine

11.1.14.1 Beschreibung des Namespace

Der Namespace SmtpMailer.CleanupEngine enthält lediglich eine Klasse, welche eine Konsolenapplikation repräsentiert. Mittels der Konsolenapplikation lässt sich die CleanupEngine starten. Die CleanupEngine arbeitet regelbasiert und bereinigt die in der Datenbank gespeicherten Nachrichten. Das heisst, es werden die speicherintensiven Attachments und Nachrichteninhalte in der Datenbank gelöscht.

11.1.15 SmtpMailerConfiguration

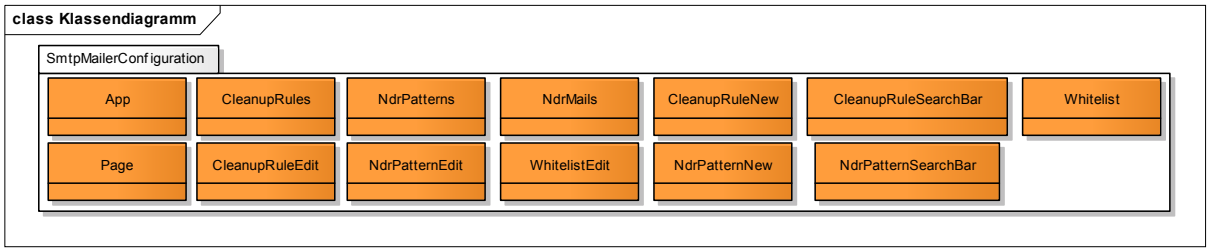


Abbildung 85 - Klassendiagramm SmtpMailerConfiguration

11.1.15.1 Beschreibung des Namespace

Der Namespace SmtpMailerConfiguration enthält die Silverlight Webapplikation, welche die Konfiguration des *Smtp Mailers* erlaubt. Dazu gehört die Verwaltung der CleanupRules, NdrPatterns, die Anzeige der unerkannten NdrMails sowie die Verwaltung der Whitelist.

11.1.16 SmtpMailerConfiguration.Web

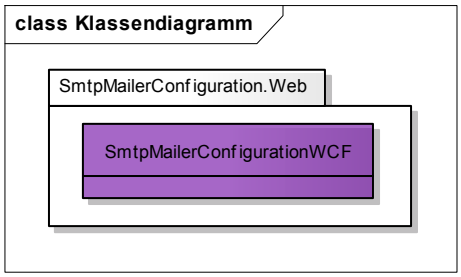


Abbildung 86 - Klassendiagramm SmtpMailerConfiguration.Web

11.1.16.1 Beschreibung des Namespace

Der Namespace SmtpMailerConfiguration.Web enthält die zur Silverlight Webapplikation SmtpMailerConfiguration gehörende ASP.NET Applikation. Die ASP.NET Applikation ruft die Silverlight Applikation auf.

11.1.17 SmtpMailerCustomerService

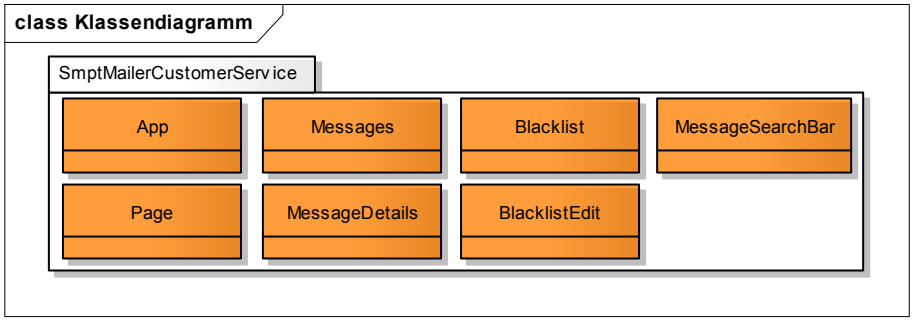


Abbildung 87 - Klassendiagramm SmtpMailerCustomerService

11.1.17.1 Beschreibung des Namespace

Der Namespace SmtpMailerCustomerService enthält die Silverlight Webapplikation, welche die Kundendienstmitarbeiter der comparis.ch AG nutzen. Dem Kundendienst ist es mit der Webapplikation möglich, nach zu versendenden oder versendeten Nachrichten zu suchen sowie die Blacklist zu konfigurieren.

11.1.18 SmtpMailerCustomerService.Web

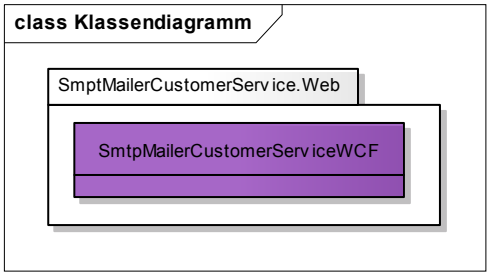


Abbildung 88 - Klassendiagramm SmtpMailerCustomerService.Web

11.1.18.1 Beschreibung des Namespace

Der Namespace SmtpMailerConfiguration.Web enthält die zur Silverlight Webapplikation SmtpMailerConfiguration gehörende ASP.NET Applikation. Die ASP.NET Applikation ruft die Silverlight Applikation auf.

11.1.19 SmtpMailerReporting

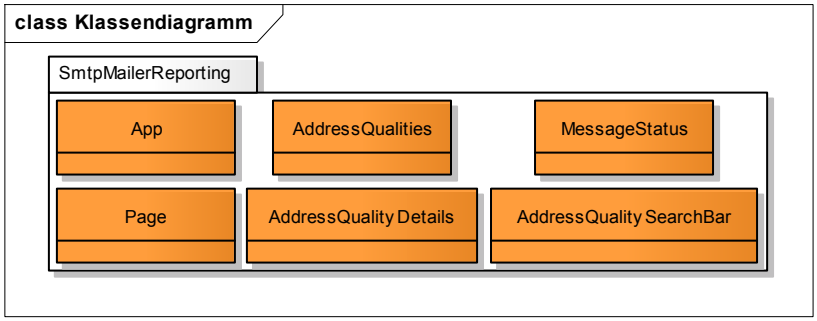


Abbildung 89 - Klassendiagramm SmtpMailerReporting

11.1.19.1 Beschreibung des Namespace

Der Namespace SmtpMailerReporting enthält die Silverlight Webapplikation, welche verschiedene Auswertungen bezüglich der Daten erlaubt, die über das *Smtp Mailer* System laufen. Dazu gehört ein Report, der grafisch aufzeigt, wie viele Nachrichten über einen Zeitraum verschickt wurden sowie zu verschicken sind. Des Weiteren kann der Qualitätsindikator von E-Mail Adressen abgefragt werden.

11.1.20 SmtpMailerReporting.Web

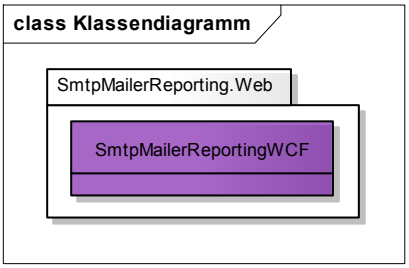


Abbildung 90 - Klassendiagramm SmtpMailerReporting.Web

11.1.20.1 Beschreibung des Namespace

Der Namespace SmtpMailerReporting.Web enthält die zur Silverlight Webapplikation SmtpMailerReporting gehörende ASP.NET Applikation. Die ASP.NET Applikation ruft die Silverlight Applikation auf.

11.2 Datenspeicherung

Das Datenmodell wird auf eine MS SQL Server 2005 Datenbank abgebildet. Sämtliche datenbankspezifischen Details werden von einem Data Access Layer gekapselt.

11.2.1 Datenmodell

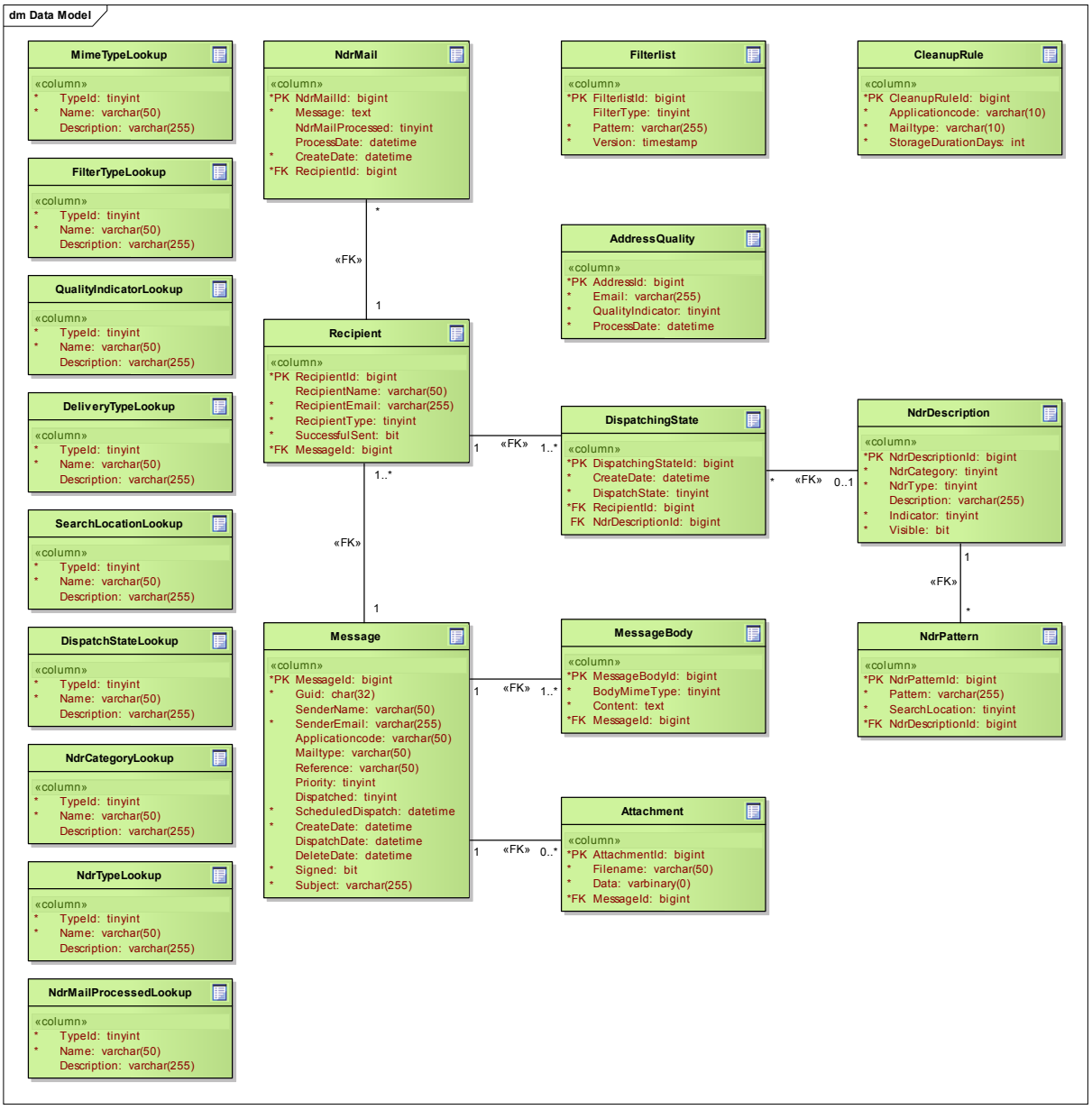


Abbildung 91 - Datenmodell

11.2.2 Tabellen

Tabellenname	*Message, MessageBody, Attachment, Recipient, DispatchingState*
Beschreibung	Die Tabelle Message definiert die eigentliche E-Mail Nachricht und enthält die Daten der zu versendenden und bereits versendeten E-Mail Nachrichten. Diese referenziert die für den Versand wichtigen Tabellen MessageBody (Inhalt), Attachment (Dateianhänge) und Recipient (Nachrichten Empfänger). Die Versandgeschichte wird in der Tabelle DispatchingState (Versandstatus) für jeden Empfänger einzeln geführt.

Tabellenname	*NdrMail, NdrDescription, NdrPattern*
Beschreibung	Die Tabelle NdrMail nimmt alle Bounce Nachrichten auf, welche noch verarbeitet und ausgewertet werden müssen. Dabei werden die in der Tabelle NdrPattern hinterlegten Signaturen benutzt, um die Bounce Nachricht (resp. den Grund für die Bounce Nachricht) eindeutig zu erkennen. Wird der Bounce Grund erkannt, wird die NdrDescription mit dem DispatchingState des Recipients assoziiert.

Tabellenname	*Filterlist*
Beschreibung	Enthält die Einträge für die Black- / Whitelist. Vor dem effektiven versenden wird in dieser Liste nachgeschlagen, ob der Versand an die E-Mail Adresse erlaubt ist.

Tabellenname	*CleanupRule*
Beschreibung	Gibt an, welche Nachrichten (Applikationscode / Mailtyp Tupel) nach welcher Zeit gelöscht werden sollen. Aufgrund dieser CleanupRules kann die CleanupEngine die Attachments und MessageBodies von Nachrichten löschen.

Tabellenname	*AddressQuality*
Beschreibung	In dieser Tabelle wird ein Qualitätsindikator für jede benutzte E-Mail Adresse geführt, welche die Zuverlässigkeit beim Versenden angibt (good, poor, failed)

Tabellenname	**Lookup*
Beschreibung	Für jede Enumeration, welche im Code verwendet wird, gibt es in der Datenbank eine Verzeichnistabelle. Diese gibt Auskunft darüber, was die Bedeutung der einzelnen Zahlenwerte sind und wie die Enumeration im Code benannt wurde.

11.2.3 Datenbankanbindung im Code

Für das Objekt Rationale Mapping wird ADO.NET verwendet. Auf LINQ to SQL wurde aus Performancegründen verzichtet sowie weil sich die Technologie bei Microsoft auf dem absteigenden Ast befindet.

Mit ADO.NET musste zwar das OR Mapping manuell implementiert werden, dafür bietet es vollständige Kontrolle über die SQL Statements. So war es erst mit ADO.NET möglich, einen ROWLOCK auf die nächste zu versendende Nachricht oder die nächste zu verarbeitende NdrMail zu setzten.

11.3 Prozesse und Threads

Teile des *Smtp Mailer* Systems wurden mittels Multi-Threading realisiert um mehr Performanz bei der Verarbeitung der Daten zu erreichen. Dies betrifft die beiden Module *SendingEngine* und *BounceEngine*. Die *SendingEngine* liest die nächste zu versendende Nachricht aus der Datenbank aus und sendet die Nachricht an einen der konfigurierten SMTP Server. Die *BounceEngine* nimmt Bounce Nachrichten von einem POP3 Server entgegen und wertet diese aus.

11.3.1 SendingEngine

Beim Start der Anwendung wird für jeden konfigurierten SMTP Server ein Kontrollerthread erzeugt. Dieser hat die Aufgabe die nächste Nachricht von der Datenbank zu erlangen, welche dieselbe Priorität hat wie für den SMTP Server konfiguriert wurde. Obwohl es nur zwei Prioritäten (hoch und tief) gibt, können pro Priorität mehrere SMTP Server zuständig sein.

Nachdem die Nachricht ausgelesen wurde, wird die Nachricht mit weiteren Versandparametern an einen Arbeitsthread (Workerthread) übergeben, welcher aus dem *Threadpool* geholt wird. Der Arbeitsthread stellt die E-Mail Nachricht anhand der übergebenen Daten zusammen und versucht die Nachricht an den SMTP Server auszuliefern.

Gibt es keine Arbeit mehr zu verrichten resp. sind keine zu versendenden Nachrichten vorhanden, wird der Kontrollerthread für eine bestimmte Zeit schlafen gelegt um nicht Ressourcen zu verbrauchen.

11.3.2 BounceEngine

Beim Start der Anwendung werden genau zwei Kontrollerthread erzeugt. Der eine Kontrollerthread ist zuständig für das Abholen der Bounce Nachrichten auf dem POP3 Server und dem anschliessenden Speichern in die Datenbank. Der zweite Kontrollerthread liest die nächste auszuwertende Bounce Nachricht von der Datenbank aus und übergibt die Daten und weitere Parameter an einen Arbeitsthread (Workerthread), der aus dem Threadpool geholt wird. Der Arbeitsthread nimmt die Auswertung vor und aktualisiert die Datenbank.

Gibt es keine Arbeit mehr zu verrichten resp. sind keine auszuwertende Bounce Nachrichten mehr vorhanden, wird der Kontrollerthread für eine bestimmte Zeit schlafen gelegt um nicht Ressourcen zu verbrauchen.

11.3.3 Strategie

11.3.3.1 Algorithmus

Die Strategie beim Einsatz von Multi-Threading war es, möglichst dort zu parallelisieren, wo ein gleichzeitiges Abarbeiten nicht mehr Ressourcen in Anspruch nimmt, die Verarbeitungzeit aber massiv beschleunigt.

Beim Einsatz von Kontrollerthreads, die selber erstellt werden müssen, waren wir zurückhaltend und haben uns entschieden diese nur für den kleinsten gemeinsamen Nenner zu benutzen. Im Fall der SendingEngine sind es die SmtpServer, wobei jeder Kontrollerthread die Warteschlange für einen SmtpServer darstellt. Bei der BounceEngine wurde die Aufteilung beim Abrufen resp. Verarbeiten gemacht. Dies hat den grossen Vorteil, dass die Anzahl der zu kontrollierenden Threads übersichtlich bleibt und die angrenzenden Schnittstellen wie Datenbank oder Mailserver nicht überlastet werden mit Verbindungsaufbauanfragen. Die Kontroller-Threads laufen in einer Endlosschleife wobei sie für eine bestimmte Zeit schlafen gelegt werden, wenn keine Arbeit ansteht.

Workerthreads aus dem *Threadpool* haben den Vorteil, dass sie nach Ablauf nicht vernichtet werden, sondern in den Threadpool zurück gelangen wo sie wieder einsatzbereit sind. Somit fällt beim Einsatz von einem Threadpool auch nicht jeweils den Erstellungsoverhead für die Erzeugung eines neuen

Threads an. Der Threadpool ist eine vorhandene API Klasse aus dem Namespace System.Threading. Diese Klasse verwaltet die Workerthreads.

11.3.3.2 Transaktionsmanagement

Die Transaktionen im *Smtp Mailer* System beziehen meistens mehrere Architektur Schichten und mehrere Threads mit ein. Daher kann die Transaktionssicherheit der Daten nicht auf Datenbank-Ebene (DAL) gelöst werden.

Der Windows Dienst *Distributed Transaction Coordinator* (*DTC*) erlaubt das Erstellen und Aufrechterhalten von Transaktionen über die oben angesprochenen Grenzen hinweg. Ein typisches Beispiel ist der Versand einer Nachricht. Gleich nach dem Auslesen des Datensatzes auf DAL Ebene muss dieser temporär gesperrt und mit einem Flag in der Datenbank markiert werden, dass der Datensatz schon in Verarbeitung ist. Gleichzeitig wird der Datensatz an einen Arbeitsthread im Core übergeben, der wiederum versucht, die Nachricht zu versenden. Je nach dem, wo Fehler passieren resp. die Verarbeitung erfolgreich war, müssen gewisse Daten wiederhergestellt werden um die Konsistenz zu bewahren (z.B. zurücksetzen des Flags in der Datenbank).

11.3.3.3 Datenbank Lock

Beim Auslesen von Datensätzen, die über eine DTC Transaktion gesteuert wird, mussten auf Datenbankebene verschiedene Locks eingesetzt werden. Die Abfragen wurden generell mit *TOP 1* ausgeführt.

Mit dem *ROWLOCK* wird verhindert, dass beim Abfragen eines Datensatzes gleich die ganze Page oder Tabelle gelockt wird. Dies würde bei einer parallelen Abarbeitung zu enormen Wartezeiten führen.

Mit dem *UPDLOCK* wird sichergestellt, dass der Lock solange erhalten bleibt bis eine Aktualisierung auf diesen Datensatz statt gefunden hat.

Mit *READPAST* wird sichergestellt, dass eine erneute Abfrage mit *TOP 1* alle zuvor gelockten Datensätze übersprungen werden und entsprechend der erste nicht gelockte Datensatz zurückgegeben wird.

11.4 Ordnerstruktur

Das gesamte Projektverzeichnis befindet sich auf einem SVN Server und wird auf den Entwicklungsrechnern ausgecheckt.

Ordner	Beschreibung
/SmtpMailer	Projektverzeichnis
/cd	Dieses Verzeichnis enthält alle Daten, welche auf die abzugebende CD kommen
/dev	Entwicklungsverzeichnis
/conf	Konfigurationsfiles wie die App.config
/dll	Verwendete Assemblies (log4net, NetXtremeBounceFilter, NUnit, SMTP, SMIME, Silverlight Toolkit etc.)
/doc	Generierte Codedokumentation
/ico	Icons für die Webapplikation
/prot	Verzeichnis mit den entwickelten Prototypen
/Architecture Prototype	Architektur Prototyp
/R04Prototype	Prototyp zur Behandlung von Risiko 04
/R05Prototype	Prototyp zur Behandlung von Risiko 05
/sql	SQL Skripts (Grundstruktur der Datenbank, Initialdaten etc.)
/src	Source Verzeichnis mit Visual Studio Solution

/doc	Dokumentationsverzeichnis
/diagrams	Enterprise Architect Files für UML Diagramme und Visio Files für GUI Designs
/icons	In den GUI Designs verwendete Icons
/images	Bilder von UML Diagrammen und GUI Designs
/presentation	Power Point Präsentationen für die gehaltenen Vorträge
/templates	Word Dokumentvorlage
/traktandenliste	Traktandenlisten von Sitzungen
/info	Dokumente, welche Informationen zur Bachelorarbeiten beinhalten (Aufgabenstellung, Dokumente des Industriepartners, etc.)

Tabelle 60 - Ordnerstruktur

11.5 Physikalische Sicht

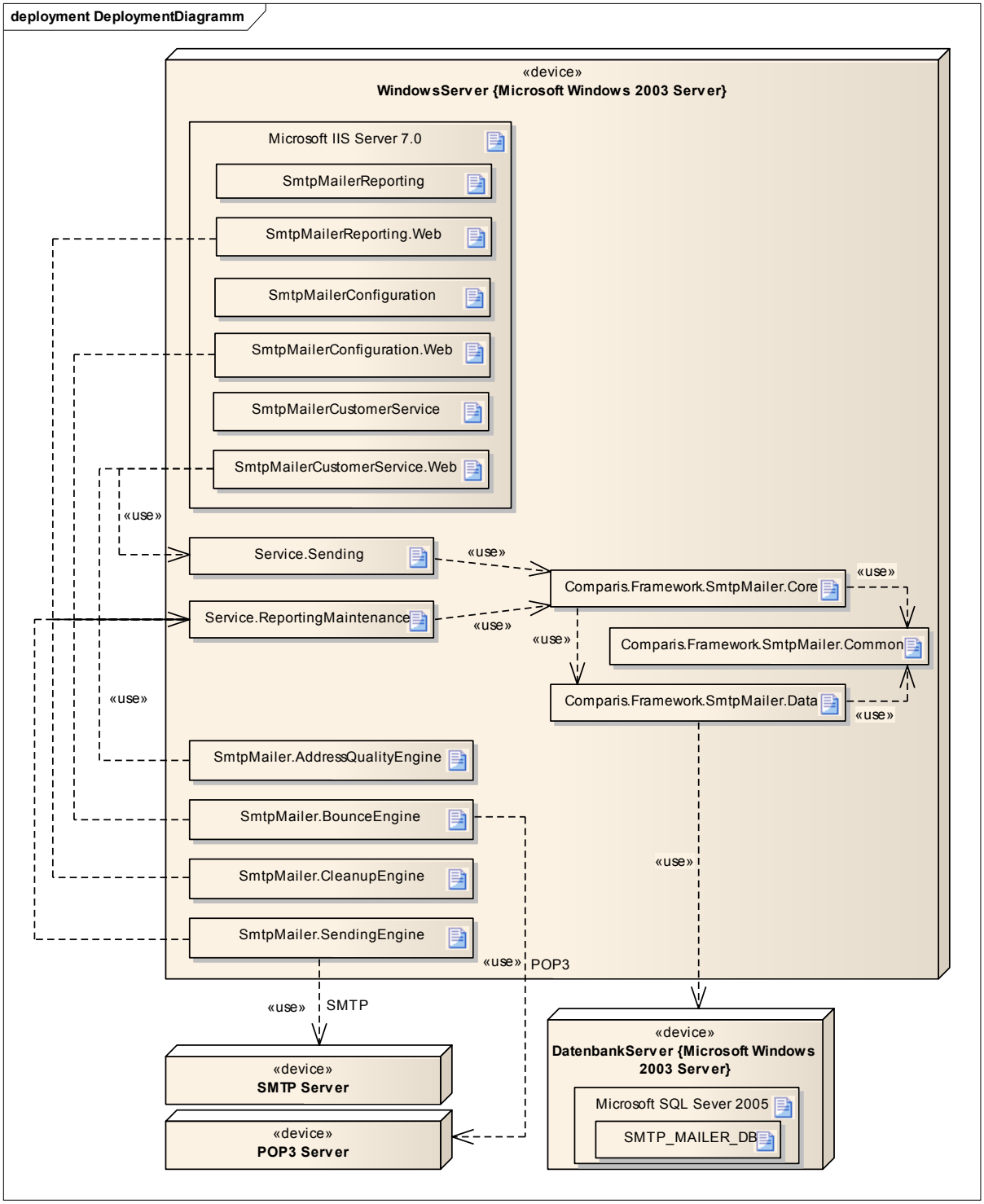


Abbildung 92 - Physikalische Sicht des Smtp Mailers

Die verschiedenen Engines des *Smtp Mailers* laufen alle auf einem Windows 2003 Server. Die AddressQualityEngine sowie die CleanupEngine laufen nicht permanent, sondern werden zu geplanten Zeitpunkten gestartet. Die SendingEngine sowie BounceEngine hingegen laufen permanent und legen sich schlafen, wenn sie nichts zu tun haben.

Die Datenbank läuft auf einem MS SQL Server 2005. Die Datenbank wird von verschiedensten Softwareteilen des *Smtp Mailers* benutzt.

Über einen SMTP Server werden die zu versendenden Nachrichten verschickt. Falls beim Versenden einer Nachricht ein Problem auftrat, so wird eine Bounce-Nachricht an eine bestimmte E-Mail Adresse zurückgeschickt. Die Bounce-Nachrichten können von der BounceEngine auf einem definierten POP3 Server abgeholt und verarbeitet werden.

Die Reporting- und Konfigurationsapplikationen, welche als Silverlight Applikationen realisiert sind (SmtpMailerCustomerService, SmtpMailerConfiguration, SmtpMailerCustomerService), laufen auf einem Microsoft Internet Information Services Server.

Jede Silverlight Applikation besitzt eine ASP.NET Applikation (SmtpMailerReporting.Web, SmtpMailerConfiguration.Web, SmtpMailerCustomerService.Web), welche die jeweilige Silverlight Applikation aufruft.

11.6 Konfigurationsfile

Sämtliche Einstellungen, welche standortabhängig konfiguriert werden müssen, befinden sich in einer App.config Konfigurationsdatei.

Bei der App.config Konfigurationsdatei handelt es sich um eine XML Textdatei. Änderungen können mit einem gewöhnlichen Texteditor vorgenommen werden. Durch das Verwenden von beschreibenden Tag- und Attributnamen kann das Konfigurationsfile praktisch ohne Anleitung abgeändert werden.

11.6.1 Grundaufbau

Der Grundaufbau eines App.config Files folgendermassen aus:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
</configuration>
```

Zwischen die beiden *configuration* Tags wird die applikationsspezifische Konfiguration in XML Syntax eingefügt.

11.6.2 Applikationsspezifische Konfiguration

Die Konfiguration, welche den *Smtp Mailer* betrifft wird zwischen die beiden *configuration* Tags platziert. Die Konfiguration zwischen den beiden *configuration* Tags besitzt folgenden Aufbau:

```
<configSections>
</configSections>

<connectionStrings>
</connectionStrings>

<smtpmailer>
    <priorityrules>
    </priorityrules>

    <threadpool/>

    <idletime/>

    <bouncehandler/>

    <addressquality>
    </addressquality>

    <licenses>
    </licenses>
</smtpmailer>
```

Nicht jede App.config Datei muss alle diese Tags beinhalten. Dies weicht je nach Assembly ab, in welcher die Konfiguration verwendet wird.

11.6.2.1 configSections

```
<configSections>
    <section
     name="smtpmailer"
     type="Comparis.Framework.SmtpMailer.Common.Configuration.ConfigurationHandler,
          Comparis.Framework.SmtpMailer.Common.Configuration"
    />
</configSections>
```

Mittels dem *configSections* Tag werden Angaben zur Konfiguration gemacht. Durch den *section* Tag wird eine Beziehung zwischen dem Konfigurationsektion-Handler und dem Konfigurationselement hergestellt.

Diesen Teil der Konfiguration darf nicht abgeändert werden.

11.6.2.2 connectionStrings

```
<connectionStrings>
 <add
  name="Comparis.Framework.SmtpMailer.Data.Properties.Settings.CONN SMTP MAILER DB"
  connectionString="Data Source=localhost\sql2008;Initial
                    Catalog=SMTP MAILER DB;Integrated Security=True"
 />
</connectionStrings>
```

Der *connectionStrings* Tag erlaubt es, die Datenbankverbindungen zu konfigurieren.

Tagname	Beschreibung
add	Definiert einen ConnectionString zu einer Datenbank
Attributname	**Beschreibung**
name	Definiert eine Identität für die Datenquelle. Mittels diesem Namen kann auf den ConnectionString zugegriffen werden.
connectionString	Definiert den Datenbankserver, die Datenbank welche verwendet werden soll, sowie allfällige Authentifizierungsinformationen (Benutzername, Passwort für Datenbankserver)

11.6.2.3 smtpmailer

```
<smtpmailer type="Comparis.Framework.SmtpMailer.Common.Configuration.SmtpMailerConfig,
Comparis.Framework.SmtpMailer.Common.Configuration">

 <smtpservers>
  <smtpserver priority="Low" deliverytype="Smtp" host="localhost" port="25" useauth="1"
              username="test" password="test" />
  <smtpserver priority="High" deliverytype="Smtp" host="localhost" port="26" useauth="0"
/>
  <smtpserver priority="High" deliverytype="Queue" pickuppath="Z:\pickup" />
  <smtpserver priority="Low" deliverytype="Queue" pickuppath="Z:\pickup" />
 </smtpservers>

 <priorityrules>
  <rule applicationcode="*" mailtype="PWFORGOTTEN" priority="High"/>
  <rule applicationcode="*" mailtype="NEWSLETTERHIGH" priority="High"/>
  <rule applicationcode="APPNL" mailtype="*" priority="Low"/>
  <rule applicationcode="APPKK" mailtype="KK" priority="High"/>
  <rule applicationcode="*" mailtype="*" priority="Low"/>
 </priorityrules>

 <whitelist active="0" />

 <threadpool min="2" max="500" />

 <idletime min="0" max="0" increasingfactor="0" />

 <bouncehandler returnpath="bounce-{0}@ba.danflash.com" pop3host="localhost"
pop3port="110"
               pop3username="bounce@ba.danflash.com" pop3password=" Zweifel " />

 <addressquality age="0"

strategy="Comparis.Framework.SmtpMailer.Core.AddressQualitySummarizeStrategy">
  <weightings>
   <weighting indicator="Ignore" value="0"  />
   <weighting indicator="Low" value="2"  />
   <weighting indicator="Medium" value="4"  />
   <weighting indicator="High" value="8"  />
  </weightings>

  <thresholds>
   <threshold indicator="Good" value="0" />
   <threshold indicator="Poor" value="8" />
   <threshold indicator="Failed" value="16" />
  </thresholds>
 </addressquality>
```

```
 <licenses>
  <license component="EasyMailSmtp" licensekey="***" />
  <license component="EasyMailSmime" licensekey="***" />
 </licenses>

</smtpmailer>
```

Im *smtpmailer* Tag werden Einstellungen vorgenommen, welche den *Smtp Mailer* betreffen.

smtpservers

Tagname	Beschreibung
smtpservers	Enthält ein bis mehrere *smtpserver* Tags, welche einen SMTP Server beschreiben

Tagname	Beschreibung
smtpserver	Beschreibt einen SMTP Server, welcher vom *Smtp Mailer* zum Versenden der E-Mail Nachrichten verwendet wird
Attributname	**Beschreibung**
priority	*High* oder *Low*. Definiert, ob über diesen SMTP Server High- oder Low-Priority E-Mail Nachrichten versendet werden
deliverytype	*Smtp* oder *Queue*. Beschreibt, wie dass die E-Mail Nachrichten dem SMTP Server übergeben werden. Mit dem Wert Smtp werden die Nachrichten per SMTP Protokoll dem SMTP Server übergeben. Mit dem Wert Queue wird die zu versendende E-Mail Nachricht in ein definiertes Verzeichnis abgelegt. Der SMTP Server holt in diesem Verzeichnis die zu versendenden E-Mail Nachrichten ab. Beim Wert Queue ist das Attribut pickuppath notwendig
host	Definiert die IP Adresse oder den Hostnamen des SMTP Servers
port	Definiert den Port, über welchen der SMTP Server angesprochen werden kann (Normalerweise Port 25)
useauth	*1* oder *0*. *1* bedeutet, dass eine Authentifizierung am SMTP Server notwendig ist. Dann müssen die Attribute username und password angegeben werden. *0* bedeutet, dass keine Authentifizierung am SMTP Server notwendig ist
username	Benutzername, um sich am SMTP Server authentifizieren
password	Passwort, um sich am SMTP Server zu authentifizieren
pickuppath	Pfadangabe des Verzeichnis, in welches die zu versendenden E-Mail Nachrichten im *.eml Format abgelegt werden

priorityrules

Tagname	Beschreibung
priorityrules	Enthält mehrere *rule* Tags, welche einem Applikationscode / Mailtyp Tupel eine Priorität zuweisen (High oder Low)

Tagname	Beschreibung
rule	Weist einem Applikationscode / Mailtyp Tupel eine Priorität zu (High oder Low)
Attributname	**Beschreibung**
applikationcode	Name des Applikationscodes oder * als Wildcard.
mailtype	Name des Mailtypes oder * als Wildcard.
priority	*High* oder *Low*. Definiert, ob die Nachricht mit hoher oder tiefer Priorität versendet werden soll

whitelist

Tagname	Beschreibung
whitelist	Definiert, ob die Whitelist aktiv ist oder nicht
Attributname	**Beschreibung**

active	*1* oder *0*. *1* bedeutet, dass die Whitelist aktiv ist. *0* bedeutet, dass die Whitelist inaktiv ist

threadpool

Tagname	Beschreibung
threadpool	Definiert die Kapazität des Threadpools
Attributname	**Beschreibung**
min	Gibt an, wie viele Workerthreads mindestens auf Vorrat im Threadpool instanziiert vorliegen
max	Gibt an, wie viele Workerthreads maximal vom Threadpool zur Ausführung genutzt werden

idletime

Tagname	Beschreibung
idletime	Definiert die Schlafzeit eines Kontrollerthreads, welcher nach neuen Aufträgen fragt und diese an Workerthreads verteilt. Wenn der Kontrollerthread nichts zu tun hat, beginnt er zu schlafen. Diese Schlafzeit wird mittels dem *idletime* Tag definiert
Attributname	**Beschreibung**
min	Minimale Schlafzeit eines Workerthreads, wenn er nichts zu tun hat.
max	Maximale Schlafzeit eines Workerthreads, wenn er nichts zu tun hat.
increasingfactor	Um diesen Faktor wächst die Schlafzeit (Multiplikation der aktuellen Schlafzeit mit dem *increasingfactor*), bis die Schlafzeit das mittels *max* definierte Maximum erreicht hat. Sobald der Kontrollerthread wieder etwas zu tun hat, fällt die Schlafzeit auf *min* zurück

bouncehandler

Tagname	Beschreibung
bouncehandler	Definiert die Eigenschaften des Bouncehandlings
Attributname	**Beschreibung**
returnpath	Definiert das Format des Returnpath der versendeten E-Mail Nachrichten. Mittels {0} wird ein Platzhalter in den Returnpath eingefügt, welcher mit der guid der jeweiligen E-Mail Nachricht abgefüllt wird
pop3host	Definiert die IP Adresse oder den Hostnamen des POP3 Servers, von welchem die Bounce Nachrichten abgerufen werden können
pop3port	Definiert den Port, über welchen der POP3 Server angesprochen werden kann (Normalerweise Port 110)
pop3username	Benutzername, um sich am POP3 Server authentifizieren
pop3password	Passwort, um sich am POP3 Server zu authentifizieren

addressquality

Tagname	Beschreibung
addressquality	Definiert die Eigenschaften des Algorithmus zur Bestimmung des Qualitätsindikators einer E-Mail Adresse. Enthält ein *weightings* Tag mit mehreren *weighting* Tags, sowie ein *thresholds* Tag mit mehreren *threshold* Tags
Attributname	**Beschreibung**
age	Jeder Qualitätsindikator enthält ein Datum, wann er zuletzt berechnet wurde. Es werden nur diejenigen Qualitätsindikatoren neu berechnet, deren letzte Berechnung länger als die mittels *age* angegebene Anzahl Tage zurückliegt
strategy	Für die Berechnung des Qualitätsindikators können verschiedene Strategien implementiert werden. Die Strategie, welche verwendet werden soll, wird hier mit dem vollen Namespace- und Klassennamen angegeben

Tagname	Beschreibung

weightings	Enthält mehrere *wheighting* Tags.

Tagname	Beschreibung
weighting	Beschreibt die Gewichtung eines aufgetretenen Bounce-Typs. Tritt ein solcher Bounce-Typ auf, so wird der unter *value* definierte Zahlenwert mit in die Berechnung des Qualitätsindikators genommen
Attributname	**Beschreibung**
indicator	Name der Gewichtung (Ignore, Low, Medium, High).
value	Gewichtung als Zahlenwert.

Die Indikatornamen dürfen nicht verändert werden.

Tagname	Beschreibung
thresholds	Enthält mehrere *threshold* Tags.

Tagname	Beschreibung
threshold	Beschreibt die Schwellenwerte für die verschiedenen Stufen, in welche der Qualitätsindikator klassiert werden kann
Attributname	**Beschreibung**
indicator	Name der Klassierungsstufe (Good, Poor, Failed)
value	Definiert den Schwellenwert als eine Zahl. Ab diesem Wert fällt der Qualitätsindikator in die mittels indicator bezeichnete Stufe

Die Indikatornamen dürfen nicht verändert werden.

Licences

Tagname	Beschreibung
thresholds	Enthält genau zwei *license* Tags

Tagname	Beschreibung
license	Beschreibt einen Software Lizenzschlüssel, welcher zur Laufzeit verwendet wird.
Attributname	**Beschreibung**
component	Name der Komponente, für welche der Lizenzschlüssel verwendet wird
licensekey	Lizenzschlüssel als Zeichenfolge

Es müssen für die Komponenten EasyMailSmtp und EasyMailSmime die Lizenzschlüssel angegeben werden. Diese Angaben dürfen nicht verändert werden.

11.6.3 Implementierung der Konfiguration

Die Umsetzung der spezifischen Konfiguration für das *Smtp Mailer* System wurde mit einer benutzerdefinierten *configurationSection* umgesetzt. Die Basis bildet ein ConfigurationHandler, welcher von *IConfigurationSectionHandler* abgeleitet ist und für die Serialisierung der Daten zuständig ist. Die Konfiguration selbst wird in Klassen abgebildet, wobei die Properties der Klassen in die XML-Datei gemappt werden. Die Klasse selbst wird mit dem *XmlRoot*-Attribut gekennzeichnet. Properties die einen alleinigen Wert repräsentieren werden mit dem *XmlAttribute*-Attribut gekennzeichnet. Jene die ein Xml-Element darstellen und weitere Attribute haben werden mit dem *XmlElement*-Attribut gekennzeichnet. Listen von Einträgen können mit dem *XmlArray*- und *XmlArrayItem*-Attribut realisiert werden.

11.7 Grössen und Leistungen

Punkt	Beschränkung	Grund
Anzahl zu verarbeitende E-Mail Nachrichten	Pro Stunde müssen 300‘000 E-Mail Nachrichten verarbeitet werden können	Dies ist eine Qualitätsanforderung gemäss der Aufgabenstellung
Anzahl zu verarbeitender Bounce Nachrichten	Pro Stunde müssen 30‘000 Bounce Nachrichten verarbeitet werden können	Dies ist eine Qualitätsanforderung gemäss der Aufgabenstellung
Anzahl *Smtp Mailer* Instanzen	Unbeschränkt	Es müssen mehrere *Smtp Mailer* Instanzen parallel betrieben werden können, welche auf dieselbe Datenbank zugreifen. Anforderung aus der Aufgabenstellung
Unterstützte Sprachen	Fehlermeldungen gegenüber dem Benutzer nur auf Deutsch, Exceptions nur auf Englisch	Bestimmung durch comparis.ch AG

Tabelle 61 - Grössen und Leistungen Smtp Mailer

11.8 Webapplikation

Die Webapplikation wurde vollständig in Silverlight Version 2 implementiert. Nachfolgend werden die wichtigsten der verwendeten Silverlight Komponenten erklärt. Danach folgen ein paar Screenshots der fertiggestellten Webapplikationen.

11.8.1 Verwendete Komponenten

In den folgenden Abschnitten werden die wichtigsten Silverlight Komponenten beschrieben, welche häufig in den realisierten Webapplikationen eingesetzt wurden:

11.8.1.1 Datagrid

Das Datagrid ordnet die Elemente tabellarisch an. Die Komponente wurde auf jeder Seite der verschiedenen Webapplikationen genutzt, um Elemente aufzulisten. Ein Nachteil der verwendeten Datagrid Komponente war es, dass kein automatisches Paging angeboten wird. So musste eine eigene Paging Komponente implementiert werden.

11.8.1.2 Stackpanel

Das Stackpanel erlaubt es, Elemente in einer Reihe horizontal oder vertikal auf der Seite anzuordnen. Weil dies häufig gebraucht wird ist das eine sehr nützliche Komponente.

11.8.1.3 Datepicker

Zur Auswahl eines Datums bietet sich eine Datepicker Komponente an. Diese erlaubt es dem Benutzer, auf komfortable Weise ein Datum zu wählen. Ein weiterer Vorteil von der Benutzung der Komponente ergibt sich durch die automatische Inputvalidierung des Datums.

11.8.1.4 Suchbar

Die Suchbar befindet sich am rechten Bildschirmrand einer Seite und kann ausgefahren werden. Die Suchbar wurde selber mittels einer Animation implementiert. Dadurch dass sich die Suchbar aus- und auch wieder einfahren lässt, bleibt mehr Platz auf der Seite für den sonstigen Inhalt.

11.8.2 Screenshots

Abbildung 93 - Screenshot CustomerService Webapplikation Nachrichtendetails

Abbildung 94 - Screenshot Customer Service Webapplikation Blacklist verwalten

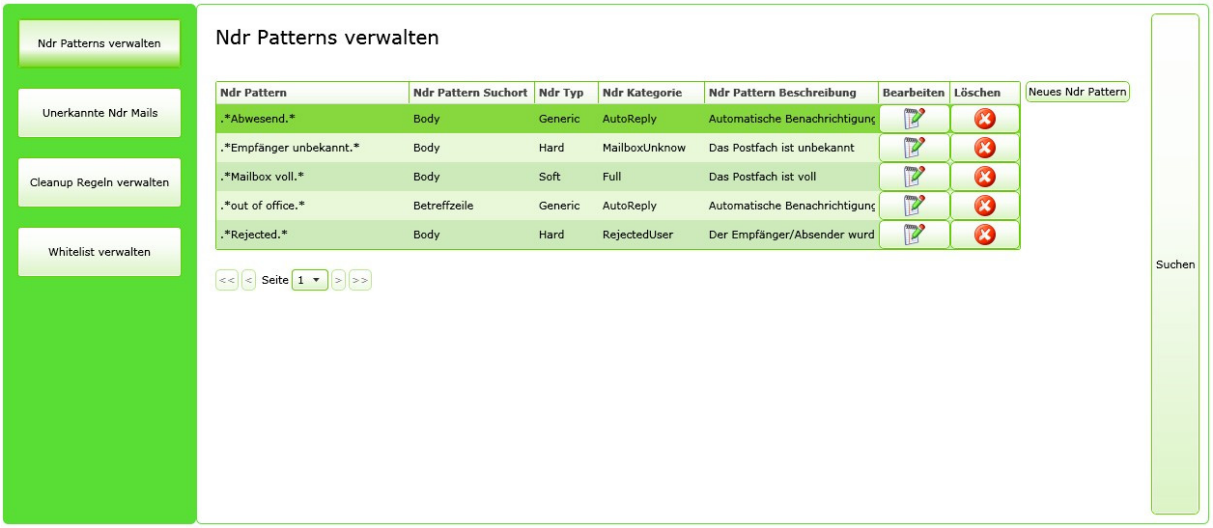


Abbildung 95 - Screenshot Configuration Webapplikation Ndr Patterns verwalten

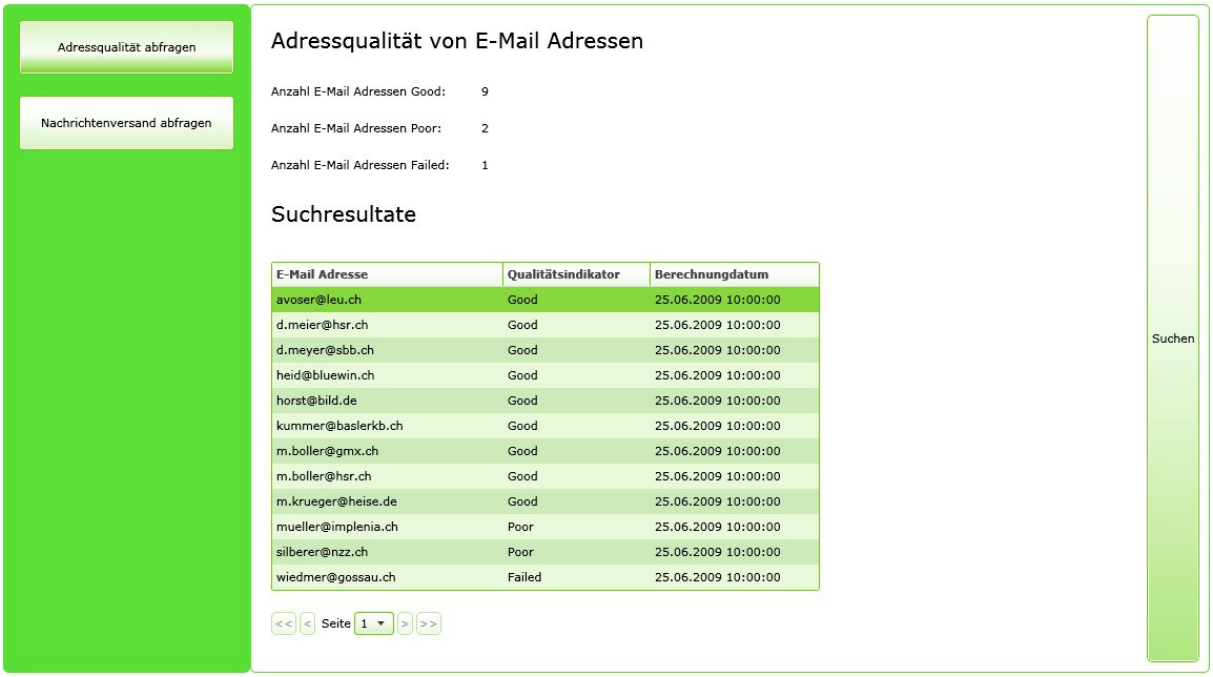


Abbildung 96 - Screenshot Reporting Webapplikation Adressqualitätsmanagement

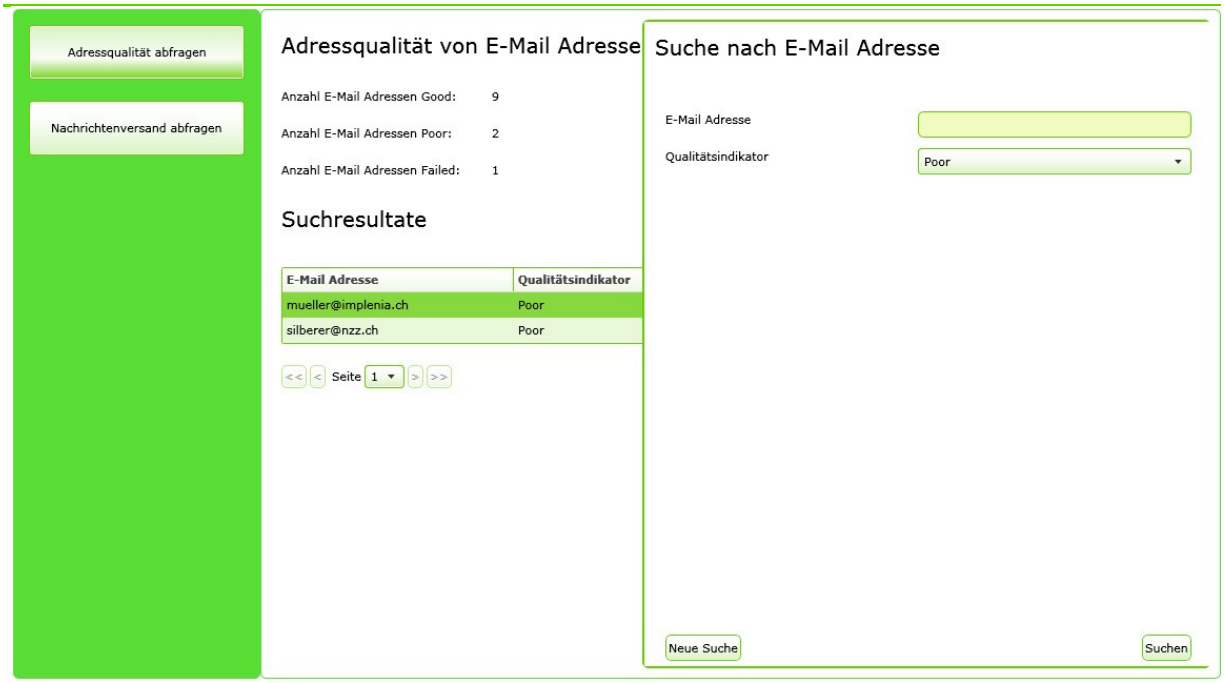


Abbildung 97 - Screenshot Reporting Webapplikation Adressqualitätsmanagement mit ausgeklappterSuchbar

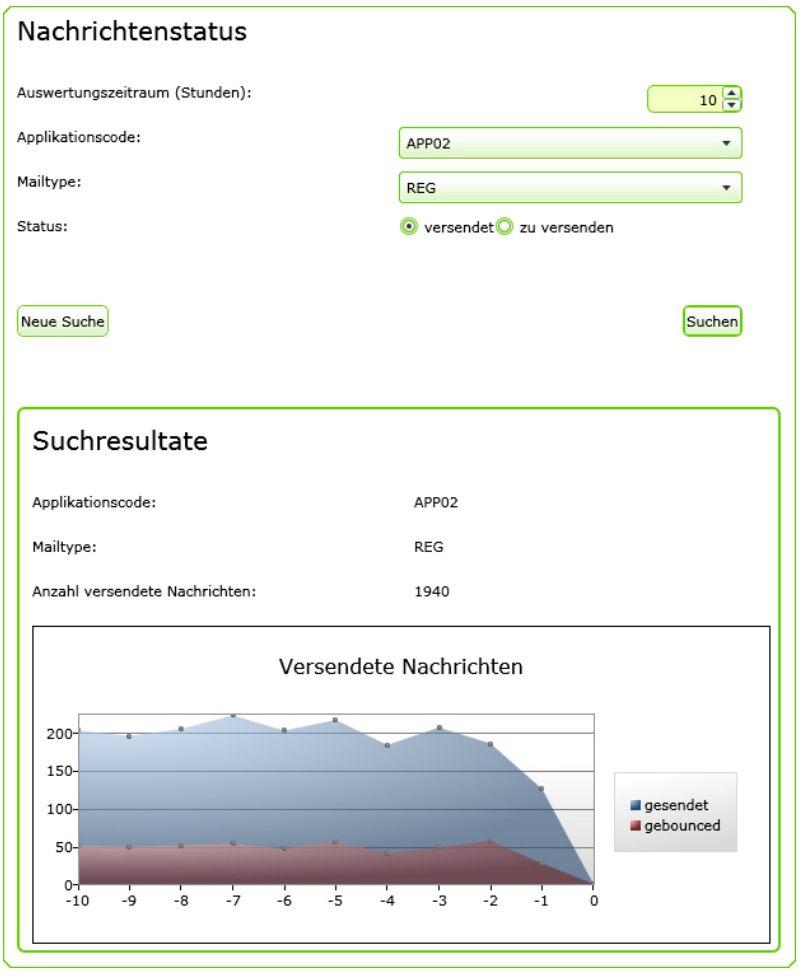


Abbildung 98 - Screenshot Reporting Webapplikation Auswertung über versendete Nachrichten

12 Test

12.1 System-Tests

12.1.1 Durchführung System-Tests KW 16

12.1.1.1 Voraussetzungen

Es sind zu versendende Nachrichten in der Datenbank vorhanden. Es ist eine App.config vorhanden, welche die Applikationscodes und Mailtypen auf Prioritäten abbildet. Beide Prioritäten (High und Low) nutzen denselben SMTP Server, um die Nachricht zu versenden. Der angegebene SMTP Server ist vorhanden. Dabei ist der SMTP Server so konfiguriert, dass er die Empfängerkonten selber hosted.

12.1.1.2 Vorbereitungen

Generieren von zu versendenden E-Mail Nachrichten in der Datenbank. Anpassung der Konfiguration (App.config).

Folgende Testkonfiguration (App.config) wurde verwendet:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <configSections>
    <section name="smtpmailer"
type="Comparis.Framework.SmtpMailer.Common.Configuration.ConfigurationHandler,
Comparis.Framework.SmtpMailer.Common.Configuration" />
  </configSections>

  <connectionStrings>
    <add name="Comparis.Framework.SmtpMailer.Data.Properties.Settings.CONN SMTP MAILER DB"
      connectionString="Data Source=localhost\sql2008;Initial Catalog=SMTP_MAILER_DB;Integrated
Security=True" />
  </connectionStrings>

  <smtpmailer type="Comparis.Framework.SmtpMailer.Common.Configuration.SmtpMailerConfig,
Comparis.Framework.SmtpMailer.Common.Configuration">
    <smtpservers>
      <smtpserver priority="Low" deliverytype="Smtp" host="localhost" port="25" useauth="0" />
      <smtpserver priority="High" deliverytype="Smtp" host="localhost" port="25" useauth="0" />
    </smtpservers>

    <priorityrules>
      <rule applicationcode="KK" mailtype=" NEWSLETTER" priority="Low" />
      <rule applicationcode="*" mailtype="NEWSLETTER" priority="High" />
      <rule applicationcode="*" mailtype="*" priority="Low" />
    </priorityrules>

    <whitelist active="0" />

    <threadpool min="2" max="500" />

    <idletime min="0" max="0" increasingfactor="0" />

    <bouncehandler returnpath="bounce-{0}@ba.danflash.com" />

    <licenses>
      <license component="EasyMailSmtp" licensekey=" ***" />
      <license component="EasyMailSmime" licensekey=" ***" />
    </licenses>
  </smtpmailer>
</configuration>
```

12.1.1.3 Testfälle

UC 1: Nachricht versenden	
Testschritt	**Resultat**
Versenden einer Nachricht an einen Empfänger mit Subject, Plaintext, ohne Attachements, ohne Signatur	Erfolgreich
Versenden einer Nachricht an mehrere Empfänger (TO) mit Subject, Plaintext, ohne Attachements, ohne Signatur	Erfolgreich
Versenden einer Nachricht an mehrere Empfänger (TO, CC, BCC) mit Subject, Plaintext, ohne Attachements, ohne Signatur	Erfolgreich

Versenden einer Nachricht an einen Empfänger mit Subject, HTML Content, mit einem Attachement, ohne Signatur	Erfolgreich
Versenden einer Nachricht an einen Empfänger mit Subject, HTML Content, mit mehreren Attachements, mit Signatur	Erfolgreich
Versenden einer Nachricht an einen nicht existenten Empfänger (Benutzername stimmt nicht, Domain jedoch schon) mit Subject, Plaintext, ohne Attachements, ohne Signatur	Erfolgreich. Der Mailserver meldet sofort, dass der Empfänger nicht existiert.
Erneutes Versenden einer bereits erstellten Nachricht mittels einer E-Mail Kennung. Diese Anforderung wurde als Nice-to-Have klassiert und konnte nicht beachtet werden.	-
Versenden einer Nachricht an einen Empfänger der auf der Blacklist steht mit Subject, Plaintext, ohne Attachements, ohne Signatur	Erfolgreich. Die Nachricht wurde auf Dispatched=1 gesetzt, ein DispatchState mit Blacklisted wurde erstellt. Die E-Mail Nachricht wird nicht zugestellt.
Versenden einer Nachricht an einen Empfänger der nicht auf der Whitelist steht obwohl sie aktiv ist mit Subject, Plaintext, ohne Attachements, ohne Signatur	Erfolgreich. Die E-Mail Nachricht wird nicht zugestellt.
Versenden einer Nachricht an einen Empfänger der auf der Whitelist steht welche aktiv ist mit Subject, Plaintext, ohne Attachements, ohne Signatur	Erfolgreich. Die E-Mail Nachricht wird zugestellt.
Versenden einer Nachricht an einen Empfänger der auf der Whitelist steht welche aktiv ist aber gleichzeitig auf der Blacklist steht mit Subject, Plaintext, ohne Attachements, ohne Signatur	Erfolgreich. Die E-Mail Nachricht wird nicht zugestellt.

Tabelle 62 - System-Test UC 1 KW 16

UC 2: Versandstatus abfragen	
Testschritt	**Resultat**
Abfrage des Versandstatus mit einer gültigen E-Mail Kennung	Erfolgreich
Abfrage des Versandstatus mit einer ungültigen E-Mail Kennung	Erfolgreich. Es wird eine ArgumentException geworfen

Tabelle 63 - System-Test UC 2 KW 16

Nicht funktionale Anforderungen	
Testschritt	**Resultat**
Versenden von 300‘000 Nachrichten. Überprüfung, ob das Ablegen der Daten in der Datenbank und das Versenden der 300‘000 Nachrichten in einer Stunde möglich ist.	Erfolgreich. Die zu versendenden Nachrichten werden dabei nicht per SMTP dem Mail Server übergeben sondern in ein Verzeichnis abgelegt.

Tabelle 64 - System-Test nicht funktionale Anforderungen KW 16

12.1.2 Durchführung von Systemtests KW 18

12.1.2.1 Voraussetzungen

Es sind zu versendende Nachrichten und bereits versendete Nachrichten in der Datenbank vorhanden. Es ist eine App.config vorhanden, welche die Applikationscodes und Mailtypen auf Prioritäten abbildet. Beide Prioritäten (High und Low) nutzen denselben SMTP Server, um die Nachricht zu versenden. Der angegebene SMTP Server ist vorhanden. Dabei ist der SMTP Server so konfiguriert, dass er die Empfängerkonten selber hosted.

Es sind NdrMails in der Datenbank eingetragen, welche noch nicht verarbeitet wurden.

12.1.2.2 Vorbereitungen

Generieren von zu versendenden und bereits versendeten E-Mail Nachrichten in der Datenbank. Generieren von NdrMails in der Datenbank. Anpassung der Konfiguration (App.config).

Folgende Testkonfiguration (App.config) wurde verwendet:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <configSections>
    <section name="smtpmailer"
type="Comparis.Framework.SmtpMailer.Common.Configuration.ConfigurationHandler,
Comparis.Framework.SmtpMailer.Common.Configuration" />
  </configSections>

  <connectionStrings>
    <add name="Comparis.Framework.SmtpMailer.Data.Properties.Settings.CONN SMTP MAILER DB"
      connectionString="Data Source=localhost\sql2008;Initial Catalog=SMTP MAILER DB;Integrated
Security=True" />
  </connectionStrings>

  <smtpmailer type="Comparis.Framework.SmtpMailer.Common.Configuration.SmtpMailerConfig,
Comparis.Framework.SmtpMailer.Common.Configuration">
    <smtpservers>
      <smtpserver priority="Low" deliverytype="Smtp" host="localhost" port="25" useauth="0" />
      <smtpserver priority="High" deliverytype="Smtp" host="localhost" port="25" useauth="0" />
    </smtpservers>

    <priorityrules>
      <rule applicationcode="KK" mailtype=" NEWSLETTER" priority="Low" />
      <rule applicationcode="*" mailtype="NEWSLETTER" priority="High" />
      <rule applicationcode="*" mailtype="*" priority="Low" />
    </priorityrules>

    <whitelist active="0" />

    <threadpool min="2" max="500" />

    <idletime min="0" max="0" increasingfactor="0" />

     <bouncehandler returnpath="bounce-{0}@ba.danflash.com" pop3host="localhost" pop3port="110"
                pop3username="bounce@ba.danflash.com" pop3password="_Zweifel_" />
    <licenses>
      <license component="EasyMailSmtp" licensekey="***" />
      <license component="EasyMailSmime" licensekey="***" />
    </licenses>
  </smtpmailer>
</configuration>
```

12.1.2.3 Testfälle

UC 4: Filterliste CRUD	
Testschritt	**Resultat**
Erstellen eines Filterlisteneintrags	Erfolgreich
Abfragen eines Filterlisteneintrags mittels der FilterAddressId	Erfolgreich
Abfragen aller Filterlisteneinträge	Erfolgreich
Aktualisieren eines Filterlisteneintrags	Erfolgreich
Löschen eines Filterlisteneintrags	Erfolgreich
Erstellen eines ungültigen Filtereintrags (Falsches AddressPattern, leeres AddressPattern, falscher Index bei FilterType)	Erfolgreich. Jedoch wird nur erkannt, ob ein AddressPattern gegeben ist. Ein falsches AddressPattern oder ein falscher FilterType wird nicht erkannt.
Aktualisieren eines Filtereintrags mit ungültigen Werten	Erfolgreich. Jedoch wird nur erkannt, ob ein AddressPattern gegeben ist. Ein falsches AddressPattern oder ein falscher FilterType wird nicht erkannt.

Tabelle 65 - System-Test UC 4 KW 18

UC 5: NDR Pattern CRUD	
Testschritt	**Resultat**
Erstellen eines NdrPatterns	Erfolgreich
Abfragen eines NdrPattern mittels NdrPatternId	Erfolgreich
Abfragen aller NdrPatterns	Erfolgreich
Aktualisieren eines NdrPatterns	Erfolgreich
Löschen eines NdrPatterns	Erfolgreich
Erstellen eines ungültigen NDRPatterns (Ungültige NdrDescription, falscher Index bei SearchLocation)	Erfolgreich. Jedoch wird nur erkannt, ob ein Pattern gegeben ist. Eine falsche SearchLocation wird nicht erkannt.
Aktualisieren eines NDRPatterns mit ungültigen Werten	Erfolgreich. Jedoch wird nur erkannt, ob ein Pattern gegeben ist. Eine falsche SearchLocation wird nicht erkannt.

Tabelle 66 - System-Test UC 5 KW 18

UC 9: Datenbankbereinigung CRUD	
Testschritt	**Resultat**
Erstellen einer Datenbankbereinigung Regel	Erfolgreich
Abfragen aller Datenbankbereinigung Regeln	Erfolgreich
Aktualisieren einer Datenbankbereinigung Regel	Erfolgreich
Löschen einer Datenbankbereinigung Regel	Erfolgreich
Erstellen einer ungültigen Datenbankbereinigung Regel	Erfolgreich. Jedoch wird nicht erkannt, dass die StorageDuration negativ ist.
Aktualisieren einer Datenbankbereinigung Regel mit ungültigen Werten	Erfolgreich. Jedoch wird nicht erkannt, dass die StorageDuration negativ ist.

Tabelle 67 - System-Test UC 9 KW 18

UC 10: NDR Nachricht verarbeiten	
Testschritt	**Resultat**
Bounce Nachrichten, welche sich in einem POP3 Postfach befinden werden dort abgeholt und erfolgreich in die Tabelle NdrMail in der Datenbank gespeichert (NdrRetrieveController)	Erfolgreich
Die NdrMails in der Datenbank werden gelesen und analysiert. Der Grund für das NdrMail d.h. der Bounce Typ und Kategorie wird erkannt und es wird ein DispatchingState mit dem Status Bounced für den entsprechenden Recipient in der Datenbank erfasst. Die NdrDescription zum erkannten Bounce Typ und Kategorie wird dem DispatchingState angehängt. Das NdrMail, welches erkannt wurde, wird in der Datenbank gelöscht Der Recipient wird auf SuccessfulSent = false gesetzt.	Erfolgreich
Falls der Grund für ein NdrMail von der Bounce Komponente nicht erkannt wird, bleibt das NdrMail in der Datenbank. Der Status des NdrMails wird auf Unrecognized gesetzt.	Erfolgreich
Anhand der DispatchingStates für einen Recipient wird ein Qualitätsindikator berechnet.	Erfolgreich

Tabelle 68 - System-Test UC 10 KW 18

Nicht funktionale Anforderungen	
Testschritt	**Resultat**
Verarbeiten von 30‘000 NDR Nachrichten. Überprüfen, ob die Verarbeitung in einer Stunde möglich ist.	Erfolgreich
Ausführen von mehreren Instanzen des *Smtp Mailers*. Die Sending Engine wird parallel auf einem Rechner laufen gelassen.	Erfolgreich.

Tabelle 69 - System-Test nicht funktionale Anforderungen KW 18

12.1.3 Durchführung von Systemtests KW 22

12.1.3.1 Voraussetzungen

Es sind zu versendende Nachrichten und bereits versendete Nachrichten in der Datenbank vorhanden. Dazu gehören auch Daten zu Empfänger, Nachrichteninhalte, Anhänge etc. Diese Daten werden als Beispieldaten für die Anzeige in der Webapplikation benötigt.

12.1.3.2 Vorbereitungen

Generieren von zu versendenden und bereits versendeten E-Mail Nachrichten in der Datenbank.

Folgende Testkonfiguration (App.config) wurde für den Test der Datenbankbereinigung (UC8) verwendet:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <configSections>
  </configSections>

  <connectionStrings>
    <add name="Comparis.Framework.SmtpMailer.Data.Properties.Settings.CONN_SMTP_MAILER_DB"
      connectionString="Data Source=localhost\sql2008;Initial Catalog=SMTP_MAILER_DB;Integrated
Security=True" />
  </connectionStrings>
</configuration>
```

12.1.3.3 Testfälle

UC 3: Nachricht abfragen	
Testschritt	**Resultat**
Abfrage einer Nachricht mit einer gültigen E-Mail Kennung	Erfolgreich (getestet über Webinterface)
Abfrage einer Nachricht mit einer ungültigen E-Mail Kennung	Erfolgreich. Es wird eine ArgumentException geworfen

Tabelle 70 - System-Test UC 3 KW 22

UC 6: Qualitätsindikator abfragen	
Testschritt	**Resultat**
Abfragen des Qualitätsindikators für eine E-Mail Adresse	Erfolgreich (getestet über Webinterface)

Tabelle 71 - System-Test UC 6 KW 22

UC 7: Report abfragen	
Testschritt	**Resultat**
Abfragen von Qualitätsindikator der E-Mail Adressen, gefiltert nach E-Mail Adresse und Qualitätsindikator. Darstellung als Übersicht	Erfolgreich (getestet über Webinterface)
Abfragen von Qualitätsindikator der E-Mail Adressen, gefiltert nach E-Mail Adresse und Qualitätsindikator. Darstellung als Liste	Erfolgreich (getestet über Webinterface)
Abfragen von Qualitätsindikator der E-Mail Adressen, gefiltert nach E-Mail Adresse und Qualitätsindikator. Darstellung als Detailanzeige	Erfolgreich (getestet über Webinterface)
Anzeigen von Report, welcher Auskunft darüber gibt, wie viele E-Mails über einen Zeitraum verschickt wurden und in welchem Status diese sind. Einschränkung des Resultates auf gewisse Applikationscodes, Mailtypen	Erfolgreich (getestet über Webinterface)

sowie Status.	
Anzeigen von Report, welcher Auskunft darüber gibt, wie viele E-Mails in den nächsten n Stunden verschickt werden sollen. Einschränkung des Resultates auf gewissen Applikationscodes und Mailtypen.	Erfolgreich (getestet über Webinterface)
Abfragen von verschiedenen Reports mit Suchkriterien, dass eine leere Resultatmenge resultiert.	Erfolgreich (getestet über Webinterface)

Tabelle 72 - System-Test UC 7 KW 22

UC 8: Datenbank bereinigen	
Testschritt	**Resultat**
Durchführen einer Datenbankbereinigung	Erfolgreich

Tabelle 73 - System-Test UC 8 KW 22

12.2 Langzeittest KW 23

Die Sending Engine und die Bounce Engine wurden mittels eines Langzeittests von 2 Tagen auf die Stabilität geprüft.

12.2.1 Voraussetzungen

Es sind zu versendende Nachrichten in der Datenbank vorhanden. Konkret wurden 100‘000 zu versendende Nachrichten generiert, welche über die 2 Tage verschickt werden müssen. Die Nachrichten sind zufällig um Attachments erweitert und werden nach dem Zufallsprinzip signiert verschickt. Es ist eine App.config vorhanden, welche die Applikationscodes und Mailtypen auf Prioritäten High und Low abbildet. Beide Prioritäten nutzen denselben SMTP Server, um die Nachricht zu versenden. Der angegebene SMTP Server ist vorhanden. Dabei ist der SMTP Server so konfiguriert, dass er die Empfängerkonten selber hosted. Ein Empfängerkonto (auto@ba.danflash.com) wurde so konfiguriert, dass beim Empfang einer Nachricht eine Out-of-Office Nachricht erstellt wird. So erhalten wir ‚Out-of-Office‘ Bounce Nachrichten. Ein weiteres Konto (full@ba.danflash.com) besitzt eine maximale Mailboxgrösse von 1 MB. So entstehen ‚Mailbox full‘ Bounce-Nachrichten. Ein weiteres Konto wurde für die Bounce Nachrichten eingerichtet. Darin werden die Bounce Nachrichten empfangen und die BounceEngine kann die Bounce Nachrichten da abholen.

12.2.2 Vorbereitungen

Generieren von zu versendenden E-Mail Nachrichten in der Datenbank. Generieren von NdrMails in der Datenbank. Anpassung der Konfiguration (App.config). Einrichtung und Konfiguration des SMTP Servers.

Folgende Testkonfiguration (App.config) wurde verwendet:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <configSections>
    <section name="smtpmailer"
type="Comparis.Framework.SmtpMailer.Common.Configuration.ConfigurationHandler,
Comparis.Framework.SmtpMailer.Common.Configuration" />
  </configSections>

  <connectionStrings>
    <add name="Comparis.Framework.SmtpMailer.Data.Properties.Settings.CONN_SMTP_MAILER_DB"
      connectionString="Data Source=localhost\sql2008;Initial Catalog=SMTP_MAILER_DB;Integrated
Security=True" />
  </connectionStrings>

  <smtpmailer type="Comparis.Framework.SmtpMailer.Common.Configuration.SmtpMailerConfig,
Comparis.Framework.SmtpMailer.Common.Configuration">
    <smtpservers>
      <smtpserver priority="Low" deliverytype="Smtp" host="localhost" port="25" useauth="0" />
      <smtpserver priority="High" deliverytype="Smtp" host="localhost" port="25" useauth="0" />
    </smtpservers>

    <priorityrules>
      <rule applicationcode="KK" mailtype=" NEWSLETTER" priority="Low" />
      <rule applicationcode="*" mailtype="NEWSLETTER" priority="High" />
      <rule applicationcode="*" mailtype="*" priority="Low" />
    </priorityrules>
```

```
    <whitelist active="0" />

    <threadpool min="2" max="500" />

    <idletime min="0" max="0" increasingfactor="0" />

     <bouncehandler returnpath="bounce-{0}@ba.danflash.com" pop3host="localhost" pop3port="110"
                pop3username="bounce@ba.danflash.com" pop3password="_Zweifel_" />
    <licenses>
      <license component="EasyMailSmtp" licensekey="***" />
      <license component="EasyMailSmime" licensekey="***" />
    </licenses>
  </smtpmailer>
</configuration>
```

12.2.3 Testfälle

Test	Resultat
Die Sending Engine wird zwei Tage laufen gelassen, um die zu versendenden Nachrichten abzuarbeiten. Dabei werden Bounce Nachrichten zurückkommen, weil wie oben beschrieben zwei Empfängerkonten speziell konfiguriert wurden (Autoreply, Full Mailbox).	OK Es entstehen jedoch ein paar wenige AccessViolationExceptions (Attempted to read or write protected memory. This is often an indication that other memory is corrupt). Diese Exceptions wurden von der EasyMail Komponente von QuikSoft geworfen. Diese Exceptions hatten jedoch keinen Einfluss auf die Stabilität der Sending Engine.
Die Bounce Engine wird parallel zur Sending Engine zwei Tage laufen gelassen. Sie holt vor zu die Bounce Nachrichten per POP3 ab und legt sie in die Datenbank ab. Die NdrMails in der Datenbank werden verarbeitet und klassifiziert.	OK

Tabelle 74 - Langzeittests KW 23

12.3 Manueller GUI Test

In der Kalenderwoche 22 wurde ein manueller GUI Test durchgeführt. Dabei wurden sämtliche Webapplikationen und deren Funktionalitäten durchgetestet. Folgende Probleme wurden dabei festgestellt und sofort behoben:

- In der Reporting Applikation stimmt die Skala in der x-Achse nicht
- Die Suchbar im NdrPatterns verwalten Screen lässt sich nicht aus- und einfahren. Wahrscheinlich besteht noch ein Problem im XAML Code

12.4 Usability-Tests

Am 06.05.2009 wurde bei der Firma comparis.ch AG mit einem realen Endbenutzer ein Paper Prototype Test durchgeführt. Dieser Paper Prototype deckte dabei die Kundendienst Webapplikation ab. Es konnten diverse Probleme und Unklarheiten beim Bedienen des Paper Prototypes festgestellt werden.

Die Kundendienst Webapplikation bietet einem Kundendienstmitarbeiter zwei Funktionalitäten:

- Suchen nach versendeten und noch zu versendenden Nachrichten und Anzeigen von Nachrichtendetails
- Verwaltung der Blacklist

Nachfolgend in Stichworten die Probleme und Unklarheiten, welche beim Paper Prototype Test festgestellt wurden.

12.4.1 Szenario Nachricht abfragen

- Beschriftung ‚Empfängername' in der Suchmaske ist nicht klar (besser getrennte Vorname und Name Suchfelder mit entsprechenden Bezeichnungen)
- Was ist, wenn in der Tabelle nur ein Empfänger für die E-Mail Nachricht dargestellt wird (weil nach ihm gesucht wird), wird dann in der Detailansicht der Nachricht auch nur dieser Empfänger dargestellt?
- Als erstes Feld in der Suche sollte die E-Mail Adresse stehen und evtl. die Benutzer-Id
- Der Applikationscode und die Referenz in der Detailansicht sind für den Kundendienstmitarbeiter uninteressant
- In der Suchmaske sind Applikationscode und Mailtyp uninteressant. Was es sicher braucht ist die E-Mail Adresse, Benutzer-Id und evtl. Vorname/Name
- Der Button ‚Zurücksetzen' soll in ‚Neue Suche' umbenannt werden sowie die Position mit dem Button ‚Suchen' tauschen
- Spalten ‚Geplanter Versand' und ‚Versanddatum' umdrehen
- Es soll bereits in der Tabelle ersichtlich sein, ob die E-Mail Nachricht ohne Probleme versendet werden konnte (Hacken) oder ob ein Problem auftrat (Ausrufezeichen)
- Umbenennung von ‚Nicht erfolgreiche Empfänger' in ‚Versandprobleme' in der Detailansicht

12.4.2 Szenario Blacklist konfigurieren

- Diese Maske kann auf das Minimum vereinfacht werden
  - Suche nach einem Blacklist Eintrag. Falls nicht vorhanden so wird ein Button angeboten ‚In Blacklist übernehmen' resp. ‚Aus Blacklist löschen'
  - Die Suche sollte auf der E-Mail Datenbank stattfinden, damit eine E-Mail Adresse gefunden werde kann (für die Übernahme in die Blacklist)

12.5 Unit-Tests

Die Unit-Tests werden mit NUnit erstellt. Zu jedem Projekt wird ein paralleles Projekt in der Visual Studio Solution erstellt, welches die Unit-Tests enthält.

12.5.1 Namensgebung für Unit-Test Projekte

Pro Projekt (Assembly), für welches Unit-Tests geschrieben werden, wird ein eigenständiges Projekt angelegt. Dieses besitzt denselben Projektname wie das zu testende Projekt, endet aber auf Unit-Tests. Z.B. werden für das Projekt Comparis.Framework.SmtpMailer.Core die Unit-Tests in einem separaten Projekt namens Comparis.Framework.SmtpMailer.Core.Unit-Tests geführt.

12.5.2 Namespace Comparis.Framework.SmtpMailer.Common.UnitTests

12.5.2.1 ValidationTest

Testmethode	Beschreibung
ValidateValidGuid	Prüft, ob die Validierung einer Guid korrekt arbeitet und gültige Guids als gültig erklärt werden
ValidateInvalidGuid	Prüft, ob die Validierung einer Guid korrekt arbeitet und ungültige Guids als ungültig erklärt werden
ValidateValidEmail	Prüft, ob die Validierung einer E-Mail Adresse korrekt arbeitet und gültige E-Mail Adressen als gültig erklärt werden
ValidateInvalidEmail	Prüft, ob die Validierung einer E-Mail Adresse korrekt arbeitet und ungültige E-Mail Adressen als ungültig erklärt werden
ValidateValidMessage	Prüft, ob die Validierung einer Message korrekt arbeitet und gültige Messages als gültig erklärt werden
ValidateInvalidMessage	Prüft, ob die Validierung einer Message korrekt arbeitet und ungültige Messages als ungültig erklärt werden
ValidateValidRecipient	Prüft, ob die Validierung eines Recipient korrekt arbeitet und gültige Recipients als gültig erklärt werden
ValidateInvalidRecipient	Prüft, ob die Validierung eines Recipient korrekt arbeitet und ungültige Recipients als ungültig erklärt werden
ValidateValidMessageBody	Prüft, ob die Validierung eines MessageBody korrekt arbeitet und gültige MessageBodys als gültig erklärt werden
ValidateInvalidMessageBody	Prüft, ob die Validierung eines MessageBody korrekt arbeitet und ungültige MessageBodys als ungültig erklärt werden
ValidateValidAttachment	Prüft, ob die Validierung einer Attachments korrekt arbeitet und gültige Attachments als gültig erklärt werden
ValidateInvalidAttachment	Prüft, ob die Validierung eines Attachments korrekt arbeitet und ungültige Attachments als ungültig erklärt werden

Tabelle 75 - Testmethoden ValidationTest

12.5.2.2 ConversionTest

Testmethode	Beschreibung
ConversionBounceAddressToGuid	Prüft, ob gültige Bounce Adressen in eine korrekte guid umgewandelt werden und ob ungültige Bounce Adressen erkannt werden (Exception wird geworfen)

Tabelle 76 - Testmethoden ConversionTest

12.5.3 Namespace Comparis.Framework.SmtpMailer.Core.UnitTests

12.5.3.1 FilterListHandlerTest

Testmethode	Beschreibung
IsWhitelisted	Prüft die Logik zur Entscheidung, ob eine E-Mail Adresse oder ein E-Mail Adress-Pattern (Adressen mit Wildcards) auf der Whitelist steht
IsBlacklisted	Prüft die Logik zur Entscheidung, ob eine E-Mail Adresse oder ein E-Mail Adress-Pattern (Adressen mit Wildcards) auf der Blacklist steht

Tabelle 77 - Testmethoden FilterListHandlerTest

12.5.3.2 AddressQualityTest

Testmethode	Beschreibung
AddressQualitySummarizeStrategy	Prüft die Summarize-Strategie auf das korrektes Berechnen des Qualitätsindikators gemäss der implementierten Strategie

Tabelle 78 - Testmethoden AddressQualityTest

12.5.3.3 CleanupControllerTest

Testmethode	Beschreibung
CleanupMessages	Prüft, ob die korrekten Anhänge und Nachrichteninhalte gelöscht werden

Tabelle 79 - Testmethoden CleanupControllerTest

12.5.3.4 NdrAnalyzeControllerTest

Testmethode	Beschreibung
ProcessAnalyze	Prüft, ob die Bounce Nachrichten korrekt erkannt und der Status aktualisiert wird

Tabelle 80 - Testmethoden NdrAnalyzeControllerTest

12.5.4 Namespace Comparis.Framework.SmtpMailer.Data.UnitTests

12.5.4.1 MessageDALCTest

Testmethode	Beschreibung
CreateMessage	Prüft, ob eine Message mit Attachments, Recipients und MessageBodies erstellt und in der Datenbank korrekt gespeichert werden kann.
ReadRecipientsWithDispatchingState	Prüft, ob alle in der Datenbank vorhandenen Recipients zusammen mit ihren jeweilig assoziierten DispatchingStates korrekt gelesen werden können.
CreateDispatchingState	Prüft, ob ein DispatchingState erstellt und in der Datenbank korrekt gespeichert werden kann.
CreateDisptachingStates	Prüft, ob mehrere DispatchingStates auf einmal erstellt und in der Datenbank korrekt gespeichert werden können.
ReadNextMessageToSend	Prüft, ob unter einer gegebenen Konfiguration (welche definiert, welche Applikationscode und Mailtyp Tupel welche Priorität besitzen) die nächste zu versendende Nachricht mit Priorität High resp. Low korrekt aus der Datenbank gelesen wird.
UpdateMessageDispatched	Prüft, ob eine Nachricht in der Datenbank auf Dispatched = true gesetzt werden kann.
UpdateRecipient	Prüft, ob ein Recipient geupdated werden kann und die Änderungen korrekt in die Datenbank übernommen werden.

DeleteCleanupMessages	Prüft ob anhand von in der Datenbank erfassten CleanupRules die korrekten Messages gewählt werden, welche bereinigt werden (Attachments, MessageBodies). Prüft, ob die Attachments und MessageBodies gelöscht werden.
ReadRecipient	Prüft, ob ein Recipient mittels der guid der Message, welche an den zu lesenden Recipient geschickt wurde und der E-Mail Adresse korrekt gelesen werden kann.
ReadRecipientsWithDispatching StateSentBounced	Prüft ob alle Empfänger mit den Versandstatus gelesen werden die entweder den Status Bounced oder Sent haben.
ReadAddresses	Prüft, ob alle AddressQualities korrekt von der Datenbank gelesen werden können
ReadAddressesWithQualityIndicator	Prüft, ob alle AddressQualities mit einem gewissen QualityIndicator korrekt von der Datenbank gelesen werden können
ReadAddressesWithAddressPattern	Prüft, ob alles AddressQualities welche auf ein bestimmtes AddressPattern passen, korrekt von der Datenbank gelesen werden können
ReadMessageWithGuid	Prüft, ob eine Message anhand der Guid korrekt von der Datenbank gelesen werden kann
ReadMessagesWithSearchParameters	Prüft, ob Messages korrekt anhand von Suchparametern (Vorname und Name des Recipients, E-Mail Adresse des Recipients, Betreff, Zeitspanne von bis) gelesen werden
ReadMessagesCount	Prüft, ob die Anzahl der zu lesenden Messages korrekt anhand von Suchparametern (Vorname und Name des Recipients, E-Mail Adresse des Recipients, Betreff, Zeitspanne von bis) gelesen wird
ReadMessagesWithCount	Prüft, ob bei Angabe einer Anzahl zu lesender Messages die korrekte Anzahl an Messages von der Datenbank gelesen wird
ReadAddressesCount	Prüft, ob die Anzahl der zu lesenden AddressQualities korrekt anhand von Suchparametern (E-Mail Adresse, Qualitätsindikator) gelesen wird
ReadMessages	Prüft, ob alle Nachrichten korrekt von der Datenbank gelesen werden können
ReadSentMessages	Prüft, ob alle gesendeten Nachrichten anhand von Suchparametern (Zeitspanne von bis, Applikationscode, Mailtyp) korrekt von der Datenbank gelesen werden können
ReadNumberOfMessagesToSendHours	Prüft, ob die Anzahl der zu sendenden Nachrichten anhand von der angegebenen Anzahl Stunden in die Zukunft korrekt von der Datenbank gelesen werden kann
ReadNumberOfMessagesTo SendHoursAppcodeMailtype	Prüft, ob die Anzahl zu sendenden Nachrichten anhand von der angegebenen Anzahl an Stunden in die Zukunft und Applikationscode, Mailtyp korrekt von der Datenbank gelesen werden kann
ReadNumberOfSentMessages SearchParameters	Prüft, ob die Anzahl aller gesendeten Nachrichten anhand von Suchparametern (E-Mail Adresse des Recipients, Vorname und Name des Recipients, Betreff, Zeitspanne von bis) korrekt von der Datenbank gelesen werden kann
ReadNumberOfSentMessages WithHours	Prüft, ob die Anzahl aller gesendeten Nachrichten anhand von der angegebenen Anzahl Stunden zurück in die Vergangenheit korrekt von der Datenbank gelesen werden kann
ReadNumberOfBouncedMessages	Prüft, ob die Anzahl aller Bounce Nachrichten anhand von Suchparametern (Anzahl Stunden zurück in die Vergangenheit, Applikationscode, Mailtyp) korrekt von

	der Datenbank gelesen werden kann
ReadAddressOverview	Prüft, ob die AddressQuality Übersicht (Anzahl E-Mail Adressen mit Qualitätsindikator good, poor, failed) korrekt von der Datenbank gelesen werden kann
ReadApplicationcodes	Prüft, ob die vorhandenen Applikationscodes korrekt von der Datenbank gelesen werden können
ReadMailtypes	Prüft, ob die vorhandenen Mailtypes korrekt von der Datenbank gelesen werden können

Tabelle 81 - Testmethoden MessageDALCTest

12.5.4.2 FilterDALCTest

Testmethode	Beschreibung
ReadFilterAddresses	Prüft, ob in der Datenbank abgelegte FilterAddresses anhand der Angabe, ob die Whitelist oder die Blacklist gelesen werden soll, korrekt gelesen werden können
CreateFilterAddress	Prüft, ob eine FilterAddress erstellt und in der Datenbank korrekt gespeichert werden kann
UpdateFilterAddress	Prüft, ob eine geänderte FilterAddress korrekt in die Datenbank übernommen wird
DeleteFilterAddress	Prüft, ob eine FilterAddress aus der Datenbank gelöscht werden kann
ReadFilterAddress	Prüft ob das Auslesen eines Filterlist-Eintrags funktioniert anhand einer Id
ReadFilterAddressesWithAddressPattern	Prüft ob die FilterAddresses welche auf ein bestimmtes AddressPattern passen, korrekt von der Datenbank gelesen werden können

Tabelle 82 - Testmethoden FilterDALCTest

12.5.4.3 NdrDALCTest

Testmethode	Beschreibung
ReadNdrPatterns	Prüft, ob in der Datenbank abgelegte NdrPatterns korrekt gelesen werden können.
ReadNdrPattern	Prüft, ob ein in der Datenbank abgelegtes NdrPattern anhand der NdrPatternId korrekt gelesen werden kann.
CreateNdrPattern	Prüft, ob ein NdrPattern erstellt und in der Datenbank korrekt gespeichert werden kann.
UpdateNdrPattern	Prüft, ob ein geändertes NdrPattern korrekt in die Datenbank übernommen wird.
DeleteNdrPattern	Prüft, ob ein NdrPattern aus der Datenbank gelöscht werden kann.
CreateNdrMail	Prüft, ob ein NdrMail erstellt und in der Datenbank korrekt gespeichert werden kann.
DeleteNdrMail	Prüft, ob ein NdrMail aus der Datenbank gelöscht werden kann.
ReadNdrMail	Prüft, ob ein in der Datenbank abgelegtes NdrMail anhand der NdrMailId korrekt gelesen werden kann
ReadNextNdrMailToAnalyze	Prüft, ob die nächste NdrMail von der Datenbank gelesen wird, die einen Status Create oder Requeue hat.
UpdateNdrMail	Prüft, ob ein geändertes NdrMail korrekt in die Datenbank übernommen wird.
ReadNdrDescription	Prüft, ob eine in der Datenbank abgelegte NdrDescription anhand des NdrTypes und der NdrCategory korrekt gelesen werden kann.
UpdateAddressQuality	Prüft, ob eine geänderte AddressQuality korrekt in die Datenbank übernommen wird.
ReadAddressQualityToUpdate	Prüft, ob die nächste zu aktualisierende AddressQuality (Qualitätsindikator) korrekt gewählt wird anhand der Angabe von Anzahl Tagen, die seit der letzten

	Aktualisierung des Qualitätsindikators zurückliegen sollen
ReadNdrMails	Prüft, ob in der Datenbank abgelegte NdrMails korrekt gelesen werden können
ReadUnrecognizedNdrMails	Prüft, ob NdrMails mit dem ProcessingState Unrecognized korrekt von der Datenbank gelesen werden können
ReadUnrecognizedNdrMailsWithCount	Prüft, ob bei Angabe einer Anzahl zu lesender NdrMails die korrekte Anzahl an NdrMails aus der Datenbank gelesen werden kann
ReadUnrecognizedNdrMails WithDateRange	Prüft, ob bei Angabe einer Zeitspanne (von bis) die korrekten NdrMails aus der Datenbank gelesen werden
ReadUnrecogniedNdrMailsCount WithDateRange	Prüft, ob bei Angabe einer Zeitspanne (von bis) die korrekten Anzahl an zu lesender NdrMails aus der Datenbank gelesen werden kann

Tabelle 83 - Testmethoden NdrDALCTest

12.5.4.4 CleanupDALCTest

Testmethode	Beschreibung
ReadCleanupRules	Prüft, ob in der Datenbank abgelegte CleanupRules korrekt gelesen werden können.
ReadCleanupRule	Prüft, ob ein in der Datenbank abgelegte CleanupRule anhand der CleanupRuleId korrekt gelesen werden kann.
CreateCleanupRule	Prüft, ob eine CleanupRule erstellt und in der Datenbank korrekt gespeichert werden kann.
UpdateCleanupRule	Prüft, ob eine geänderte CleanupRule korrekt in die Datenbank übernommen wird.
DeleteCleanupRule	Prüft, ob eine CleanupRule aus der Datenbank gelöscht werden kann.

Tabelle 84 - Testmethoden CleanupDALCTest

12.6 Anforderungserfüllung

Dieser Abschnitt soll aufzeigen, welche Anforderungen mit welchem Test abgedeckt sind und somit als erfüllt betrachtet werden können. Die Anforderungen wurden aus der Anforderungsspezifikation übernommen siehe Abschnitt 7 Anforderungsspezifikation auf Seite 30.

12.6.1 Funktionale Anforderungen

Anforderung	Anforderungserfüllungstest	Erledigt
Versenden von E-Mail Nachrichten im Plaintext oder MIME Format	Geprüft mit System-Test UC 1 (KW 16)	OK
Versenden von E-Mail Nachrichten an ein bis mehrere Empfänger	Geprüft mit System-Test UC 1 (KW 16)	OK
Klassierung von Empfänger nach TO, CC und BCC möglich	Geprüft mit System-Test UC 1 (KW 16)	OK
Versenden von E-Mail Nachrichten zeitgesteuert über Angabe ‚nicht früher als'	Geprüft mit System-Test UC 1 (KW 16)	OK
Versandpriorisierung durch zuweisen von Applikationscode und Mailtypen zu Mailprioritäten	Geprüft mit System-Test UC 1 (KW 16). Dabei bezieht sich die Versandpriorisierung auf die Konfiguration im App.config File.	OK
Unterstützung von mehreren SMTP Server, welche die Nachrichten in der Priorität High resp. Low abarbeiten	Geprüft mit System-Test UC 1 (KW 16). Dabei bezieht sich die Versandpriorisierung auf die Konfiguration im App.config File.	OK
Unterstützung einer Blacklist, welche auch Wildcards erlaubt	Geprüft mittels System-Test UC 1 (KW 16)	OK
Unterstützung einer Whitelist, welche auch Wildcards erlaubt. Über die Konfiguration kann die Whitelist aktiviert und deaktiviert werden	Geprüft mittels System-Test UC 1 (KW 16)	OK
Abfragen des Versandstatus einer Nachricht mittels Guid	Geprüft mittels System-Test UC 2 (KW 16)	OK
Anbieten von zwei getrennten APIs. Eine API für das Versenden von Nachrichten und Abfragen des aktuellen Versandstatus und eine API für Reporting und Maintenance Aufgaben.	Implizit erfüllt, indem zwei Service Assemblies angeboten werden (Service.ReportingMaintenance und Service.Sending)	OK
Ablegen der E-Mail Nachrichten in einer Datenbank	Geprüft mittels System-Test UC 1 (KW 16) und Unit-Test (CreateMessage Testmethode im MessageDALCTest)	OK
Pro E-Mail Nachricht und Empfänger wird ein Status abgelegt (nicht versendet, versendet, blacklisted, failed)	Geprüft mittels System-Test UC 1 (KW 16)	OK
Pro E-Mail Adresse wird ein Qualitätsindikator abgelegt (good, poor, failed)	Geprüft mittels System-Test UC 1 (KW 16)	OK
Pro E-Mail Nachricht werden Metadaten in die Datenbank abgelegt (Applikationscode, Mailtyp, Referenz)	Geprüft mittels System-Test UC 1 (KW 16)	OK
Über das Webinterface ist die Konfiguration der Blacklist möglich (hinzufügen, löschen und bearbeiten von Einträgen)	Geprüft mittels System-Test UC 4 (KW 18), Geprüft mittels manuellem GUI Test (KW 22)	OK

Über das Webinterface ist die Konfiguration der Whitelist möglich (hinzufügen, löschen und bearbeiten von Einträgen)	Geprüft mittels System-Test UC 4 (KW 18), Geprüft mittels manuellem GUI Test (KW 22)	OK
Über das Webinterface ist die Konfiguration des Bounce-Handlings möglich (hinzufügen, löschen und bearbeiten von NdrPatterns anhand von nicht erkannten NdrMails)	Geprüft mittels System-Test UC 5 (KW 18), Geprüft mittels manuellem GUI Test (KW 22)	OK
Über das Webinterface ist die Konfiguration der Datenbankbereinigung möglich (Bereinigung pro Applikationscode und Mailtyp konfigurierbar)	Geprüft mittels System-Test UC 9 (KW 18), Geprüft mittels manuellem GUI Test (KW 22)	OK
Über das Konfigurationsfile des *Smtp Mailers* ist es möglich, die Anzahl Prozesse zu definieren, welche High Priority Nachrichten abarbeiten resp. Low Priority Nachrichten abarbeiten	Geprüft mittels System-Test UC 1 (KW 16)	OK
Über das Konfigurationsfile des *Smtp Mailers* ist es möglich, einem Applikationscode und Mailtyp Tupel eine Priorität zuzuweisen (High, Low)	Geprüft mittels System-Test UC 1 (KW 16)	OK
Über das Konfigurationsfile des *Smtp Mailers* ist es möglich, die Übergabemethode zwischen dem *Smtp Mailer* und dem versendenden SMTP Server zu definieren (SMTP Protokoll oder Message Queue)	Geprüft mittels System-Test UC 1 (KW 16)	OK
Über das Konfigurationsfile des *Smtp Mailers* ist es möglich, die Whitelist zu aktivieren resp. deaktivieren	Geprüft mittels System-Test UC 1 (KW 16)	OK
Der *Smtp Mailer* bietet ein automatisches Bounce-Handling an, welches Bounce Nachrichten auswertet und das Ergebnis der Auswertung in den Qualitätsindikator einfliessen lässt	Geprüft mittels System-Test UC 10 (KW 18)	OK
Der *Smtp Mailer* unterscheidet verschiedene Bounce Typen (Hard, Soft, etc.)	Implizit erfüllt durch die Verwendung von der NetXtremeBounceFilter Komponente von Safabyte	OK
Über das Webinterface ist es möglich, Bounce-Nachrichten abzufragen, welche nicht erkannt wurden	Geprüft mittels manuellem GUI Test	OK
Über das Webinterface ist es möglich, E-Mail Nachrichten anzuzeigen (als Liste oder als Zusammenfassung) gefiltert nach Datum / Uhrzeit, E-Mail Adresse(n), Betreff, Text, Status, Applikationscode und Mailtyp	Geprüft mittels System-Test UC 3 (KW 22), Geprüft mittels manuellem GUI Test (KW 22)	OK
Über das Webinterface ist es möglich, eine Detailansicht von einzelnen E-Mail Nachrichten zu erhalten	Geprüft mittels System-Test UC 3 (KW 22), Geprüft mittels manuellem GUI Test (KW 22)	OK
Über das Webinterface ist das Anzeigen von Qualitätsindikatoren von E-Mail Adressen möglich, gefiltert nach E-Mail Adresse und Qualitätsindikator. Die Darstellung ist als Übersicht, List oder Detailansicht möglich sein	Geprüft mittels manuellem GUI Test (KW 22)	OK

Über das Webinterface ist das Anzeigen von Reports möglich (Wie viele E-Mail Nachrichten wurden über einen Zeitraum verschickt, wie viele E-Mail Nachrichten werden in den nächsten n Stunden verschickt)	Geprüft mittels manuellem GUI Test (KW 22)	OK
Über das Webinterface ist es möglich, E-Mail Nachrichten nochmals zu versenden, welche wegen einer Bounce-Nachricht als nicht versendet angeschaut werden (Nice-To-Have)	Nicht geprüft, weil diese Anforderung als Nice-to-Have angeschaut werden kann und nicht implementiert wurde	-
Der *Smtp Mailer* bietet ein regelmässiges Bereinigen der Datenbank (Löschen von Content und Attachments) gemäss CleanupRules an	Geprüft mittels System-Test UC 8 (KW 18)	OK

Tabelle 85 - Anforderungserfüllung funktionale Anforderungen

12.6.2 Nicht funktionale Anforderungen

Anforderung	Anforderungserfüllungstest	Erledigt
Die Software muss auf einem Dual-Core XEON x64-Architektur lauffähig sein	Konnte nicht getestet werden, weil keine entsprechende Hardware zur Verfügung stand	-
Die Software muss auf einer Microsoft Windows Server 2003 Standard Umgebung lauffähig sein	Laufenlassen des *Smtp Mailers* auf dem virtuellen Server der HSR	OK
Die Datenspeicherung muss in einer Microsoft SQL Server 2005 Datenbank erfolgen	Bereits während der Entwicklung wurde mit einem Microsoft SQL Server 2005 gearbeitet	OK
Der *Smtp Mailer* muss mit einer dazugekauften Mail Komponente und einer dazugekauften Ndr Komponente zusammenwirken können	Integration der Drittherstellerkomponenten während der Entwicklung	OK
Bei Fehler werden Exceptions geworfen. Exceptions welche an menschliche Benutzer gehen enthalten eine textuelle Beschreibung des Fehlers	System-Tests der verschiedenen Engines (Sending Engine, Bounce Engine, Cleanup Engine, AddressQualityEngine). Exceptions vom *Smtp Mailer* Core werden im WCF Service in menschenlesbare Fehlermeldungen übersetzt, welche in der Silverlight Applikation angezeigt werden	OK
Bei einem Fehler im System gehen keine Nachrichten verloren	Geprüft mit System-Test UC 1 (KW 16). Durch das Absichern mittels Transaktionen kann eine zum Versenden gelesene Nachricht nicht durch einen Fehler verloren gehen, dass die Nachricht nie verschickt wird.	OK
Die Weboberfläche muss intuitiv zu bedienen sein. Für einen Benutzer mit Weberfahrung muss die Webapplikation einfach zu handhaben sein	Mittels dem Paper Prototype Test wurde die Usability anhand eines realen Benutzers sichergestellt. Zudem nahm Benedikt Unold bei der Meilenstein 15 Sitzung die Webapplikation ab.	OK

Die Webapplikation muss nach kurzer Einarbeitungszeit (1/2 Tag) verstanden werden	Wurde nicht geprüft. Weil jedoch die Webapplikation auf die Endbenutzer zugeschnitten ist und dank einer QuikCard für den Kundendienst dürfte diese Anforderung erfüllt sein	OK
Die Webapplikation muss vollständig mit der Maus bedienbar sein	Geprüft mit einem manuellen GUI Test	OK
Die Versandleistung des *Smtp Mailers* muss 2‘000‘000 E-Mail Nachrichten pro Tag und 300‘000 E-Mail Nachrichten pro Stunde betragen	Geprüft mittels System-Test (KW 16)	OK
Die Verarbeitungsleistung des *Smtp Mailers* muss 200‘000 NDR Nachrichten pro Tag und 30‘000 NDR Nachrichten pro Stunde erreichen	Geprüft mittels System-Test (KW 18)	OK
Die Business Logik besitzt 100% Statement Coverage über Unit Tests	Alle Public Methoden sowie Methoden, welche in Interfaces angeboten werden, besitzen eine Unit-Test Testmethode	OK
Es können mehrere Instanzen des *Smtp Mailers* gleichzeitig betrieben werden	Geprüft mittels System-Test (KW 18)	OK

Tabelle 86 - Anforderungserfüllung nicht funktionale Anforderungen

13 Projektmanagement

13.1 Projektübersicht

Das Projekt umfasst drei Teilprojekte mit folgendem Umfang:

- Entwickeln eines Mail Moduls für das Versenden von priorisierbaren und zeitgesteuerten E-Mails. Die E-Mails müssen im Plaintext oder Multipart E-Mail-Format erstellt werden können. Es müssen Anhänge und Signaturen unterstützt werden. Zusätzlich muss das Mail Modul über eine White- / Blacklist verfügen um effizient die E-Mails zu filtern. Der Versand kann über mehrere SMTP Server erfolgen und hat eine Zielgrösse von 300‘000 E-Mails pro Stunde. Versendete E-Mails müssen selektiv gelöscht oder archiviert werden können. Das Mail Modul stellt eine Schnittstelle in Form einer Klassenbibliothek und Webservices für ausgewählte Funktionen bereit.
- Für zurückgewiesene E-Mails muss der Non Delivery Report (NDR) ausgewertet werden, um den Fehlertyp und den Versandstatus zu eruieren. Diese Information fliesst in das Adressqualitätsmanagement ein. So kann ein Qualitätsindikator für alle verwendeten E-Mail Adressen ermittelt werden.
- Die Administration, Konfiguration und das Reporting erfolgen über ein Webinterface. Es sollen diverse Auswertungen erstellt werden können. Die Webapplikation soll mittels der Silverlight Technologie entwickelt werden.

13.1.1 Zweck und Ziel

In diesem Projekt geht es primär um die Verbesserung und Ablösung der bestehenden Mail-Lösung NLNMailer. Das bisherige System verfügte über keine zentrale Schnittstelle für das Versenden von E-Mails. Damit war keine unternehmensweite Konfiguration und zentrales Logging möglich. Auch betreffend Funktionalität ist die bestehende Lösung eingeschränkt. Bisher konnten keine Multipart E-Mails oder signierte E-Mails versendet werden. Durch fehlendes Bounce-Handling konnte der E-Mailadressstamm nicht automatisiert gepflegt werden. Zudem hat das bisherige System mit instabilen und hängenden Prozessen zu kämpfen.

Die Ziele dieser Arbeit sind die Konzeption, das Design, die Umsetzung sowie Funktions- und Performancetests des *Smtp Mailers*.

13.1.2 Annahmen und Einschränkungen

Für das Projekt wird mit einer Richtzeit von 400 Stunden pro Teammitglied geplant. Dies entspricht einem wöchentlichen Aufwand pro Teammitglied von 25 Stunden.

Dieser Projektplan wird in der Inception-Phase gemäss RUP entwickelt. In dieser Startphase können nur grobe Planungen gemacht werden und es müssen Annahmen getroffen werden. Deshalb können zwischenzeitlich Abweichungen zu den angenommenen Terminen und Daten entstehen. Der Projektplan wird jedoch während des Projekts aktuell gehalten.

13.1.3 Ausbaufähigkeiten

Das Mail Modul wird so entwickelt, dass es um weitere Eigenschaften ergänzt werden kann. Denkbar wäre die verschlüsselte Übertragung mittels SSL wie auch die Verschlüsselung der E-Mails selbst. Somit müssen solche Eigenschaften bereit bei der Analyse der zu verwendenden Mail Komponente eines Drittherstellers beachtet werden.

Die Möglichkeit der Ausbaufähigkeit für eine Verschlüsselung des E-Mail Verkehrs ist vom Industriepartner erwünscht, dieses Feature ist aber als Nice-to-Have einzustufen.

13.2 Projektorganisation

13.2.1 Projektmitarbeiter

13.2.1.1 Teammitglieder

Das Projektteam besteht aus zwei gleichberechtigten Personen:

- Danny Meier, dmeier@hsr.ch
- Micha Boller, mboller@hsr.ch

13.2.1.2 Betreuer

Die Betreuung dieser Bachelorarbeit wird übernommen von:

- Prof. Dr. Markus Stolze, Institut für Software, Hochschule für Technik, Rapperswil, mstolze@hsr.ch

13.2.1.3 Industriepartner

Dieses Projekt wird für die Firma comparis.ch AG entwickelt. Als Ansprechpartner gilt:

- Benedikt Unold, CTO, comparis.ch AG, Zürich, benedikt.unold@comparis.ch

13.2.1.4 Experte

Der Experte dieser Arbeit ist:

- Dr. Marcel Graf, IBM Research GmbH, Rüschlikon ZH

13.2.1.5 Gegenleser

Der Gegenleser dieser Arbeit ist:

- Prof. Hansjörg Huser, .NET Competence Center, Hochschule für Technik, Rapperswil, hhuser@hsr.ch

13.2.2 Software Engineering Vorgehensmodell

Dieses Projekt wird nach dem Rational Unified Process (RUP) abgehandelt. Als Vorgabe zum Vorgehen und für Detailinformationen wird Bezug auf das Lehrbuch von Craig Larman (Larman, 2005) genommen.

Sämtliche Teilprojekte in dieser Bachelorarbeit werden in einem Zyklus erarbeitet und entsprechend wird jede der vier RUP-Phasen nur einmal durchlaufen. Daher wird der Grossteil der Anforderungen bereits in der Inception- / Elaboration 1 – Phase spezifiziert und zusätzliche Anforderungen werden in späteren Phasen nachgeführt.

Desweiteren werden sämtliche Systemtests gleich nach der Spezifizierung der Anforderungen ausgearbeitet und bilden alle relevanten Use Cases ab. Dieses Vorgehen ist dem V-Modell nachempfunden und erlaubt einen hohen Deckungsgrad zwischen Anforderungen und Systemtests.

13.3 Managementabläufe

13.3.1 Projektkosten

Dem Projekt stehen 400 Stunden pro Teammitglied zur Verfügung. Dies entspricht einer totalen Arbeitszeit von 800 Stunden.

Der Projektstart erfolgt am 16.02.2009 und das Projektende wird auf den 12.06.2009 festgelegt.

13.3.2 Projektplan

13.3.2.1 Zeitplan

Die Zeitplanung wird durch das iterative Vorgehen gemäss RUP vor zu für die anstehenden Iterationen ergänzt und verfeinert.

Die Zeitplanung wird mittels der Projektmanagement- und Projektplanungssoftware dotProject erstellt. Die Software ist webbasiert und kann unter https://ba.danflash.com/dotproject/ erreicht werden. Für die Projektbetreuung und den Industriepartner wurde ein Gastaccount eingerichtet. Die Zugangsdaten findet man unter dem Abschnitt Zugänge auf Seite 203.

Um den Zeitplan des Projektes anzuschauen geht man folgendermassen vor:

1. Login mit dem genannten Benutzername und Passwort
2. Klick auf *Projects* im Menü am oberen Bildrand
3. Klick auf *SmtpMailer* in der Projekttabelle
4. Die Arbeitspakete mit den Soll- und Ist-Zeiten sind ersichtlich

13.3.2.2 Iterationsplan

Der Iterationsplan zeigt die Aufteilung der Projektlaufzeit in die RUP Iterationen auf. Es ist ersichtlich, welchen Inhalt die verschiedenen Iterationen haben. Genauere Informationen zu den Iterationen können aus den Arbeitspaketen (Abschnitt Arbeitspakete auf Seite 190) oder im Zeitplan, welcher mit dotProject verwaltet wird, entnommen werden (Abschnitt Zeitplan auf Seite 186).

Iteration	Beschreibung	Ende der Iteration	Dauer in Wochen
Inception	Kickoff Meeting, Unterschreiben von Verträgen, Projektplan erstellen, Beginn der Anforderungsspezifikation (Use Cases im Brief-Format), Dokumentenvorlagen erstellen, Einrichten der Umgebung	KW 8	1
Elaboration 1	Anforderungsspezifikation erstellen, Domainanalyse erstellen, Datenmodell entwickeln, Komponenten Evaluation durchführen, Studium Silverlight Technologie, System-Tests planen, Integration-Tests planen	KW 10	2
Elaboration 2	Design des Architektur Prototyps, Implementierung des Architektur Prototyps, Implementierung des Prototyps mit der evaluierten Mail Komponente und Performance-Test (Risiko 04), Implementierung des Prototyp mit der Datenbank und Performance-Test (Risiko 05)	KW 12	2
Construction 1	Design des Mail Moduls und API, Implementierung des Mail Moduls und API, Unit-Tests für Mail Modul	KW 15	3
Construction 2	Design des NDR Moduls, Implementierung des NDR Moduls, Unit-Tests für NDR Modul	KW 18	3
Construction 3	Design der Webapplikation, Entwicklung von Paper Prototypes, Usability Tests durchführen, Implementierung Web Applikation, Test der Webapplikation	KW 21	3
Transition 1	System-Tests durchführen, Integration-Tests durchführen	KW 22	1
Transition 2	Abnahme *Smtp Mailer* durch Industriepartner, Dokumentation abschliessen, Verfassung Kurzbericht, Erstellung A0-Poster, Abschlussessen	KW 24	2

Tabelle 87 - Iterationsplanung

Abbildung 99 - Iterationsplanung

In Abbildung 99 - Iterationsplanung sind die RUP Iterationen ersichtlich. Es wird aufgezeigt, während welchen Kalenderwochen die Iterationen stattfinden.

13.3.2.3 Meilensteine

Meilenstein	Woche	Datum	Ergebnis	Teilnehmer
MS 1	KW 9	27.02.09	Projektplan	Markus Stolze, Benedikt Unold
MS 2	KW 10	06.03.09	Anforderungsspezifikation	Benedikt Unold
MS 3	KW 10	06.03.09	Komponenten Evaluation	Benedikt Unold
MS 4	KW 11	13.03.09	Domainanalyse	Benedikt Unold
MS 5	KW 11	13.03.09	Datenmodell	Benedikt Unold
MS 6	KW 11	13.03.09	Design Architektur Prototyp	Benedikt Unold
MS 7	KW 13	27.03.09	Architektur Prototyp implementiert	Markus Stolze
MS 8	KW 13	27.03.09	Prototyp mit der evaluierten Mail Komponente implementiert (Risiko 04), Prototyp mit der Datenbank implementiert (Risiko 05)	Markus Stolze
MS 9	KW 14	03.04.09	Design Mail Modul	Benedikt Unold
MS 10	KW 16	17.04.09	Mail Modul implementiert	Markus Stolze, Benedikt Unold
MS 11	KW 17	24.04.09	Design NDR Modul	Benedikt Unold
MS 12	KW 19	08.05.09	NDR Modul implementiert	Markus Stolze, Benedikt Unold
MS 13	KW 20	15.05.09	User Interface Evaluation	Markus Stolze
MS 14	KW 21	22.05.09	Webapplikation Prototyp	Markus Stolze
MS 15	KW 22	29.05.09	Webapplikation implementiert	Markus Stolze, Benedikt Unold
MS 16	KW 23	05.06.09	System-Tests und Integration-Tests abgeschlossen, Abnahme	Markus Stolze, Benedikt Unold
MS 17	KW 24	12.06.09	Schlussabgabe	

Tabelle 88 - Meilensteine

Meilenstein	Beschreibung
MS 1	Projektplan erstellt / gereviewed und von Betreuer / Industriepartner abgenommen
MS 2	Anforderungsspezifikation erstellt / gereviewed und von Industriepartner abgenommen
MS 3	Komponenten von Drittherstellern (Mail- und NDR-Komponente) evaluiert und Evaluation / Komponentenentscheid durch Industriepartner abgenommen
MS 4	Domainanalyse erstellt / gereviewed und von Industriepartner abgenommen
MS 5	Datenmodell erstellt / gereviewed und von Industriepartner abgenommen
MS 6	Design von Architektur Prototyp erstellt und von Industriepartner abgenommen
MS 7	Demonstration von Architektur Prototyp beim Betreuer / Industriepartner
MS 8	Demonstration von Prototyp mit der evaluierten Mail Komponente (Risiko 04), Prototyp mit der Datenbank (Risiko 05) beim Betreuer / Industriepartner
MS 9	Design Mail Modul erstellt / gereviewed und von Industriepartner abgenommen
MS 10	Projektfortschrittsziel erreicht
MS 11	Design NDR Modul erstellt / gereviewed und Industriepartner abgenommen
MS 12	Projektfortschrittsziel erreicht
MS 13	User Interface Evaluation erstellt / gereviewed und von Betreuer abgenommen
MS 14	Demonstration von Webapplikation Prototyp beim Betreuer
MS 15	Abnahme von Webapplikation durch Betreuer / Industriepartner
MS 16	System-Tests abgeschlossen, Integration-Tests abgeschlossen
MS 17	Abgabe sämtlicher Dokumente und Abschluss der Bachelorarbeit

Tabelle 89 - Beschreibung Meilensteine

13.3.2.4 Release- und Reviewtermine

Datum	Woche	Dokument-Review (intern)	Code-Review (intern)	Release
25.02.09	KW 9	Review 1 Projektplan		
04.03.09	KW 10	Review 2 Anforderungsspezifikation		
11.03.09	KW 11	Review 3 Domainanalyse		
11.03.09	KW 11	Review 4 Datenmodell		
11.03.09	KW 11	Review 5 Design Architektur Prototyp		
25.03.09	KW 13		Review 1 Architektur Prototyp	Release 1 Architektur Prototyp
01.04.09	KW 14	Review 6 Design Mail Modul		
15.04.09	KW 16		Review 2 Mail Modul	Release 2 Mail Modul
15.04.09	KW 16	Review 7 Design NDR Modul		
06.05.09	KW 19		Review 3 NDR Modul	Release 3 NDR Modul
13.05.09	KW 20	Review 8 User Interface Evaluation		
20.05.09	KW 21		Review 4 Prototyp Webapplikation	Release 4 Prototyp Webapplikation
27.05.09	KW 22		Review 5 Webapplikation	Release 5 Webapplikation
03.06.09	KW 23	Review 9 System-Tests, Integration-Tests		

Tabelle 90 - Release- und Reviewtermine

13.3.2.5 Besprechungen

Wöchentlich findet eine Projektfortschrittssitzung mit dem Betreuer statt. Projektintern findet jeweils freitags eine Sitzung statt, bei welcher Dokumente und Code gereviewt werden und die nächste Iteration geplant wird. Zudem werden dabei die Todo Listen im Wiki erstellt.

Bei Meilensteinen findet eine Sitzung mit dem Industriepartner und dem Betreuer statt. Diese Sitzungen werden gemäss der Meilensteinplanung angesetzt und es ist vorgängig eine Traktandenliste zu erstellen.

Wochentag	Zeit	Teilnehmer	Inhalt
Mittwoch	08:10	Betreuer (Prof. Dr. Markus Stolze, HSR) Danny Meier Micha Boller	Projektfortschrittssitzung Klärung von Fragen, Problemen und Besprechung vom Projektfortschritt und Risiken.
Freitag	08:10	Danny Meier Micha Boller	Teaminterne Sitzung Besprechung von den nächsten Arbeitsschritten, Aufstellen von Todo Listen im Wiki, Review von Code und Dokumenten, Planung der nächsten Iteration
Gemäss Meilenstein-Planung		Industriepartner (Benedikt Unold, comparis.ch AG) Betreuer (Prof. Dr. Markus Stolze, HSR) Danny Meier Micha Boller	Besprechung von Dokumenten, Prototypen gemäss Meilenstein

Tabelle 91 - Besprechungen

13.4 Arbeitspakete

Die Arbeitspakete sind parallel auch im Online Projektmanagement-Tool dotProject (http://ba.danflash.com/dotproject) erfasst.

Die Arbeitspakete gliedern sich in die RUP Disziplinen Requirements, Analyse, Design, Implementation und Test ein. Zusätzlich wurden die Disziplinen Meetings, Projektmanagement, Infrastruktur und Technologiestudium eingeführt. Jedes Arbeitspaket wird von einem Projektmitglied bearbeitet. Es werden die Kürzel *dm* (Danny Meier) und *mb* (Micha Boller) eingesetzt. Arbeiten beide Teammitglieder am Arbeitspaket, so wird das Paket mit *Team* vermerkt. Priorität 1 hat höchste Priorität.

Nr.	Name	Inhalt / Resultat	Iteration	Verantwortlichkeit	Soll (in h)	Ist (in h)	Priorität (1, 2, 3)
1 Meetings							
101	Kickoff Meeting	Projekt starten	Inception	Team	2	3.5	1
102	Projektfortschrittssitzungen	Besprechung von Projektfortschritt, Problemen, Fragen, Zeitplan, Risiken, nächste Schritte, Termine mit dem Betreuer	Inception bis Transition	Team	32	13.5	1
103	Interne Sitzungen	Besprechung projektintern von nächsten Schritten, Durchführen von Code Reviews	Inception bis Transition	Team	32	27.5	1
104	Meilensteinsitzungen	Besprechung von Dokumenten und Releases zusammen mit dem Betreuer und Industriepartner	Inception bis Transition	Team	28	29.5	1
105	Sitzungsprotokoll verfassen	Dokumentation der Sitzung, indem das Besprochene, Erkenntnisse, Termine dokumentiert werden. Versenden der Protokolle an den Betreuer und evtl. Industriepartner	Inception bis Transition	dm	20	16.5	2
106	Sitzungen vorbereiten	Schreiben von Traktandenlisten, Versenden von Traktandenlisten, Vorbesprechung der Sitzung	Inception bis Transition	Team	14	19.75	1
2 Projektmanagement							
201	Projektplan erstellen, überarbeiten	Verfassung des Projektplanes (Projektbeschreibung, Projektorganisation, Managementabläufe, Risikoanalyse, Infrastruktur, Qualitätsmassnahmen)	Inception, Elaboration 1	Team	30	33.25	1
202	Review Projektplan	Review des Dokumentes projektintern	Inception	Team	2	2	2
203	Dokumentationsvorlagen erstellen	Erstellen einer einheitlichen Dokumentationsvorlage	Inception	Team	2	1	1
204	Zeitplan erstellen	Zeitplan erstellen und vor jeder Iteration für die nächste Iteration ergänzen und verfeinern	Inception bis Transition	Team	5	2.75	1

205	Arbeitspakete erfassen und ergänzen	Erfassung der Arbeitspakete im Projektplan und in dotProject	Inception bis Transition	mb	7	3.75	2
206	Dokumentation abschliessen	Review von Dokumenten, Gesamtdokument zusammenfügen, Inhaltsverzeichnis erstellen, etc.	Transition 2	Team	50	84.5	1
3 Infrastruktur							
301	Virtueller Server einrichten	Basis-Infrastruktur installieren, konfigurieren (Trac, SVN, dotProject)	Inception	dm	8	12.5	1
302	SVN einrichten	SVN Verzeichnisstruktur einrichten und auf persönlichen Notebooks konfigurieren	Inception	Team	2	1	1
303	Server Backup	Einrichten des Server Backups	Inception	dm	1	1	1
304	Testumgebung	Einrichten der Testumgebung (Datenbank, Silverlight Umgebung, Testdaten)	Inception bis Transition	Team	8	5	1
4 Technologiestudium							
401	Studium Codierungsrichtlinen	Einlesen in die Codierungsrichtlinien von Microsoft	Inception	Team	4	4	
402	Studium Silverlight Technologie	Einarbeiten in die Silverlight Technologie. Lesen von Artikeln, Tutorials, Büchern	Elaboration 1 bis Construction 3	Team	20	19.5	1
403	Studium Log4Net	Einarbeiten in die Logging Technologie. Verständnis entwickeln für Konfigurationsfile	Construction 1	Team	10	2.5	1
4 Requirements							
401	Use Cases brief	Von den wichtigsten Use Cases eine brief Beschreibung erstellen. Dokumentation in der Anforderungsspezifikation	Inception	Team	2	0	1
402	Use Cases fully dressed	Use Cases fully dressed ausgearbeitet. Dokumentation in der Anforderungsspezifikation	Elaboration 1	Team	10	12.50	1
403	Use Case Model	Ausarbeiten eines Use Case Models mit den Aktoren	Inception	Team	2	4	2
404	Funktionale Anforderungen	Ausarbeiten der funktionalen Anforderungen. Dokumentation in der Anforderungsspezifikation	Elaboration 1	Team	2	1	1
405	Supplementary Specification	Ausarbeiten der nicht funktionalen Anforderungen. Dokumentation in der Anforderungsspezifikation	Elaboration 1	Team	5	2.5	1
406	Komponenten Evaluation	Durchführen der Komponenten Evaluation für eine Mailer-Komponente und eine Bounce-Handling Komponente	Elaboration 1	Team	10	34.5	1

407	Review Anforderungsspezifikation	Review des Dokumentes Anforderungsspezifikation projektintern	Elaboration 1	Team	4	6.25	2
408	Anforderungsspezifikation erstellen	Erstellen von Dokument, Verfassung von allgemeinen Kapitel	Elaboration 1	Team	5	8	2
5 Analyse							
501	Domain Model	Erstellen eines Domain Models. Dokumentation in Domainanalyse	Elaboration 1	Team	10	11.5	1
502	Konzeptbeschreibung	Erstellen der Konzeptbeschreibungen basierend auf dem Domain Model. Dokumentation der Konzepte in der Domainanalyse	Elaboration 1	Team	4	3	2
503	System Sequenzdiagramme und Contracts	Erstellen der System Sequenzdiagramme basierend auf den Use Cases. Erstellen von Contracts für die System Operationen.	Elaboration 1	Team	15	19.5	1
504	Review Domainanalyse	Review des Dokumentes Domainanalyse projektintern	Elaboration 1	Team	2	2.5	2
6 Design							
601	Datenmodell entwickeln	Erstellen und Ausarbeiten eines Datenmodells für die Datenbank	Elaboration 1	Team	20	7.25	1
602	Design Architektur Prototyp	Dokumentieren eines Architektur Prototyps im Software Architektur Dokument	Elaboration 1	Team	10	13.25	1
603	Design Prototyp Risiko 04	Dokumentation der Bestandteile des Prototyps	Elaboration 2	Team	5	6.75	1
604	Design Prototyp Risiko 05	Dokumentation der Bestandteile des Prototyps	Elaboration 2	Team	10	17.75	1
605	Design Mail Modul	Erstellen von Klassendiagrammen, Beschreibung von Namespaces, Zeichnen von konkreten System Sequenzdiagrammen	Construction 1	Team	15	12	1
606	Design API	Erarbeiten von API's und den dazugehörigen Methoden mit Rückgabewerten und Parameterwerten	Construction 1	Team	12	6	1
607	Software Architecture Document	Verfassen vom Software Architecture Document	Construction 1 bis Construction 3	Team	10	22.75	1
608	Design NDR Modul	Ergänzen von Klassendiagramm, Beschreibung von Namespaces, Zeichnen von konkreten System Sequenzdiagrammen	Construction 2	Team	15	9.25	1
609	Design Qualitätsindikator Algorithmus	Aufzeigen vom Ablauf mittels System Sequenzdiagramm, Festlegen von Regeln für den Entscheid und die Berechnung des Qualitätsind.	Construction 2	Team	8	9	1

610	Paper Prototype entwickeln	Erstellen des Paper Prototypes für die Usability-Tests	Construction 3	Team	15	14	1
611	Design Webapplikation	Dokumentation von Personas und Szenarios, Vorbereitung Contextual Inquiry	Construction 3	Team	20	20.75	1
7 Implementierung							
701	Implementierung Architektur Prototyp	Implementierung des vertikalen Architektur Prototyps gemäss Design	Elaboration 2	Team	20	21	1
702	Implementierung Risiko 04	Entwicklung eines Prototyps, welcher die Mail Komponente nutzt. Durchführen von Tests. Die Tests sollen die Möglichkeit der Erfüllung der Performance-Anforderungen mit der Mail Komponente beweisen	Elaboration 2	Team	12	16	1
703	Implementierung Risiko 05	Entwicklung eines Prototyps, welcher die Daten der zu versendenden E-Mail Nachrichten in die Datenbank schreibt. Die Tests sollen die Möglichkeit der Erfüllung der Performance-Anforderungen auf der gegebenen Hardware beweise.	Elaboration 2	Team	40	85.5	1
704	Erstellen von Datenbank Skript	Erstellen des Datenbank Skripts, welches sämtliche Tabellen und Beziehungen aufbaut	Elaboration 2	Team	4	3.75	2
705	Implementierung Mail Modul	Implementierung von Business Logik und Data Access Layer für Mail Modul. Implementierung von Exception Handling, Logging	Construction 1	Team	50	73.5	1
706	Implementierung API	Implementierung des Service Layers	Construction 1	Team	5	10.5	2
707	Implementierung NDR Modul	Implementierung von Business Logik und Data Access Layer für NDR Modul. Implementierung von Exception Handling, Logging	Construction 2	Team	50	49	1
708	Implementierung CleanupEngine	Implementierung der Engine, welche den CleanupProzess anstösst	Construction 1	Team	5	4	2
709	Implementierung AddressQuality Engine	Implementierung der Engine, welche den AddressQuality Berechnungsalgorithmus anstösst	Construction 2	Team	5	4	2
710	Implementierung WCF Services	Implementierung der WCF Schnittstelle für den Datenaustausch zwischen Silverlight und Core	Construction 3	Team	20	17	1

711	Implementierung Webapplikation	Implementierung der Silverlight Webapplikation	Construction 3	Team	70	178	1
712	Code abschliessen	Fertigstellung von Code und Kommentaren	Transition 2	Team	20	10.25	1
8 Test							
801	System-Tests planen	Aufgrund der Use Cases die System-Tests planen	Elaboration 1	Team	5	10.5	1
802	System-Tests durchführen für Mail Modul	Systemtests durchführen, dabei werden die Use Cases geprüft, welche auf das Mail Modul zutreffen	Construction 1	Team	10	11	1
803	Integration-Tests durchführen für Mail Modul	Durchführen von Integration-Tests (manuell) d.h. ein Zusammenspiel aller Komponenten über alle Software Schichten testen	Construction 1	Team	3	1	1
804	Unit Tests Mail Modul	Implementierung von Unit Tests für die Core- und DAL Klassen	Construction 1	Team	20	15.5	1
805	System-Tests durchführen für NDR Modul	Systemtests durchführen, dabei werden die Use Cases geprüft, welche auf das NDR Modul zutreffen	Construction 2	Team	8	18	1
806	Unit-Tests NDR Modul	Implementierung von Unit Tests für die Core- und DAL Klassen	Construction 2	Team	20	33	1
807	Usability-Test mit Paper Prototype	Durchspielen der Usability-Tests mit einem Kundendienstmitarbeiter. Besprechung der Resultate mit Designverantwortlichen bei comparis.ch AG	Construction 3	Team	4	2.5	1
808	GUI-Test (manuell)	Testen der Silverlight Benutzeroberfläche	Construction 3	Team	8	3.75	1
809	System-Tests durchführen für Webapplikation	Testen der Webapplikation mit den anderen Bestandteilen des *Smtp Mailer*s	Construction 3	Team	12	8.5	1

Tabelle 92 – Arbeitspakete mit Soll- und Ist Zeiten

13.5 Risikomanagement

13.5.1 Risiken

Die Risiken werden in die Prioritäten gering, mittel und hoch klassiert. Je nachdem ist eine frühzeitige Behandlung des Risikos sinnvoll.

ID	Risiko	Auswirkung	Massnahme	Kosten der Massnahme	Max. Schaden	Eintritts-Wahrscheinlich-keit	Gewichteter Schaden	Priorität
R01	Es gibt keine Mailer-Komponente, welche sämtlichen Anforderungen entspricht (Anbieten von Plaintext, MIME Format, Attachmets, Signaturen, geforderte Performance, Verschlüsselung etc.)	Funktionalitäten, welche die Mailer-Komponente nicht anbietet, müssten selbst implementiert werden. Es ist mit Verzögerungen zu rechnen	Abklärung des Funktionsumfangs der möglichen Mailer-Komponenten. Besprechung der Evaluierung mit dem Industriepartner	10 h (1 AT)	50 h	5 %	2.5 h	Hoch
R02	Anforderungen werden vom Kunden nicht genügend genau kommuniziert	Die nicht erkannten Anforderungen können nicht erfüllt werden. Der Kunde ist unzufrieden und es kommt zu Verzögerungen, weil die nicht erkannten Anforderungen evtl. grössere Änderungen und Mehraufwand mit sich bringen und	Anforderungsspezifikation erstellen und mit Industriepartner / Betreuer besprechen und in späteren Phasen Prototypen zur Demonstration bilden	50 h (5 AT)	50 h	20 %	10 h	Hoch
R03	Datenverlust	Teile der Arbeit gehen verloren (Code, Dokumente, etc.)	Einsatz eines Versionsverwaltungs-Systems, Konfiguration eines Backupsystems, regelmässiges Backup (täglich) ab dem virtuellen Server auf ein anderes Speichermedium an	2 h (1 AT)	40 h	5 %	2 h	Gering

			einem anderen räumlichen ort					
R04	Qualitätsanforderungen bezüglich Performance können softwaremässig nicht eingehalten werden. Das heisst, die Mailer-Komponente schränkt die Performance ein (2 Mio. E-Mails pro Tag, 300‘000 E-Mails pro Stunde)	Qualitätsanforderungen können nicht eingehalten werden. Es muss eine alternative Mailer-Komponente evaluiert werden. Es ist mit Verzögerungen zu rechnen	Entwicklung eines Prototyps, welcher die Mailer-Komponente nutzt. Durchführen von Tests. Die Tests sollen die Möglichkeit der Erfüllung der Performance-Anforderungen mit der Mailer-Komponente beweisen	20 h (2 AT)	20 h	5 %	1 h	Mittel
R05	Qualitätsanforderungen bezüglich Performance können hardwaremässig (Disk-Geschwindigkeit) nicht eingehalten werden (2 Mio. E-Mails pro Tag, 300‘000 E-Mails pro Stunde)	Es muss entsprechend leistungsfähige Hardware angeschafft werden. Zusätzliche Kosten entstehen	Entwicklung eines Prototyps, welcher die Daten der zu versendenden E-Mail Nachrichten in die Datenbank schreibt und für die Versendung liest. Die Tests sollen die Möglichkeit der Erfüllung der Performance-Anforderungen auf der gegebenen Hardware beweisen	30 h (3 AT)	20 h	15 %	3 h	Mittel
R06	Zeitplan kann nicht eingehalten werden	Die Arbeiten verzögern sich und der Abgabetermin kann nicht eingehalten werden	Einplanung von Pufferzeiten und frühzeitige Abdeckung von Risiken	5 h (1 AT)	30 h	10 %	3 h	Mittel
R07	Performance kann nicht ausreichend oder genau getestet werden	Mögliche Performanceprobleme sind erst im Betrieb ersichtlich	Absprechen mit Industriepartner, dass Performancetests in einer realen Umgebung durchgeführt werden können	10 h (1 AT)	50 h	5 %	2.5 h	Gering
R08	Benutzeroberfläche der	Endbenutzer vermissen	Durchführen von	30 h (3 AT)	20 h	10 %	2 h	Hoch

	Webapplikation entspricht nicht der Arbeitsweise und Anforderungen der Endbenutzer	Funktionalitäten oder arbeiten nicht gern mit der Webapplikation, weil sie nicht der Arbeitsweise des Endbenutzers entspricht	Interviews (Contextual Inquiry), Usability Tests mit realen Endbenutzern sowie Entwicklung von Paper Prototypes. Besprechung der Designresultate mit der Usability Verantwortlichen von comparis.ch AG (Annette Sulzbacher)					
R09	Anforderungen ändern sich während der Projektlaufzeit	Anforderungen können nicht mehr beachtet werden, weil Projekt zu weit fortgeschritten ist	Arbeiten mittels inkrementellem Vorgehensmodells RUP. Anforderungsänderungen oder Ergänzungen sind kein Problem	10 h (1 AT)	30 h	50 %	15 h	Hoch
R10	NDR können zu wenig zuverlässig ausgewertet werden	NDR Modul kann zu wenig zum Adressqualität-Management beitragen	Analyse der verschiedenen NDR Nachrichtenformate und Evaluierung einer bestehenden NDR Lösung, welche evtl. ergänzt werden muss	50 h (5 AT)	30 h	20 %	6 h	Hoch
R11	Priorisierte E-Mails werden nicht zeitgemäss versendet	Zeitkritische E-Mails werden zu spät versendet.	Entwicklung von Prototypen, welche die zeitgemässe Versendung beweisen.	30 h (3 AT)	50 h	15 %	7.5 h	Hoch
R12	Algorithmus zur Bestimmung des Qualitätsindikators arbeitet zu restriktiv oder zu träge	E-Mail Adressen wechseln zu schnell resp. zu träge in den Zustand poor oder failed. Die Adressqualität leidet darunter	Entwicklung eines Regelwerkes, nach welchem der Algorithmus arbeitet. Absprechen des Regelwerkes mit dem Industriepartner	15 h (2 AT)	15 h	10 %	1.5 h	Mittel

R13	Die für uns neue Technologie Silverlight kommt für die Webapplikation zum Einsatz	Bei der Entwicklung der Webapplikation wird zu viel Zeit für die Einarbeitung benötigt. Zudem kann es zu Problemen kommen, weil Möglichkeiten und Einschränkungen der Technologie nicht klar sind	Frühzeitiges Einarbeiten in die Technologie Silverlight. Entwicklung von Prototypen.	20 h (2 AT)	30 h	20 %	6 h	Mittel
R14	Der Daemon Prozess (Mail Modul) läuft nicht stabil	Der Daemon Prozess stürzt ab oder arbeitet nur noch mit verminderter Performance	Frühzeitiges Langzeittesten von Prototypen in einer realen Umgebung	40 h (4 AT)	50 h	10 %	5 h	Mittel
R15	Das Datenbankmodell ist nicht ausreichend auf Performance ausgelegt	Die Performance kann nicht erreicht werden. Die Auslastung ist unnötig höher als bei einem optimalen Datenbankmodell. Das Datenbankmodell muss neu modelliert werden	Besprechung des Datenbankmodells mit dem Datenbankarchitekt von comparis.ch AG (Christian Tarnutzer)	4 h (1 AT)	50 h	10 %	5 h	Mittel
	Total Kosten in Arbeitspaketen enthalten			**326 h**				
	Total Rückstellungen						**72 h**	

Tabelle 93 - Risiken

Erläuterungen zur Tabelle
Die Massnahme beschreibt die konkreten Schritte, welche vorgenommen werden, um das Risiko zu beseitigen respektive zu minimieren. Die Spalten „Kosten der Massnahme“ und „Maximaler Schaden“ sind Schätzungen. Ebenso eine Schätzung ist die Eintrittswahrscheinlichkeit.
Der Wert der Spalte „Gewichteter Schaden“ berechnet sich gemäss folgender Formel: *Maximaler Schaden * Eintrittswahrscheinlichkeit = Gewichteter Schaden*.
Diese Zahl ist umso höher, je höher und gewichtiger das Risiko ist. Eine frühe Behandlung der Risiken mit hohem gewichtetem Schaden ist also sinnvoll.

13.5.2 Behandlungsplan

Die Risiken werden je nach gewichtetem Schaden in einer früheren oder späteren Iteration behandelt. Das Risiko wird durch konkrete Schritte, beschrieben in der Spalte Massnahme, minimiert.

ID	Behandlung Iteration	Behandelt
R01	Elaboration 1 (Dokumentation der Evaluation in Analyse, Besprechung der Evaluation in MS 3)	Ok
R02	Elaboration 1 (Erstellung einer Anforderungsspezifikation, Besprechung der Anforderungsspezifikation in MS 2)	Ok
R03	Inception (Einrichten einer Backuplösung)	Ok
R04	Elaboration 2	Ok
R05	Elaboration 2	Ok
R06	Inception	Ok
R07	Construction 1	Ok
R08	Construction 3	Ok
R09	Inception (Behandlung durch Projektplanung mittels RUP)	Ok
R10	Elaboration 1 (Abgedeckt mittels Komponenten Evaluation)	Ok
R11	Elaboration 2	Ok
R12	Construction 2	Ok
R13	Elaboration 2	Ok
R14	Construction 1	Ok
R15	Elaboration 1	Ok

Tabelle 94 - Behandlungsplan von Risiken

13.5.3 Dokumentation der Behandlung

Die Risiken werden in den oben genannten Phasen und Iterationen behandelt. Im Zeitplan wird die Behandlung eines Risikos als Arbeitspaket ausgewiesen. Nachfolgend eine Dokumentation der zur Behandlung von Risiken vorgenommenen Massnahmen.

13.5.3.1 Risiko R06

Behandelt von	Phase, Iteration	Datum der Dokumentation
Danny Meier **Micha Boller**	Incpetion	19.02.2009

Erstellen einer Risikoanalyse und Aufstellen eines Behandlungsplanes der Risiken. So wird die Behandlung der Risiken in eine sinnvolle Reihenfolge gebracht und die Behandlung der Risiken kann überwacht werden.

13.5.3.2 Risiko R03

Behandelt von	Phase, Iteration	Datum der Dokumentation
Danny Meier	Inception	20.02.2009

Aufsetzen eines virtuellen Servers. Installation und Konfiguration von SVN und Backupsystem (Batchscript, Syncback Software)

13.5.3.3 Risiko R09

Behandelt von	Phase, Iteration	Datum der Dokumentation
Danny Meier **Micha Boller**	Inception	20.02.2009

Erstellen eines Projektplanes, welcher die Planung nach RUP dokumentiert. In der Arbeits- und Zeitplanung wurden die RUP Iterationen miteinbezogen.

13.5.3.4 Risiko R02

Behandelt von	Phase, Iteration	Datum der Dokumentation
Danny Meier **Micha Boller**	Elaboration 1	06.03.09

Verfassung einer Anforderungsspezifikation und Besprechung der Anforderungsspezifikation mit Benedikt Unold, comparis.ch AG.

13.5.3.5 Risiko R01

Behandelt von	Phase, Iteration	Datum der Dokumentation
Danny Meier **Micha Boller**	Elaboration 1	13.03.09

Aufstellen eines Anforderungskataloges und Evaluierung der Eignung der Komponenten am Markt auf die Anforderungen. Durchführung und Dokumentation von Performancetests zur Überprüfung der Einhaltung der Qualitätsanforderungen.

13.5.3.6 Risiko R04

Behandelt von	Phase, Iteration	Datum der Dokumentation
Danny Meier	Elaboration 1	13.03.2009

Erstellen eines Prototyps, welcher die Zeit misst, um eine bestimmte Anzahl von E-Mails zu versenden. Dabei wurden beide Übergabemethoden für die Übergabe von zu versendende Mails zwischen Mail Komponente und SMTP Server (SMTP-Protokoll / Queue) berücksichtigt.

13.5.3.7 Risiko R10

Behandelt von	Phase, Iteration	Datum der Dokumentation
Danny Meier **Micha Boller**	Elaboration 2	15.03.2009

Aufstellen eines Anforderungskataloges. Analyse der verschiedenen NDR Komponenten am Markt. Analyse der Kategorisierung von Bounce Messages der verschiedenen NDR Komponenten. Evaluierung der Erkennungsgenauigkeit mittels Erkennungstests (Erkennungsgenauigkeit von Bounce Messages).

13.5.3.8 Risiko R15

Behandelt von	Phase, Iteration	Datum der Dokumentation
Danny Meier **Micha Boller**	Elaboration 2	16.03.2009

Das von uns entwickelte Datenmodell wurde mit Benedik Unold und Christian Tarnutzer von der Firma comparis.ch AG besprochen. Es wurden einige Änderungen vorgenommen (Z.B. Einführen von Redundanz damit eine bessere Performance erreicht werden kann, Beachtung von Namenskonventionen der comparis.ch AG)

13.5.3.9 Risiko R07

Behandelt von	Phase, Iteration	Datum der Dokumentation
Danny Meier **Micha Boller**	Construction 1	25.03.2009

Weil die Performanceanforderungen bereits auf einem performancemässig schwächeren System und in einem langsameren Netzwerk erreicht wurden, entfällt dieses Riskio. Die Systeme in der realen Umgebung sind wesentlich performanter. Die Anforderungen sollten also übertroffen werden.

13.5.3.10 Risiko R05

Behandelt von	Phase, Iteration	Datum der Dokumentation
Danny Meier **Micha Boller**	Elaboration 2	25.03.2009

Es wurde ein Prototyp entwickelt, welcher die Erreichung der Performanceanforderungen (Verarbeitete Mails pro Stunde resp. Sekunde) beweist.

13.5.3.11 Risiko R13

Behandelt von	Phase, Iteration	Datum der Dokumentation
Danny Meier **Micha Boller**	Elaboration 2	25.03.2009

Es wurde ein Architektur Prototyp entwickelt, welcher eine Silverlight Komponente besitzt. Die grundsätzlichen Entwicklungsvorgehensweisen wurden so erkannt. Zudem wurden Möglichkeiten für die Kommunikation zwischen Silverlight Applikation und Service Layer vom *Smtp Mailer* aufgezeigt. Ein Technologiestudium der Silverlight-Technologie hat einiges zum Verständnis beigetragen.

13.5.3.12 Risiko R11

Behandelt von	Phase, Iteration	Datum der Dokumentation
Danny Meier **Micha Boller**	Construction 2	28.04.2009

Die Angabe eines ScheduledDispatch (also geplanter Versandzeitpunkt, d.h. ‚nicht früher als') wird beim Versenden beachtet. Die Versandpriorisierung kann über die Zuweisung von Applikationscode und Mailtyp Tupel zu einer Priorität erreicht werden.

13.5.3.13 Risiko R12

Behandelt von	Phase, Iteration	Datum der Dokumentation
Danny Meier **Micha Boller**	Construction 2	03.05.2009

Es wurde ein Qualitätsindikator Algorithmus entwickelt und dieser mit Benedikt Unold an einer Meilensteinsitzung besprochen. Zudem wurde dieser Algorithmus mittels Strategy Pattern (Wikipedia, 2009) implementiert. Einen Austausch des Qualitätsindikators ist also einfach möglich, falls nun der implementierte Algorithmus sich nicht wunschgemäss verhält.

13.5.3.14 Risiko R08

Behandelt von	Phase, Iteration	Datum der Dokumentation
Danny Meier **Micha Boller**	Construction 3	14.05.2009

Es wurde mit einem realen Endbenutzer bei der comparis.ch AG ein Contextual Inquiry und Paper Prototype Tests durchgeführt. Die Erkenntnisse liessen wir in die Persona- und Szenariobeschreibung einfliessen sowie in die konkreten Low-Fi Prototypes, welche als Grundlage für das Design der Webapplikation dienten. Die Resultate wurden mit Annette Sulzbacher, der Usability-Verantwortlichen von comapris.ch AG besprochen.

13.5.3.15 Risiko R14

Behandelt von	Phase, Iteration	Datum der Dokumentation
Danny Meier **Micha Boller**	Construction 3	05.06.2009

Es wurden Langzeittests von der Dauer von 2 Tagen auf den Rechnern im Labor durchgeführt. Der Prozess (SendingEngine) lief stabil und es traten keine Exceptions auf. Auch wurde das Log Level hoch eingestellt und das Logfile beobachtet. Es sind keine Fehler aufgetreten.

13.6 Infrastruktur

13.6.1 Hardware

Im Rahmen der Bachelorarbeit wird folgende Hardware verwendet:

- Private Notebooks (Windows XP Professional)
- Virtual Server (Windows Server 2003), zur Verfügung gestellt durch die HSR

13.6.2 Software

Im Rahmen der Bachelorarbeit wird folgende Software verwendet:

Umgebung

- Microsoft Windows XP / Microsoft Windows Server 2003
- .NET Framework 3.5 SP1
- ASP.NET 3.5 SP1

Server

- Trac 0.11.3
- Subversion 1.5.5
- dotProject 2.1.2
- CruiseControl.NET
- MySQL 5.1

Entwicklung

- Microsoft Visual Studio 2008
- Microsoft SQL Server 2005
- JetBrains Resharper 4.1.933
- VisualSVN 1.6.1
- NUnit 2.4.8
- Apache log4net 1.2.10
- Red Gate SQL Data Generator 1.2.0.286
- NDepend 2.12.1 (Trial Version)

Dokumentation

- Microsoft Office 2007
- Sandcastle (Codedokumentation)
- Enterprise Architect 7.1

13.6.3 Backup

Daten des virtuellen Servers werden täglich in der Nacht vollständig gesichert. Die Datensicherung umfasst:

- Datenbanken (MySQL (für dotProject) und MSSQL)
- SVN Repository
- Dateien und Konfigurationen

Die Datensicherung wird auf einen externen Onlinespeicher vorgenommen, welcher dauerhaft zur Verfügung steht und über eine breitbandige Anbindung verfügt.

13.6.4 Kommunikation

Folgende Kommunikationsmittel werden eingesetzt um einen erfolgreichen Projektablauf zu unterstützen:

- Wiki (Trac, http://ba.danflash.com/trac)
- E-Mail
- Persönliche Besprechungen
- Skype

13.6.5 Räumlichkeiten

Der grösste Teil der Projektarbeit findet in dem durch die HSR zugewiesen Bachelorarbeitszimmer (1.258) statt.

Besprechungen mit dem Projektbetreuer finden in den Räumlichkeiten des Instituts für Software (6.108) statt. Jene mit dem Industriepartner am Firmensitz der comparis.ch AG in Zürich (Stampfenbachstrasse 48).

13.6.6 Zugänge

Dem Projektbetreuer sowie dem Industriepartner wird durch die folgenden Zugangsdaten die Möglichkeit geboten, sich in der Projektinfrastruktur und Projektdokumentation umzusehen.

Wiki

Benutzername: Gast
Passwort: BaSmtpMailer
Url: https://ba.danflash.com/trac

Subversion

Benutzername: Gast
Passwort: BaSmtpMailer
Url: https://ba.danflash.com/svn/SmtpMailer

dotProject

Benutzername: Gast
Passwort: BaSmtpMailer
Url: http://ba.danflash.com/dotproject

13.7 Qualitätsmassnahmen

13.7.1 Projektfortschrittssitzungen

In einer wöchentlichen Besprechung mit dem Projektbetreuer sollen der aktuelle Stand der Arbeit sowie Probleme und Fragen besprochen werden. Durch diese Besprechungen kann die Qualität der entwickelten Software weiter gesteigert werden, weil so sich zusätzlich eine weitere Instanz mit der Arbeit und dem Projekt beschäftigt.

13.7.2 Sitzungsprotokolle

Von allen Sitzungen wird ein Sitzungsprotokoll erstellt. So werden automatisch Vorschläge, Anregungen, Kritik und Vereinbartes schriftlich festgehalten. Dies garantiert, dass nichts in Vergessenheit gerät und minimiert Missverständnisse.

13.7.3 Dokumentation

Es wird bei diesem Projekt grossen Wert darauf gelegt, dass die verschiedenen Dokumente aktuell gehalten werden. So kann sich jeder Projektteilnehmer auf den Inhalt in den Dokumenten verlassen.

Der Code wird zwar kommentiert, aber die Kommentare auf ein Minimum reduziert. Der Code soll selbsterklärend geschrieben werden.

13.7.4 Styleguides

Für die Dokumentation wird eine einheitliche Dokumentationsvorlage erstellt, damit sämtliche Dokumente die gleiche Aufmachung und Formatierung besitzen.

13.7.5 Naming Guidelines

Bei comparis.ch AG wird intern bei Entwicklungsprojekten der MSDN Naming Guideline (Microsoft Guidelines for Names, 2009) verwendet. Dieser Guideline ist auch für dieses Projekt von comparis.ch AG vorgeschrieben.

13.7.6 Todo Listen

Auf dem Wiki (Trac, http://ba.danflash.com/trac) für jedes Projektmitglied Todo Listen geführt. So dient das Wiki als Sammlung und Übersicht aller Todo‘s. Die Änderungen und Aktualisierungen sind so für alle Projektteilnehmer immer sofort ersichtlich. Zudem ist die gemeinsame Arbeit am Wiki unproblematischer als mit Word Dokumenten und SVN.

13.7.7 Projekt- und Zeitplan aktualisieren

Der Projektplan wird über die gesamte Dauer des Projektes auf aktuellem Stand gehalten. Auch der Zeitplan wird dauernd aktualisiert und ergänzt. Im Zeitplan werden die Arbeitszeiten von den Projektmitgliedern vor zu erfasst, damit man jederzeit die Kontrolle über Soll- und Ist-Arbeitszeiten besitzt.

13.7.8 Entwicklung von Prototypen

Von kritischen Softwareteilen werden Prototypen entwickelt. Diese können mit dem Kunden angeschaut und besprochen werden. Dies hilft, fehlende oder fehlerhafte Anforderungen resp. Funktionalitäten schon in einem frühen Entwicklungsstadium zu erkennen.

13.7.9 Reviews

13.7.9.1 Codereviews

Mit Codereviews werden verschiedene Ziele verfolgt. Zum einen werden die Qualität des Codes und die Einhaltung der Naming Guidelines überprüft. Zum anderen wird das Verständnis des Codes für das Projektmitglied erhöht, welches den Code nicht geschrieben hat. Die regelmässigen Reviews

erlauben es, Programmiertechniken auszutauschen und geben Motivation für die Erstellung von wartbarem und leicht verständlichem Code.

Wird eine Quellcodedatei zum Review freigegeben, so wird dieser Code vom anderen Projektmitglied überprüft.

Generell sollen bei einem Codereview folgende Kriterien überprüft werden:

- Existenz von Code Smells
- Einhaltung von Codierungsrichtlinen
- Verständlichkeit und Nachvollziehbarkeit des Codes
- Unit-Tests des Codes

13.7.9.2 Dokumentenreview

Nach der Fertigstellung eines Dokumentes wird von jedem Projektmitglied der Teil des Dokumentes überprüft, welcher nicht selber verfasst wurde. Vorzunehmende Änderungen werden gemeinsam besprochen.

Vor einer Abgabe eines Dokumentes ist das Dokument von jedem Projektmitglied zu prüfen. Bei grösseren Dokumenten können Reviews in Form eines Walkthroughs angeordnet werden. So wird das Dokument gemeinsam im Team durchgeschaut.

Nach einem Review wird das vollzogene Review in der Dokumentreview History eingetragen. Zudem wird im Sitzungsprotokoll festgehalten, welche Dokumente gereviewed wurden.

13.7.10 Tests

13.7.10.1 Unit-Tests

Wo sinnvoll wird nach dem Test-First Prinzip entwickelt. Das heisst, es werden zuerst die Unit-Tests zum zu entwickelnden Code erstellt und erst danach der eigentliche Code. Es darf kein Code eingecheckt werden, bei welchem die Unit-Tests fehlschlagen.

Die Unit-Tests werden mit dem Testframework NUnit erstellt.

13.7.10.2 System-Tests

Die System-Tests werden aufgrund von den Use Cases durchgeführt und protokolliert. Weil die Use Cases funktionale Anforderungen definieren, kann so überprüft werden, ob die Software den Anforderungen entspricht.

13.7.10.3 Usability-Tests

Usability-Tests werden durchgeführt, um sicherzustellen, dass die Benutzeroberfläche der Arbeitsweise und den Bedürfnissen der tatsächlichen Benutzer entspricht. Für die Usability-Tests müssen potenzielle Benutzer gefunden werden. Diese Benutzer geben ein Feedback zum Prototyp und der Benutzeroberfläche. Dieses Feedback wird ausgewertet und fliesst in nächste Iterationen ein.

13.7.10.4 Performance-Tests

Weil Performance ein kritischer Faktor ist für die zu erstellende Software, werden Performance-Tests durchgeführt und protokolliert. Dadurch ist es möglich, zu prüfen, ob die Software die Performancekriterien erfüllt.

13.7.10.5 Automatisierung

Mit einem Skript werden der Buildvorgang und die Unit-Tests automatisiert. Dies erleichtert eine von der Entwicklungsumgebung unabhängige Überprüfung des Codes ob er kompilier- und lauffähig ist.

14 Projektmonitoring

14.1 Soll – Ist Zeitvergleich

14.1.1 Überblick

Diese Bachelorarbeit nahm total 1106.25 Arbeitsstunden in Anspruch. Ausgehend von der offiziellen Soll-Zeit von 360 Stunden pro Person haben wir zu Zweit einen Arbeitsstundenüberschuss von 386.25 Stunden erarbeitet.

14.1.2 Rückblick

Zu dieser enormen Mehrleistung ist es gekommen, da wir die verschiedenen Administrationsoberflächen mit Silverlight gestalten mussten. Silverlight war für uns beide eine neue Technologie, in die wir uns erst einmal einarbeiten mussten. Zum einen handelte es sich um eine völlig andere Entwicklungsart als bei gängigen Webseiten mittels HTML und ASP.NET. Des Weiteren sind die vorhandenen Komponenten noch sehr Fehleranfällig und bieten oftmals nicht die Funktionalität und Flexibilität, wie man es von anderen .NET Projekten gewohnt ist. Dies resultierte in weiterer Arbeit für das Recherchieren und Umsetzen. Silverlight bietet mit den Styles und Templates eine gute Möglichkeit das Aussehen individuell zu beeinflussen, setzt aber sehr tiefe XAML Kenntnisse voraus, welche wir auch nach Abschluss dieser Arbeit nicht vorweisen können. Daher griffen wir auf Microsoft Expression Blend 2 zurück. Diese Software wiederum verursachte beim Erzeugen der Styles mehr Fehler als dass sie hilfreich war. Alles in allem ist unser Fazit zu Silverlight 2 folgendes: Es ist fehleranfällig, noch nicht ausgereift und kostete uns eine Menge Zeit.

Das Austesten und Sicherstellen der Performance war eines der Risiken, dass uns sehr viel Zeit gekostet hat. Das Hauptproblem bestand beim Abfragen der Daten aus der Datenbank, wenn darin bereits viele Datensätze vorhanden sind. Unsere Performancetests zeigten einen exponentiellen Anstieg der Abfragezeit mit zunehmender Datensatzanzahl. Durch viele Tests und Optimierungen konnten wir uns der erhofften Performance annähern. Unterstützung bot uns dabei auch immer wieder Benedikt Unold und Christian Tarnutzer von der comparis.ch AG.

14.1.3 Meilensteine

Alle unsere Meilensteine konnten wir termingerecht erreichen und die Resultate dem Betreuer so wie dem Industriepartner demonstrieren. Da wir der Situation entsprechend unser Arbeitspensum anpassen konnten, sind wir zu keiner Zeit in Rückstand oder Zeitnot geraten.

14.1.4 Auswertung

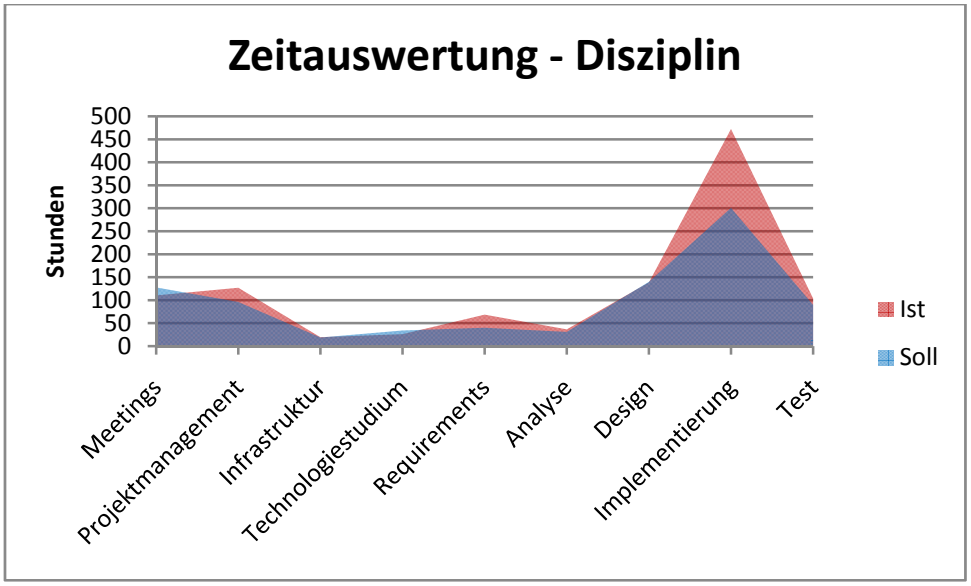


Abbildung 100 - Zeitauswertung nach Disziplinen

Dieses Diagramm (Abbildung 100 - Zeitauswertung nach Disziplinen) zeigt deutlich auf, dass die grössten Abweichungen im Bereich der Implementierungen zu Stande gekommen sind, wie dies bereits weiter oben ausgeführt wurde.

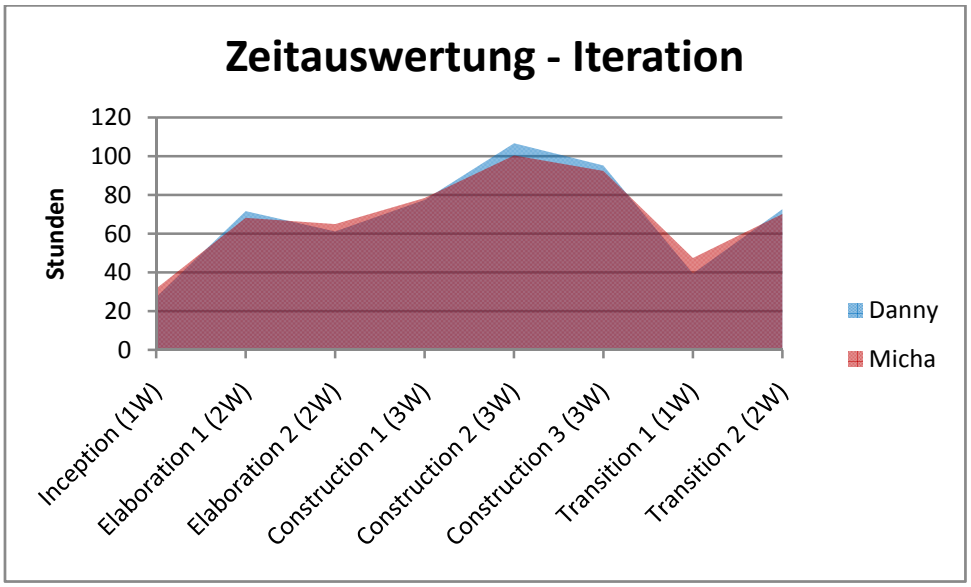


Abbildung 101 - Zeitauswertung nach Iterationen

Die Arbeitsteilung im Projektteam funktionierte bestens, die geleisteten Stunden sind sehr ausgeglichen auf die beiden Projektmitglieder verteilt (siehe Abbildung 101 - Zeitauswertung nach Iterationen). Beide waren bereit, bis an das Ende der Bachelorarbeit überdurchschnittlich viel Zeit in die Bachelorarbeit zu investieren.

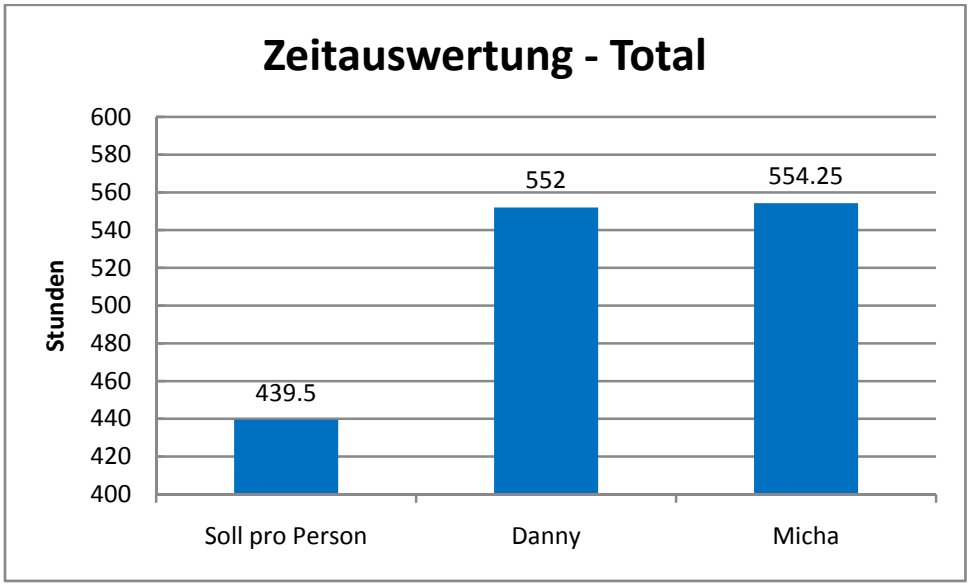


Abbildung 102 - Zeitauswertung Total

Die totale Arbeitszeit umfasst 1106.25 Stunden. Diese wurde wie bereits erwähnt mit sehr ausgeglichenem Arbeitsaufwand von Danny Meier und Micha Boller erreicht. Im Schnitt haben beide Parteien 114 Stunden Mehrarbeit geleistet als für die Bachelorarbeit offiziell vorgesehen ist Der Sollwert von 439.5 Stunden errechnet sich aus dem geplanten Stundenaufwand der Arbeitspakete. Eine Erreichung der Bachelorarbeit in den verlangten 360 Stunden pro Person war also nicht vorgesehen.

14.1.5 Fazit

Mit dem Abschluss der Construction 2 Iteration hatten wir die verlangten Arbeitsstunden (360 Stunden pro Person) bereits erreicht. Der deutliche Stundenüberschuss entstand durch das Erstellen der Silverlight 2 Webapplikation und den dazugehörigen Usability-Tests.

Diese Mehrleistung war grösstenteils nur möglich, da sich das Projektteam auch vor Wochenendarbeiten nicht abschrecken liess, um die jeweiligen Meilensteine zu erreichen.

Bei einer zukünftigen Arbeit ist darauf zu achten, dass bekannte Technologien zum Einsatz kommen, wenn vorhersehbar ist, dass die Projektzeit nahezu ausgeschöpft wird. Oder es sind Kürzungen im Umfang vorzunehmen.

14.2 Codestatistiken

Der Code wurde mit dem Tool NDepend analysiert. NDepend erlaubt es, verschiedenste Auswertungen und Codestatistiken zu generieren.

Nachfolgende Tabelle enthält verschiedenste Merkmale zum erstellten Code:

Merkmal	Zahl
Anzahl Codezeilen (ohne Unit-Tests)	8164
Anzahl Codezeilen Unit-Tests	3392
Anzahl Unit-Test Methoden	74
Anzahl Kommentarzeilen	4611
Anzahl Assemblies	18
Anzahl Typen	248
Anzahl Interfaces	20

Tabelle 95 - Codestatistik

15 Softwaredokumentation

15.1 Maintenance Anleitung

15.1.1 Einleitung

Diese Benutzeranleitung zeigt auf, welche Massnahmen getroffen werden müssen um das *Smtp Mailer* System einzusetzen. Dabei steht die Konfiguration der einzelnen Applikationen im Vordergrund. Eine Deployment Anleitung, wie die einzelnen Bestandteile auf eine produktive Umgebung migriert werden können, wird hier bewusst ausgelassen, da diese abhängig von der Infrastruktur und den internen Prozessen der comparis.ch AG ist.

15.1.2 Installation

15.1.2.1 Voraussetzungen

Allgemein

Folgende Voraussetzungen müssen in jedem Fall geschaffen werden, um den *Smtp Mailer* zu entwickeln, testen oder zu betreiben:

- Microsoft Windows XP SP3 / Microsoft Windows 2003 R2 oder höher
- Microsoft .NET Framework 3.5 SP1
- Microsoft Silverlight 2 Runtime
- Microsoft MS-SQL Server 2005 oder höher
- Tabellenstruktur und Initialdaten für die Datenbank (SmtpMailer.sql und SmtpMailerData.sql)
- Safabyte NetXtremeBounceFilter 2.0 Assemblies
- Quiksoft EasyMail Assemblies 3.0 inkl. S/MIME Plug-In
- Installierte Zertifikate im Zertifikatsspeicher von Windows für alle Absenderadressen, welche signiert werden sollen
- Distributed Transaction Coordinator (DTC) Dienst gestartet

Sollte der DTC-Dienst nicht gestartet sein, kann dieser unter Start -> Einstellungen -> Systemsteuerung -> Verwaltung -> Dienste gestartet werden. Ohne aktiven DTC sind die beiden Module SendingEngine und BounceEngine nicht lauffähig.

Entwicklung

Für die Entwicklung des *Smtp Mailers* sind folgende zusätzliche Komponenten notwendig:

- Microsoft Visual Studio 2008 SP1 oder höher
- Microsoft Silverlight 2 SDK
- *Microsoft Expression Blend 2 (optional)*

Produktiv

Für den produktiven Einsatz sind folgende zusätzliche Komponenten notwendig:

- WCF Service Hosting
  - Microsoft IIS (Internet Information Server) 5.1 oder höher
  - Microsoft WAS (Windows Activation Services)
  - Eigenständiges Hosting mittels Konsolen oder Dienstanwendung
- Webseite
  - Microsoft IIS (Internet Information Server) 5.1 oder höher

15.1.2.2 Datenbank

Die Datenbank wird von allen Modulen des *Smtp Mailer* Systems über den Verbindungsnamen ***CONN_SMTP_MAILER_DB*** resp. ***CONN_SMTP_MAILER_DB_TEST*** bei UnitTests angesprochen. Dieser ist entsprechend in den Konfigurationsdateien *App.config*, *Web.config* oder *Machine.config* zu definieren. Genauere Informationen sind dem Unterabschnitt Datenbank im Abschnitt 15.1.2.3 Konfiguration auf Seite 211 zu entnehmen.

Damit eine sinnvolle Nutzung möglich ist, ist es absolut notwendig, neben der Tabellenstruktur *SmtpMailer.sql* (Struktur) auch die Initialdaten *SmtpMailerData.sql* (Struktur und Daten) in die Datenbank zu importieren. Diese enthält sämtliche Datensätze zu den Lookup-Tabellen, welche die verwendeten Enumerationen dokumentieren, sowie Basisdaten für das Erkennen und Zuordnen von Bounce-Nachrichten.

Bezugnehmend auf die Anforderungsspezifikation ist die Datenbankverbindung auf den Datenbanknamen ***SMTP_MAILER_DB*** resp. ***SMTP_MAILER_DB_TEST*** bei Unit-Tests vorkonfiguriert.

15.1.2.3 Konfiguration

Allgemein

Alle standortabhängigen Konfigurationen werden mittels einer Applikationskonfiguration in XML erfasst. Diese kann wahlweise in der *App.config* (Windowsapplikation) / *Web.config* (Webapplikation) erfolgen oder wenn die Konfiguration applikationsübergreifend erfolgen soll kann dies in der *Machine.config* erfolgen.

Auf folgende Vorlage beziehen sich alle weiteren Konfigurationsanweisungen:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <configSections>
    <section
      name="smtpmailer"
      type="Comparis.Framework.SmtpMailer.Common.Configuration.ConfigurationHandler,
            Comparis.Framework.SmtpMailer.Common.Configuration"
    />
  </configSections>

  <connectionStrings>
    <add
      name="Comparis.Framework.SmtpMailer.Data.Properties.Settings.CONN SMTP MAILER DB"
      connectionString="<connectionString>"
    />
  </connectionStrings>

  <smtpmailer type="Comparis.Framework.SmtpMailer.Common.Configuration.SmtpMailerConfig,
                    Comparis.Framework.SmtpMailer.Common.Configuration"
  />
</configuration>
```

Datenbank

Der Verbindungsnamen ***CONN_SMTP_MAILER_DB*** resp. ***CONN_SMTP_MAILER_DB_TEST*** ist in der Applikationskonfiguration zu definieren und mit dem effektiven Connectionstring zu versehen. Eine gute Übersicht dazu bietet http://www.connectionstrings.com.

Logging

Über das gesamte *Smtp Mailer* System hinweg wurde das Logging-Framework *log4net* verwendet. Genaue Konfigurationsanweisungen sind entsprechend der Projekt-Dokumentation unter http://logging.apache.org/log4net zu entnehmen.

Schnell und ohne grossen Konfigurationsaufwand kann jedoch das Log-Level je nach aktuellem Einsatzzweck verändert werden. Die Konfiguration ist in der Datei *log4net.config* zu tätigen.

Level	Beschreibung
ALL	Loggt alle Meldungen unabhängig vom Logging-Level
DEBUG	Loggt Meldungen vom Logging-Level DEBUG und alle schwerwiegenderen. Debug Meldungen geben die kleinsten Details über die aktuelle Tätigkeit des Systems aus – wann wurde etwas in der Datenbank gemacht, wann wurde welcher Wert gesetzt. Dieser Level sollte nur zum Debuggen verwendet werden
INFO	Loggt Meldungen vom Logging-Level INFO und alle schwerwiegenderen. Info Meldungen geben wissenswerte Informationen über das System aus – wann es gestartet wurde, wie viele Threads erzeugt wurden, welche Konfigurationen geladen wurden. Dieser Level kann verwendet werden um Informationen zum aktuellen Systemstatus zu erhalten
WARN	Loggt Meldungen vom Logging-Level WARN und alle schwerwiegenderen. Warn Meldungen geben wichtige Hinweise zu Konfigurationsfehlern und fehlerhaften Teilmodulen, die möglichst behoben werden sollten. Das System ist jedoch weiterhin lauffähig und behält seinen konsistenten Zustand bei. Jedoch ist es möglich, dass nicht mehr alle Operationen aufgrund der Konfigurationsfehler oder fehlerhaften Teilmodulen durchgeführt werden können
ERROR	Loggt Meldungen vom Logging-Level ERROR und alle schwerwiegenderen. Error Meldungen geben Fehler aus, die unverzüglich behandelt werden müssen. Wird dies nicht getan, ist ein Weiterarbeiten nicht möglich. Das System sollte in einem kontrollierten und konsistenten Zustand abstürzen
FATAL	Loggt alle Meldungen vom Logging-Level FATAL. Fatal Meldungen folgen immer auf einen unkontrollierten Absturz. Je nach Grund kann es möglich sein das System manuell in einen konsistenten Zustand zurückzusetzen.
OFF	Loggt keine Meldungen.

Tabelle 96 - Logging-Levels

```
<?xml version="1.0" encoding="utf-8" ?>
<log4net>

  <root>
    <level value="<logginglevel>" />
    <appender-ref ref="LogFileAppender" />
  </root>

  <appender name="LogFileAppender" type="log4net.Appender.FileAppender" >
    <file value="SmtpMailer.log" />
    <appendToFile value="true" />
    <lockingModel type="log4net.Appender.FileAppender+MinimalLock" />
    <layout type="log4net.Layout.PatternLayout">
      <conversionPattern value="%date [%thread] %-5level %logger - %message%newline" />
    </layout>
  </appender>

</log4net>
```

Nachricht versenden

Beim Versenden einer Nachricht über das Sending API wird ermittelt welche Priorität eine Nachricht hat.

Attribute die **fett** geschrieben sind, sind zwingend notwendig, *kursiv* geschriebene Attribute sind optional.

Nachfolgende Konfiguration wird zusätzlich zur Vorlage benötigt und ist innerhalb des Elements *<smtpmailer></smtpmailer>* zu beschreiben.

Element	Attribut	Beschreibung
priorityrules / rules (kann zur Laufzeit geändert werden)	**applicationcode**	Applikationscode oder * als Wildcard
	mailtype	Mailtyp oder * als Wildcard
	priority	*High* für hohe Priorität *Low* für tiefe Priorität

```
<priorityrules>
  <rule applicationcode="<applicationcode>"
        mailtype="<mailtype>"
        priority="<priority>"
  />
  <rule applicationcode="*"
        mailtype="*"
        priority="<priority>"
  />
</priorityrules>
```

AddressQualityEngine

Attribute die **fett** geschrieben sind, sind zwingend notwendig, *kursiv* geschriebene Attribute sind optional.

Nachfolgende Konfiguration wird zusätzlich zur Vorlage benötigt und ist innerhalb des Elements *<smtpmailer></smtpmailer>* zu beschreiben.

Element	Attribut	Beschreibung
addressquality	**age**	Mindestdauer in Tagen, die seit der letzten Berechnung des Qualitätsindikators vergangen sein muss, dass für eine Adresse der Qualitätsindikator neu berechnet wird
	sentoffset	Zeitspanne in Tagen, welche definiert, innerhalb welcher Zeit auf eine versendete Nachricht eine Bounce Nachricht zurückerhalten wird. Diese Zeitspanne wird bei der Berechnung des Qualitätsindikators berücksichtigt. Normalerweise erfolgt eine Bounce Nachricht innerhalb von 48 Stunden
	numberofmsg	Anzahl der Nachrichten die in die Berechnung des Qualitätsindikators für eine E-Mail Adresse miteinbezogen werden
	strategy	Strategie die für die Berechnung des Qualitätsindikators verwendet werden soll. Der voll qualifizierte Namen muss dabei angegeben werden (Namespace + Klasse)
addressquality / weightings / weighting	**indicator**	*Nicht ändern!*
	value	Einen positiven Zahlenwert (Ganzzahl) der angibt, wie stark eine Bounce Nachricht mit dieser Einstufung bewertet werden soll
addressquality / thresholds / threshold	**indicator**	*Nicht ändern!*
	value	Einen positiven Zahlenwert (Ganzzahl) der angibt, ab wann die nächst schlechtere Qualitätsstufe erreicht wird

Tabelle 97 - Konfigurationseigenschaften für die AddressQualityEngine

```
<smtpmailer type="Comparis.Framework.SmtpMailer.Common.Configuration.SmtpMailerConfig,
                  Comparis.Framework.SmtpMailer.Common.Configuration">

  <addressquality age="<age>"
                  sentoffset="<sentoffset>"
                  numberofmsg="<numerofmsg>"
                  strategy="<strategy>">

    <weightings>
      <weighting indicator="Ignore" value ="<value>" />
      <weighting indicator="Low" value ="<value>" />
      <weighting indicator="Medium" value ="<value>" />
      <weighting indicator="High" value ="<value>" />
    </weightings>

    <thresholds>
      <threshold indicator="Good" value="<value>" />
      <threshold indicator="Poor" value="<value>" />
      <threshold indicator="Failed" value="<value>" />
    </thresholds>

  </addressquality>

</smtpmailer>
```

BounceEngine

Attribute die **fett** geschrieben sind, sind zwingend notwendig, *kursiv* geschriebene Attribute sind optional.

Nachfolgende Konfiguration wird zusätzlich zur Vorlage benötigt und ist innerhalb des Elements *<smtpmailer></smtpmailer>* zu beschreiben.

Element	Attribut	Beschreibung
threadpool	**min**	Anzahl der Threads die im Threadpool auf Vorrat gehalten werden sollen
	max	Anzahl der Threads die maximal aktiv sein können
idletime (kann zur Laufzeit geändert werden)	**min**	Minimale Zeit die gewartet wird bis der nächste Versuch unternommen wird
	max	Maximale Zeit die gewartet wird bis der nächste Versuch unternommen wird
	increasingfactor	Gibt einen Faktor an, um den die aktuelle Wartezeit erhöht wird, bis das Maximum erreicht wird
bouncehandler (kann zur Laufzeit geändert werden)	**returnpath**	Definiert die Rücksendeadresse als Muster, wobei {0} als Platzhalter dient für die Guid. Dieses Muster muss übereinstimmen mit dem Return-Path im Abschnitt 0 SendingEngine auf Seite 215 z.B. *bounce-{0}@comparis.ch*
	pop3host	Hostname oder IP des POP3-Servers, welcher die Bounce Nachrichten enthält
	pop3port	Port des POP3-Servers, welcher die Bounce Nachrichten enthält
	pop3username	Benutzernamen des POP3-Servers, welcher die Bounce Nachrichten enthält
	pop3password	Passwort des POP3-Servers, welcher die Bounce Nachrichten enthält

licenses / license	**component**	*EasyMailSmtp* für die EasyMail Smtp Komponente *EasyMailSmime* für das EasyMail S/MIME Plug-In
	licensekey	Lizenzschlüssel für die angegebene Komponente

Tabelle 98 - Konfigurationseigenschaften für die BounceEngine

```
<smtpmailer type="Comparis.Framework.SmtpMailer.Common.Configuration.SmtpMailerConfig,
                  Comparis.Framework.SmtpMailer.Common.Configuration">

  <threadpool min="<min>" max="<max>"/>

  <idletime min="<min>" max="<max>" increasingfactor="<increasingfactor>"/>

  <bouncehandler returnpath="<returnpath>"
                 pop3host="<pop3host>"
                 pop3port="<pop3port>"
                 pop3username="<pop3username>"
                 pop3password="<pop3password>"
  />

  <licenses>
    <license component="<component>" licensekey="<licensekey>"/>
  </licenses>

</smtpmailer>
```

CleanupEngine

Die CleanupEngine benötigt keine zusätzliche Konfiguration. Die Daten aus der Vorlage im Abschnitt 0 Allgemein auf Seite 211 reichen aus.

Konfiguriert wird die CleanupEngine über die *SmtpMailerConfiguration* Webapplikation, indem die CleanupRules verwaltet werden.

SendingEngine

Attribute die **fett** geschrieben sind, sind zwingend notwendig, *kursiv* geschriebene Attribute sind optional.

Nachfolgende Konfiguration wird zusätzlich zur Vorlage benötigt und ist innerhalb des Elements *<smtpmailer></smtpmailer>* zu beschreiben.

Element	**Attribut**		**Beschreibung**
smtpservers / smtpserver	**priority**		*High* für hohe Priorität *Low* für tiefe Priorität
	deliverytype		*Smtp* für Auslieferung mittels SMTP Protokoll *Queue* für die Auslieferung in ein Verzeichnis (IMail Pickup)
	Smtp	**host**	Hostname oder IP des SMTP Servers
		port	Port des SMTP Servers
		useauth	Wenn eine Authentifizierung verlangt wird *1* sonst *0*
		useauth=1 *username*	Benutzername für die Authentifizierung
		useauth=1 *password*	Passwort für die Authentifizierung

	Queue	**pickuppath**	Verzeichnispfad des Pickup-Verzeichnisses des IMail SMTP Servers
whitelist	**active**		Wenn die Whitelist aktiv sein soll *1* sonst *0*
threadpool	**min**		Anzahl der Threads die im Threadpool auf Vorrat gehalten werden sollen
	max		Anzahl der Threads die maximal aktiv sein können
idletime (kann zur Laufzeit geändert werden)	**min**		Minimale Zeit die gewartet wird bis der nächste Versuch unternommen wird
	max		Maximale Zeit die gewartet wird bis der nächste Versuch unternommen wird
	increasingfactor		Gibt einen Faktor an, um den die aktuelle Warte Zeit erhöht wird, bis das Maximum erreicht wird
bouncehandler (kann zur Laufzeit geändert werden)	**returnpath**		Definiert die Rücksendeadresse als Muster, wobei {0} als Platzhalter dient für die Guid. Dieses Muster muss übereinstimmen mit dem Return-Path im Abschnitt 0 SendingEngine auf Seite 215 z.B. *bounce-{0}@comparis.ch*
licenses / lincense	**component**		*EasyMailSmtp* für die EasyMail Smtp Komponente *EasyMailSmime* für das EasyMail S/MIME Plug-In
	licensekey		Lizenzschlüssel für die angegebene Komponente

Tabelle 99 - Konfigurationseigenschaften für die SendingEngine

```
<smtpmailer type="Comparis.Framework.SmtpMailer.Common.Configuration.SmtpMailerConfig,
                  Comparis.Framework.SmtpMailer.Common.Configuration">

  <smtpservers>

    <smtpserver priority="<priority>"
                deliverytype="Smtp"
                host="<host>"
                port="<port>"
                useauth="0"
    />
    <smtpserver priority="<priority>"
                deliverytype="Smtp"
                host="<host>"
                port="<port>"
                useauth="1"
                username="<username>"
                password="<password>"
    />
    <smtpserver priority="<priority>"
                deliverytype="Queue"
                pickuppath="<pickuppath>"
    />

  </smtpservers>

  <priorityrules>
    <rule applicationcode="<applicationcode>"
          mailtype="<mailtype>"
          priority="<priority>"
    />
    <rule applicationcode="*"
          mailtype="*"
          priority="<priority>"
    />
  </priorityrules>

  <whitelist active="<active>"/>

  <threadpool min="<min>" max="<max>"/>

  <idletime min="<min>" max="<max>" increasingfactor="<increasingfactor>"/>

  <bouncehandler returnpath="<returnpath>"/>

  <licenses>
    <license component="component" licensekey="<licensekey>"/>
  </licenses>

</smtpmailer>
```

15.1.3 Administration

Alle über die Silverlight Webapplikation durchführbaren Konfigurationen werden direkt zur Laufzeit übernommen und angewendet.

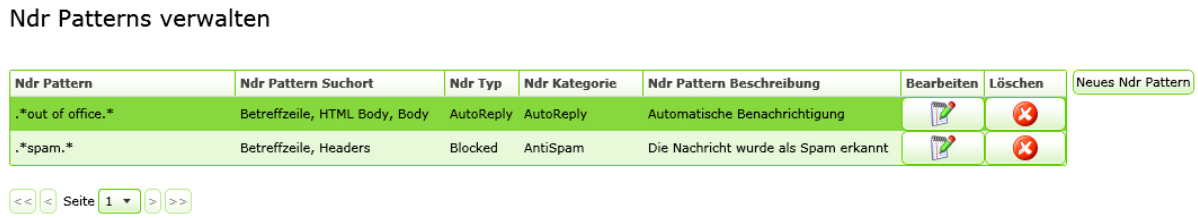
15.1.3.1 Ndr Patterns verwalten

Die selbst definierbaren Ndr Patterns erweitern die vordefinierten Muster der NetXtremeBounceFilter Komponente von Safabyte und helfen die Bounce Nachrichten genauer erkennen zu können.

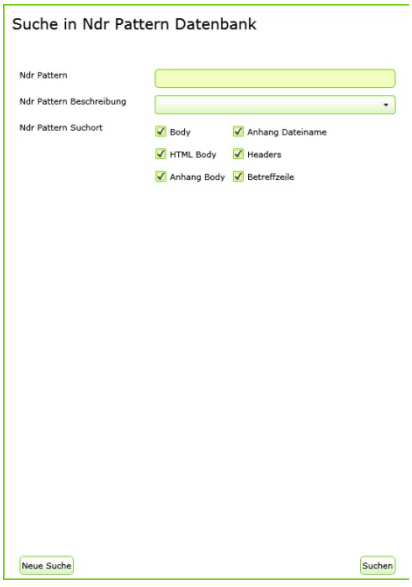
Benutzung

Übersicht

Listet alle selbst definierten Ndr Patterns auf und bietet die Möglichkeit, neue Muster zu erstellen, bestehende zu bearbeiten oder zu löschen. Auf der rechten Seite bietet die Suche die Möglichkeit gezielt nach Patterns zu suchen, um auch bei vielen Einträgen den Überblick zu bewahren.



Suchen



Die Suche erlaubt es gezielt nach einem Ndr Pattern (genauer Wortlaut oder Teil davon) zu suchen, nur nach einem Typ (Pattern Beschreibung) zu suchen oder nur den Suchort des Ndr Patterns anzugeben. Natürlich ist auch eine Kombination der drei Suchkriterien möglich, wobei diese als Und-Kriterien zusammen-gefasst werden.

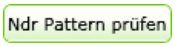
Erfassen / Bearbeiten

Durch Klicken der Schaltfläche 

resp.  erfolgt der Dialog um ein neues Ndr Pattern zu erfassen oder ein bestehendes zu bearbeiten.

Das Ndr Pattern ist als „regulären Ausdruck“ zu erfassen und umfasst die komplette Syntax. Zudem ist eine Beschreibung auszuwählen, welche den Bounce Nachrichten zugeordnet wird, die auf dieses Pattern passen. Als dritten obligatorischen

Parameter muss ausgewählt werden, wo in der Bounce Nachricht nach diesem Pattern gesucht werden soll.

Um die Auswirkungen des Musters gleich sichtbar zu machen, kann durch Betätigen der Schaltflache 

überprüft werden, welche und wie viele bisher unerkannte Bounce Nachrichten mit dem neuen Ndr Pattern erkannt werden.

Damit die unerkannten Bounce Nachrichten erneut verarbeitet werden, kann auf die Seite *Unerkannte Ndr Mails* gewechselt werden, welche im Abschnitt 15.1.3.2 Unerkannte Ndr Mails auf Seite 219 beschrieben wird.

Löschen

Um ein Eintrag zu löschen, muss die Schaltfläche  der entsprechenden Zeile betätigt werden. Dabei ändert sich die Schaltfläche von  nach  . Um ein versehentliches Löschen eines Eintrages zu vermeiden, muss die Löschaktion durch erneutes Klicken auf die Schaltfläche  bestätigt werden. Somit ist der Eintrag gelöscht.

Erweiterung und Wartung

Zurzeit sind alle möglichen Kombinationen aus Bounce-Typ und Bounce-Kategorie in der Datenbank als Basisdaten erfasst, welche die Safabyte NetXtremeBounceFilter Komponente in ihren vordefinierten Signaturen (Ndr Patterns) bereitstellt.

Zukünftige Versionen der Komponente können weitere Kombinationen umfassen. Die dazugekommenen Kombinationen müssten daher in der Datenbank in der Tabelle *NdrDescription* nacherfasst werden, damit diese in der Webapplikation auswählbar sind. Die Spalte *Indicator* gibt dabei an, wie stark diese Bounce Beschreibung in Bezug auf die Berechnung des Qualitätsindikators gewichtet werden soll. Die Spalte *Visible* gibt an, ob dieser Eintrag in der Silverlight Webapplikation beim Erstellen und Bearbeiten von Ndr Patterns ersichtlich sein soll.

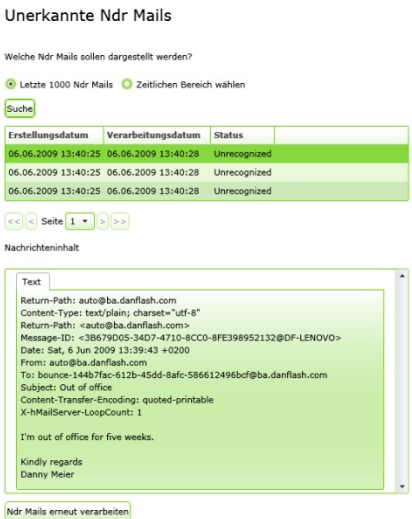
Indicator

Wert	Beschreibung
0	Wert wird ignoriert
1	Tief
2	Mittel
3	Hoch

Visible

Wert	Beschreibung
0	Nicht Sichtbar
1	Sichtbar

15.1.3.2 Unerkannte Ndr Mails



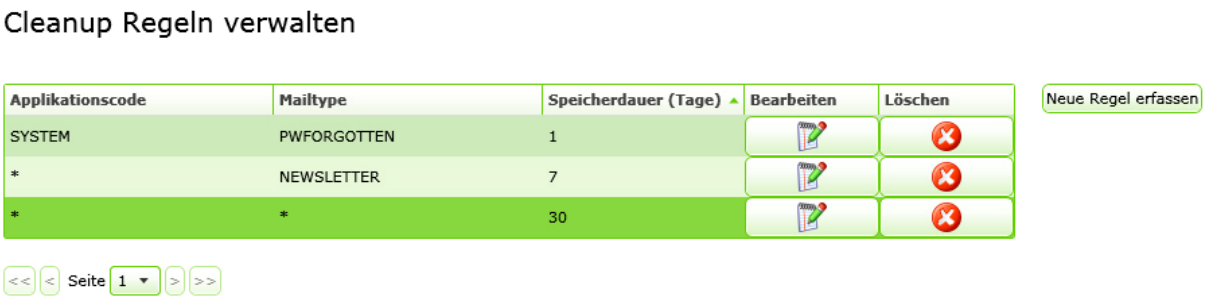
Auf dieser Seite können unerkannte Ndr Mails (Bounce Nachrichten) zurückgesetzt werden, um sie nochmals durch den Erkennungsalgorithmus abarbeiten zu lassen. Dies ist z.B. interessant, wenn ein eigenes Ndr Pattern erfasst wurde. Die Auswahl, welche Nachrichten zurückgesetzt werden sollen, kann entweder über einen Datumsbereich eingeschränkt werden oder es können die letzten 1000 Ndr Mails (bezogen auf das Erstellungsdatum) verarbeitet werden.

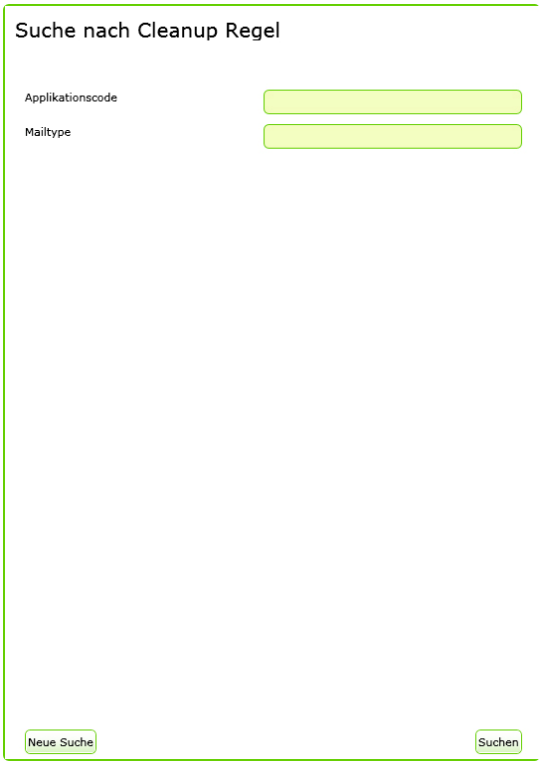
15.1.3.3 Cleanup Rules verwalten

Benutzung

Übersicht

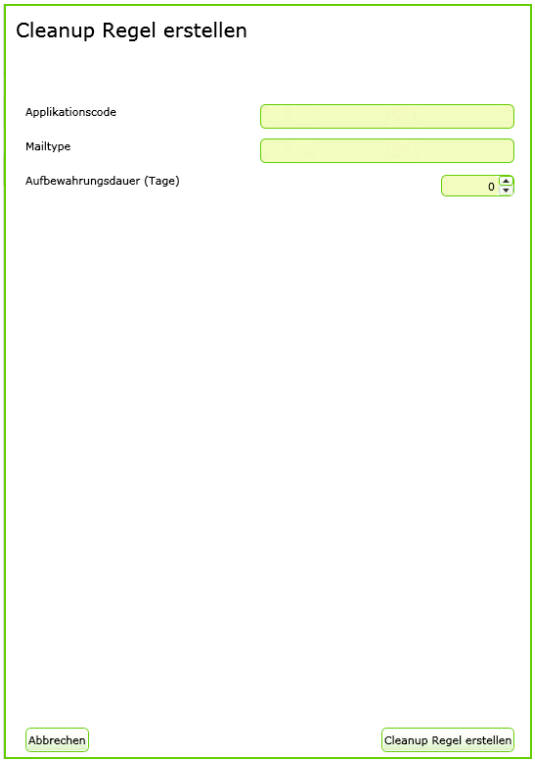
Listet alle selbst erfassten Cleanup Rules auf und bietet die Möglichkeit, neue Regeln zu erstellen, bestehende zu bearbeiten oder zu löschen. Auf der rechten Seite bietet die Suche die Möglichkeit gezielt nach Regeln zu suchen um auch bei vielen Einträgen den Überblick zu bewahren.





15.1.3.4 Suchen

Die Suche erlaubt es nach Applikationscode und Mailtyp zu suchen. Die Suchkriterien werden als Und-Bedingungen behandelt.



Erfassen / Bearbeiten

Beim Erfassen oder Bearbeiten von Einträgen sind sowohl der Applikationscode wie auch der Mailtyp auszufüllen. Für beide Felder ist jedoch die Wildcard * erlaubt.

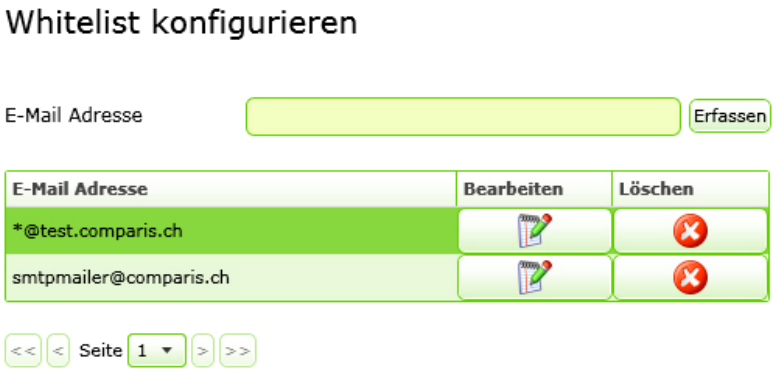
Löschen

Um ein Eintrag zu löschen, muss die Schaltfläche  der entsprechenden Zeile betätigt werden. Dabei ändert sich die Schaltfläche von  nach  . Um ein versehentliches Löschen eines Eintrages zu vermeiden, muss die Löschaktion durch erneutes Klicken auf die Schaltfläche  bestätigt werden. Somit ist der Eintrag gelöscht.

15.1.3.5 White- /Blacklist konfigurieren

Benutzung

Übersicht



Hinweis: Die Konfiguration der Blacklist gestaltet sich analog zur Konfiguration der Whitelist.

Suchen

Durch eingeben eines Zeichenstrings in die Textbox, wird die Tabelle automatisch gefiltert und zeigt nur noch jene Einträge an, welche mit der eingegeben E-Mail Adresse übereinstimmen.

Erfassen

Nachdem eine E-Mail Adresse in das Textfeld eingegeben wurde (als Wildcard dient der *) kann durch einfaches betätigen der Schaltfläche  ein neuer Eintrag hinzugefügt werden.

Bearbeiten

Im Bearbeiten-Dialog kann die eingegebene E-Mail Adresse editiert und anschliessend gespeichert werden.

Löschen

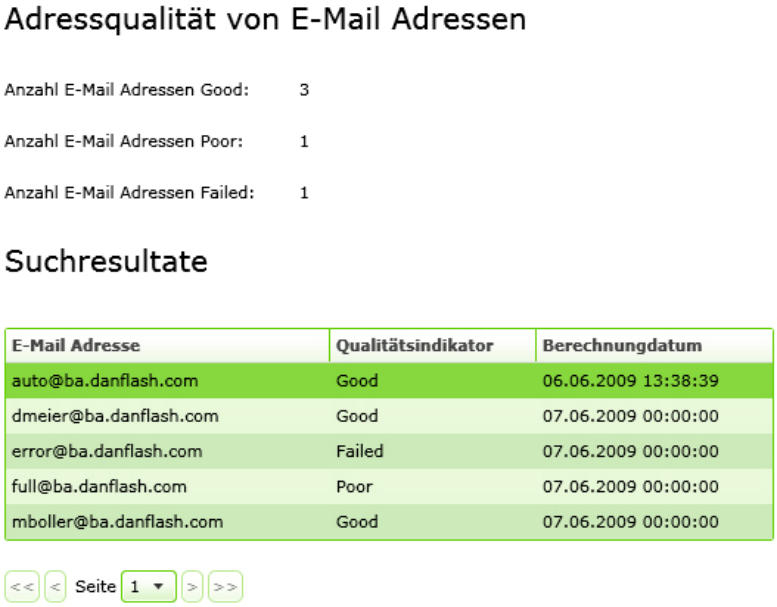
Um ein Eintrag zu löschen, muss die Schaltfläche  der entsprechenden Zeile betätigt werden. Dabei ändert sich die Schaltfläche von  nach  . Um ein versehentliches Löschen eines Eintrages zu vermeiden, muss die Löschaktion durch erneutes Klicken auf die Schaltfläche  bestätigt werden. Somit ist der Eintrag gelöscht.

15.1.4 Auswertung

15.1.4.1 Adressqualität von E-Mail Adressen abfragen

Benutzung

Übersicht



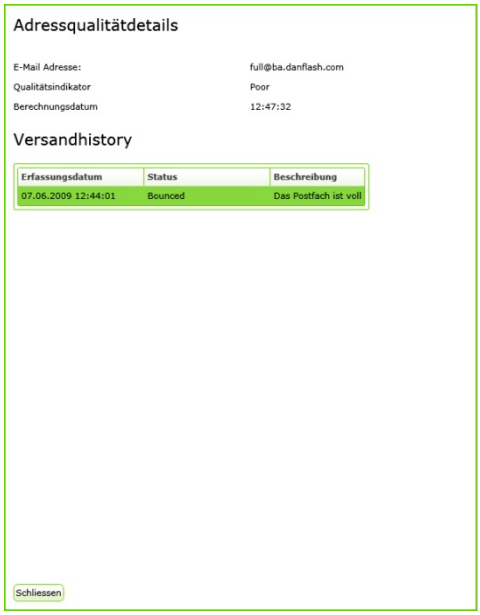
Zeigt auf einer übersichtlichen Seite wie viele E-Mail Adressen einen guten, schlechten oder fehlerhaften Qualitätsindikator haben.

Die Tabelle zeigt maximal 1000 Einträge an. Für weitere Einträge muss spezifischer nach einem Teil der E-Mail Adresse oder dem Qualitätsindikator gesucht werden.

Details

Wenn mit der Maus auf eine Ergebniszeile geklickt wird, öffnet sich die Detailansicht zu dieser E-Mail Adresse. In der Detailansicht ist die Versandhistory für die entsprechende E-Mail Adresse sichtbar.

Anhand dieser Angaben berechnet sich der Qualitätsindikator für die Adresse. So kann die Berechnung des Qualitätsindikators nachvollzogen werden.



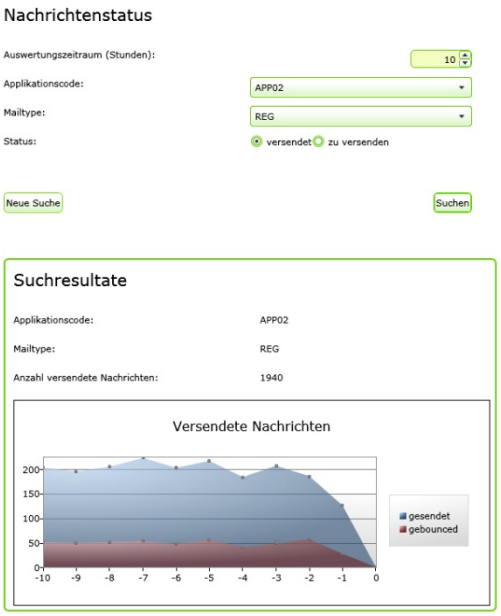
Suchen

Die Suche erlaubt es nach einem Teil der E-Mail Adresse und nach dem Qualitätsindikator zu suchen. Die beiden Kriterien werden als Und-Bedingungen ausgewertet.

15.1.4.2 Nachrichtenstatus

Benutzung

Übersicht



Diese Seite erlaubt es, eine Auswertung über den E-Mail Verkehr zu erhalten, welcher über das *Smtp Mailer* System läuft.

Nach Eingabe des Auswertungszeitraums in Stunden, dem Applikationscode und dem Mailtyp kann gewählt werden, ob die zu versendenden Nachrichten oder die bereits gesendeten Nachrichten statistisch dargestellt werden sollen.

Bei den zu versendenden Nachrichten gilt die angegebene Stundenzahl (Auswertungszeitraum) als Anzahl Stunden in die Zukunft, welche ausgewertet werden sollen. Bei den versendeten Nachrichten gilt die angegebene Stundenzahl als Anzahl Stunden in die Vergangenheit, welche ausgewertet werden sollen.

Bei der Auswertung der gesendeten Nachrichten ist ersichtlich, wie viele Bounce Nachrichten (dunkel rot) auf das gesamte Versandvolumen (blau) erzeugt wurden.

15.2 Quikcard für Kundendienst

Quickcard Kundendienst

Nachricht suchen

Übersicht

Nachrichtenliste

Die *Nachrichtenliste* zeigt die letzten 1000 E-Mail Nachrichten an, sortiert nach dem geplanten Versanddatum. Dabei zeigt das Icon *Status* an, ob der Versand stattgefunden hat und das Icon *Erfolg*, ob alle Empfänger die Nachricht fehlerfrei erhalten haben. signalisiert, dass es bei mindestens einem Empfänger beim Versand Probleme gab. Ist jedoch das Icon vorhanden, dann war der Versand an alle Empfänger erfolgreich.

Details

Nachdem mit der linken Maustaste eine Zeile in der Tabelle angeklickt wurde, erscheint eine Detailübersicht der gewählten Nachricht mit der genauen Versandgeschichte aller Empfänger und wieso es unter Umständen zu einem Fehler beim Versand kam.

Suche

Die Resultate können auf E-Mail Adressen, Empfängernamen, den Betreff und einen Datumsbereich (bezogen auf das geplante Versanddatum) eingeschränkt werden.

Um wieder die aktuellsten 1000 Nachrichten in der Tabelle zu sehen, ist in der Suche die Schaltfläche Neue Suche zu betätigen und anschliessend Suchen.

Abbildung 103 - Quickcard Nachricht suchen

Quickcard Kundendienst

Blacklist konfigurieren

Übersicht

Die *Blacklist* enthält alle E-Mail Adressen (oder Teile davon) die absolut keine E-Mails mehr von comparis.ch AG wünschen.

Suche

Durch Eingeben einer Zeichenfolge in die Textbox *E-Mail Adresse* wird die Tabelle automatisch gefiltert und zeigt nur noch jene Einträge an, welche mit der eingegeben E-Mail Adresse übereinstimmen.

Erfassen

Nachdem eine E-Mail Adresse in das Textfeld eingegeben wurde (als Wildcard dient der *) kann durch einfaches Bestätigen der Schaltfläche Erfassen die eingegebene E-Mail Adresse der Blacklist hinzugefügt werden.

Bearbeiten

Durch Klicken auf das Bearbeiten-Symbol gelangt man in den Bearbeiten-Dialog. Dort kann die gewählte E-Mail Adresse editiert und anschliessend gespeichert werden.

Löschen

Um ein Eintrag zu löschen, muss die Schaltfläche der entsprechenden Zeile betätigt werden. Dabei ändert sich die Schaltfläche von nach . Um ein versehentliches Löschen eines Eintrages zu vermeiden, muss die Löschaktion durch erneutes Klicken auf die Schaltfläche bestätigt werden. Somit ist der Eintrag gelöscht.

Abbildung 104 - Quickcard Blacklist konfigurieren

Abbildungsverzeichnis

Tabellenverzeichnis

Quellen- und Literaturverzeichnis

Advanced Intellect ListNanny. (2009). Von http://www.listnanny.net/help/ abgerufen

Anforderungen SMTP Mailer für comparis.ch. (2009).

Crispin, L. (2006). *IEEE*. Abgerufen am 2. Mai 2009 von http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4012627&isnumber=4012608

Dean. (11. Februar 2009). *ButtonChrome*. Abgerufen am 30. Mai 2009 von http://blog.buttonchrome.co.uk/post/Silverlight-DataGrid-e28093-A-Simple-Pager-Control.aspx

DTC MSDN. (2009). Abgerufen am 7. Mai 2009 von http://msdn.microsoft.com/en-us/library/ms229976(VS.80).aspx

Janzen, D. (2005). *ACM - Association for Computing Machinery*. Abgerufen am 2. Mai 2009 von http://doi.acm.org/10.1145/1094855.1094954

Ji-Ye Mao, K. V. (2002). *ACM - Association for Computing Machinery*. Abgerufen am 29. April 2009 von http://doi.acm.org/10.1145/503376.503460

Larman, C. (2005). *Applying UML and patterns.* Pearson Education.

Microsoft Guidelines for Names. (2009). Abgerufen am 26. Februar 2009 von http://msdn.microsoft.com/en-us/library/ms229002.aspx

QuikSoft BounceBuster. (2009). Von http://www.quiksoft.com/bouncebuster/docs/ abgerufen

Safabyte NetXtremeBounceFilter. (2009). Von http://safabyte.com/documentation/bouncefilter/AtAGlance.html abgerufen

Wikipedia. (2009). Abgerufen am 14. Mai 2009 von http://en.wikipedia.org/wiki/Strategy_pattern

Wikipedia. (2009). Abgerufen am 14. Mai 2009 von http://en.wikipedia.org/wiki/Data_Transfer_Object

Wikipedia. (2009). Abgerufen am 14. Mai 2009 von http://en.wikipedia.org/wiki/Adapter_pattern

A Inhalt der abgegebenen CD

Alle für diese Arbeit relevanten Daten befinden sich auf der CD-ROM.

Ordner	Beschreibung
/Dokumentation	Enthält die Dokumentation der Bachlelorarbeit
/Diagramme	UML-Diagramme, LowFi-Prototypen
/Originale	Alle selbst erstellten Dokumente im .docx Format
/Präsentationen	Präsentation, die während der Bachelorarbeit erstellt wurde
/Evaluation	Daten und Informatoinen die für die Evaluation wichtig waren
/Test E-Mails	22 Bounce Nachrichten
/Information	Richtlinien, Aufgabenstellung, keine selbst erarbeiteten Dokumente
/Smtp Mailer	Smtp Mailer System
/dll	Benötigte und referenzierte Klassenbibliotheken
/doc	Generiete Code Dokumentation
/ico	Benutze Icons in den Formaten 24x24 und 32x32
/prot	Erstellte Prototypen
/ArchitecturePrototype	Architektur Prototyp
/R04Prototype	Risiko 04 Prototyp
/R05Prototype	Risiko 05 Prototyp
/sql	Sql Skripte (Struktur und Daten)
/src	Quellcode
/Smtp Mailer	Quellcode zu sämtlichen Modulenb

Tabelle 100 - CD Inhalt

B Glossar

Begriff, Abkürzung	Erklärung
NDR	Non Delivery Report. Eine NDR Nachricht wird von einem Mailserver generiert, wenn eine E-Mail nicht zustellbar ist. Die NDR wird an den Absender der E-Mail zurück geschickt.
Bounce-Handling	Das Behandeln von Bounce Nachrichten resp. NDR Nachrichten. Dazu gehören die Auswertung der NDR und das Vornehmen von entsprechenden Massnahmen im Adressqualitätsmanagement.
RUP	Rational Unified Process. Es handelt sich um ein iteratives Vorgehensmodell in der Softwareentwicklung welches in Phasen und Iterationen abgehandelt wird. RUP ist ein kommerzielles Produkt von IBM.
SVN	Steht als Abkürzung für Subversion. Bei Subversion handelt es sich um eine Versionsverwaltung, welche als Open-Source-Software zur Verfügung gestellt wird.
Code Smell	Code Smells sind bekannte Unschönheiten oder Fehler im Aufbau des Codes. Code Smells weisen auf ein Problem hin und liefern eine Lösung zur Umarbeitung des Codes um den Code Smell zu beheben.
Blacklist	Die Blacklist enthält Empfängeradressen, welche keine E-Mails von comparis.ch erhalten dürfen. Darauf werden E-Mail Adressen gesetzt, welche z.B. einen Newsletter abbestellt haben. Weil aber die E-Mail Adresse noch irgendwo im CRM sein könnte, wird die Adresse als zusätzliche Sicherheit auf die Blacklist gesetzt. Die Blacklist muss zur Laufzeit konfigurierbar sein.
Whitelist	Die Whitelist wird für Test- und Debuggingzwecke verwendet. Dabei gehen E-Mails nur an E-Mail Adressen, welche sich auf der Whitelist befinden. So sind beim E-Mail Versand nur gewisse E-Mail Domains erreichbar.
ISO 9126	Bei der ISO-Norm 9126 handelt es sich um ein Modell, um die Softwarequalität sicherzustellen.
MTA	Mail Transfer Agent. Ein MTA ist ein SMTP Server, welcher für das Versenden von E-Mail Nachrichten zuständig ist.
ACM	Association for Computing Machinery. Veröffentlicht informationstechnologische Publikationen und Zeitschriften.
Regulärer Ausdruck	Ein regulärer Ausdruck dient der Beschreibung von gewissen Mengen von Zeichenketten. Reguläre Ausdrücke kommen in Bezug auf den *Smtp Mailer* bei den NdrPatterns zum Einsatz.
Assembly	Bei einem Assembly handelt es sich um eine Sammlung von übersetzter .NET Codedateien in einer Art Container.
Guid	Abkürzung für Globally Unique Identifier. Es handelt sich dabei um eine eindeutige Zahl in hexadezimaler Form mit 16 Bytes. Jede Message im *Smtp Mailer* besitzt eine eindeutige Guid. Diese wird z.B. als Teil der Returnpath Adresse eingesetzt.
Contextual Inquiry	Beim Contextual Inquiry handelt es sich um eine User Centred Design Methode. Dabei handelt es sich um eine Benutzerbeobachtung am Arbeitsplatz des Benutzers. Dabei sind Befragungen bei Unklarheiten erlaubt. Das Ziel ist es, den Benutzer und seine Arbeitsweise sowie Arbeitsabläufe zu verstehen.
V-Modell	Beim V-Modell handelt es sich um eine Projektmanagement Methode. Speziell ist, dass zu einem Arbeitsschritt immer gleich das erwartete Ergebnis dokumentiert wird
Test Driven Development	Unit-Tests werden vor der zu testenden Komponente erstellt

Tabelle 101 - Glossar

C Fragenkatalog

44	Besprochen am 03.06.2009
Adressat	Prof. Dr. Markus Stolze, HSR
Frage	Gibt es eine Vorlage oder Beispiele, wie das Management Summary gestaltet werden muss?
Antwort	Prof. Dr. Markus Stolze teilt uns den Link zur Webseite mit.

43	Besprochen am 03.06.2009
Adressat	Prof. Dr. Markus Stolze, HSR
Frage	Darf die Aufgabenstellung elektronisch kopiert werden und danach von allen Parteien neu Unterschrieben, damit das Layout durchgängig ist?
Antwort	Nein, die abgegebene Aufgabenstellung ist im Original zu verwenden. Entweder als gescanntes Bild auf eine Seite mit unserem Layout oder direkt in die gedruckte Dokumentation legen.

42	Besprochen am 03.06.2009
Adressat	Prof. Dr. Markus Stolze, HSR
Frage	Wo soll die Methodenreflexion und das Silverlightlog dokumentiert werden?
Antwort	Der von uns vorgeschlagene Ort „Resultate“ geht in Ordnung.

41	Besprochen am 26.05.2009
Adressat	Benedikt Unold, comparis.ch AG
Frage	Wie soll die Abgabe der Software erfolgen?
Antwort	Als build-fähige VisualStudio-Solution.

40	Besprochen am 26.05.2009
Adressat	Prof. Dr. Markus Stolze, HSR
Frage	Wie viele Dokumentation sollen abgegeben werden und in welcher Form?
Antwort	Drei gedruckte Versionen für den Betreuer, Experte und Gegenleser sowie eine elektronische Version an Industriepartner.

39	Besprochen am 13.05.2009
Adressat	Prof. Dr. Markus Stolze, HSR
Frage	Wie sollen in der Tabelle vom Screen 'Nachrichten suchen' mehrere Empfänger dargestellt werden? Was soll beim Sortieren passieren?
Antwort	Ist nicht sortierbar. Angezeigt wird der erste Empfänger und in Klammern wie viele weitere Empfänger es gibt.

38	Besprochen am 06.05.2009
Adressat	Benedikt Unold, comparis.ch AG
Frage	Herr Stolze hat einen Vorschlag für die Qualitätssicherung von eigenen Signaturen gemacht. Soll diese Idee umgesetzt werden?
Antwort	Nein. Aber was sicher implementiert werden soll ist, dass beim Erfassen der Signatur diese Signatur auf die Top 100 Nachrichten angewendet wird. Nachrichten, auf welche die Signatur passt werden dargestellt.

37	Besprochen am 06.05.2009
Adressat	Benedikt Unold, comparis.ch AG
Frage	Welche Concurrency-Control Strategie soll umgesetzt werden im DAL bezüglich Webapplikation?
Antwort	Optimistic Concurrency. Dies aber nur bei der Kundendienst Webapplikation. Die Admin Webapplikation ist eine Single User Webapplikation.

36	Besprochen am 06.05.2009
Adressat	Benedikt Unold, comparis.ch AG
Frage	Es gibt auf der Webapplikation verschiedene Benutzerrollen, welche verschiedene Funktionalität benötigt (Kundendienst, Admin, Entwickler). Müssen diese Rollen getrennt werden können?
Antwort	Es sollen drei getrennte Webapplikationen implementiert werden, welche über verschiedene URL's erreichbar sind (Tracking, Reporting, Administration)

35	Besprochen am 06.05.2009
Adressat	Benedikt Unold, comparis.ch AG
Frage	Wie weiss man, dass eine E-Mail Nachricht tatsächlich korrekt verschickt wurde? (In Bezug auf die Bestimmung des Qualitätsindikators)
Antwort	Unsere Strategie ist es, Nachrichten die in den letzten 48 Stunden versendet wurden, nicht für die Berechnung zu berücksichtigen

34	Besprochen am 06.05.2009
Adressat	Benedikt Unold, comparis.ch AG
Frage	Was soll passieren, wenn beim Versenden einer E-Mail Nachricht ein Fehler auftritt? Momentan wird für den Recipient, bei welchem der Fehler auftrat, Successful Sent = false und ein Dispatching State mit Dispatch State = failed in die Datenbank eingetragen.
Antwort	Das ist gut so.

33	Besprochen am 22.04.2009
Adressat	Prof. Dr. Markus Stolze, HSR
Frage	Herr Stolze hat gesagt, unsere Dokumentation muss mindestens ein Buch, ein IEEE Paper und ein ACM referenzieren. Was ist ein ACM?
Antwort	Es ist ein Association for Computing Machinery (ACM) Standard gemeint (www.acm.org)

32	Besprochen am 16.04.2009
Adressat	Benedikt Unold, comparis.ch AG
Frage	Sollen die in der Config angegebenen Strings (Applikationscode, Mailtyp, etc.) case-sensitiv behandelt werden?
Antwort	Ja. Alle Strings, welche intern verglichen werden immer zuerst mit der Methode ToLower() in Kleinbuchstaben wandeln.

31	Besprochen am 16.04.2009
Adressat	Benedikt Unold, comparis.ch AG
Frage	Wie soll reagiert werden, wenn eine E-Mail signiert versendet werden soll, sich aber kein passendes Zertifikat im Zertifikatspeicher finden lässt?
Antwort	Es soll eine Exception geworfen werden.

30	Besprochen am 31.03.2009
Adressat	Benedikt Unold, comparis.ch AG
Frage	Was soll mit zu versendenden E-Mail Nachrichten passieren, welche aufgrund eines Fehlers im *Smtp Mailer* nicht versendet werden konnten (Wenn z.B. Mailserver direkt auf eine zu versendende E-Mail mit einem Fehlercode antwortet)?
Antwort	Dies muss in der Message oder im Recipient abgelegt werden, je nachdem wie die Mail Komponente reagiert. Auf den Qualitätsindikator hat das keinen Einfluss, weil der SMTP Server der comparis.ch immer jegliche Nachricht entgegennimmt.

29	Besprochen am 31.03.2009
Adressat	Benedikt Unold, comparis.ch AG
Frage	Wie geschieht das Testen des Data Access Layers?
Antwort	Die sauberste und sicherste Variante ist es, vor jedem Unit Test eine Test Datenbank zu bereinigen, mit Testdaten befüllen, den Test durchführen und schliesslich die Daten prüfen ob sie den Erwartungen entsprechen.

28	Besprochen am 31.03.2009
Adressat	Benedikt Unold, comparis.ch AG
Frage	Wie gross ist die Wahrscheinlichkeit, dass das Logging Framework ausgetauscht werden muss? Muss eine vollständige Kapselung vom konkreten Logging Framework erfolgen?
Antwort	Das Logging Framework muss in einer zentralen Klasse gekapselt werden, damit der Logging Mechanismus ausgetauscht werden kann. Das Logging soll dann nicht aufzeigen, in welcher Klasse der Fehler aufgetreten ist sondern in welchem Projekt.

27	Besprochen am 11.03.2009
Adressat	Benedikt Unold, comparis.ch AG
Frage	Müssen die Fehlermeldungen und Meldungstexte nur in Deutsch geliefert werden oder müssen mehrere Sprachen unterstützt werden?
Antwort	Texte an Webbapplikationsbenutzer in Deutsch, Exception-Messages in Englisch

26	Besprochen am 11.03.2009
Adressat	Benedikt Unold, comparis.ch AG
Frage	Worauf bezieht sich die Archivierungszeit? Was ist, wenn die Nachricht für gewisse Empfänger noch nicht versendet werden konnte (z.B. Softbounce)?
Antwort	Auf komplett versendete Mails. Dazu gelten auch Mails welche mit einem Hard- oder Softbounce zurückgekommen sind.

25	Besprochen am 11.03.2009
Adressat	Benedikt Unold, comparis.ch AG
Frage	In welcher Einheit wird die Storage Duration angegeben (für CleanupRules)?
Antwort	In Tagen (int)

24	Besprochen am 11.03.2009
Adressat	Benedikt Unold, comparis.ch AG
Frage	Muss jeder Entwickler die App.config selbst erstellen mit den Zugangsdaten zum MSSQL Server und sonstigen Konfigurationen? Die Assemblies sind ja verteilt.
Antwort	Es muss einfach auf die Konfigurationsvariable CONN_SMTP_MAILER_DB verwiesen werden. Diese wird durch die globale Konfiguration mitgegeben

23	Besprochen am 11.03.2009
Adressat	Benedikt Unold, comparis.ch AG
Frage	Soll mit den LINQ Entities auf allen Layern garbeitet werden oder soll auf Service und BLL mit Business Entities gearbeitet werden und ein Mapping auf die LINQ Entitites im DAL stattfinden?
Antwort	Es soll nicht mit LINQ to SQL gearbeitet werden (besser Entity Framework wegen Performance). Es findet eine Kapselung der DAL Entities auf dem DAL statt.

22	Besprochen am 11.03.2009
Adressat	Benedikt Unold, comparis.ch AG
Frage	Wie werden die E-Mail Nachrichten von der Software an den SMTP Server übergeben bei der bestehenden Lösung NLNMailer (Queue oder SMTP)?
Antwort	Bei der bestehenden Lösung werden die Nachrichten per SMTP übergeben. Der *Smtp Mailer* soll beide Varianten anbieten (per SMTP oder per Ablage in einem Directory) und per Konfiguration eine Umstellung erlauben.

21	Besprochen am 04.03.2009
Adressat	Prof. Dr. Markus Stolze, HSR
Frage	Sollen die nicht funktionalen Anforderungen in der Anforderungsspezifikation nach ISO 9126 spezifiziert werden?
Antwort	Wir sollen projektintern die verschiedenen Punkte der ISO 9126 Norm durchgehen und diejenigen Punkte in die Dokumentation aufnehmen welche noch fehlen und für welche es Anforderungen zu spezifizieren gibt.

20	Besprochen am 03.03.2009
Adressat	Benedikt Unold, comparis.ch AG
Frage	Welche Lizenzen für EasyMail von QuikSoft wurden von comparis.ch AG gekauft?
Antwort	Es ist eine Lizenz für die Full Edition vorhanden.

19	Besprochen am 03.03.2009
Adressat	Benedikt Unold, comparis.ch AG
Frage	Sollen die zurückerhaltenen Bounce-Messages abgelegt werden? So könnte man die Bounce-Message anzeigen lassen um weitere Details herauszulesen falls der Bouncetype zu allgemein ist.
Antwort	Ja. Dies wird benötigt, falls die von der NDR Komponente zugewiesene NDR Kategorie oder NDR Typ zu wenig aussagekräftig ist.

18	Besprochen am 03.03.2009
Adressat	Benedikt Unold, comparis.ch AG
Frage	Sollen die nicht funktionalen Anforderungen in der Anforderungsspezifikation nach ISO 9126 spezifiziert werden?
Antwort	Für Benedikt Unold sind die nicht funktionalen Anforderungen so genügend detailliert spezifiziert und vollständig.

17	Besprochen am 03.03.2009
Adressat	Benedikt Unold, comparis.ch AG
Frage	Gibt es verfügbare Bounce-Messages, welche wir für die Evaluierung einer NDR Komponente als Testeingabe verwenden können?
Antwort	Ja. Benedikt Unold wird uns diese zukommen lassen..

16	Besprochen am 25.02.2009
Adressat	Benedikt Unold, comparis.ch AG
Frage	Können wir die bisherigen erhaltenen Dokumente auch noch elektronisch erhalten? Für Ablage in Projektordner und evtl. Abgabe in Dokumentation.
Antwort	Benedikt Unold wird uns die bisherig abgegebenen Dokumente elektronisch zustellen.

15	Besprochen am 25.02.2009
Adressat	Benedikt Unold, comparis.ch AG
Frage	Muss die Blacklist und die Whitelist aktiviert / deaktiviert werden können? Muss beim Whitelist Betrieb auch die Blacklist beachtet werden resp. beim Blacklist Betrieb auch die Whitelist?
Antwort	Die Blacklist ist immer aktiv. Die Whitelist kann aktiviert / deaktiviert werden. Die Blacklist hat dabei eine stärkere Gewichtung als die Whitelist.

14	Besprochen am 25.02.2009
Adressat	Benedikt Unold, comparis.ch AG
Frage	Was versteht man unter 'Anzeige als Zusammenfassung' im Anforderungsdokument im Abschnitt Debugging- und Reporting-Tool?
Antwort	Die Anzahl von Mails z.B. eines gewissen Applikationscodes, Mailtyps und Datums.

13	Besprochen am 25.02.2009
Adressat	Benedikt Unold, comparis.ch AG
Frage	Müssen die E-Mails eines gewissen Applikationscodes oder Mailtyps auch auf mehrere SMTP Server verteilt werden können?
Antwort	Nein. Ein Applikationscode / Mailtyp Tupel wird einem SMTP Server zugewiesen. Es sollen aber Wildcard erlaubt sein, dass man z.B. alle Mails eines Mailtyps einem SMTP Server zuweisen kann. Dabei werden eingetragene Konfigurationen nach dem first-one-wins Prinzip berücksichtigt.

12	Besprochen am 25.02.2009
Adressat	Benedikt Unold, comparis.ch AG
Frage	Soll Sitzungsprotokoll auch an Benedikt Unold gesendet werden?
Antwort	Ja.

11	Besprochen am 25.02.2009
Adressat	Benedikt Unold, comparis.ch AG
Frage	Gibt es E-Mails die nicht gelöscht werden dürfen? Unter welchen Voraussetzungen dürfen E-Mails gelöscht werden?
Antwort	Die Datenbankbereinigung soll ein automatischer Prozess sein. Es soll für ein Applikationscode / Mailtyp Tupel definiert werden können, nach welcher Zeit der Content gelöscht werden soll. Der Header wird nie gelöscht.

10	Besprochen am 25.02.2009
Adressat	Benedikt Unold, comparis.ch AG
Frage	Gibt es eine weitere Qualitätsanforderung neben dem 'nicht früher als' beim zeitgesteuerten Versand? Muss zu diesem angegeben Zeitpunkt die E-Mail sofort versendet werden?
Antwort	Nein. Es soll nach best effort gearbeitet werden.

9	Besprochen am 25.02.2009
Adressat	Benedikt Unold, comparis.ch AG
Frage	Was ist mit der Referenz und Mailtyp (was gibt es für Mailtypen?) bei E-Mail Metadaten gemeint?
Antwort	Eine Referenz ist ein String und wird über die API übergeben. Die Applikationscodes und Mailtypen müssen nirgends abgelegt werden resp. verwaltet werden

8	Besprochen am 25.02.2009
Adressat	Benedikt Unold, comparis.ch AG
Frage	Wie verläuft die Datenbankbereinigung? Muss diese manuell über das Webinterface ausgelöst werden? Wie verläuft die Konfiguration?
Antwort	Die Datenbankbereinigung soll als automatischen Prozess ablaufen. Die Datenbankbereinigung ist über das Webinterface konfigurierbar.

7	Besprochen am 25.02.2009
Adressat	Benedikt Unold, comparis.ch AG
Frage	Wie funktioniert der Versand von Personalisierten Massenemails?
Antwort	Für jedes einzelne Mail wird die API Methode zum Versenden eines Mails aufgerufen. Wir müssen uns nicht darum kümmern.

6	Besprochen am 25.02.2009
Adressat	Benedikt Unold, comparis.ch AG
Frage	Wie muss die E-Mail gespeichert werden (generierte E-Mail oder übergebene Parameter)?
Antwort	Nur die übergebenen Parameter. Beim erneuten Versenden müssen einfach alle Informationen vorhanden sein, um das gleiche E-Mail wieder zu versenden können.

5	Besprochen am 25.02.2009
Adressat	Benedikt Unold, comparis.ch AG
Frage	Werden beim Löschen einer E-Mail nur die Datenbankfelder gelöscht oder gesamte Nachricht. Was ist mit in Beziehung stehenden Daten?
Antwort	Nur der Content.

4	Besprochen am 25.02.2009
Adressat	Benedikt Unold, comparis.ch AG
Frage	Dürfen Artefakte, Mindmaps etc. im Bachelorarbeitszimmer aufgehängt werden?
Antwort	Ja. Das ist kein Problem, solange keine Geschäftsgeheimnisse von comparis.ch AG offengelegt werden.

3	Besprochen am 25.02.2009
Adressat	Markus Stolze, HSR
Frage	Gibt es eine offizielle, unterschriebene Aufgabenstellung?
Antwort	Eine offizielle Aufgabenstellung (unterschrieben) haben wird am 25.02.2009 erhalten.

2	Besprochen am 25.02.2009
Adressat	Prof. Dr. Markus Stolze, HSR
Frage	Wie soll die Anforderungsspezifikation erstellt werden? Sollen Teilprojekte gebildet werden oder zu Beginn die Anforderungsspezifikation über das gesamte Projekt erstellt werden?
Antwort	Zu Beginn die Anforderungsspezifikation erstellen soweit möglich. Und RUP in einem Zyklus durcharbeiten.

1	Besprochen am 25.02.2009
Adressat	Prof. Dr. Markus Stolze, HSR
Frage	Das Projektmanagement findet mit dotProject statt. Müssen die dort erfassten Arbeitspakete in den Projektplan übernommen werden?
Antwort	Ja. Im Projektplan sollten ebenfalls alles was in dotProject ersichtlich ist, aufgeführt sein.

D Sitzungsprotokolle

1 – Kickoff Meeting mit Industriepartner, 17.02.2009

Teilnehmer

- Benedikt Unold, comparis.ch AG
- Danny Meier, HSR
- Micha Boller, HSR

Ort / Zeit

Zürich, 15:00 – 16:15

Zweck

- Unterschreiben der Verträge betreffend Geheimhaltung und Immaterialgüterrechte
- Besprechung von Anforderungskatalog aufgestellt durch Benedikt Unold

Was wurde besprochen

- Verträge betreffend Geheimhaltung und Immaterialgüterrechte
- Unterzeichnung der Verträge in vier Ausführungen (Exemplar für comparis.ch AG, Danny Meier, Micha Boller und HSR)
- Besprechung der durch Benedikt Unold aufgelisteten Anforderungen an den *Smtp Mailer*
    - Allgemeines:
        - „zur Laufzeit konfigurierbar“ heisst: Konfigurierbar über Webinterface
        - API kann frei entwickelt werden ohne Einschränkungen
        - Verwendung von einzukaufenden Komponenten OK, aber bei comparis.ch AG Erlaubnis einholen
        - Für EasyMail besitzt comparis.ch AG Lizenzen
    - Versand:
        - Die Metadaten Applikationscode, Mailtyp und Referenz pro Message werden nur in die Datenbank abgelegt
        - Pro Mailtyp / Applikationscode kann definiert werden, welcher SMTP Server verwendet werden soll
        - Die Verwaltung der Whitelist kann z.B. über ein App.config geschehen
        - Die Qualitätsindikatoren poor und failed klassieren Hardbounces (failed) und Softbounces (poor)
        - Qualitätsindikator soll History über bereits an eine E-Mail Adresse versendete E-Mails beachten
        - Ist ein Empfänger nicht erreichbar (poor), so muss das E-Mail nicht automatisch neu versendet werden
        - Bei aktivierter Whitelist können nur E-Mails an die Adressen / Domains gesendet werden, welche in der Whitelist stehen
        - Der Algorithmus für den Qualitätsindikator darf weder zu restriktiv noch zu träge sein
    - Errorhandling
        - SMTP Server als Server für Bounce Nachrichten verwenden und per POP 3 auf Bounce Nachrichten zugreifen
    - Debugging- und Reporting Tool (Webapplikation)
        - Verwendung von Silverlight für Reports
    - Nice-to-Have (optional):
        - Verschlüsselung von E-Mails vorsehen
    - API:
        - Eine API für Applikationen, um ein E-Mail zu verschicken. Zudem muss der E-Mail Status abgefragt werden können
        - Eine API für erweiterte Tools wie CRM, Cockpit, etc.
- Klärung von Fragen aus dem Fragenkatalog

- Benedikt Unold ist in der KW 12 abwesend

Entscheide

- Die von uns erstellte Software dürfen wir nach Beendigung der Arbeit kommerziell nutzen
- Zur Generierung der API-Dokumentation kann irgendein Tool verwendet werden. Nur die Dokumentation sollte sich als XML-Kommentar im Quellcode befinden
- Artefakte und Diagramme dürfen im Bachelorarbeitszimmer aufgehängt werden, sofern keine Firmengeheimnisse (insbesondere Kennzahlen) offenbart werden

Nächste Schritte

- Entwicklung einer Anforderungsspezifikation basierend auf der Auflistung der Anforderungen durch Benedikt Unold

Nächste Termine

- Besprechung von Projektplan
  Mittwoch 25.02.2009, 17:00 an der HSR

2 – Kickoff Meeting mit Betreuer, 18.02.2009

Teilnehmer

- Prof. Dr. Markus Stolze, HSR
- Danny Meier, HSR
- Micha Boller, HSR

Ort / Zeit

Rapperswil, 08:10 – 08:30

Zweck

- Projekt offiziell starten
- Vorgehen klären
- Fragen besprechen

Was wurde besprochen

- Besprechung der bis jetzt vorgenommenen Tätigkeiten
- Besprechung des Meetings vom 17.02.2009
- Anwendung der RUP Vorgehensweise auf unser Projekt mit mehreren Teilprojekten
- Bürokratisches
    - Traktandenliste vor einer Sitzung versenden für Sitzungen mit der comparis.ch AG und für Sitzungen, welche eine Vorbereitung durch den Betreuer erfordern
    - Entscheidungs- resp. Sitzungsprotokoll nach jeder Sitzung an den Betreuer zukommen lassen
    - Anleitung von Prof. Stefan Keller angesprochen und abgemacht, dass auch Teile wegelassen werden dürfen, welche aus unserer Sicht nichts bringen. Diese Änderungen müssen aber mit dem Betreuer besprochen werden

Entscheide

- Einigung auf Vorgehensweise nach RUP
- Am 25.02.2009 sollte eine erste Version vom Projektplan und der Anforderungsspezifikation besprochen werden können

Nächste Schritte

- Projektplan fertigerstellen
- Anforderungsspezifikation basierend auf Anforderungskatalog vom Industriepartner beginnen

Nächste Termine

- Besprechung vom Projektplan und einer ersten Version der Anforderungsspezifikation am Mi, 25.02.2009, 17:00 an der HSR

3 – MS 1 Sitzung, 25.02.2009

Teilnehmer

- Prof. Dr. Markus Stolze, HSR
- Benedikt Unold, comparis.ch AG
- Danny Meier, HSR
- Micha Boller, HSR

Ort / Zeit

Rapperswil, 17:00 – 18:15

Zweck

- Besprechen des Projektplanes
- Fragen besprechen

Was wurde besprochen

- Zeitplanung
  - Die Zeitplanung und Phasendauer der Construction1 sollte nochmals überdacht werden
  - Einplanen der Entwicklung eines Architektur Prototypen
- Meilensteine
  - Einen Meilenstein erstellen für den Architektur Prototypen
  - Einen Meilenstein erstellen für den Integrationstest aller Bestandteile des *Smtp Mailers*
  - Einen Meilenstein erstellen für die Besprechung und Abnahme des Datenmodells
  - Bei folgenden Meilensteinen möchte Benedikt Unold gerne involviert sein:
    - Anforderungsspezifikation
    - Domain Analyse
    - Datenmodell
    - Architekturprototyp
    - Sonstige Prototypen
- Risiken
  - Hauptrisiko bezüglich Performance werden die Zugriffe auf die Festplatte sein, beim Schreiben / Lesen der Daten
  - Die beiden Performancerisiken (Software R04, Hardware R05), werden mit entsprechenden Testprototypen beseitigt
- Umsetzung
  - Das Datenmodell sollte mit Benedikt Unold und Christian Tarnutzer angeschaut werden um Schwachstellen und Designfehler zu erkennen und die Integration in die comparis.ch Umgebung zu gewährleisten
  - Es soll ein Architekturprototyp erstellt werden, damit die reibungslose Kommunikation der verschiedenen Module frühzeitig geklärt ist
  - Für die Usability Evaluation können wir die Usability Spezialistin von comparis.ch AG (Annette Sulzbacher) zu Rate ziehen
  - Die Usability Tests sollten mit mehreren Paper Prototypes erfolgen
- Projektmanagement
- Der Datenbankbereinigungsprozess ist ein automatischer Prozess

Entscheide

- Benedikt Unold möchte keine Dokumente und Informationen erhalten, welche das reine Projektmanagement betreffen. Zum Beispiel Arbeitspaket Planung
- Als Modultest Framework ist NUnit zu verwenden. Das Projekt, welches die Unit-Tests enthält trägt den gleichen Namen wie das zu testende Projekt, endet aber mit *.UnitTests* und befindet sich in der gleichen Solution

- Die in den Use Cases vorkommenden Aktoren sind wie folgt:
  - Kundendienst
  - Entwickler
  - Admin
  - comparis.ch Applikation
- Bei der Datenbankbereinigung soll nur der Content inklusive Anhang gelöscht werden können
- Applikationscodes und Mailtypen müssen nicht geführt werden, man kann sie auf der API einfach entgegennehmen
- Zu besprechende Dokumente müssen frühzeitig bei Prof. Dr. Markus Stolze und Benedikt Unold eintreffen damit ihnen genügend Vorbereitungszeit bleibt (min. 2 Tage zuvor)

Nächste Schritte

- Überarbeiten des Projektplanes gemäss Feedback
- Fertigstellen der Anforderungsspezifikation
- Fertigstellen der Domain Analyse
- Erstellen des Datenmodells
- Komponenten Evaluieren

Nächste Termine

- Besprechung der Anforderungsspezifikation und der Komponenten Evaluation
  Di, 03.03.2009, 15:00 in Zürich
- Besprechung der Domain Analyse und des Datenmodells
  Mi, 11.03.2009, 16:00 in Zürich

4 – MS 2 / MS 3 Sitzung, 03.03.2009

Teilnehmer

- Benedikt Unold, comparis.ch AG
- Danny Meier, HSR
- Micha Boller, HSR

Ort / Zeit

Zürich, 15:00 – 17:00

Zweck

- Besprechen der Anforderungsspezifikation
- Besprechen der Komponenten Evaluation
- Fragen besprechen

Was wurde besprochen

- Projektplan
  - Benedikt Unold möchte bei den Modul-Abgaben Meilensteinen anwesend sein, jedoch nicht bei den Prototypen
- Anforderungsspezifikation
  - Bei dem zu entwickelnden API sollte eine Trennung vorliegen zwischen der Schnittstelle für das Versenden und Abfragen von E-Mails und der Schnittstelle für das Reporting, Konfiguration und Maintenance
  - Der Produkt Perspektive ist die Aussage „enthält keine Framework Integration“ zu entfernen, da diese vorhanden ist
  - Änderungen in der App.config sollten automatisch vom System erkennt und angewendet werden (Nice-To-Have)
  - E-Mail Nachrichten, dessen Empfänger auf der Blacklist steht, sollte entsprechend markiert werden und bereinigt werden können. E-Mail Adressen die auf der Blacklist stehen haben keinen Einfluss auf den Qualitätsindikator
  - Der Report zur Anzeige der Anzahl E-Mails über einen bestimmten Zeitraum sollte auch über den Status gefiltert werden können
  - Die verwendete x64 Hardwarearchitektur ist den nichtfunktionalen Anforderungen hinzuzufügen
  - Die Benennung des Namespace ist den nichtfunktionalen Anforderungen unter *Naming Guidelines* hinzuzufügen
  - Loggen
    - Es kann ein Log-Framework wie log4net verwendet werden. Jedoch sollte das Loggen gekapselt werden um ein späteres Ersetzten des Logging Mechanismus einfach zu ermöglichen
    - Die Logeinträgen sollten Aufschluss darüber geben, welche Parameter zum Fehler geführt haben und nicht nur der Fehler selbst
  - Ausnahmebehandlung
    - Wo möglich ist immer mit Exceptions zu arbeiten. System-Exceptions können über alle Schichten bis an den Entwickler weitergereicht werden (kein kapseln, z.B. SQLException)). Bei selbst geworfenen Exceptions sind Custom-Exceptions zu verwenden
    - Wenn möglich sollte auf jegliche Rückgabe von Null-Werten verzichten werden
  - Erstellen eines Use Cases für die NDR Verarbeitung
- Komponenten Evaluation

- Bei den nichtfunktionalen Anforderungen ist die Performance für die Verarbeitung der Bounce Mails hinzuzufügen. Diese ist auf 30‘000 Mails / Stunde zu definieren
- Die beiden in Frage kommenden NDR Komponenten sind noch auf Performance (siehe oben) und Erkennungsgenauigkeit zu testen (Produkte ListNanny und NetXtremeBounceFilter)
- Die Firma comparis.ch AG verfügt über Lizenzen für das Produkt EasyMail.Net Full Edition

Entscheide

- Die beiden APIs müssen nur per Assembly zugänglich gemacht werden
- In die Datenbank werden nur Konfigurationen gespeichert, welche Unabhängig vom Standort gebraucht werden können. Die standortabhängigen Konfigurationen sind in einer App.config zu definieren.
  - Datenbank
    - Blacklist
    - Regeln zur Datenbankbereinigung,
    - NDR Konfigurationen
    - Whitelist
  - App.config
    - SMTP Server Zuordnung
- Bei der Datenbankbereinigung sind alle E-Mail Nachrichten (definiert durch Applikationscode / Mailtyp) mit einzubeziehen die nicht den Status *Neu* oder S*oft Bounce* haben.
- Der Namespace für den *Smtp Mailer* lautet: Comparis.Framwork.SmtpMailer.*
- Es ist die Mail Komponente EasyMail.Net Full Edition zu verwenden

Nächste Schritte

- Beim Projektplan die Meilensteine anpassen
- Die Anforderungsspezifikation gemäss Feedback überarbeiten
- Datenmodell entwickeln
- Architekturprototyp entwickeln
- Domain Analyse fertigstellen

Nächste Termine

Prof. Dr. Markus Stolze

Projektfortschrittssitzung
Mi, 04.03.2009, 08:10 in Rapperswil

Benedikt Unold

Besprechung der Domain Analyse, des Datenmodells und des Architekturprototyps
Mi, 11.03.2009, 16:00 in Zürich

5 – Projektfortschrittssitzung, 04.03.2009

Teilnehmer

- Prof. Dr. Markus Stolze, HSR
- Danny Meier, HSR
- Micha Boller, HSR

Ort / Zeit

Rapperswil, 08:15 – 09:00

Zweck

- Besprechen Anforderungsspezifikation Feedback
- Besprechen Projektfortschritte

Was wurde besprochen

- Anforderungsspezifikation
  - Die Aktoren sollten eine unterschiedliche Grafik haben, je nachdem ob es sich um einen menschlichen Aktor oder eine Applikation handelt
  - Nochmals die Punkte der nichtfunktionalen Anforderungen gemäss ISO 9126 durchgehen und wenn noch sinnvolle Kriterien auftauchen diese dokumentieren
  - Die Use Case Beschreibungen um die nichtfunktionalen Anforderungen ergänzen, sofern diese von den globalen dokumentierten Anforderungen abweichen
- Mitteilen der Kontaktdaten an Prof. Dr. Markus Stolze, um eine Benachrichtigung zu ermöglichen bei Unvorhergesehenem

Entscheide

- -

Nächste Schritte

- Die Anforderungsspezifikation gemäss Feedback überarbeiten
- Datenmodell entwickeln
- Architekturprototyp entwickeln
- Domain Analyse fertigstellen

Nächste Termine

Prof. Dr. Markus Stolze

Projektfortschrittssitzung
Mi, 11.03.2009, 08:10 in Rapperswil

Benedikt Unold

Besprechung der Domain Analyse, des Datenmodells und des Architekturprototyps
Mi, 11.03.2009, 16:00 in Zürich

6- Projektfortschrittssitzung, 11.03.2009

Teilnehmer

- Prof. Dr. Markus Stolze, HSR
- Danny Meier, HSR
- Micha Boller, HSR

Ort / Zeit

- Rapperswil, 08:10 – 08:40

Zweck

- Besprechen Projektfortschritte

Was wurde besprochen

- Domain Analyse
  - Erklärung der Ndr-Klassen
- Analyse
  - Genauer erklären wieso gerade 22 E-Mails für die Ndr-Komponenten Tests verwendet wurden (weil uns so viele Ndr Mails von comparis.ch zur Verfügung gestellt wurden)
  - Hinweis hinzufügen, wieso die Beschreibungen bei der Safabyte NetXtremeBounceFilter-Komponente leer sind (weil diese von Safabyte nicht dokumentiert sind)

Entscheide

- -

Nächste Schritte

- Architekturprototyp implementieren
- Risiko 04 / Risiko 05 Prototyp implementieren
- Design beginnen

Nächste Termine

Prof. Dr. Markus Stolze

Projektfortschrittssitzung
Mi, 18.03.2009, 08:10 in Rapperswil

Benedikt Unold

Besprechung der Domain Analyse, des Datenmodells und des Architekturprototyps
Mi, 11.03.2009, 16:00 in Zürich

7 – MS 4 / MS 5 / MS 6 Sitzung, 11.03.2009

Teilnehmer

- Benedikt Unold, comparis.ch AG
- Danny Meier, HSR
- Micha Boller, HSR

Ort / Zeit

- Zürich, 16:00 – 18:15

Zweck

- Besprechen und Entscheiden der zu verwendenden NDR Komponente
- Besprechen des Datenmodells
- Besprechen der Domain Analyse

Was wurde besprochen

- Technologie
  - LINQ to SQL wird nicht weiter entwickelt. Wenn von der Performance her zulässig auf das Entity Framework umsteigen oder mit ADO.NET direkt arbeiten
- Datenmodell
  - Um eine effiziente sich nicht schneidende Verarbeitung der Nachrichten Warteschlange zu gewährleisten sind folgende Locks notwendig:
    - ReadPast
    - UpdateLock
    - RowLock
  - In den Attributen sollten keine Underscore (_) vorkommen. Somit sind die Primärschlüssel nach dem Schema *<TabellenName>Id* und die Fremdschlüssel nach dem Schema *<ReferenzTabellenName>Id* zu benennen
  - Verwendete Enumerationen im Code (z.B. Typen-Attribute) sollten über eine Lookup-Tabelle in der Datenbank verfügen mit folgenden Attributen:
    - *TypeId* Zahlenwert
    - *Name* Name, wie er in der Enumeration verwendet wird
    - *Description* Eine Beschreibung was der Typ bedeutet
  - Datums Attribute sollten einheitlich benennt werden und aussagekräftig benannt werden, also nicht nur *Date* sondern *CreateDate*
  - Allen Attributen einen für sich selbst sprechenden Namen geben
  - Die Tabelle *Message* benötigt ein zusätzliches Attribut *guid*. welches eine eindeutige Kennung für die Nachricht ist. Nur diese wird nach aussen weiter gegeben. Intern wird nach wie vor eine *MessageId* verwendet
  - Die Assoziation zwischen *Message* und *NdrMail* ist zu ändern in eine Assoziation zwischen *NdrMail* und *Recipient*. Da ein eine Bounce Nachricht per Empfänger zurück kommt
  - Der Adressqualitätsindikator wird in einer eigenen unabhängigen Tabelle verwaltet. Neu ist die E-Mail Adresse für den Versand direkt in der *Recipient* Tabelle enthalten

- Anforderungsspezifikation
  - Die x64 Architektur sollte vom Kapitel 4.2.1 (Hardware) in das Kapitel 4.2.2 (Software) verschoben werden
  - Die parallele Nutzung im Kapitel 4.8.2 muss genauer beschrieben werden, welche Kombinationen unterstützt werden
  - Es ist konfigurierbar, wie die Übermittlung der Nachrichten an den MTA erfolgen sollen (per SMTP Protokoll oder Ablage der zu versendenden Nachrichten in einem definierten Verzeichnis)
  - Das Tupel aus Applikationscode und Mailtyp gibt die Priorität vor (Low und High). Ist keine explizite Zuordnung vorhanden, gilt High-Priority als Standard
  - Es ist konfigurierbar, wie viele Low-/High-Priority Instanzen für den Versand zur Verfügung stehen
  - Jede Low-/High-Priority Instanz verfügt über eine SMTP Server Zuordnung
- Domain Analyse
  - Beim Use Case 8 die korrekte SSD Grafik im Dokument einsetzen
  - Der Name *Datenbankbereinigung* ist in *SmtpMailerCleanup* umzubenennen
- Analyse
  - Auswertung der Komponenten Evaluation besprochen und Entscheidung gefällt
  - Lizenzschlüssel für die EasyMail.NET Komponente erhalten
- Diverses
  - Soft-Bounces müssen Beim berechnen des Qualitätsindikators auch miteinbezogen werden, werden jedoch leichter gewichtet als Hard-Bounces
  - Soft-Bounces müssen je nach Typ unterschieden werden für die Qualitätsindikatorberechnung
  - Diskutiert, wie die Warteschlange funktionieren und arbeiten könnte.

Entscheide

- Die Datenbank hat den Namen *SMTP_MAILER_DB*
- Es wird die NDR-Komponente NetXtremeBounceFilter von Safabyte verwendet
- Endbenutzermeldungen sollten in deutsch erscheinen, alle anderen Sprachstrings und Beschreibungen sind in englisch zu halten
- Versendete Nachrichten (unabhängig ob es noch Soft- oder Hard-Bounces hat) werden bei der Bereinigung gelöscht
- Die Dauer nachdem Nachrichten gelöscht werden soll in Tagen festgelegt werden (StorageDuration)
- Der Connection-String ist unter dem Namen *CONN_SMTP_MAILER_DB* in der Konfiguration abzulegen

Nächste Schritte

- Vorhandene Dokumente gemäss Änderungen aktualisieren
- Architektur Prototyp implementieren
- Risiko 04 / Risiko 05 Prototyp implementieren
- Design beginnen

Nächste Termine

Prof. Dr. Markus Stolze

Projektfortschrittssitzung
Mi, 18.03.2009, 08:10 in Rapperswil

8 - Projektfortschrittssitzung, 18.03.2009

Teilnehmer

- Prof. Dr. Markus Stolze, HSR
- Danny Meier, HSR
- Micha Boller, HSR

Ort / Zeit

- Rapperswil, 08:10 – 08:40

Zweck

- Besprechen Projektfortschritte

Was wurde besprochen

- Besprechung des aktuellen Stands
- Expertenbesuch
    - Die Besprechung mit dem Experten finden am 06.05.2009 um 09:00 statt
    - Am 06.05.2009 findet keine Projektfortschrittssitzung statt um 08:10
    - Es soll eine Präsentation vorbereitet werden, welche dem Experten aufzeigt, um was es in der Bachelorarbeit geht
    - Der Experte kann Fragen stellen und erläutern, auf was er achtet und was er gerne noch hätte
- MS 7 / MS 8 Sitzung
    - Prof. Dr. Markus Stolze geht der Termin vom Dienstag, 31.3. jedoch erst ab 16:15 Uhr. Die Sitzung würde folglich bis 18:15 dauern.
- Architektur Prototyp
    - Es soll dokumentiert werden, welche Einschränkungen bei Silverlight bezüglich WCF vorliegen

Entscheide

- -

Nächste Schritte

- Risiko 05 Prototyp fertigstellen
- Design beginnen

Nächste Termine

Prof. Dr. Markus Stolze

Projektfortschrittssitzung
Mi, 25.03.2009, 08:10 in Rapperswil

Besprechung der Prototypen für Risiko 04, 05 und Architektur Prototyp
Di, 31.03.2009, 16:15 – 18:15 in Rapperswil

Benedikt Unold

Besprechung der Prototypen für Risiko 04, 05 und Architektur Prototyp
Di, 31.03.2009, 16:15 – 18:15 in Rapperswil

9 – Projektfortschrittssitzung, 25.03.2009

Teilnehmer

- Prof. Dr. Markus Stolze, HSR
- Danny Meier, HSR
- Micha Boller, HSR

Ort / Zeit

- Rapperswil, 08:10 – 08:30

Zweck

- Besprechen Projektfortschritte

Was wurde besprochen

- Besprechung des aktuellen Stands
- UI Requirements / Design
    - Paper Prototype ist bei den Anforderungsspezifikationen zu dokumentieren. Ausser wenn sehr detaillierte Elemente dazukommen können diese auch im Design Dokument genauer dokumentiert werden
    - Es sind Testszenarien zu dokumentieren, welche die Paper Prototype Tests beschreiben
    - Szenarien und Personas sind nach eigenem Ermessen zu dokumentieren oder wegzulassen

Entscheide

- -

Nächste Schritte

- Termin mit Benedikt Unold ausmachen für die Usability-Tests in der ‚Construction 3'-Iteration
- Design Mail Modul beginnen
- Prototypen dokumentieren
- Codereview bei den Prototypen durchführen

Nächste Termine

Prof. Dr. Markus Stolze

Besprechung der Prototypen für Risiko 04, 05, Architektur Prototyp und Design Mail Modul
Di, 31.03.2009, 16:15 – 18:15 in Rapperswil

Projektfortschrittssitzung
Mi, 01.04.2009, 08:10 in Rapperswil

Benedikt Unold

Besprechung der Prototypen für Risiko 04, 05, Architektur Prototyp und Design Mail Modul
Di, 31.03.2009, 16:15 – 18:15 in Rapperswil

10 - MS 7 / MS 8 / MS 9 Sitzung, 31.03.2009

Teilnehmer

- Prof. Dr. Markus Stolze, HSR
- Benedikt Unold, comparis.ch AG
- Danny Meier, HSR
- Micha Boller, HSR

Ort / Zeit

- Rapperswil, 16:15 – 17:30

Zweck

- Besprechen der Prototypen Architektur, R04, R05
- Besprechen des Designs vom Mail Modul
- Besprechen Projektfortschritte

Was wurde besprochen

- Prototyp R05
  - Das Performance Problem in der Variante 1 könnte damit zusammenhängen, dass der Query Optimizer die gesetzten Indexes nicht nutzt, weil ihm ein Full Scan schneller erscheint
  - Als Workaround für das Problem kann man versuchen mit 'WITH INDEX *(<indexname>)*' das Benutzen des Index zu erzwingen
  - Variante 1 soll auf das geprüft werden, ob sich so das Performanceproblem lösen lässt
  - Im Ausführungsplan des Querys kann überprüft werden, ob der Index genutzt wird
  - Christian Tarnutzer, comparis.ch AG, steht bei Fragen zur Verfügung
  - Die Performancemessungen müssen nochmals mit der fertig implementierten Variante 1 durchgeführt werden
- Transaktionsmanagement
  - Allgemein sollte das Lesen einer zu versendenden Nachricht und das Updaten der Nachricht in einer Transaktion ablaufen
  - So kann zwar kein *Exactly Once* erreicht werden aber sicher ein *At Least Once*
  - Dazu gibt man der Business Komponente die Transaction mit in den Business Layer, damit bei einer Exception ein Rollback auf der Transaktion ausgelöst werden kann. Oder es kann auch ein Transaction Objekt beim Aufruf in den DAL verlangt werden.
  - Als weitere Alternative kann der DTC (Distributed Transaction Coordinator) in Betracht gezogen werden
- Mail Modul Design
  - Der Namespace Comparis.Framework.SmtpMailer.Web soll in SmtpMailerAdmin umbenannt werden
  - Der Namespace Comparis.Framework.SmtpMailer.SendingEngine soll in SmtpMailer.SendingEngine umbenannt werden
  - Der Namespace Comparis.Framework.SmtpMailer.BounceEngine soll in SmtpMailer.BounceEngine umbenannt werden
  - Der Namespace Comparis.Framework.SmtpMailer.CleanupEngine soll in SmtpMailer.CleanupEngine umbenannt werden
- Konfigurationsdatei
  - Es soll die integrierte App.config genutzt werden und nicht ein eigen entwickeltes XML File

    - Dazu können ConfigurationSectionHandler implementiert werden, welche ein Stück XML aus dem App.config verarbeiten können
    - Mittels Setzen von Attributen im ConfigurationSectionHandler können die Daten gar automatisch abgefüllt werden
- Paper Prototype Tests und Besprechung mit Annette Sulzbacher, comparis.ch AG
    - Benedikt Unold kümmert sich um einen Termin mit einem Kundendienstmitarbeiter (Paper Prototype) und mit Annette Sulzbacher

Entscheide

- Wenn möglich soll die Variante 1 vom Prototyp Risiko 05 als Basis für das Mail Modul verwendet werden
- App.config als Konfigurationsdatei für den *Smtp Mailer* verwenden

Nächste Schritte

- Risiko 05 Prototyp weiterentwickeln und Performancemessungen durchführen
- Mail Modul fertig implementieren

Nächste Termine

Prof. Dr. Markus Stolze

Projektfortschrittssitzung
Mi, 07.04.2009, 08:10 in Rapperswil

Besprechung des fertigimplementierten Mail Moduls (Meilenstein 10)
Do, 16.04.2009, 16:30 in Zürich

Benedikt Unold

Besprechung des fertigimplementierten Mail Moduls (Meilenstein 10)
Do, 16.04.2009, 16:30 in Zürich

11 – Projektfortschrittssitzung, 07.04.2009

Teilnehmer

- Prof. Dr. Markus Stolze, HSR
- Danny Meier, HSR
- Micha Boller, HSR

Ort / Zeit

- Rapperswil, 11:00 – 11:15

Zweck

- Besprechen Projektfortschritte

Was wurde besprochen

- Performance Probleme erläutert
  - Kombinierter Indexschlüssel
  - Fill Factor auf 20%
- Die Szenarien und der Paper Prototype wird eine Woche vor dem UI-Test mit Prof. Dr. Markus Stolze besprochen während der wöchentlichen Projektfortschrittssitzung (29.04.2009)

Entscheide

- -

Nächste Schritte

- Fertigimplementierung von Mail Modul

Nächste Termine

Prof. Dr. Markus Stolze

Projektfortschrittssitzung
Mi, 15.04.2009, 08:10 in Rapperswil

Meilensteinsitzung MS9 / M10
Do, 16.04.2009, 16:30 in Zürich

Benedikt Unold

Meilensteinsitzung MS9 / MS10
Do, 16.04.2009, 16:30 in Zürich

12 – Projektfortschrittssitzung, 15.04.2009

Teilnehmer

- Prof. Dr. Markus Stolze, HSR
- Danny Meier, HSR
- Micha Boller, HSR

Ort / Zeit

- Rapperswil, 08:10 – 08:30

Zweck

- Besprechen Projektfortschritte
- Besprechen der Vorstellungen bezüglich Dokumentation von Prof. Dr. Markus Stolze

Was wurde besprochen

- Prof. Dr. Markus Stolze hat eine Prüfung des Zwischenstands durchgeführt und uns seine Vorstellungen bezüglich Dokumentation mitgeteilt
    - Der Projektbericht soll sich in einem Dokument befinden
    - Es soll ein Inhaltsverzeichnis vorhanden sein, welches sich über die gesamte Dokument zieht
    - Ein Management Summary muss vorhanden sein
    - Abschliessend muss eine Webseite gestaltet werden, welche unser Projekt beschreibt. Es soll sich dabei um ein etwas ausführlicheres und mit Bildern versehendes Management Summary handeln, welches auf die Wiki Seite des Software Institutes geschaltet wird. Unsere Namen dürfen auch darauf sein
    - Es soll mittels Referenzen angegeben werden, wie sich unsere Bachelorarbeit in die Welt einordnet. Dazu müssen mindestens ein Buch, ein IEEE Paper und ein ACM referenziert werden.
    - Die Aufgabenstellung (Original mit Unterschrift) soll eingescannt in die Dokumentation eingebunden werden
    - Es muss eine Methodenreflexion in der Dokumentation stattfinden. Das heisst, es muss genannt werden, welche Methoden der Software Entwicklung geklappt haben und welche nicht. Zudem muss angegeben werden, wieso sie geklappt haben resp. wieso nicht.
    - Bei der Entwicklung der Silverlight Applikation soll ein Log geführt werden. Dieses soll dokumentieren, was für Probleme aufgetreten sind und wie diese gelöst wurden. Es soll auch darstellen, was gut geklappt hat.
    - Der Paper Prototype soll im Analyse Dokument dokumentiert werden
    - Im Testplan soll ein Abschnitt Anforderungserfüllung verfasst werden, welcher aufzeigt, welche Anforderungen mit welchem Test getestet wurden. So kann bewiesen werden, dass die Anforderungen abgedeckt sind
    - Die Anleitung für den Benutzer soll sich eher als Maintenance Anleitung repräsentieren statt als Installationsanleitung
    - Für die Benutzer der Webapplikation (Kundendienst) soll eine Quickcard verfasst werden. Das Format soll A5 sein. Diese Quickcard soll erklären, wie man die Webapplikation bedient und man mit ihr arbeitet.
- Bezüglich Code gab uns Prof. Dr. Markus Stolze auch noch ein paar Tipps
    - Es werden verschiedene Code Metriken angeschaut
    - Es wird überprüft, ob es irgendwelche zyklische Abhängigkeiten im Code gibt

Entscheide

- -

Nächste Schritte

- Fertigimplementierung von Mail Modul
- Weiterfahren mit dem Designs für das NDR Modul

Nächste Schritte

Prof. Dr. Markus Stolze

Besprechung des fertigimplementierten Mail Moduls (Meilenstein 10)
Do, 16.04.2009, 16:30 in Zürich

Benedikt Unold

Besprechung des fertigimplementierten Mail Moduls (Meilenstein 10)
Do, 16.04.2009, 16:30 in Zürich

13 - MS 10 Sitzung, 16.04.2009

Teilnehmer

- Prof. Dr. Markus Stolze, HSR
- Benedikt Unold, comparis.ch AG
- Danny Meier, HSR
- Micha Boller, HSR

Ort / Zeit

- Zürich, 16:30 – 18:00

Zweck

- Demonstration des implementierten Mail Moduls
- Besprechung der Architektur und des Codes

Was wurde besprochen

- Demonstration des Mail Moduls
- Besprechung der Konfiguration
    - Die Konfiguration soll um den Lizenzschlüssel der Mail und NDR Komponente ergänzt werden
- Die Sending Engine soll in Form einer Konsolen Applikation und nicht als Windows Dienst implementiert werden. Wenn die Engine gestartet wird soll eine Konsolenausgabe getätigt werden
- Besprechung des Log-Files
    - Die Logmeldung ‚Message sent over message queue' muss um die Id der betroffenen Message ergänzt werden
    - Die Logmeldung ‚Message has no recipient' im Fall wenn die Blacklist den Empfänger gefiltert hat und die Message nun keinen Empfänger mehr besitzt ist missverständlich
    - WARN nur für die Log Meldungen verwenden, welche wirklich helfen, auf einen Missstand aufmerksam zu machen
    - Das Logging findet später in die Datenbank statt, deshalb muss das Programm abstürzen, wenn die Datenbank nicht mehr verfügbar ist (Exceptions weiterwerfen)
    - Die Log Meldungen müssen allgemein überarbeitet werden, weil sie zum Teil missverständlich sind
- Das Transaction Management beim Versenden der Message über DTC soll manuell mit dem Debugger getestet werden
- Code Kommentare
    - Es sollen bei komplexeren Codestellen auch im Code Kommentare eingefügt werden (innerhalb der Methoden)
    - Auch private Methoden müssen kommentiert werden
- Exception Handling
    - Wird beim Validieren bemerkt, dass ein Feld ausgefüllt ist welches nicht ausgefüllt sein soll, so muss eine Exception geworfen werden
    - Ist die Datenbank z.B. nicht vorhanden, so soll die Exception weitergeworfen werden. Somit stürzt das Programm ab

Entscheide

- Log Messages müssen überarbeitet werden
- Exceptions werden weitergeworfen damit das Programm abstürzt und nicht nur loggt
- Bereits beim Erstellen einer Message wird entschieden, ob die Message die Priorität High oder Low besitzt. Dies wird in der Datenbank vermerkt. Somit verliert man zwar die Flexibilität, zur Laufzeit die Prioritäten zu verändern (also auch bei bereits in die Datenbank eingetragenen Messages), dafür wird die Implementierung vereinfacht und man erreicht eine höhere Performance

Nächste Schritte

- Mail Modul abschliessen gemäss besprochenen Punkten
- Design Ndr Modul erstellen und implementieren

Nächste Termine

Prof. Dr. Markus Stolze

Projektfortschrittssitzung
Mi, 22.04.2009, 08:10 in Rapperswil

Besprechung eines ersten Entwurfes des Paper Prototypes
Mi, 29.04.2009, 08:10 in Rapperswil

Benedikt Unold

Besprechung des Designs vom Ndr Modul (Meilenstein 11) und Besprechung des Algorithmus für die Bestimmung des Qualitätsindikators für eine E-Mail Adresse
Di, 21.04.2009, 16:00 in Zürich

14 – MS 11 Sitzung, 21.04.2009

Teilnehmer

- Benedikt Unold, comparis.ch AG
- Danny Meier, HSR
- Micha Boller, HSR

Ort / Zeit

- Zürich, 16:00 – 17:15

Zweck

- Strategie und Regeln definieren wie der Qualitätsindikator funktioniert
- Ndr Modul Design besprechen

Was wurde besprochen

- Qualitätsindikator
  - Die Berechnung des Adress-Qualitätsindikators soll einmal wöchentlich als Batch-Job ablaufen. Die Umsetzung soll daher als Konsolenprogramm erfolgen
  - Der Indikator wird nur mittels Batch-Job ausgeführt und soll nicht in die Bounce-oder Sending-Engine integriert werden
  - Die Tabelle „AddressQuality“ soll um ein Attribut „ProcessDate“ erweitert werden, welches das Datum der letzten Berechnung des Qualitätsindikators enthält
  - Um eine Anhäufung von fehlerhaften E-Mail nicht eine zu starke Gewichtung zu verleihen, ist die Anzahl der Nachrichten mit einzubeziehen (z.B. indem man den berechneten Wert durch die Anzahl Nachrichten teilt)
  - Der Algorithmus soll als Strategy Pattern implementiert werden, damit weitere Berechnungsstrategien zu einem späteren implementiert werden können

Entscheide

- Algorithmus als Strategy Pattern implementieren
- AddressQualityEngine ist als Konsolenprogramm zu entwickeln

Nächste Termine

Prof. Dr. Markus Stolze

Projektfortschrittssitzung
Mi, 22.04.2009, 08:10 in Rapperswil

Besprechung eines ersten Entwurfes des Paper Prototypes
Mi, 29.04.2009, 08:10 in Rapperswil

Benedikt Unold

Demonstration Ndr Modul (MS 12)
Mi, 06.05.2009, 14:00 in Zürich

15 – Projektfortschrittssitzung, 22.04.2009

Teilnehmer

- Prof. Dr. Markus Stolze, HSR
- Danny Meier, HSR
- Micha Boller, HSR

Ort / Zeit

- Rapperswil, 08:10 – 08:25

Zweck

- Besprechen Projektfortschritte

Was wurde besprochen

- Besprochen, welche Pendenzen seit dem letzten Treffen aufgearbeitet und erledigt wurden
- Der Paper Prototyp soll den Mainflow abbilden und ein paar wenige Alternativen
    - Es können alle Komponenten / Elemente benutzt werden, die auch Silverlight zur Verfügung stellt
    - Es soll überlegt werden, wie mit Wörtern umgegangen wird die falsch geschrieben wurden (z.B. Meier, Maier, Meyer)
    - Es soll überlegt werden, wo Autocomplete Mechanismen zum Einsatz kommen sollen

Entscheide

- -

Nächste Schritte

- Ndr Modul abschliessen
- Paper Prototyp vorbereiten

Nächste Termine

Prof. Dr. Markus Stolze

Besprechung eines ersten Entwurfes des Paper Prototypes
Mi, 29.04.2009, 08:10 in Rapperswil

Benedikt Unold

Demonstration Ndr Modul (MS 12)
Mi, 06.05.2009, 14:00 in Zürich

16 – Projektfortschrittssitzung, 29.04.2009

Teilnehmer

- Prof. Dr. Markus Stolze, HSR
- Danny Meier, HSR
- Micha Boller, HSR

Ort / Zeit

- Rapperswil, 08:15 – 09:15

Zweck

- Besprechen Paper-Prototypen (E-Mail Nachricht abfragen / Blacklist Eintrag hinzufügen)
- Besprechen Projektfortschritte

Was wurde besprochen

- Contextual Inquiry
  - Den Kundendienstmitarbeiter fragen, mit welchen Applikationen er sich auskennt, um daraus auf die gewohnte Oberfläche zu schliessen
  - Beobachten, wie der Kundendienstmitarbeiter nach einer spezifischen Nachricht in der Applikation Outlook sucht (sortieren, suchen, filtern, …)
  - Abklären wie ein effektives Blacklist-Szenario aussehen könnte (Tragt tatsächlich ein Kundendienstmitarbeiter eine E-Mail Adresse in die Blacklist ein?)
- Paper Prototype „E-Mail Nachricht abfragen“
  - Die Datumsspalte der Tabelle hinzufügen, damit bekannt ist, wann die E-Mail versandt wurde
  - Um eine volle Tabelle zu simulieren, könnte man anstatt drei effektiven Nachrichten alle Zeilen mit „~“ – Platzhaltern füllen um eine ausgefüllte nicht geordnete Tabelle vorzugeben
  - Durch einen „~“ – Platzhalter kann umgangen werden, dass der Benutzer von Anfang an eine Sortierfunktion nutzen möchte auf die Tabelle
  - Bei der Realisierung der übergeblendeten Fenstern (Popup) sollte der Hintergrund ausgegraut werden und mit Transparenz versehen
  - Bei der Detailansicht einer Nachricht könnten folgende Optimierung vorgenommen werden:
    - Fehlerhafte Mails (roter Umschlag) bereits mit Details öffnen
    - Zwei Überschriften machen, einmal mit „Erfolgreich versendete Nachrichten“ und einmal mit „Fehlerhafte Nachrichten“
  - Alternativen zur Suchmaske wären:
    - Eine zentrales Textfeld um eine Suche a la Google zu ermöglichen
    - Oberhalb jeder Spalte ein Textfeld anbieten um direkt zu filtern
- Paper Prototype „Blacklist Eintrag hinzufügen“
  - Eventuell die Möglichkeit hinzufügen, direkt aus der Nachrichtenübersicht ein Empfänger auf die Blacklist zu setzen
  - Beim Hinzufügen eines neuen Musters auch ein Auto-Complete zur Verfügung stellen mit den bekannten Adressen aus der Datenbank
- Suche Allgemein
  - Abklären wie das Suchen nach sprachspezifischen Umlauten erfolgen soll (z.B. é, è, ê, …)

- Erstes Experten-Gespräch
  - Der Inhalt der Präsentation sollte in etwa folgende Schwerpunkte besitzen:
    - Warum wurde das Thema gewählt und wird die Arbeit durchgeführt?
    - Anforderungen
    - Architektur und Architektur Entscheide
    - Evaluation
    - Aktueller Stand der Arbeit
  - Die Dauer sollte unter 30 Minuten liegen, bevorzugt bei ca. 20 Minuten
- Bachelorarbeit Präsentation und Gespräch
  - Wird beim nächsten Treffen vereinbart
  - Herr Huser wird Gegenleser unserer Arbeit sein
- Neben einem IEEE und einem Buch sollte auch ein ACM-Artikel (acm.org) referenziert werden
- In den Anforderungsspezifikationen die Personas auf eine reduzieren, diese jedoch mit „Behavioral Variables“ genauer beschreiben

Entscheide

- -

Nächste Schritte

- Prof. Dr. Markus Stolze die aktuelle Version der Anforderungsspezifikationen zustellen
- Paper Prototype überarbeiten gemäss obigem Feedback
- Präsentation für Experte vorbereiten
- Interview und Ablauf für Contextual Inquiry vorbereiten
- Systemtests für das Ndr Modul

Nächste Termine

Prof. Dr. Markus Stolze

Besprechen des implementierten Ndr Moduls
Mi, 06.05.2009, 08:10 in Rapperswil

Präsentation mit Experte
Mi, 06.05.2009, 09:00 in Rapperswil

Benedikt Unold

Demonstration Ndr Modul (MS 12)
Mi, 06.05.2009, 14:00 in Zürich

17 – Projektfortschrittssitzung, 06.05.2009

Teilnehmer

- Prof. Dr. Markus Stolze, HSR
- Danny Meier, HSR
- Micha Boller, HSR

Ort / Zeit

- Rapperswil, 08:10 – 08:40

Zweck

- Demonstration des NDR Moduls
- Besprechen Projektfortschritte

Was wurde besprochen

- Demonstration des NDR Moduls
  - Eine Möglichkeit zur Qualitätssicherung von selbst erfassten Signaturen zur NDR Erkennung wäre es, dass einzelne Nachrichten, welche aufgrund einer eignen Signatur erkannt wurden, dem Benutzer vorgelegt werden und dieser überprüfen muss, ob das NDR Mail zum erkannten NDR Typ und NDR Kategorie passt
- Besprechung der Resultatdokumentation vom Paper Prototype Test
  - Die Erkenntnisse aus dem Paper Prototype sollten in einem Szenario dokumentiert werden
- Besprechung der Resultatdokumentation vom Contextual Inquiry
  - Die Resultate und Erkenntnisse sollten in die Persona Beschreibung einfliessen
- Problem des Optimistic Lockings erläutert
  - Bei comparis.ch AG die Anforderungen abklären, ob solche eine Strategie überhaupt notwendig und gefordert ist

Entscheide

- Abklären ob Qualitätssicherung von eigenen Signaturen gewünscht ist

Nächste Schritte

- Webapplikation Design fertigerstellen
- Webapplikation implementieren

Nächste Termine

Prof. Dr. Markus Stolze

Projektfortschrittssitzung
Mi, 13.05.2009, 08:10 in Rapperswil

Benedikt Unold

Demonstration Ndr Modul (MS 12)
Mi, 06.05.2009, 14:00 in Zürich

18 – Präsentation für Experte, 06.05.2009

Teilnehmer

- Prof. Dr. Markus Stolze, HSR
- Dr. Marcel Graf, IBM Research
- Danny Meier, HSR
- Micha Boller, HSR

Ort / Zeit

- Rapperswil, 09:00 – 10:00

Zweck

- Experte soll wissen, wer wir sind
- Experte soll wissen, um was es bei dieser Bachelorarbeit geht
- Vereinbarung eines Termins für die Bachelorarbeit Schlusspräsentation

Was wurde besprochen

- Vorführung einer Power Point Präsentation, welche sämtliche Themen der Bachelorarbeit und das Umfeld erläutert
- Besprechung von Unklarheiten und Fragen
- Demonstration des Paper Prototyps
- Tipps von Dr. Marcel Graf für die Schlusspräsentation
    - Detailliertere und graphische Darstellung von Resultaten
        - Transaktionsverhalten darstellen
        - Evaluation und Evaluationsmatrix präsentieren
        - Konkretes Beispiel eines Unit Tests zeigen
        - System-Tests beschreiben, Unterschied zu einem Unit Test beschreiben
        - Handout verteilen

Entscheide

- Der Termin für die Bachelorarbeit Schlusspräsentation wurde auf Freitag, 10.07.2009 09:00 Uhr festgelegt

Nächste Schritte

- Webapplikation Design fertigerstellen
- Webapplikation implementieren

Nächste Termine

Prof. Dr. Markus Stolze

Projektfortschrittssitzung
Mi, 13.05.2009, 08:10 in Rapperswil

Dr. Marcel Graf

Bachelorarbeit Schlusspräsentation
Fr, 10.07.2009, 09:00 in Rapperswil

19 – MS 12 Sitzung, 06.05.2009

Teilnehmer

- Benedikt Unold, comparis.ch AG
- Danny Meier, HSR
- Micha Boller, HSR

Ort / Zeit

- Zürich, 14:00 – 15:00

Zweck

- Demonstration des implementierten Ndr Moduls
- Abnahme des implementierten Ndr Moduls
- Besprechung der implementierten Qualitätsindikator Berechnungs-Strategie

Was wurde besprochen

- Demonstration des Ndr Moduls
  - Es muss beim Abspeichern einer abgerufenen Bounce-Nachricht in der Datenbank zuerst geprüft werden, ob die Guid überhaupt in der Datenbank vorhanden ist (die Guid wird über die Returnpath Adresse aus der Bounce-Nachricht gelesen). Dies muss wegen möglichen Denial-of-Service Attacken gemacht werden
  - Der Status Requeued, welcher eine NdrMail einnimmt, falls sie nochmals verarbeitet werden soll, ist überflüssig. Stattdessen kann der Status Created nochmals eingenommen werden. Dabei ist aber das ProcessDate bereits gesetzt. Deshalb kann eine bereits verarbeitete NdrMail von einer neuen NdrMail unterschieden werden
  - Erläuterung der verschiedenen Qualitätsindikatoren
    - E-Mail Nachrichten mit dem Qualitätsindikator Failed werden von einem Newsletter Versand ausgenommen
    - E-Mail Adressen mit dem Qualitätsindikator Poor oder Failed werden von Umfragemails ausgenommen
- Besprechung der verschiedenen Varianten der Qualitätsindikatoralgorithmus
  - Es soll nicht immer die gesamte Vergangenheit mit in die Berechnung genommen werden sondern es soll eine konfigurierbare Anzahl an Mails beachtet werden

Entscheide

- Anzahl Mails, welche in die Qualitätsindikatorberechnung einfliessen, muss konfigurierbar gemacht werden
- Der Status Requeued, welcher eine NdrMail einnehmen kann, ist überflüssig
- Vor dem Abspeicherung einer NdrMail zuerst in der Datenbank prüfen, ob die Guid vorhanden ist (DOS Attacke verhindern)

Nächste Schritte

- Webapplikation Design fertigerstellen
- Webapplikation implementieren

Nächste Termine

Benedikt Unold

Besprechung Usability Testresultate
Mi, 13.05.2009, 10:00 in Zürich

20 – Paper Prototype Test und Contextual Inquiry, 06.05.2009

Teilnehmer

- Annette Sulzbacher, comparis.ch AG
- Sven, comparis.ch AG
- Danny Meier, HSR
- Micha Boller, HSR

Ort / Zeit

- Zürich, 15:00 – 16:00

Zweck

- Durchführen von Paper Prototype Test
- Durchführen von Contextual Inquiry

Was wurde besprochen

- Durchspielen von Szenario 1 (Nachricht suchen)
    - Beschriftung ‚Empfängername' nicht klar in der Suchmaske (besser getrennte Vorname und Name Suchfelder mit entsprechenden Bezeichnungen)
    - Was ist, wenn in der Tabelle nur ein Empfänger für die E-Mail Nachricht dargestellt wird (weil nach ihm gesucht wird), wird dann in der Detailansicht auch nur dieser Empfänger dargestellt?
    - Als erstes Feld in der Suche sollte die E-Mail Adresse stehen und evtl. die Benutzer-Id.
    - Der Applikationscode und die Referenz in der Detailansicht sind uninteressant
    - In der Suchmaske sind Applikationscode und Mailtyp uninteressant. Was es sicher braucht ist die E-Mail Adresse, Benutzer-Id und evtl. Vorname/Name
    - Der Button ‚Zurücksetzen' soll in ‚Neue Suche' umbenannt werden sowie die Position mit dem Button ‚Suchen' tauschen
    - Spalten ‚Geplanter Versand' und ‚Versanddatum' umdrehen
    - Es soll bereits in der Tabelle ersichtlich sein, ob die E-Mail Nachricht ohne Probleme versendet werden konnte (Hacken) oder ob ein Problem auftrat (Ausrufezeichen)
    - Umbenennung von ‚Nicht erfolgreiche Empfänger' in ‚Versandprobleme' in der Detailansicht
- Durchspielen von Szenario 2 (Blacklist CRUD)
    - Diese Maske kann auf das Minimum vereinfacht werden
        - Suche nach einem Blacklist Eintrag. Falls nicht vorhanden so wird ein Button angeboten ‚In Blacklist übernehmen' resp. ‚Aus Blacklist löschen'
        - Die Suche sollte auf der E-Mail Datenbank stattfinden, damit eine E-Mail Adresse gefunden werde kann (für die Übernahme in die Blacklist)

Entscheide

- Design von Masken entsprechend den Erkenntnissen (wie oben dokumentiert) abändern
- Design für restliche Masken in Visio vorbereiten für die Sitzung am 13.05.2009

Nächste Schritte

- Webapplikation Design fertigerstellen
- Webapplikation implementieren

Nächste Termine

Benedikt Unold

Besprechung Usability Testresultate
Mi, 13.05.2009, 10:00 in Zürich

20 – Projektfortschrittssitzung, 13.05.2009

Teilnehmer

- Prof. Dr. Markus Stolze, HSR
- Danny Meier, HSR
- Micha Boller, HSR

Ort / Zeit

- Rapperswil, 08:10 – 08:30

Zweck

- Besprechen Projektfortschritte

Was wurde besprochen

- Silverlight
  - Beim Deployen muss evtl. das .xap Image editiert werden, da es noch Referenzen zum Debugger enthalten kann
  - Ein mögliches Problem könnte entstehen wenn eine abgefüllte Dropdown-Box mit neuen Werten gefüllt werden soll
- Präsentation
  - Bei der Präsentation sollte ein Handout verteilt werden (z.B. 6 Folien pro Seite)
  - Es sollten keine Informationen auf den Folien erscheinen, die während der Präsentation nicht auch angesprochen werden
  - Wie bereits Dr. Marcel Graf erwähnt hat, sollte aus der Fülle von Informationen eines herausgenommen werden und dieses im Detail erklärt werden
- Contextual Inquiry
  - Das Contextual Inquiry sollte möglichst in der Requirements-Phase erfolgen und in die Anforderungsspezifikation mit einfliessen
  - Das Contextual Inquiry ist nicht dazu gedacht, irgend etwas zu testen
- Paper Prototype
  - Das Erstellen und Durchspielen eines Paper-Prototypen sollte wenn möglich auch sehr früh geschehen (Requirements-Phase), damit man die gewonnenen Informationen in die Anforderungsspezifikation mit aufnehmen kann
- Suchbar
  - Es sollte nochmals mit Annette Sulzbacher besprochen werden, ob die Suchbar ständig eingeblendet werden soll oder eingefahren wird

Entscheide

- -

Nächste Schritte

- Paper-Prototypen in Silverlight umsetzen
- Restliche Low-Fi Prototypen mit Annette Sulzbacher besprechen

Nächste Termine

Prof. Dr. Markus Stolze

Projektfortschrittssitzung
Mi, 20.05.2009, 08:10 in Rapperswil

21 – Besprechung der Low-Fi Prototypen, 13.05.2009

Teilnehmer

- Annette Sulzbacher, comparis.ch AG
- Danny Meier, HSR
- Micha Boller, HSR

Ort / Zeit

- Zürich, 10:00 – 11:00

Zweck

- Besprechen aller Low-Fi Prototypen
- Festlegen auf ein endgültiges Design

Was wurde besprochen

- Jeden erstellten Low-Fi Prototype wurde besprochen. Anmerkungen wurden gleich auf der ausgedruckten Version erfasst
- Das Naming und die Darstellung sollte über alle Ansichten hinweg konsistent gehalten werden

Entscheide

- Es wurde festgelegt, wie die einzelnen Ansichten definitiv aussehen sollen

Nächste Schritte

- Zusenden der Visio-Shapes (Sketch GUI) an Annette Sulzbacher
- Paper-Prototypen sowie Low-Fi Prototypen in Silverlight umsetzen

Nächste Termine

- -

22 – Projektfortschrittssitzung, 20.05.2009

Teilnehmer

- Prof. Dr. Markus Stolze, HSR
- Danny Meier, HSR
- Micha Boller, HSR

Ort / Zeit

- Rapperswil, 08:10 – 08:40

Zweck

- Besprechen Projektfortschritte

Was wurde besprochen

- Erste Abgabe der Dokumentation zur Durchsicht bis 29.05.2009
- Die Schlusspräsentation sollte zwischen 15 – 20 min lang sein
- E-Mail an Herr Huser senden um evtl. einen Termin zu vereinbaren für eine kleine Präsentation

Entscheide

- -

Nächste Schritte

- Silverlight Webapplikation implementiere
- Dokumentation zum Abschluss bringen

Nächste Termine

Prof. Dr. Markus Stolze

Meilensteinsitzung 15
Di, 26.05.2009, 08:10 in Rapperswil

23 – MS 15, 26.05.2009

Teilnehmer

- Prof. Dr. Markus Stolze, HSR
- Benedikt Unold, comparis.ch AG
- Danny Meier, HSR
- Micha Boller, HSR

Ort / Zeit

- Rapperswil, 08:10 – 09:30

Zweck

- Demonstration Silverlight Webapplikation
- Offene Fragen abklären

Was wurde besprochen

- Allgemein
  - Suchabfragen, welche tendenziell sehr viele Abfragen liefern können, sollten folgendes Verhalten aufweisen:
    - Bis zu 1000 Datensätze: Diese im Datagrid mit einem clientseitigen Paging anzeigen
    - Mehr als 1000 Datensätze: Eine Meldung anzeigen, dass die Suchkriterien verfeinert werden sollen
  - In den Hauptmenus sollen alle Buttons zum Anlegen eines neuen Eintrags entfernt werden, da dies bereits über die jeweilige Übersichtsseite erledigt werden kann
  - Die Hauptmenu-Buttons sollen jeweils die aktuell gewählte Seite farblich kennzeichnen
- Ndr Patterns verwalten
  - Das Zurücksetzen von Ndr Mails soll anhand von Kriterien eingegrenzt werden können (z.B. Zeitperiode), damit nicht zu viele auf einmal wieder verarbeitet werden.
- Adressqualitätsindikator
  - Die Kombination aus Qualitätsindikator und E-Mail Adresse bei der Suche muss als Und-Kriterium erfolgen, da sonst zu viele Treffer möglich sind
- Reporting – versendete / zu versendende Nachrichten
  - Die beiden Auswertungen sollen auf einer Seite getätigt werden können, wobei über eine Auswahl die entsprechende Auswertung selektiert wird
  - Beide Auswertungen werden über die Anzahl Stunden vorgenommen
  - Bei den gesendeten Nachrichten ist zusätzlich eine zweite Serie einzublenden, welche die Bounce Nachrichten anzeigt

Entscheide

- Alle Datagrids sollten mit einem clientseitigen Paging versehen werden
- Bei allen Löschoperationen soll zuerst eine Nachfrage erfolgen, ob wirklich gelöscht werden will
- Das Erfassen von Ndr Patterns und das Zurücksetzen von Ndr Mails soll getrennt werden auf verschiedene Seiten

Nächste Schritte

- Silverlight Webapplikation abschliessen
- Dokumentation abschliessen

Nächste Termine

Prof. Dr. Markus Stolze

Projektfortschrittssitzung
Mi, 03.06.2009, 08:10 in Rapperswil

Gemeinsames Abschlusstreffen
Di, 09.06.2009, 12:00 in Rapperswil

Benedikt Unold

Gemeinsames Abschlusstreffen
Di, 09.06.2009, 12:00 in Rapperswil

24 – Projektfortschrittssitzung, 03.06.2009

Teilnehmer

- Prof. Dr. Markus Stolze, HSR
- Danny Meier, HSR
- Micha Boller, HSR

Ort / Zeit

- Rapperswil, 08:10 – 08:20

Zweck

- Besprechen Projektfortschritte

Was wurde besprochen

- Das Management Summary sowie das Inhaltsverzeichnis kann vorab Prof. Dr. Markus Stolze zur Durchsicht abgegeben werden um ein Feedback zu erhalten
- Die Methodik-Reflexion und das Silverlight-Log können im Teil ‚Resultate' dokumentiert werden
- Die Aufgabenstellung ist im Original zu übernehmen, wobei folgende Optionen bestehen:
  - Scannen der Aufgabenstellung und als Bild in die Dokumentation einfügen
  - Aufgabenstellung im Original in die gedruckte Version einfügen
- Prof. Dr. Markus Stolze gibt uns den Link der Webseite bekannt, wo das Institut spezifische Management Summary veröffentlicht

Entscheide

- Prof. Dr. Markus Stolze spricht sich mit Herr Benedikt Unold bezüglich Organisation eines Restaurants für das Abschlusstreffen ab

Nächste Schritte

- Dokumentation abschliessen

Nächste Termine

Prof. Dr. Markus Stolze

Gemeinsames Abschlusstreffen
Di, 09.06.2009, 12:00 in Rapperswil

Benedikt Unold

Gemeinsames Abschlusstreffen
Di, 09.06.2009, 12:00 in Rapperswil

25 – Sitzung zur Besprechung von Dokumenten, 08.06.2009

Teilnehmer

- Prof. Dr. Markus Stolze, HSR
- Micha Boller, HSR

Ort / Zeit

- Rapperswil, 15:00 – 15:30

Zweck

- Besprechen von verschiedenen Dokumenten

Was wurde besprochen

- Besprechen von Abstract
- Besprechen von Management Summary
  - Die Ausgangslage kann etwas kürzer verfasst werden
- Besprechen der Kurzfassung
  - Die Kurzfassung ist gut so und kann abgegeben werden
- Besprechen des Inhaltsverzeichnis für die Gesamtdokumentation
  - Das Inhaltsverzeichnis ist in Ordnung
- Die Bachelorarbeit wird nach dem Bewertungsraster der HSR bewertet
- Angelegenheit bezüglich HSR Forum
  - Eventuell kann Prof. Dr. Markus Stolze aushandeln, dass ein Video unserer Bachelorarbeit und eine vertonte Präsentation vor unserem Plakat gezeigt werden kann
  - Dazu wurde die Software Camtasia Studio zur Verfügung gestellt
  - Dieser Video müsste einfach vor dem HSR Forum erstellt werden
- Präsentation für Herr Huser
  - Eventuell wird vor der Abschlusspräsentation eine kurze Präsentation für Herr Huser gehalten

Entscheide

-

Nächste Schritte

- Dokumentation abschliessen
- Abgabe von Dokumentation
- Abgabe von Kurzbeschreibung
- Abgabe von A0 Poster

Nächste Termine

Prof. Dr. Markus Stolze

Gemeinsames Abschlusstreffen
Di, 09.06.2009, 12:00 in Rapperswil

Benedikt Unold

Gemeinsames Abschlusstreffen
Di, 09.06.2009, 12:00 in Rapperswil